Universidade Técnica de Lisboa
Instituto Superior de Economia e Gestão

Master Thesis in Information Systems Management

Artificial Society Simulation

An Agent-based Experiment

by

Luís Filipe dos Reis Pereira

Supervised by: António Palma dos Reis

September 2011

RESUMO

Esta tese é resultado de um trabalho de investigação feito para a obtenção do grau de mestre em Gestão de Sistemas de Informação. É uma experiência com uma sociedade artificial baseada num modelo de agentes. Foi programado um simulador de sociedades artificiais em JAVA RTS e com ele foram desenhadas e realizadas experiências. A sociedade artificial é baseada num mercado onde é apenas transaccionado um produto. O objectivo foi perceber qual seria a diferença de comportamento dos agentes, sendo estes muito ou pouco reactivos a variações de preço do produto. O resultado foi um conjunto de indicações sobre como calibrar a sociedade artificial de forma a maximizar a sua prosperidade.

ABSTRACT

This thesis is the result of a work done to obtain the degree of Master in Information Systems Management. It is an experiment with an artificial society created with an agent based model. An artificial societies simulator was programmed in JAVA RTS and with it, experiments were designed and performed. The artificial society is based on a market with only one product. The objective was to understand what would be the difference between the behavior of the agents if they have high or low reaction to product price variation. The result was a set of guidelines about how to calibrate the artificial society maximizing its prosperity.

Contents

Chapter 1

Introduction

This work is the design and experiment of an artificial society of agents that interact with each other generating transactions. To achieve this objective an artificial market was created and, in it, experiments were conduced. The work done in this thesis is similar to what was done by (Darley & Outkin, 2007). In their work they have built a simulator and then simulated a financial market. This is the approach followed in this research, although the intention is to simulate a product market.

1.1 The research

Several authors have simulated artificial societies including (Muis, 2010) and (Younger, 2010). Other authors have published guidelines to these experiments such as (Axelrod, 2003) and (Sawyer, 2003). This work is based in trading simulation and social simulation.

1.1.1 Research question

The research question which this work intents to answer is: **"What is the difference between a market with high reaction agents and a market with low reaction agents."** The market is a products market with only one product with which the agents buy and sell. The parameters in analysis are:

1. Number of Agents
2. Stored Product
3. Stored Wealth
4. Age

5. Price

1.2 Agent description

There are two classes of agents: Agent and Government.

Agent is the basis of this work. It is as simple as possible while maintaining enough properties to simulate a living entity. It can be born, grow, reproduce and die. To be able to participate in a market four more properties were added. The ability to produce the product, the ability to sell the product, the ability to store the product and the ability to store money. This agent has enough intelligence to determine the amount of product it delivers to the market and the amount of product it buys in the market. It has one primary objective, to ask Market to transform product into money, and one secondary objective, to clone itself. An Agent destroys a finite amount of product to maintain its existence.

Government is the agent responsible for producing money. It can participate in the market only to buy product. It does not have any other property.

1.3 Working basis

To create the model the platform chosen was Java RTS 2.2. Java was chosen for the simulation of an artificial market because the trading system was based on work of (Bruno & Bollella, 2009). Although the trading system developed has few similarities with the one proposed by (Bruno & Bollella, 2009), the Real-Time capabilities of the Java RTS virtual machine allow determinism on the time events happening during the simulation, such as the amount of product each agent consumes over time. Another reason for choosing this platform was the ability to program with multiple execution threads. Multi-threading allows each agent to have its own thread, which permits agent isolation and the utilization of all the hardware available. The graphical user interface of the simulator is presented in figure 1.1. Since the software developed generates large amounts of data that must be analyzed by external software, two different types of clipboard flavors were programmed: tab-separated values and a special clipboard flavor that allows copy-paste do MATLAB. MATLAB was used to perform a Kolmogorov-Smirnov (Lilliefors version) test on the experiment's residues.

1.4 Introduction to agent based models

What is an agent? According to (Macal & North, 2007) there is no universal definition of what an agent is. Several authors such as (Bonabeau, 2002) argue that an agent should be any kind of component. This includes expressions, formulas and algorithms. Other authors like (Casti & Andersen, 1997) argue that an agent

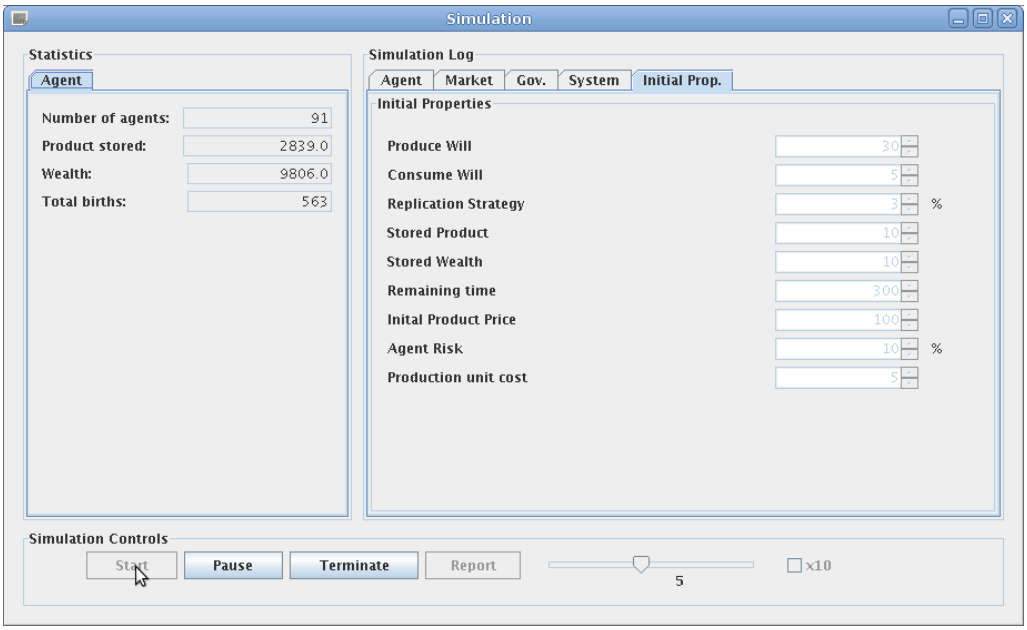

Figure 1.1: Simulators Graphical User Interface

should be able to adapt to the external environment. Most authors agree that an agent must be able to make decisions and be active. Another important aspect of an agent is it's identification. Although anonymous interaction is possible, it is undesirable because it hides the identity of the agent and hides the property of assets in the artificial society, making impossible to answer simple questions like how many agents there are in the artificial society. Another characteristic of an agent is the impossibility to exist outside an environment. Outside an artificial society the agent cannot interact and therefore it is a collection of procedures, algorithms and variables. Other properties of an agent are intrinsic objectives. When an agent is designed, it must be given a purpose and defined the vector of intentions of the agent. Without objectives, the agent does not interact with the society. Also, every agent must be autonomous. Autonomy does not mean that the agent exists in isolation, it means that the agent can decide how to modify the environment or itself without the help from other agents. Autonomy has two implicit properties. The first is the ability to be influenced by itself or by the environment that surrounds it. The second is the ability to influence itself or the environment that surrounds it. Researchers also agree that an agent has rules and norms that describe the behavior of the agent.

1.5 Introduction to artificial societies

What is an artificial society? An artificial society is an environment where agents exist. An agent must have a support to exist and its support is the artificial soci-

ety. Researchers use artificial societies to operationalize concept and mechanisms of interaction between agents with the intention to understand an identifiable phenomena. With the model built, researchers can make modification to it and conduce a scenario analysis to better understand, explain and predict the phenomena. This is done by creating computer programs with representations of entities and procedures of relations between entities. One of the first things to do when a researcher is creating an artificial society is to set it's target. The target is the identified phenomena which the researcher wants to study. In this thesis the phenomena in study is trading. Another thing that must be set prior to the development of the artificial society is the objective, which in this case is the research question. Artificial societies have many applications, including Stock market simulation(LeBaron, 2002), (Levy, Levy, & Solomon, 2000) (Bruno & Bollella, 2009) supply chain simulation (Kimbrough, Wu, & Zhong, 2002)

1.6 Introduction to simulation

According to (Lane, Mansour, & Harpell, 1993) between 1973 and 1988 simulation was on the top three operation research techniques. One of the most important reasons for simulation to be so popular is the number of applications it has. It was used in various productive processes analysis, logistics requirements determination, determination of hardware, software and network requirements, design of roads, ports, airports and railways, business processes reengineering and analysis of financial and economic systems. In this work the simulation technique is applied to artificial societies. Simulation is used in this work because the system designed is too complex to have an easy to achieve exact analytical solution. The simulation presented in this work is dynamic because it changes with time, hence the requirement of precise time control and the adoption of a real-time programming platform. This simulation is a discrete event simulation because it has a countable number of events that happen in a limited time frame. The time advance mechanism chosen is a fixed increment time advance because the transactions happen synchronously. A transaction must be made with two agents and no events can happen when the transaction is being processed. When should simulation be used? (Law & Kelton, 1991) presented an illustrative figure to answer this question. In Figure 1.2 we can see that there are several ways to understand systems. Simulation is one of the last resources available to researchers and it has no impact on reality because the system simulated is a representation of reality. Another remark on Figure 1.2 is that (Law & Kelton, 1991) present no alternative to understand a system without experimentation.

Another idea presented by (Law & Kelton, 1991) is a process to build a simulation. One of the most important steps is the design of the experiments, so the next section is an introduction to design of experiments.

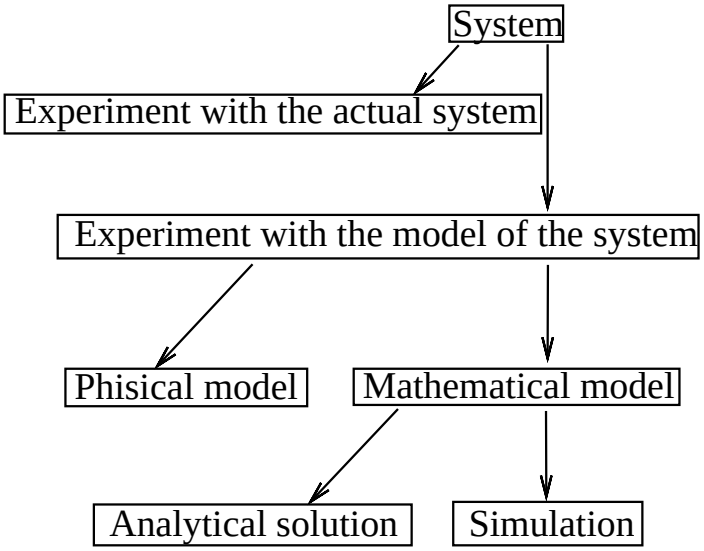

Figure 1.2: Ways to study a system

(Law & Kelton, 1991, p. 4)

1.7 Introduction to design of experiments

According to (Montgomery, 2005), Ronald A. Fisher created the design of experiments theory while working in the Rothamsted agricultural experiment station in the 20's and 30's of the 20th century. He published in 1935 the first book about the subject called the "The Design of Experiments". George Box advanced the design of experiments introducing the Response surface methodology in the 50's and another major advance in design of experiments was introduced by Genichi Taguchi in the 80's. What is design of experiments? Design of experiments is a technique for conducing structured scientific experiments on a process.

Figure 1.3 shows a general process. The design of experiments technique can be applied to generic process if there are control variables and a measurable input and output. Optionally there can be uncontrolled factors; uncontrolled factors are called noise. The objectives of the design of experiments technique are:

1. Determine which factors most influence a response y;
2. Determine which levels of factors x_n maximize y;
3. Determine which levels of factors x_n minimize y;
4. Determine which levels of factors x_n minimize noise.

To work with design of experiments seven steps should be taken to achieve meaningful results: The first is recognition and statement of the problem. In this

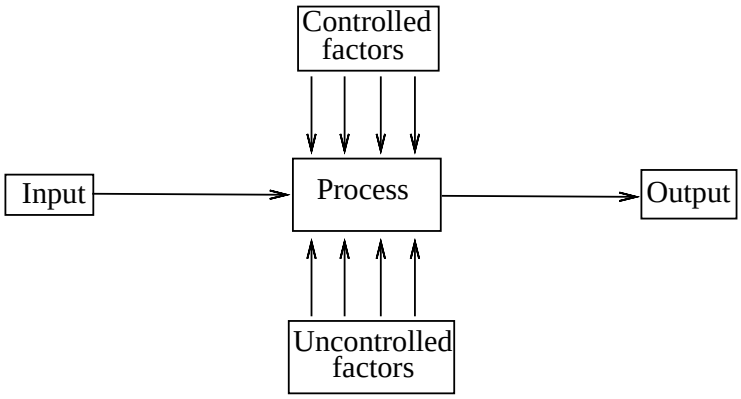

Figure 1.3: General model of process or system

(Montgomery, 2005, p. 2)

step the problem is found and the cause identified. The second is selection of response variables. In this step the researcher chooses what is to be controlled. The third step is choice of factors, levels and ranges. In this step the researcher chooses the controlled factors and their levels. In this step screening experiments should be taken. The fourth is the selection of the experimental design. In this step, variables like time to perform the experiment, cost of the experiment and amount of collected data, should be considered in the decision to adopt a design. The fifth step is the performance of the experiment. In this step the researcher performs the experiments and collects the data for further analysis. The sixth step is statistical analysis of the data. In this step the analysis of the data is performed depending on the type chosen for the experiment. The seventh and last step is conclusion where conclusions and recommendations are reached. This was the methodology followed in this work.

Chapter 2

Creating the agent-based model

2.1 The design of the model

This artificial society model is based on a product market. For simplicity, every agent can only produce one type of product, so there is, intentionally, only one product in this simulation. The market is where the agents exchange the product they produce and each transaction has a fixed quantity of one unit of product and a variable price. At each moment, each agent decides to buy, sell, sit(do nothing), produce or reproduce. First the agent tries to produce the product because it consumes a part of the product it produces. Second, the agent tries to reproduce if there are enough resources to reproduce, because reproduction has a cost in terms of stored product and in terms of stored money. Even if there are stored product and stored money available, they only reproduce with a certain probability. If the condition to reproduce is false and there is enough product stored, the agent tries to sell or buy the product. Since the agent can only exchange one unit of product in each transaction, all the agent has to do is set the price to buy or sell. The price is determined with the following expression:

$$price_i = price_{t-1}\left(\frac{price_{t-1}}{price_{t-2}} - \xi \times \rho\right)$$

Where $price_i$ is the price which each agent uses to ask or bid, $price_{t-1}$ is the price of the last transaction made in the market, $price_{t-2}$ is the price of the second last transaction made in the market, ξ is a random parameter introduced to simulate a perception error each agent has about the market prices and ρ a reactivity sensibility parameter. The reactivity sensibility parameter is used to make the agents more or less tolerant to market price variation. The transaction is always done by the lowest price. For example, if one agent tries to sell for 100 and another agent tries to buy for 110, the transaction is done by 100. This

expression was created to make the agents buy or sell based only in the rising or falling of the price of the product. It was reached after several experiments and was adopted to the simulation because it allowed the agents to increased their stored money. Another important property of Agents it reproduction. Each agent has a window of opportunity to clone itself. Before the window of opportunity the probability of an Agent to reproduce is very little and after the window has passed, the probability of an agent to reproduce is very little. The decision to reproduce is computed as follows: First a random number is generated with Gaussian standard distribution. This number is used to create differences between each agent's decision. Next a cyclic function is used to create a time window of opportunity for reproduction. The complete expression for an agent to be in the "reproduction window" is:

$$a < noise \times \left| sin\left(\frac{\text{SimulationTime} - \text{AgentBornTime}}{K}\right)\right|$$

In this expression a and K are constants tuned to producer the effect of repetition in the reproduction opportunity. a is set to 0,6 and K to 5000 milliseconds. Time is not the only constraint for a Agent to reproduce. An agent must have enough product and money stored. The condition is simple:

$$\text{stored money} > \text{stored product} \times \text{current price} \times \text{reproduction threshold}$$
$$\wedge$$
$$\text{stored product} > \text{produce will} \times \text{production threshold}$$

Reproduction threshold and production threshold are parameters configured at the beginning of the simulation. This procedure to decide if an agent should reproduce was reached by trial and error and was adopted to this simulation because it produced a stable number of agents participating in the market. The values for a and K were discovered during preliminary experiments and provided a way to stabilize the number of agents participating in the market and were not changed during the experimentations. The decider of the agents has the following Activity Diagram:

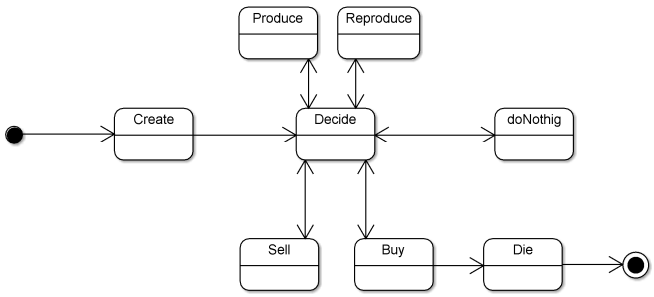

Figure 2.1: State Activity Diagram for the Agent agent

There is a government in this artificial society. It's function is just to maintain price stability. It only buys product so that prices will not go high. This agent also is the only one that can produce money.

Chapter 3

The Experiment

3.1 The hypothesis

In this work it is believed that the market behavior is different if all Agents react fast and if all react slow to market price changes. Therefore a simulation based approach to the hypothesis was conduced. Several authors have already developed simulations with artificial societies. (Izumi & Ueda, 1999) created an artificial market an conduced experiments with the intent to understand what is the effect of interest rates in the stabilization of yen-dollar rates in 1998. Another relevant experiment has been done with artificial societies by (Vidal & Durfee, 1996); they try to understand what is the effect of having a class of agents that can model the behavior of their competitors. The research done is similar to what has already been done and oriented to answer the research question: **"What is the difference between a market with high reaction agents and a market with low reaction agents."**

3.2 The design

In this experiment the 2^k factorial design was adopted. k is the number of factors. The design of this experiment is made with three independent variables. The independent variables and their respective maximum and minimum values are:

1. Variable A: Produce will, $MAX = 35, MIN = 30$
2. Variable B: Replication strategy, $MAX = 5\%, MIN = 1\%$
3. Variable C: Agent reaction, $MAX = 10\%, MIN = 5\%$

The rage for all variables was reached in preliminary experiments. Outside these ranges the artificial society either collapses (the simulation ends because all agents are bankrupt) or expands to a very high number of agents making impossible for the agents to decide in time to seize trading opportunities because of hardware limitations. During preliminary experiments, simulations with more than 250 agents participating in the market caused some agents to lose trading opportunities. To avoid this effect, the experimentations ranges were chosen to provide a significant number of agents (to avoid premature market collapse) while maintaining the number of agents controlled to allow all agents an opportunity to trade if the agent decides to do so.

The dependent variables are:

1. Average number of Agents
2. Average stored money
3. Average stored product
4. Average agent age
5. Average transaction price

In this experiment the following design table was used. In table 3.1 the signs + and − represent the maximum and minimum values of the variables.

Table 3.1: Design table

	A	B	C
(1)	-	-	-
a	+	-	-
b	-	+	-
ab	+	+	-
c	-	-	+
ac	+	-	+
bc	-	+	+
abc	+	+	+

The experiment was repeated two times to have a better estimate on the factors sum of squares.

3.3 The Experimentations

This experimentation had several results. Each result is investigated as a separate response variable. For each response variable, the response data, the ANOVA table, the regression expression and the response surface or response line are presented. For all experiments a significance level of $\alpha = 0,05$ is adopted and the K statistic for the Kolmogorov-Smirnov (Lilliefors version) is presented.

3.3.1 Experimentations results

The result of the experimentation are charts. Charts that contain the time series of each response variable. One of these charts is presented in figure 3.1. In this

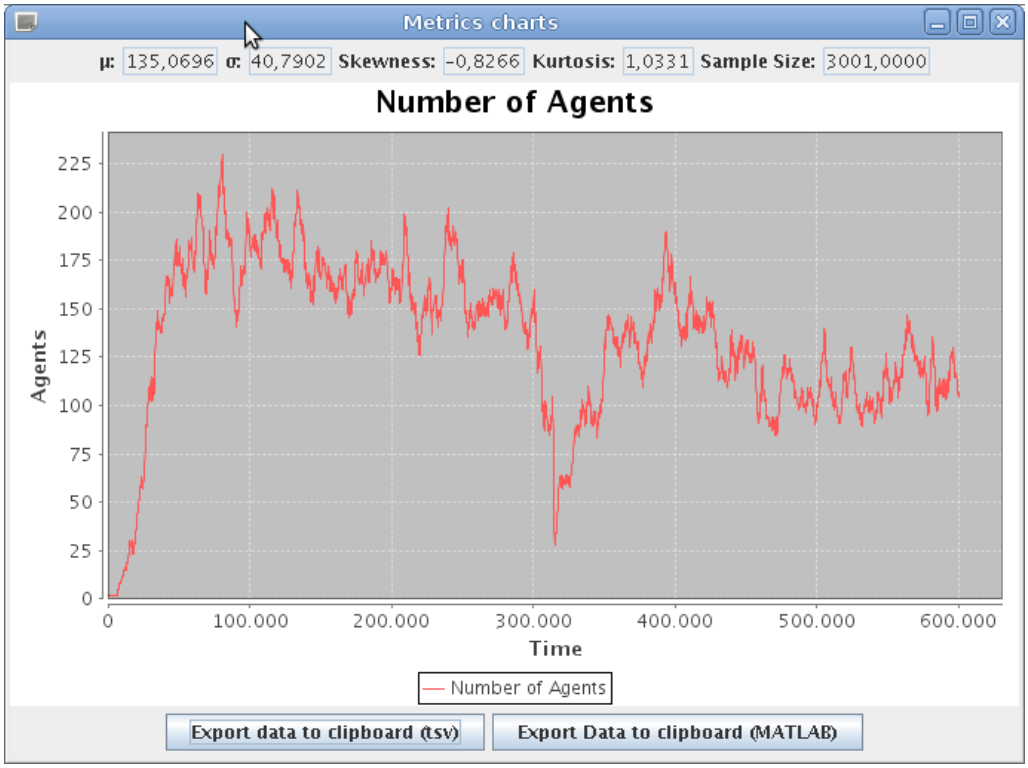

Figure 3.1: Number of agents time series chart

chart we can see that for each response variable several metrics are collected:

1. Average
2. Variance
3. Skewness
4. Kurtosis
5. Sample size

For each chart only the average is analyzed and it is considered the response variable. A time series analysis is not performed, although we can understand several things by visual inspection in this type of charts. After 300 seconds there is a drastic reduction on the number of agents. This reduction is consistent with what happened in all other charts. In Figure 3.2 we can see a drastic reduction in the society's stored product at about the same time the number of agents

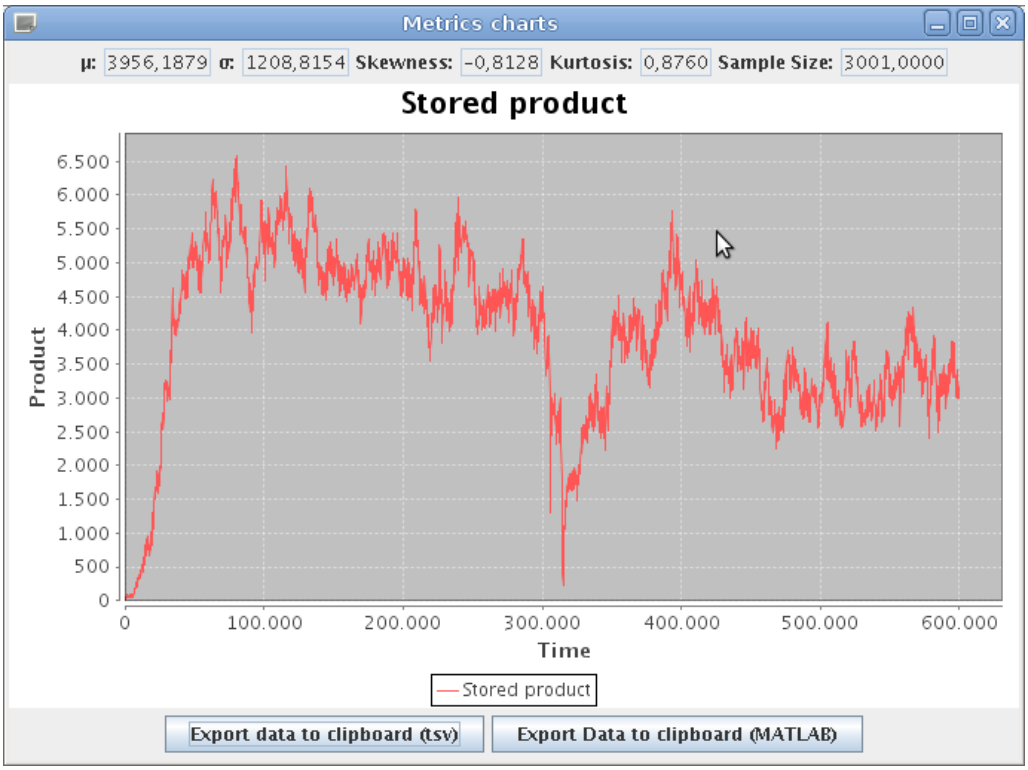

Figure 3.2: Stored product time series chart

is reduced. And at the same time the society's average age chart, presented in figure 3.3 has a peak. This phenomena suggests a bankrupt of a large number of very young players in the market.

3.3.2 The number of Agents response variable

In this sub section, the number of agents response variable is analyzed and some conclusions are presented.

For the agent response variable a detailed analysis is presented. The data is analyzed with the following procedure.

1. Compute the contrasts for each factor;
2. Compute the sum of squares for each factor;
3. Compute the total sum of squares;
4. Build the ANOVA table;
5. Determine the significant factors;
6. Estimate the regression model;

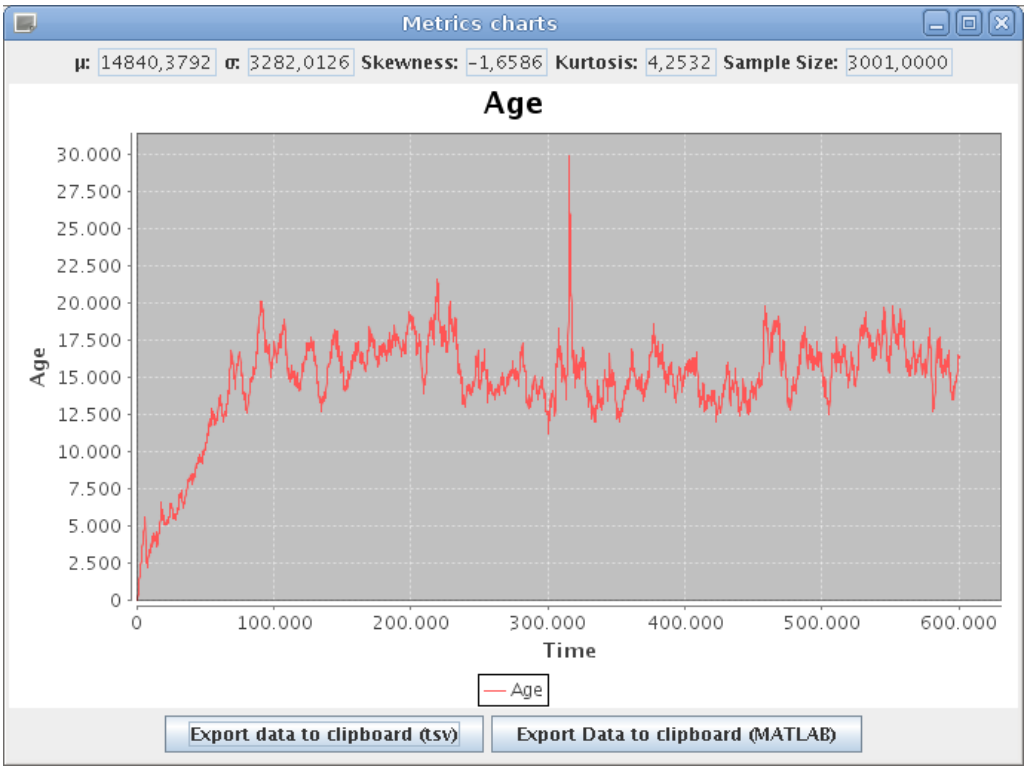

Figure 3.3: Age time series chart

7. Compute the estimated values for each experiment;
8. Compute the observation residues;
9. Verify that the observation residues have a Gaussian distribution;
10. Build the response surface.

Table 3.2: Response data for the number of agents variable

Experiment	Replicate 1	Replicate 2	Sum
(1)	135,0696	144,6180	279,6876
a	190,4538	174,9070	365,3608
b	131,8827	131,5681	263,4508
ab	184,3076	184,3202	368,6278
c	129,2046	122,6714	251,8760
ac	177,2093	179,4099	356,6192
bc	126,0873	122,4808	248,5681
abc	168,7118	177,9090	346,6208

To compute the contrasts for factor A the expression

$$A = \sum y_{A^+} - \sum y_{A^-} = a - (1) + ab - b + ac - c + abc - bc$$

was used. Numerically, with the data from table 3.2, we have the contrast of factor A computed as follows:

$$A = 365,3608 - 279,6876 + 368,6278 - 263,4508 +$$

$$+356,6192 - 251,8760 + 346,6208 - 248,5681 = 393,6461$$

For factors B and C the procedure is the same. To compute the sum of squares we have the expression

$$SS = \frac{contrast^2}{runs \times replicates}$$

which numerically for factor A is

$$SS_A = \frac{393,6461^2}{8 \times 2} = 9684,8283$$

For all factors we have table 3.3.

Table 3.3: Contrasts and sum of square table

Factor	Contrast	Sum of squares
a	393,6461	9684,8283
b	-26,2761	43,1521
c	-73,4429	337,1162
ab	12,8133	10,2613
ac	11,9457	8,9187
bc	-0,3365	0,0071
abc	-26,1943	42,8838

The sum of squares is then used to build the ANOVA table which is presented in table 3.4

Table 3.4: ANOVA table for the number of agents variable

Factor	SS	df	MS	F	p
a	9684,8283	1	9684,8283	324,1141	9,3E-08*
b	43,1521	1	43,1521	1,4441	0,263835
c	337,1162	1	337,1162	11,2820	0,009948*
ab	10,2613	1	10,2613	0,3434	0,574026
ac	8,9187	1	8,9187	0,2985	0,59974
bc	0,0071	1	0,0071	0,0002	0,988098
abc	42,8838	1	42,8838	1,4352	0,26521
Error	239,0474	8	29,8809		
Total	10366,2149	15			

From the ANOVA table it was inferred that the significant factors are (a) - Produce will - and (c) - Agent reaction because they have a p-value less than 0,05.

To build the regression expression we use the expression

$$y = \frac{\sum x}{\text{number of runs}} + \frac{contrast_A}{\text{number of runs}} \times x_a + \frac{contrast_C}{\text{number of runs}} \times x_c$$

which, numerically for the number of Agents response variable is computed as follows

$$y = \beta_0 + \beta_1 \times x_a + \beta_2 \times x_c$$

$$y = 155,1 + 24,6 \times x_a - 4,6 \times x_c$$

In the previous expressions x_a and x_c might assume values between –1 and 1. To use the parameters from the experiment, the following expression should be used:

$$y = 155,1 + (x_a - 32,5) \times 24,6 - (x_c - 7,5) \times 4,6$$

Now we are in condition to validate the model with the residues analysis. In this analysis we compute for every observation the respective estimate from the regression expression and then we can calculate the residue for each observation subtracting the estimated value to the observed value. With this we have the residues table on table 3.5

Table 3.5: Residues table for the number of agents variable

Order	Observed Value	Estimation	Residual
1	135,0696	135,0380	0,0316
2	190,4538	184,2438	6,2100
3	131,8827	135,0380	-3,1553
4	184,3076	184,2438	0,0638
5	129,2046	125,8576	3,3470
6	177,2093	175,0634	2,1459
7	126,0873	125,8576	0,2297
8	168,7118	175,0634	-6,3516
9	144,6180	135,0380	9,5800
10	174,9070	184,2438	-9,3368
11	131,5681	135,0380	-3,4699
12	184,3202	184,2438	0,0764
13	122,6714	125,8576	-3,1862
14	179,4099	175,0634	4,3465
15	122,4808	125,8576	-3,3768
16	177,9090	175,0634	2,8456

Computing the Kolmogorov-Smirnov (Lilliefors version) test statistic we have $k = 0,1276$ and $p - value = 0,2128$ for sample size 16. Since $k < p - value$ there is not enough evidence to reject the test null hypothesis; we can conclude that the residues have Gaussian distribution.

With the regression expression the response surface of the model was built an presented in Figure 3.4

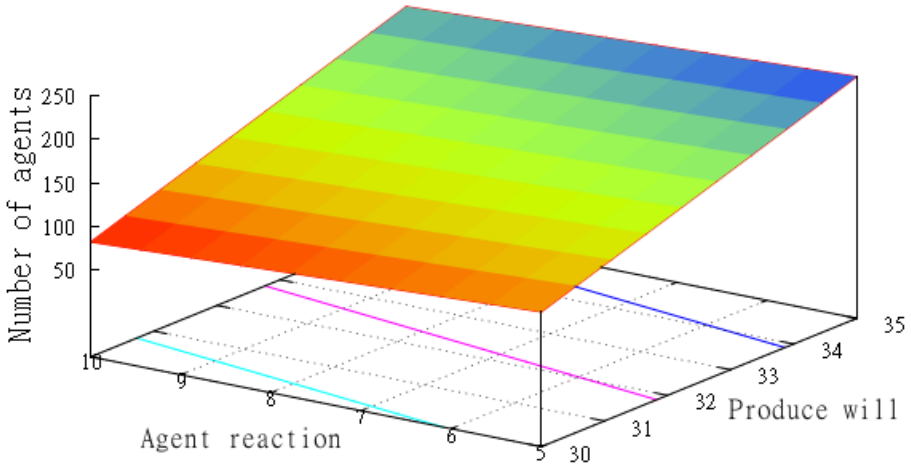

Figure 3.4: Response surface of the Number of agents variable

From this surface we can see how the variable Agent's produce will affects the number of agents. We can also see that the agent reaction has a small negative effect on the number of agents.

3.3.3 The stored product response variable

In this sub section, the stored product response variable is analyzed and some conclusions are presented.

Table 3.6: Response data for the stored product variable

Experiment	Replicate 1	Replicate 2
(1)	3956,1879	4202,7551
a	6656,9537	5817,9560
b	3936,9330	3938,2779
ab	6556,5698	6620,8404
c	3751,3965	3555,6365
ac	6178,8047	6266,6268
bc	3764,3412	3641,0167
abc	6042,6188	6381,2582

From the ANOVA table it was inferred that there is only one significant factor: (a) - Produce will. Therefore the regression expression is:

$$y = \beta_0 + \beta_1 \times x_a$$

Table 3.7: ANOVA table for the stored product variable

Factor	SS	df	MS	F	p
a	24440870,7117	1	24440870,7117	413,9165	3,56E-08*
b	15347,4189	1	15347,4189	0,2599	0,623942
c	276879,7047	1	276879,7047	4,6891	0,062261
ab	46910,4917	1	46910,4917	0,7944	0,398772
ac	18140,8304	1	18140,8304	0,3072	0,59454
bc	7310,1645	1	7310,1645	0,1238	0,734037
abc	76467,7348	1	76467,7348	1,2950	0,288057
Error	472382,6589	8	59047,8324		
Total	25354309,7156	15			

$$y = 5079,3 + 1235,9 \times x_a$$

In the previous expressions x_a might assume values between -1 and 1. To use the parameters from the experiment, the following expression should be used:

$$y = 5079,4 + (x_a - 32,5) \times 1235,9$$

The model reached was validated performing a K-S test on the residues of the estimate, which returned a value of 0,1055. Since the critical value of this test is 0,2128 we can conclude that the residues have Gaussian distribution. With the regression expression the response line of the model was built an presented in Figure 3.5 From this line we can see how the number of agents changes the

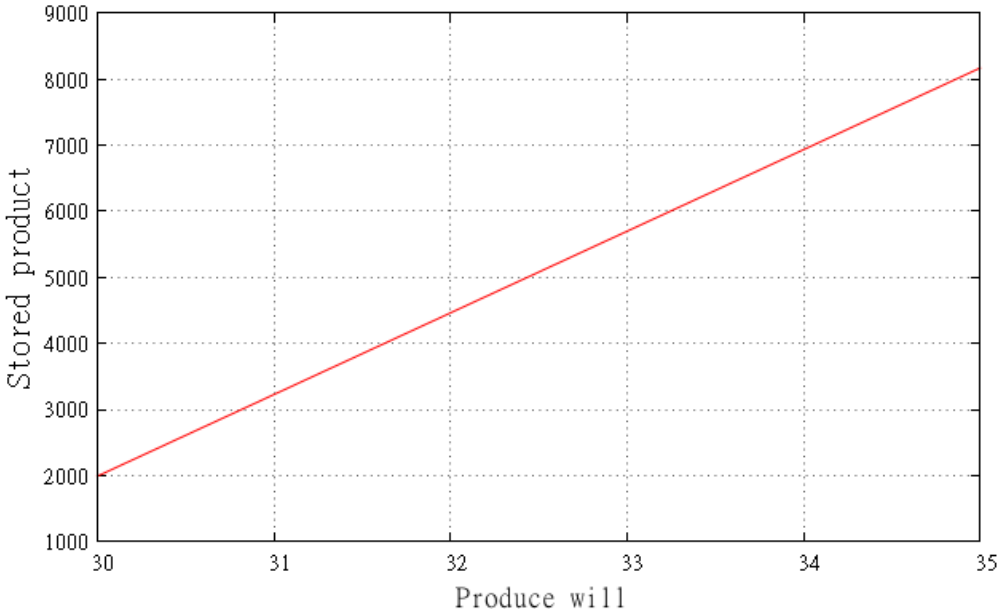

Figure 3.5: Response line of the product stored variable

product stored. An increase in Agent's produce will, has a positive effect on the stored product.

3.3.4 The stored wealth response variable

In this sub section, the stored wealth response variable is analyzed and some conclusions are presented.

Table 3.8: Response data for the stored wealth variable

Experiment	Replicate 1	Replicate 2
(1)	5451,5022	5775,1023
a	8434,1476	6565,4039
b	6779,0883	6823,9194
ab	7716,1303	9813,5641
c	4924,9054	4412,2309
ac	6410,7184	7014,1899
bc	5994,4528	6039,1110
abc	8172,9823	8718,7884

Table 3.9: ANOVA table for the stored wealth variable

Factor	SS	df	MS	F	p
a	17317276,1768	1	17317276,1768	31,0447	0,000527*
b	7658829,3167	1	7658829,3167	13,7300	0,005996*
c	2010354,6280	1	2010354,6280	3,6040	0,094198
ab	53381,1682	1	53381,1682	0,0957	0,76496
ac	97086,0223	1	97086,0223	0,1740	0,687511
bc	98712,8112	1	98712,8112	0,1770	0,685069
abc	23769,5606	1	23769,5606	0,0426	0,841615
Error	4462535,0309	8	557816,8789		
Total	31721944,7147	15			

From the ANOVA table it was inferred that the significant factors are (a) - Produce will - and (b) - Replication Strategy. Therefore the regression expression is

$$y = \beta_0 + \beta_1 \times x_a + \beta_2 \times x_b$$

$$y = 6815,4 + 1040,4 \times x_a - 691,9 \times x_b$$

In the previous expressions x_a and x_b might assume values between -1 and 1. To use the parameters from the experiment, the following expression should be used:

$$y = 6815,4 + (x_a - 32,5) \times 1040,4 + (x_b - 2,5) \times 691,9$$

The model reached was validated performing a K-S test on the residues of the estimate, which returned a value of 0,1508. Since the critical value of this test is

0,2128 we can conclude that the residues have Gaussian distribution. With the regression expression the response surface of the model was built and presented in Figure 3.6

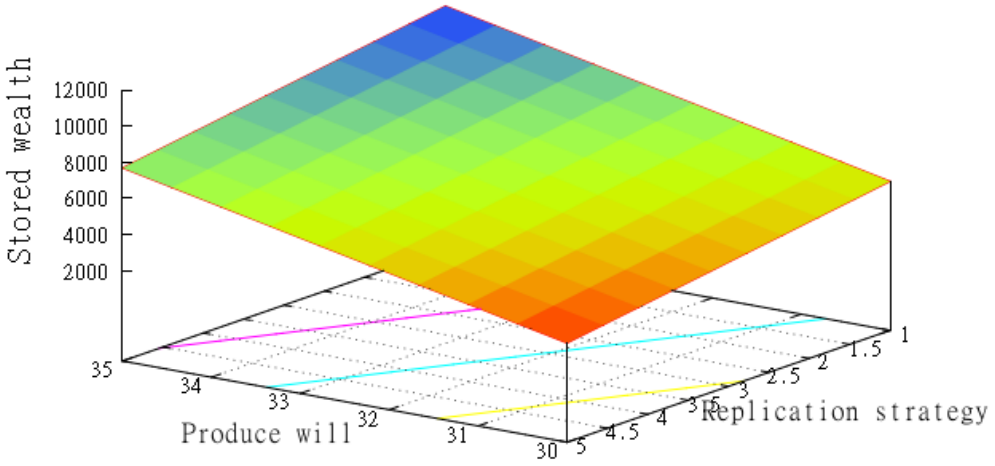

Figure 3.6: Response surface of the Wealth variable

From this surface we can see how the variables Agent's Produce will and Replication Strategy affects the number of agents. Both variables have a positive effect on the wealth.

3.3.5 The age response variable

In this sub section, the Age response variable is analyzed and some conclusions are presented.

Table 3.10: Response data for the Age variable

Experiment	Replicate 1	Replicate 2
(1)	14840,3792	14191,3542
a	19546,6661	16323,3975
b	22188,0010	21462,1066
ab	19416,8134	26778,3859
c	14031,3582	13968,6608
ac	16321,5145	19295,7244
bc	20073,4798	20537,6101
abc	25342,1223	26436,9227

Table 3.11: ANOVA table for the Age variable

Factor	SS	df	MS	F	p
a	49591865,7072	1	49591865,7072	10,4687	0,011957*
b	180340638,8494	1	180340638,8494	38,0695	0,000268*
c	99270,5070	1	99270,5070	0,0210	0,88848
ab	34456,9469	1	34456,9469	0,0073	0,934129
ac	5524560,0846	1	5524560,0846	1,1662	0,311657
bc	916503,2742	1	916503,2742	0,1935	0,671683
abc	3845495,2129	1	3845495,2129	0,8118	0,39391
Error	37897113,3078	8	4737139,1635		
Total	278249903,8899	15			

From the ANOVA table it was inferred that the significant factors are (a) - Produce will - and (b) - Replication Strategy. Therefore the regression expression is

$$y = \beta_0 + \beta_1 \times x_a + \beta_2 \times x_b$$

$$y = 19422,2 + 1760,5 \times x_a + 3357,3 \times x_b$$

In the previous expressions x_a and x_b might assume values between −1 and 1. To use the parameters from the experiment, the following expression should be used:

$$y = 19422,2 + (x_a - 32,5) \times 1760,5 - (x_b - 2,5) \times 3357,3$$

The model reached was validated performing a K-S test on the residues of the estimate, which returned a value of 0,1443. Since the critical value of this test is 0,2128 we can conclude that the residues have Gaussian distribution. With the regression expression the response surface of the model was built an presented in Figure 3.7

From this surface we can see how the variables Produce will and Replication Strategy affects the number of agents. Both variables have a positive effect on the wealth.

3.3.6 The price response variable

In this sub section, the price response variable is analyzed and some conclusions are presented.

From the ANOVA table it was inferred that there are two significant factors (a) - Production will and (c) - Agent reaction. Therefore the regression expression is

$$y = \beta_0 + \beta_1 \times x_a + \beta_2 \times x_c$$

$$y = 92,6 + 0,07259 \times x_a - 2,5 \times x_c$$

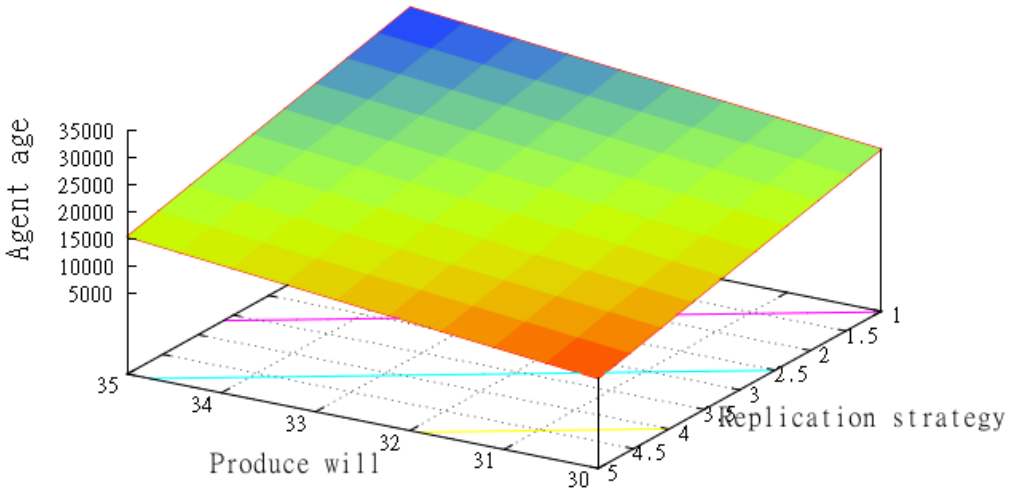

Figure 3.7: Response surface of the Age variable

Table 3.12: Response data for the Price variable

Experiment	Replicate 1	Replicate 2
(1)	95,0237	95,1052
a	95,0753	95,0595
b	94,9043	94,8492
ab	95,2104	94,9778
c	90,0038	89,9899
ac	90,2294	90,2004
bc	89,9064	90,0649
abc	90,1044	90,1517

In the previous expressions x_a might assume values between −1 and 1, and x_c might assume values between −1 and 1. To use the parameters from the experiment, the following expression should be used:

$$y = 92,6 + (x_a - 32,5) \times 0,07259 - (x_c - 7,5) \times 2,5$$

The model reached was validated performing a K-S test on the residues of the estimate, which returned a value of 0,0835. Since the critical value of this test is 0,2128 we can conclude that the residues have Gaussian distribution. With the regression expression the response line of the model was built and presented in Figure 3.8

From Figure 3.8 it is possible to infer that the increase in the Agent reaction has a negative effect on the price of the product, lowering its average value.

Table 3.13: ANOVA table for the Price variable

Factor	SS	df	MS	F	p
a	0,0843	1	0,0843	14,5966	0,005086*
b	0,0168	1	0,0168	2,9043	0,126745
c	97,7849	1	97,7849	16927,9790	1,35E-14*
ab	0,0048	1	0,0048	0,8332	0,388034
ac	0,0049	1	0,0049	0,8501	0,383493
bc	0,0010	1	0,0010	0,1715	0,689662
abc	0,0210	1	0,0210	3,6410	0,092797
Error	0,0462	8	0,0058		
Total	97,9640	15			

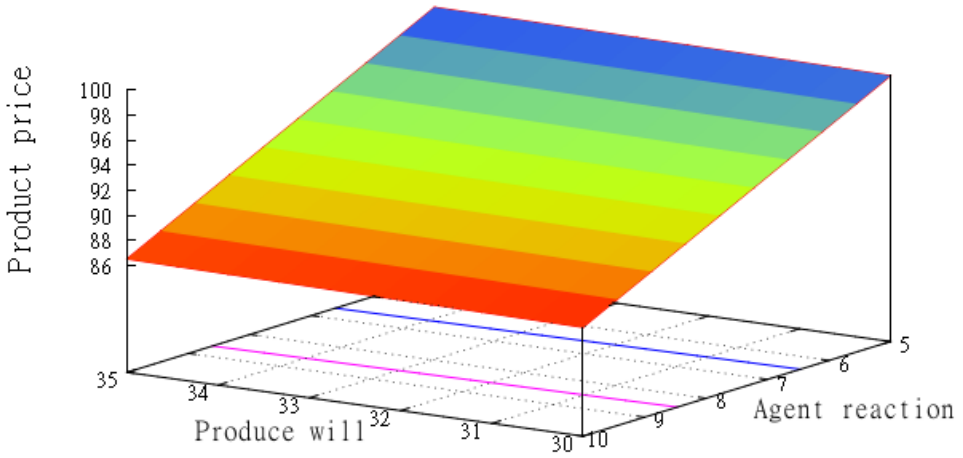

Figure 3.8: Response surface of the Price variable

In Chapter 4 a synthesis of conclusions reached is presented and a comparison between all conclusions reached is made.

Chapter 4

Conclusions

This chapter has the conclusions reached with this work. Some conclusions lead to more questions which could not be addressed in this work due to time and size limitations. Therefore this chapter is divided in two subsections. One with the the conclusions reached an the other with questions that are not answered and suggestions for further research.

4.1 Conclusions of this work

This thesis showed that artificial societies simulation is a technique that can be used to model societies. In this case, a product market was simulated. This market was controlled through a set of parameters called factors. The levels of these factors were changed to see how the society reacted and several conclusions were reached: The most important factor for all experiments is A. The Agent's produce will has a positive effect on all responses. B and C, Replication strategy and agent reaction, respectively, are also important although they have no influence on the stored product response variable. B has always a positive effect while C has always a negative effect. None of the experimented factors were found neglectable and no significant interaction between factors was found. The model created is very reactive to small changes in the experiment's parameters. We can see that a 5% amplitude for factors B and C and a 5 points amplitude for factor A is enough to make significant changes in the artificial society. For this artificial society, if the desired objectives are:

1. maximizing the number of Agents;
2. maximizing stored money;
3. maximizing stored product;

4. maximizing age;

5. minimizing transaction price;

then the correct levels for the factors should be:

1. A Produce will set to its maximum: 35

2. B Replication strategy set to its maximum: 5 %

3. C Agent reaction set no its minimum: 5%

There is a contradiction in the correct level for variable A. Setting the level of variable A to it's maximum has a small undesirable effect on the transaction price minimization objective. Although setting the variable A to it's maximum has the desired effect for all other objectives. So it is recommended to set it to it's maximum. This also was a contradiction on known market laws. Maximizing production, drops market prices in a real market, although this market has a maximum price set by the artificial society government. The effect of the maximization of the production is more than 1000 times smaller than the average price and 35 times smaller than the effect of agent reaction. This means that the effect of produce will on the price of the product is almost neglectable in this market.

4.2 Further research

A direct application of the simulator developed is its use to simulate a real society. The simulation of a real society would lead to conclusions that can be applied to reality. This was not done due to time and size limitations and is left for further research. The simulator developed is also useful to societies design. A simple society was designed, in this case a market, although it could be changed, adapted and calibrated to represent other types of Agent-Based models. Another idea for further research is, to conduct political experiments. The created market has a government which has only one policy: to create money as requested by the agents. Other more complex policies, could be simulated.

Chapter 5

References

This chapter contains all references used in this thesis. All references presented in this chapter are cited in text and all used references are presented in this chapter.

Bibliographic References

Axelrod, R. (2003). Advancing the Art of Simulation in the Social Sciences. *Journal of the Japan Society for Management Information*, *12*(3), 3–16.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(Suppl 3), 7280.

Bruno, E., & Bollella, G. (2009). *Real-Time Java Programming: With Java RTS*. Crawfordsville, Indiana, United States: Prentice Hall.

Casti, J., & Andersen, R. (1997). Would-be worlds: How simulation is changing the frontiers of science. *New York, J*.

Darley, V., & Outkin, A. (2007). *A NASDAQ market simulation: insights on a major market from the science of complex adaptive systems*. Singapore: World Scientific Pub Co Inc.

Izumi, K., & Ueda, K. (1999). Analysis of exchange rate scenarios using an artificial market approach. In *Proceedings of the international conference on artificial intelligence* (Vol. 2, pp. 360–366).

Kimbrough, S., Wu, D., & Zhong, F. (2002). Computers play the beer game: can artificial agents manage supply chains? *Decision Support Systems*, *33*(3), 323–333.

Lane, M., Mansour, A., & Harpell, J. (1993). Operations research techniques: A longitudinal update 1973-1988. *Interfaces*, 63–68.

Law, A., & Kelton, W. (1991). *Simulation modeling and analysis* (Vol. 2). McGraw-Hill New York.

LeBaron, B. (2002). Building the santa fe artificial stock market. *Physica A*.

Levy, M., Levy, H., & Solomon, S. (2000). *The microscopic simulation of financial markets: from investor behavior to market phenomena*. Academic Pr.

Macal, C., & North, M. (2007). Agent-based modeling and simulation: Desktop abms. In *Simulation conference, 2007 winter* (pp. 95–106).

Montgomery, D. C. (2005). *Design and analysis of experiments* (6th Edition ed.). John Wiley & Sons, Inc.

Muis, J. (2010). Simulating political stability and change in the netherlands (1998-2002): an agent-based model of party competition with media effects empirically tested. *Journal of Artificial Societies and Social Simulation*, *13*(2), 4. Available from `http://jasss.soc.surrey.ac.uk/13/2/4.html`

Sawyer, R. (2003). Artificial Societies. *Sociological Methods & Research*, *31*(3), 325.

Vidal, J., & Durfee, E. (1996). The impact of nested agent models in an information economy. In *Proceedings of the second international conference on multi-agent systems* (pp. 377–384).

Younger, S. (2010). Leadership in small societies. *Journal of Artificial Societies and Social Simulation*, *13*(3), 5. Available from `http://jasss.soc.surrey.ac.uk/13/3/5.html`

Appendices

Appendix A

Simulator source code

In this appendix is presented all source code used to construct the artificial society simulator.

Although all the software source code is copyrighted with ASSOFT number 1837/D/11 the reader is invited to download it from `https://aquila2.iseg.utl.pt/aquila/homepage/l37314/simulador-de-sociedades-artificiais`

Agent.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8
9  import java.util.ArrayList;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12 import javax.realtime.RelativeTime;
13 import org.apache.commons.math.MathException;
14 import org.apache.commons.math.random.JDKRandomGenerator↙
       ↪ ;
15 import org.apache.commons.math.distribution.↙
       ↪ NormalDistributionImpl;
```

```
16
17
18
19 /**
20  *
21  * @author Lus F. Reis Pereira
22  */
23 public class Agent implements Runnable{
24     private static volatile boolean SIM_STOP = false;
25     private double produceWill = 0; //Factor controlling↙
       ↪  the amount of product this agent produces
26     private double consumeWill = 1; //Factor controlling↙
       ↪  the amount of product this agent consumes
27                                     //1 = one monetary ↙
                                          ↪ unit per second
28     private double replicationStrategy = 0; //Factor ↙
       ↪ controlling the replication strategy
29     private double replicationIntensity = 0; //Factor ↙
       ↪ controlling the replication Intensity
30     private double storedProduct = 0; //The amount of ↙
       ↪ stored product
31
32
33     private double storedWealth = 0; //The amount of ↙
       ↪ wealth the agent
34     private int AgentID = 0;
35     private long parentAgentID  = 0;
36     private long saudade = 0;
37     private Market market;
38     private final Pool pool;
39     private Log log;
40     private double lastTransactionPrice = 0;
41     private double currentPrice = 0;
42     private double productionThreshold = 0.95;
43     private double reproductionThreshold = 1.05;
44
45     private static int Decision_DoNothig = 0;
46     private static int Decision_Sell = 1;
47     private static int Decision_Buy = 2;
48     private static int Decision_Produce = 3;
49     private static int Decision_Reproduce = 4;
```

```
50      private DashBoard dashboard;
51      private Thread itSelf;
52      private God god;
53      private RelativeTime lastTime  = new RelativeTime↙
           ↪ (0,0);
54      private ArrayList<Agent> AgentList;
55      private double perceptionError=0;
56      private JDKRandomGenerator RandomGenerator;
57      int seed = AgentID;
58      private static volatile int NextID =0;
59      private RelativeTime bournTime ;
60      private RelativeTime deathTime ;
61      private Recorder recorder;
62
63      private static volatile long transactionID = 0;
64
65      public Agent(
66              double inheritedProduceWill,
67              double inheritedConsumerWill,
68              double inheritedReplicationStrategy,
69              double inheritedReplicationIntensity,
70              double inheritedProduct,
71              double inheritedWealth,
72              long inheritedparentAgentID,
73              int SimSpeedValue,
74              Market market,
75              Pool pool, Log log, DashBoard dashBoard,
76              God god, ArrayList<Agent> AgentList,
77              Recorder recorder)
78      {
79          produceWill = inheritedProduceWill;
80          consumeWill = inheritedConsumerWill;
81          reproductionThreshold = ↙
               ↪ inheritedReplicationStrategy / 100;
82          replicationIntensity = ↙
               ↪ inheritedReplicationIntensity;
83          storedProduct = inheritedProduct;
84          storedWealth = inheritedWealth;
85          parentAgentID = inheritedparentAgentID;
86          this.market = market;
87          this.pool = pool;
```

```
88          this.log = log;
89          this.dashboard = dashBoard;
90          this.god=god;
91          this.AgentList = AgentList;
92          this.recorder = recorder;
93          lastTransactionPrice = market.getCurrentPrice();
94
95          addID();
96
97          //connection=log.getConnection();
98
99      }
100     private synchronized void addID(){
101         synchronized(Lock.AgentIDlock)
102         {
103             AgentID = NextID;
104             NextID++;
105         }
106
107     }
108
109     public void run() {
110
111         int decision = 0;
112         double decisionIntensity = 0.0;
113         double P=0;
114         synchronized(dashboard.getjTextArea2()){
115                 dashboard.getjTextArea2().append(this.↙
                        ↪ itSelf.getName() + "****BORN****\n"↙
                        ↪ );
116                 dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard.↙
                        ↪ getjTextArea2().getDocument().↙
                        ↪ getLength());
117                 bournTime = new RelativeTime(God.↙
                        ↪ getSimulationTime());
118         }
119         Boolean firstrun =true;
120         while(storedProduct > 0 )
121         {
122             if(!firstrun){
```

```
123                 god.incrementSemaphore();
124             }
125
126             else
127                 firstrun =false;
128
129
130                 synchronized(Lock.timelock){
131                         try {
132                             while(lastTime.equals(God.↙
                                    ↪ getSimulationTime()))
133                             {Lock.timelock.wait();}
134                         } catch (InterruptedException ex↙
                                ↪ ) {
135                             Logger.getLogger(Agent.class↙
                                    ↪ .getName()).log(Level.↙
                                    ↪ SEVERE, null, ex);
136                         }
137
138                 }
139                 lastTime.set(God.getSimulationTime());
140
141
142             if(SIM_STOP){
143
144                 synchronized(dashboard.getjTextArea2()){
145                 dashboard.getjTextArea2().append(this.↙
                        ↪ itSelf.getName() + "****PAUSED****\↙
                        ↪ n");
146                 dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard.↙
                        ↪ getjTextArea2().getDocument().↙
                        ↪ getLength());
147                 }
148                 synchronized(Lock.lock){
149                         try {
150                             while(SIM_STOP){Lock.lock.↙
                                    ↪ wait();}
151                         } catch (InterruptedException ex↙
                                ↪ ) {
```

```
152                             Logger.getLogger(Agent.class↙
                                    ↪ .getName()).log(Level.↙
                                    ↪ SEVERE, null, ex);
153                         }
154
155                 }
156                 synchronized(dashboard.getjTextArea2()){
157                 dashboard.getjTextArea2().append(this.↙
                        ↪ itSelf.getName() + "****UNPAUSED↙
                        ↪ ****\n");
158                 dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard.↙
                        ↪ getjTextArea2().getDocument().↙
                        ↪ getLength());
159                 }
160             }
161
162             decision = 0;
163             decisionIntensity = 0.0;
164             // Decider
165             //default decision : do Nothig
166             decision = Agent.Decision_DoNothig;
167             if(currentPrice == 0.0)
168             {
169                 currentPrice = Double.parseDouble(↙
                        ↪ dashboard.getjSpinner11().getValue↙
                        ↪ ().toString());
170             }
171             else
172             {
173                 currentPrice = market.getCurrentPrice();
174             }
175             lastTransactionPrice = market.↙
                    ↪ getLastTransactionPrice();
176             if(lastTransactionPrice == 0.0)
177             {
178                 lastTransactionPrice=currentPrice;
179             }
180             RandomGenerator= new JDKRandomGenerator();
181             RandomGenerator.setSeed(seed++);
```

```
182             perceptionError=RandomGenerator.nextGaussian↙
                    ↪ ();
183             try {
184                 P =  currentPrice * (  currentPrice / ↙
                        ↪ lastTransactionPrice
185                             - perceptionError * Double.↙
                                ↪ parseDouble(dashboard.↙
                                ↪ getjSpinner2().getValue().↙
                                ↪ toString())/100);
186                 P = Math.round(P);
187
188             } catch (Exception ex) {
189                 Logger.getLogger(Agent.class.getName()).↙
                        ↪ log(Level.SEVERE,
190                             "CurrentPrice:␣{0}␣↙
                                ↪ LastTransactionPrice:{1}␣↙
                                ↪ perceptionError␣:{2}",
191                             new Object[]{currentPrice, ↙
                                ↪ lastTransactionPrice, ↙
                                ↪ perceptionError});
192                 Logger.getLogger(Agent.class.getName()).↙
                        ↪ log(Level.SEVERE, null, ex);
193             }
194             if( P > 0)
195             {
196                 decision = Agent.Decision_Sell;
197                 decisionIntensity = P;
198             }
199             else if( P < 0)
200             {
201                 if (storedWealth > -P) {
202                     decision = Agent.Decision_Buy;
203                     decisionIntensity = P;
204                 }
205                 else
206                     decision = Agent.Decision_DoNothig;
207             }
208             else if ( P == 0 )
209             {
210                 decision = Agent.Decision_DoNothig;
211             }
```

```
212                 if(storedWealth > storedProduct * ↙
                        ↪ currentPrice * reproductionThreshold &&
213                         storedProduct > produceWill*↙
                                ↪ productionThreshold)
214                 {
215                     Double fertility = RandomGenerator.↙
                            ↪ nextGaussian();
216                     //RelativeTime timeFactor = God.↙
                            ↪ getTimePassed().subtract(new ↙
                            ↪ RelativeTime(6750, 0));
217                     //double fertilityOportunity = Math.abs↙
                            ↪ ((bournTime.subtract(God.↙
                            ↪ getSimulationTime()).↙
                            ↪ getMilliseconds() - 6750L) / God.↙
                            ↪ getTimePassed().getMilliseconds());
218                     double fertilityOportunity = Math.abs(↙
                            ↪ Math.sin(bournTime.subtract(God.↙
                            ↪ getSimulationTime()).↙
                            ↪ getMilliseconds() / 5000L) );
219                     Double NormalVariable = fertility*↙
                            ↪ fertilityOportunity;
220 //                    double probability = 0;
221 //                    try {
222 //                        probability = new ↙
    ↪ NormalDistributionImpl(0.0, 1.0).↙
    ↪ cumulativeProbability(NormalVariable);
223 //                        //System.out.println("prob: " + ↙
    ↪ probability+ "fert: " + fertility + "fertOpo: " + ↙
    ↪ fertilityOportunity);
224 //                    } catch (MathException ex) {
225 //                        Logger.getLogger(Agent.class.↙
    ↪ getName()).log(Level.SEVERE, null, ex);
226 //                    }
227
228
229                     //if (0.25 > probability || probability ↙
                            ↪ > 0.35)
230                     if(NormalVariable >0.6)
231                     {
232                         decision = Agent.Decision_Reproduce;
```

```
233                 decisionIntensity = storedProduct * ↙
                        ↪ currentPrice
234                         * (reproductionThreshold * ↙
                                ↪ 100);
235             }
236         }
237         //produce to sell
238         if((storedWealth < storedProduct * ↙
                ↪ currentPrice * productionThreshold &&
239                 //produce to survive
240                 storedProduct < produceWill*↙
                        ↪ productionThreshold//)
241                 //Must have money to produce
242                 && storedWealth > 0)
243                 )
244         {
245             decision = Agent.Decision_Produce;
246             decisionIntensity = storedProduct * ↙
                    ↪ currentPrice * ( 1 - ↙
                    ↪ productionThreshold);
247         }
248
249         if( decision == Agent.Decision_Buy)
250         {
251             // TODO log decision
252             synchronized(dashboard.getjTextArea2()){
253                 dashboard.getjTextArea2().append(↙
                        ↪ this.itSelf.getName() + "****↙
                        ↪ BUY****@" + Math.abs(P) + "\n"↙
                        ↪ );
254                 dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard.↙
                        ↪ getjTextArea2().getDocument().↙
                        ↪ getLength());
255               }
256             Transaction transaction = new ↙
                    ↪ Transaction();
257             transaction.setAgentIndex(this.AgentID);
258             transaction.setAmount(1L);
259             transaction.setPrice(P);
260             transaction.setTYPE(Transaction.BUY);
```

```
261                 synchronized(this){
262                     transaction.setID(""+transactionID);
263                     transactionID++;
264                 }
265                 transaction.setCreationTime(God.↙
                        ↪ getSimulationTime());
266                 synchronized(pool){pool.setElement(↙
                        ↪ transaction);}
267             }
268             if( decision == Agent.Decision_DoNothig)
269             {
270                 // TODO log decision
271                 synchronized(dashboard.getjTextArea2()){
272                         dashboard.getjTextArea2().↙
                            ↪ append(this.itSelf.getName↙
                            ↪ () + "****DoNothig****" + ↙
                            ↪ P + "\n");
273                         dashboard.getjTextArea2().↙
                            ↪ setCaretPosition(dashboard↙
                            ↪ .getjTextArea2().↙
                            ↪ getDocument().getLength())↙
                            ↪ ;
274                     }
275             }
276             if( decision == Agent.Decision_Produce)
277             {
278                 // TODO log decision
279                 this.storedWealth = this.storedWealth -
280                         Double.parseDouble(dashboard.↙
                            ↪ getjSpinner3().getValue().↙
                            ↪ toString());
281                 synchronized(dashboard.getjTextArea2()){
282                         dashboard.getjTextArea2().↙
                            ↪ append(this.itSelf.getName↙
                            ↪ () + "****Produce****\n");
283                         dashboard.getjTextArea2().↙
                            ↪ setCaretPosition(dashboard↙
                            ↪ .getjTextArea2().↙
                            ↪ getDocument().getLength())↙
                            ↪ ;
284                     }
```

```
285
286                 storedProduct=storedProduct+produceWill;
287                 // Todo Log decision Result
288             }
289             if( decision == Agent.Decision_Sell)
290             {
291                 // TODO log decision
292                 synchronized(dashboard.getjTextArea2()){
293                         dashboard.getjTextArea2().↙
                            ↪ append(this.itSelf.getName↙
                            ↪ () + "****SELL****@" + P +↙
                            ↪  "\n");
294                         dashboard.getjTextArea2().↙
                            ↪ setCaretPosition(dashboard↙
                            ↪ .getjTextArea2().↙
                            ↪ getDocument().getLength())↙
                            ↪ ;
295                     }
296                 // Place ask
297                 Transaction transaction = new ↙
                        ↪ Transaction();
298                 transaction.setAgentIndex(this.↙
                        ↪ AgentID);
299                 transaction.setAmount(1L);
300                 transaction.setPrice(P);
301                 transaction.setTYPE(Transaction.SELL↙
                        ↪ );
302                 synchronized(this){
303                     transaction.setID(""+↙
                            ↪ transactionID);
304                     transactionID++;
305                 }
306                 transaction.setCreationTime(God.↙
                        ↪ getSimulationTime());
307                 synchronized(pool){pool.setElement(↙
                        ↪ transaction);}
308
309
310                 // Todo Log decision Result
311             }
312             if( decision == Agent.Decision_Reproduce)
```

```
313             {
314
315                 Agent tmpAgent = new Agent(
316                 Double.parseDouble(dashboard.↙
                        ↪ getjSpinner1().getValue().toString↙
                        ↪ ()),
317                 Double.parseDouble(dashboard.↙
                        ↪ getjSpinner6().getValue().toString↙
                        ↪ ()),
318                 Double.parseDouble(dashboard.↙
                        ↪ getjSpinner8().getValue().toString↙
                        ↪ ()),
319                 0.0,
320                 Double.parseDouble(dashboard.↙
                        ↪ getjSpinner9().getValue().toString↙
                        ↪ ()),
321                 Double.parseDouble(dashboard.↙
                        ↪ getjSpinner7().getValue().toString↙
                        ↪ ()),
322                 0,
323                 0,
324                 market,
325                 pool, log, dashboard, god, God.↙
                        ↪ getAgentList(), recorder);
326
327                 synchronized(Lock.AgentListlock){
328                     God.getAgentList().add(tmpAgent);
329                     Thread tmpThread = new Thread(↙
                            ↪ tmpAgent, "Thread␣for␣Agent␣↙
                            ↪ number␣" +God.getAgentList().↙
                            ↪ size());
330                     tmpAgent.setItSelf(tmpThread);
331                     tmpThread.start();
332                 }
333
334                 synchronized(this)
335                 {
336                     storedProduct = storedProduct - ↙
                            ↪ Double.parseDouble(dashboard.↙
                            ↪ getjSpinner9().getValue().↙
                            ↪ toString());
```

```
337                     }
338
339
340                     synchronized(dashboard.getjTextArea2()){
341                             dashboard.getjTextArea2().↙
                                ↪ append(this.itSelf.getName↙
                                ↪ () + "****REPRODUCE****␣↙
                                ↪ Created␣Agent␣:" + ↙
                                ↪ tmpAgent.getAgentID() +"\n↙
                                ↪ ");
342                             dashboard.getjTextArea2().↙
                                ↪ setCaretPosition(dashboard↙
                                ↪ .getjTextArea2().↙
                                ↪ getDocument().getLength())↙
                                ↪ ;
343                         }
344                     // TODO reproduce
345                     // Todo Log decision Result
346                 }
347               god.decrementSemaphore();
348             }
349
350             synchronized(Lock.AgentListlock)
351             {
352                 AgentList.set(AgentID, null);
353             }
354             synchronized(dashboard.getjTextArea2()){
355                     dashboard.getjTextArea2().append(this.↙
                        ↪ itSelf.getName() + "****DIES***\n")↙
                        ↪ ;
356 //                  synchronized(Lock.Recorderlock)
357 //                  {
358 //                      deathTime = new RelativeTime(God.↙
    ↪ getSimulationTime());
359 //                      recorder.addValue(Recorder.AGE, ↙
    ↪ new MetricPoint(God.getSimulationTime(), deathTime.↙
    ↪ subtract(bournTime).getMilliseconds()));
360 //                  }
361
362                     synchronized(this)
363                     {
```

```
364                         try {
365                             finalize();
366                         } catch (Throwable ex) {
367                             Logger.getLogger(Agent.class.↙
                                    ↪ getName()).log(Level.SEVERE↙
                                    ↪ , null, ex);
368                         }
369                     }
370 
371 //                  synchronized(Lock.AgentListlock)
372 //                  {
373 //                      synchronized(Lock.AgentIDlock)
374 //                      {
375 //                          AgentList.set(AgentID, null);
376 //                      }
377 //
378 //                  }
379 
380 //                  synchronized(AgentList.get(AgentID))
381 //                  {
382 //                      AgentList.set(AgentID, null);
383 //
384 //                  }
385             }
386         }
387 
388         @Override
389         protected void finalize() throws Throwable
390         {
391             super.finalize();
392             synchronized(Lock.AgentListlock)
393             {
394                 synchronized(Lock.AgentIDlock)
395                 {
396                     AgentList.set(AgentID, null);
397                 }
398 
399             }
400         }
401             public double getConsumeWill() {
402             return consumeWill;
```

```
403     }
404     public double getProduceWill() {
405         return produceWill;
406     }
407     public double getReplicationIntensity() {
408         return replicationIntensity;
409     }
410     public double getReplicationStrategy() {
411         return replicationStrategy;
412     }
413     public double getStoredProduct() {
414         synchronized(Lock.transactionlock)
415         {
416             return storedProduct;
417         }
418     }
419
420     public long getAgentID() {
421         return AgentID;
422     }
423     public long getParentAgentID() {
424         return parentAgentID;
425     }
426
427     public double getStoredWealth() {
428         synchronized(Lock.transactionlock)
429         {
430             return storedWealth;
431         }
432     }
433     public long getSaudade() {
434         return saudade;
435     }
436
437     public static long getTransactionID() {
438         return transactionID;
439     }
440     public static void setSIM_STOP(boolean SIM_STOP) {
441         Agent.SIM_STOP = SIM_STOP;
442     }
443     public Thread getItSelf() {
```

```
444         return itSelf;
445     }
446     public void setItSelf(Thread itSelf) {
447         this.itSelf = itSelf;
448     }
449 
450     public static boolean isSIM_STOP() {
451         return SIM_STOP;
452     }
453 
454     public void setStoredProduct(double storedProduct) {
455         synchronized(Lock.transactionlock)
456         {
457             this.storedProduct = storedProduct;
458         }
459     }
460 
461     public static void setTransactionID(long ↙
        ↪ transactionID) {
462         Agent.transactionID = transactionID;
463     }
464 
465     public void setStoredWealth(double storedWealth) {
466         synchronized(Lock.transactionlock)
467         {
468             this.storedWealth = storedWealth;
469         }
470 
471     }
472 
473     public RelativeTime getBournTime() {
474         return bournTime;
475     }
476 
477 
478 }
```

DashBoard.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  /*
7   * DashBoard.java
8   *
9   * Created on 5/Out/2010, 14:10:35
10  */
11
12 package marketsim;
13
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16 import javax.swing.JButton;
17 import javax.swing.JFrame;
18 import javax.swing.JOptionPane;
19 import javax.swing.JSlider;
20 import javax.swing.JSpinner;
21 import javax.swing.JTextArea;
22 import javax.swing.JTextField;
23
24
25 /**
26  *
27  * @author lfp
28  */
29 public class DashBoard extends javax.swing.JFrame {
30
31     God god;
32     lauchSim lauchSim;
33     /** Creates new form DashBoard */
34     public DashBoard(God god, lauchSim lauchSim) {
35         this.god = god;
36         this.lauchSim=lauchSim;
37         initComponents();
38         jTabbedPane2.setSelectedIndex(4);
39         jSpinner1.setValue(30);
40         jSpinner6.setValue(5);
```

```
41          jSpinner8.setValue(3);
42          jSpinner9.setValue(10);
43          jSpinner7.setValue(10);
44          jSpinner10.setValue(10);
45          jSpinner11.setValue(100);
46          jSpinner3.setValue(5);
47          jSpinner2.setValue(100);
48
49
50          jLabel3.setVisible(false);
51          jLabel4.setText("" + jSlider1.getValue());
52
53  //          System.out.println(System.getProperty("java.↙
        ↪ version"));
54
55      }
56
57      /** This method is called from within the ↙
            ↪ constructor to
58       * initialize the form.
59       * WARNING: Do NOT modify this code. The content of ↙
             ↪ this method is
60       * always regenerated by the Form Editor.
61       */
62      @SuppressWarnings("unchecked")
63      // <editor-fold defaultstate="collapsed" desc="↙
            ↪ Generated Code">//GEN-BEGIN:initComponents
64      private void initComponents() {
65
66          jPanel1 = new javax.swing.JPanel();
67          jButton1 = new javax.swing.JButton();
68          jButton2 = new javax.swing.JButton();
69          jButton3 = new javax.swing.JButton();
70          jButton4 = new javax.swing.JButton();
71          jSlider1 = new javax.swing.JSlider();
72          jCheckBox1 = new javax.swing.JCheckBox();
73          jLabel3 = new javax.swing.JLabel();
74          jLabel4 = new javax.swing.JLabel();
75          jPanel2 = new javax.swing.JPanel();
76          jTabbedPane1 = new javax.swing.JTabbedPane();
77          jPanel4 = new javax.swing.JPanel();
```

```
78          jLabel1 = new javax.swing.JLabel();
79          jLabel2 = new javax.swing.JLabel();
80          jLabel11 = new javax.swing.JLabel();
81          jLabel12 = new javax.swing.JLabel();
82          jTextField1 = new javax.swing.JTextField();
83          jTextField2 = new javax.swing.JTextField();
84          jTextField4 = new javax.swing.JTextField();
85          jTextField6 = new javax.swing.JTextField();
86          jPanel3 = new javax.swing.JPanel();
87          jTabbedPane2 = new javax.swing.JTabbedPane();
88          jPanel8 = new javax.swing.JPanel();
89          jScrollPane2 = new javax.swing.JScrollPane();
90          jTextArea2 = new javax.swing.JTextArea();
91          jPanel10 = new javax.swing.JPanel();
92          jScrollPane3 = new javax.swing.JScrollPane();
93          jTextArea3 = new javax.swing.JTextArea();
94          jPanel11 = new javax.swing.JPanel();
95          jScrollPane4 = new javax.swing.JScrollPane();
96          jTextArea4 = new javax.swing.JTextArea();
97          jPanel9 = new javax.swing.JPanel();
98          jScrollPane5 = new javax.swing.JScrollPane();
99          jTextArea5 = new javax.swing.JTextArea();
100         jPanel13 = new javax.swing.JPanel();
101         jLabel29 = new javax.swing.JLabel();
102         jLabel30 = new javax.swing.JLabel();
103         jLabel33 = new javax.swing.JLabel();
104         jLabel34 = new javax.swing.JLabel();
105         jLabel35 = new javax.swing.JLabel();
106         jLabel36 = new javax.swing.JLabel();
107         jSpinner1 = new javax.swing.JSpinner();
108         jSpinner6 = new javax.swing.JSpinner();
109         jSpinner7 = new javax.swing.JSpinner();
110         jSpinner8 = new javax.swing.JSpinner();
111         jSpinner9 = new javax.swing.JSpinner();
112         jSpinner10 = new javax.swing.JSpinner();
113         jSpinner11 = new javax.swing.JSpinner();
114         jLabel28 = new javax.swing.JLabel();
115         jLabel6 = new javax.swing.JLabel();
116         jSpinner2 = new javax.swing.JSpinner();
117         jLabel7 = new javax.swing.JLabel();
118         jSpinner3 = new javax.swing.JSpinner();
```

```
119         jLabel8 = new javax.swing.JLabel();
120         jLabel9 = new javax.swing.JLabel();
121
122         setDefaultCloseOperation(javax.swing.↙
               ↪ WindowConstants.EXIT_ON_CLOSE);
123         setTitle("Simulation");
124
125         jPanel1.setBorder(javax.swing.BorderFactory.↙
               ↪ createTitledBorder("Simulation Controls"));
126
127         jButton1.setText("Start");
128         jButton1.addActionListener(new java.awt.event.↙
               ↪ ActionListener() {
129             public void actionPerformed(java.awt.event.↙
                   ↪ ActionEvent evt) {
130                 StartSimHandler(evt);
131             }
132         });
133
134         jButton2.setText("Pause");
135         jButton2.setEnabled(false);
136         jButton2.addActionListener(new java.awt.event.↙
               ↪ ActionListener() {
137             public void actionPerformed(java.awt.event.↙
                   ↪ ActionEvent evt) {
138                 PauseResumeHandler(evt);
139             }
140         });
141
142         jButton3.setText("Terminate");
143         jButton3.setEnabled(false);
144         jButton3.addActionListener(new java.awt.event.↙
               ↪ ActionListener() {
145             public void actionPerformed(java.awt.event.↙
                   ↪ ActionEvent evt) {
146                 TerminateHandler(evt);
147             }
148         });
149
150         jButton4.setText("Report");
151         jButton4.setEnabled(false);
```

```
152         jButton4.addActionListener(new java.awt.event.↙
                ↪ ActionListener() {
153             public void actionPerformed(java.awt.event.↙
                    ↪ ActionEvent evt) {
154                 ReportHandler(evt);
155             }
156         });
157 
158         jSlider1.setForeground(new java.awt.Color(255, ↙
                ↪ 0, 0));
159         jSlider1.setMajorTickSpacing(1);
160         jSlider1.setMaximum(10);
161         jSlider1.setMinimum(1);
162         jSlider1.setMinorTickSpacing(1);
163         jSlider1.setSnapToTicks(true);
164         jSlider1.setToolTipText("<html>Simulation␣Speed<↙
                ↪ br>␣<h3>Number␣of␣seconds␣in␣one␣second␣</↙
                ↪ h3></br></html>");
165         jSlider1.setValue(1);
166         jSlider1.addChangeListener(new javax.swing.event↙
                ↪ .ChangeListener() {
167             public void stateChanged(javax.swing.event.↙
                    ↪ ChangeEvent evt) {
168                 jSlider1StateChanged(evt);
169             }
170         });
171         jSlider1.addPropertyChangeListener(new java.↙
                ↪ beans.PropertyChangeListener() {
172             public void propertyChange(java.beans.↙
                    ↪ PropertyChangeEvent evt) {
173                 jSlider1PropertyChange(evt);
174             }
175         });
176 
177         jCheckBox1.setText("x10");
178         jCheckBox1.setToolTipText("<html><h1>Can␣crash␣↙
                ↪ your␣system</h1></html>");
179         jCheckBox1.setSelectedIcon(new javax.swing.↙
                ↪ ImageIcon(getClass().getResource("/↙
                ↪ marketsim/600px-Icon-Warning-Red.svg.png"))↙
                ↪ ); // NOI18N
```

```
180         jCheckBox1.addItemListener(new java.awt.event.↙
                ↪ ItemListener() {
181             public void itemStateChanged(java.awt.event.↙
                    ↪ ItemEvent evt) {
182                 FullSpeedHandler(evt);
183             }
184         });
185
186         jLabel3.setIcon(new javax.swing.ImageIcon(↙
                ↪ getClass().getResource("/marketsim/600px-↙
                ↪ Icon-Warning-Red.svg.png"))); // NOI18N
187         jLabel3.setToolTipText("<html><h1>Can␣crash␣your↙
                ↪ ␣system</h1></html>");
188
189         jLabel4.setText("jLabel4");
190
191         org.jdesktop.layout.GroupLayout jPanel1Layout = ↙
                ↪ new org.jdesktop.layout.GroupLayout(jPanel1↙
                ↪ );
192         jPanel1.setLayout(jPanel1Layout);
193         jPanel1Layout.setHorizontalGroup(
194             jPanel1Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
195             .add(org.jdesktop.layout.GroupLayout.↙
                    ↪ TRAILING, jPanel1Layout.↙
                    ↪ createSequentialGroup()
196                 .add(54, 54, 54)
197                 .add(jButton1, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 71, Short↙
                        ↪ .MAX_VALUE)
198                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
199                 .add(jButton2, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 78, Short↙
                        ↪ .MAX_VALUE)
200                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
201                 .add(jButton3, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 105, ↙
                        ↪ Short.MAX_VALUE)
```

```
202                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
203                 .add(jButton4, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 62, Short↙
                        ↪ .MAX_VALUE)
204                 .add(jPanel1Layout.createParallelGroup(↙
                        ↪ org.jdesktop.layout.GroupLayout.↙
                        ↪ LEADING, false)
205                     .add(jPanel1Layout.↙
                            ↪ createSequentialGroup()
206                         .add(18, 18, 18)
207                         .add(jSlider1, org.jdesktop.↙
                                ↪ layout.GroupLayout.↙
                                ↪ PREFERRED_SIZE, org.↙
                                ↪ jdesktop.layout.GroupLayout↙
                                ↪ .DEFAULT_SIZE, org.jdesktop↙
                                ↪ .layout.GroupLayout.↙
                                ↪ PREFERRED_SIZE)
208                         .add(18, 18, 18))
209                     .add(org.jdesktop.layout.GroupLayout↙
                            ↪ .TRAILING, jPanel1Layout.↙
                            ↪ createSequentialGroup()
210                         .addPreferredGap(org.jdesktop.↙
                                ↪ layout.LayoutStyle.RELATED,↙
                                ↪  org.jdesktop.layout.↙
                                ↪ GroupLayout.DEFAULT_SIZE, ↙
                                ↪ Short.MAX_VALUE)
211                         .add(jLabel4)
212                         .add(89, 89, 89)))
213                 .add(jCheckBox1)
214                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
215                 .add(jLabel3)
216                 .add(148, 148, 148))
217         );
218         jPanel1Layout.setVerticalGroup(
219             jPanel1Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
220             .add(org.jdesktop.layout.GroupLayout.↙
                    ↪ TRAILING, jPanel1Layout.↙
                    ↪ createSequentialGroup()
```

```
221                 .add(jPanel1Layout.createParallelGroup(↙
                        ↪ org.jdesktop.layout.GroupLayout.↙
                        ↪ CENTER)
222                   .add(jButton1)
223                   .add(jButton2)
224                   .add(jButton3)
225                   .add(jButton4)
226                   .add(jSlider1, org.jdesktop.layout.↙
                          ↪ GroupLayout.PREFERRED_SIZE, org↙
                          ↪ .jdesktop.layout.GroupLayout.↙
                          ↪ DEFAULT_SIZE, org.jdesktop.↙
                          ↪ layout.GroupLayout.↙
                          ↪ PREFERRED_SIZE)
227                   .add(jCheckBox1)
228                   .add(jLabel3))
229                 .add(21, 21, 21))
230             .add(org.jdesktop.layout.GroupLayout.↙
                    ↪ TRAILING, jPanel1Layout.↙
                    ↪ createSequentialGroup()
231                 .add(jLabel4)
232                 .addContainerGap())
233         );
234
235         jPanel2.setBorder(javax.swing.BorderFactory.↙
               ↪ createTitledBorder("Statistics"));
236
237         jLabel1.setText("Number of agents:");
238
239         jLabel2.setText("Product stored:");
240
241         jLabel11.setText("Wealth:");
242
243         jLabel12.setText("Total births:");
244
245         jTextField1.setEditable(false);
246         jTextField1.setHorizontalAlignment(javax.swing.↙
               ↪ JTextField.RIGHT);
247         jTextField1.setText("0");
248         jTextField1.addActionListener(new java.awt.event↙
               ↪ .ActionListener() {
```

```
249             public void actionPerformed(java.awt.event.↙
                    ↪ ActionEvent evt) {
250                 jTextField1ActionPerformed(evt);
251             }
252         });
253 
254         jTextField2.setEditable(false);
255         jTextField2.setHorizontalAlignment(javax.swing.↙
                ↪ JTextField.RIGHT);
256         jTextField2.setText("0");
257 
258         jTextField4.setEditable(false);
259         jTextField4.setHorizontalAlignment(javax.swing.↙
                ↪ JTextField.RIGHT);
260         jTextField4.setText("0");
261 
262         jTextField6.setEditable(false);
263         jTextField6.setHorizontalAlignment(javax.swing.↙
                ↪ JTextField.RIGHT);
264         jTextField6.setText("0");
265 
266         org.jdesktop.layout.GroupLayout jPanel4Layout = ↙
                ↪ new org.jdesktop.layout.GroupLayout(jPanel4↙
                ↪ );
267         jPanel4.setLayout(jPanel4Layout);
268         jPanel4Layout.setHorizontalGroup(
269             jPanel4Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
270             .add(jPanel4Layout.createSequentialGroup()
271                 .addContainerGap()
272                 .add(jPanel4Layout.createParallelGroup(↙
                        ↪ org.jdesktop.layout.GroupLayout.↙
                        ↪ LEADING)
273                     .add(jPanel4Layout.↙
                            ↪ createSequentialGroup()
274                         .add(jLabel1)
275                         .addPreferredGap(org.jdesktop.↙
                                ↪ layout.LayoutStyle.RELATED)
276                         .add(jTextField1, org.jdesktop.↙
                                ↪ layout.GroupLayout.↙
                                ↪ DEFAULT_SIZE, 128, Short.↙
```

```
                                    ↪ MAX_VALUE))
277                         .add(jPanel4Layout.↙
                                ↪ createSequentialGroup()
278                             .add(jLabel2)
279                             .add(33, 33, 33)
280                             .add(jTextField2, org.jdesktop.↙
                                    ↪ layout.GroupLayout.↙
                                    ↪ DEFAULT_SIZE, 129, Short.↙
                                    ↪ MAX_VALUE))
281                         .add(jPanel4Layout.↙
                                ↪ createSequentialGroup()
282                             .add(jPanel4Layout.↙
                                    ↪ createParallelGroup(org.↙
                                    ↪ jdesktop.layout.GroupLayout↙
                                    ↪ .LEADING)
283                                 .add(jLabel11)
284                                 .add(jLabel12))
285                             .add(55, 55, 55)
286                             .add(jPanel4Layout.↙
                                    ↪ createParallelGroup(org.↙
                                    ↪ jdesktop.layout.GroupLayout↙
                                    ↪ .LEADING)
287                                 .add(jTextField6, org.↙
                                        ↪ jdesktop.layout.↙
                                        ↪ GroupLayout.↙
                                        ↪ DEFAULT_SIZE, 130, ↙
                                        ↪ Short.MAX_VALUE)
288                                 .add(jTextField4, org.↙
                                        ↪ jdesktop.layout.↙
                                        ↪ GroupLayout.↙
                                        ↪ DEFAULT_SIZE, 130, ↙
                                        ↪ Short.MAX_VALUE))))
289                 .addContainerGap())
290         );
291         jPanel4Layout.setVerticalGroup(
292             jPanel4Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
293             .add(jPanel4Layout.createSequentialGroup()
294                 .addContainerGap()
295                 .add(jPanel4Layout.createParallelGroup(↙
                        ↪ org.jdesktop.layout.GroupLayout.↙
```

```
                ↪ BASELINE)
296               .add(jLabel1)
297               .add(jTextField1, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.DEFAULT_SIZE↙
                      ↪ , org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE))
298             .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
299             .add(jPanel4Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
300               .add(jLabel2)
301               .add(jTextField2, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.DEFAULT_SIZE↙
                      ↪ , org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE))
302             .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
303             .add(jPanel4Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
304               .add(jLabel11)
305               .add(jTextField4, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.DEFAULT_SIZE↙
                      ↪ , org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE))
306             .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
307             .add(jPanel4Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
308               .add(jLabel12)
309               .add(jTextField6, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE, org.jdesktop.↙
```

```
                    ↪ layout.GroupLayout.DEFAULT_SIZE↙
                    ↪ , org.jdesktop.layout.↙
                    ↪ GroupLayout.PREFERRED_SIZE))
310                 .addContainerGap(218, Short.MAX_VALUE))
311         );
312
313         jTabbedPane1.addTab("Agent", jPanel4);
314
315         org.jdesktop.layout.GroupLayout jPanel2Layout = ↙
                ↪ new org.jdesktop.layout.GroupLayout(jPanel2↙
                ↪ );
316         jPanel2.setLayout(jPanel2Layout);
317         jPanel2Layout.setHorizontalGroup(
318             jPanel2Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
319             .add(jTabbedPane1)
320         );
321         jPanel2Layout.setVerticalGroup(
322             jPanel2Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
323             .add(jTabbedPane1, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 399, Short.↙
                    ↪ MAX_VALUE)
324         );
325
326         jPanel3.setBorder(javax.swing.BorderFactory.↙
                ↪ createTitledBorder("Simulation Log"));
327
328         jTextArea2.setColumns(20);
329         jTextArea2.setEditable(false);
330         jTextArea2.setRows(5);
331         jTextArea2.setAutoscrolls(true);
332         jScrollPane2.setViewportView(jTextArea2);
333
334         org.jdesktop.layout.GroupLayout jPanel8Layout = ↙
                ↪ new org.jdesktop.layout.GroupLayout(jPanel8↙
                ↪ );
335         jPanel8.setLayout(jPanel8Layout);
336         jPanel8Layout.setHorizontalGroup(
337             jPanel8Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
```

```
338             .add(jScrollPane2, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 541, Short.↙
                    ↪ MAX_VALUE)
339         );
340         jPanel8Layout.setVerticalGroup(
341             jPanel8Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
342             .add(jScrollPane2, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 356, Short.↙
                    ↪ MAX_VALUE)
343         );
344
345         jTabbedPane2.addTab("Agent", jPanel8);
346
347         jTextArea3.setColumns(20);
348         jTextArea3.setEditable(false);
349         jTextArea3.setRows(5);
350         jScrollPane3.setViewportView(jTextArea3);
351
352         org.jdesktop.layout.GroupLayout jPanel10Layout =↙
                ↪  new org.jdesktop.layout.GroupLayout(↙
                ↪ jPanel10);
353         jPanel10.setLayout(jPanel10Layout);
354         jPanel10Layout.setHorizontalGroup(
355             jPanel10Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
356             .add(jScrollPane3, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 541, Short.↙
                    ↪ MAX_VALUE)
357         );
358         jPanel10Layout.setVerticalGroup(
359             jPanel10Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
360             .add(jScrollPane3, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 356, Short.↙
                    ↪ MAX_VALUE)
361         );
362
363         jTabbedPane2.addTab("Market", jPanel10);
364
365         jTextArea4.setColumns(20);
```

```
366             jTextArea4.setEditable(false);
367             jTextArea4.setRows(5);
368             jScrollPane4.setViewportView(jTextArea4);
369
370             org.jdesktop.layout.GroupLayout jPanel11Layout =↙
                    ↪  new org.jdesktop.layout.GroupLayout(↙
                    ↪ jPanel11);
371             jPanel11.setLayout(jPanel11Layout);
372             jPanel11Layout.setHorizontalGroup(
373                 jPanel11Layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING)
374                 .add(jScrollPane4, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 541, Short.↙
                        ↪ MAX_VALUE)
375             );
376             jPanel11Layout.setVerticalGroup(
377                 jPanel11Layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING)
378                 .add(jScrollPane4, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 356, Short.↙
                        ↪ MAX_VALUE)
379             );
380
381             jTabbedPane2.addTab("Gov.", jPanel11);
382
383             jTextArea5.setColumns(20);
384             jTextArea5.setEditable(false);
385             jTextArea5.setRows(5);
386             jScrollPane5.setViewportView(jTextArea5);
387
388             org.jdesktop.layout.GroupLayout jPanel9Layout = ↙
                    ↪ new org.jdesktop.layout.GroupLayout(jPanel9↙
                    ↪ );
389             jPanel9.setLayout(jPanel9Layout);
390             jPanel9Layout.setHorizontalGroup(
391                 jPanel9Layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING)
392                 .add(jScrollPane5, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 541, Short.↙
                        ↪ MAX_VALUE)
393             );
```

```
394             jPanel9Layout.setVerticalGroup(
395                 jPanel9Layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING)
396                 .add(jScrollPane5, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, 356, Short.↙
                        ↪ MAX_VALUE)
397             );
398 
399             jTabbedPane2.addTab("System", jPanel9);
400 
401             jPanel13.setBorder(javax.swing.BorderFactory.↙
                    ↪ createTitledBorder("Initial Properties"));
402 
403             jLabel29.setText("Produce Will");
404 
405             jLabel30.setText("Consume Will");
406 
407             jLabel33.setText("Replication Strategy");
408 
409             jLabel34.setText("Stored Product");
410 
411             jLabel35.setText("Stored Wealth");
412 
413             jLabel36.setText("Remaining time");
414 
415             jLabel28.setText("Inital Product Price");
416 
417             jLabel6.setText("%");
418 
419             jLabel7.setText("Agent Reaction");
420 
421             jLabel8.setText("Production unit cost");
422 
423             jLabel9.setText("%");
424 
425             org.jdesktop.layout.GroupLayout jPanel13Layout =↙
                    ↪  new org.jdesktop.layout.GroupLayout(↙
                    ↪ jPanel13);
426             jPanel13.setLayout(jPanel13Layout);
427             jPanel13Layout.setHorizontalGroup(
```

```
428                 jPanel13Layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING)
429                 .add(jPanel13Layout.createSequentialGroup()
430                     .addContainerGap()
431                     .add(jPanel13Layout.createParallelGroup(↙
                            ↪ org.jdesktop.layout.GroupLayout.↙
                            ↪ LEADING)
432                         .add(jLabel35)
433                         .add(jLabel36)
434                         .add(jLabel33)
435                         .add(jLabel34)
436                         .add(jPanel13Layout.↙
                                ↪ createParallelGroup(org.↙
                                ↪ jdesktop.layout.GroupLayout.↙
                                ↪ LEADING)
437                             .add(jLabel29)
438                             .add(org.jdesktop.layout.↙
                                    ↪ GroupLayout.TRAILING, ↙
                                    ↪ jLabel30))
439                         .add(jLabel28)
440                         .add(jLabel7)
441                         .add(jLabel8))
442                     .addPreferredGap(org.jdesktop.layout.↙
                            ↪ LayoutStyle.RELATED, 207, Short.↙
                            ↪ MAX_VALUE)
443                     .add(jPanel13Layout.createParallelGroup(↙
                            ↪ org.jdesktop.layout.GroupLayout.↙
                            ↪ LEADING, false)
444                         .add(org.jdesktop.layout.GroupLayout↙
                                ↪ .TRAILING, jSpinner3)
445                         .add(org.jdesktop.layout.GroupLayout↙
                                ↪ .TRAILING, jSpinner2)
446                         .add(org.jdesktop.layout.GroupLayout↙
                                ↪ .TRAILING, jSpinner11)
447                         .add(org.jdesktop.layout.GroupLayout↙
                                ↪ .TRAILING, jSpinner1, org.↙
                                ↪ jdesktop.layout.GroupLayout.↙
                                ↪ DEFAULT_SIZE, 129, Short.↙
                                ↪ MAX_VALUE)
448                         .add(org.jdesktop.layout.GroupLayout↙
                                ↪ .TRAILING, jSpinner6, org.↙
```

```
                    ↪ jdesktop.layout.GroupLayout.↙
                    ↪ DEFAULT_SIZE, 129, Short.↙
                    ↪ MAX_VALUE)
449                 .add(org.jdesktop.layout.GroupLayout↙
                    ↪ .TRAILING, jSpinner8, org.↙
                    ↪ jdesktop.layout.GroupLayout.↙
                    ↪ DEFAULT_SIZE, 129, Short.↙
                    ↪ MAX_VALUE)
450                 .add(org.jdesktop.layout.GroupLayout↙
                    ↪ .TRAILING, jSpinner9, org.↙
                    ↪ jdesktop.layout.GroupLayout.↙
                    ↪ DEFAULT_SIZE, 129, Short.↙
                    ↪ MAX_VALUE)
451                 .add(org.jdesktop.layout.GroupLayout↙
                    ↪ .TRAILING, jSpinner7, org.↙
                    ↪ jdesktop.layout.GroupLayout.↙
                    ↪ DEFAULT_SIZE, 129, Short.↙
                    ↪ MAX_VALUE)
452                 .add(org.jdesktop.layout.GroupLayout↙
                    ↪ .TRAILING, jSpinner10, org.↙
                    ↪ jdesktop.layout.GroupLayout.↙
                    ↪ DEFAULT_SIZE, 129, Short.↙
                    ↪ MAX_VALUE))
453             .addPreferredGap(org.jdesktop.layout.↙
                ↪ LayoutStyle.RELATED)
454             .add(jPanel13Layout.createParallelGroup(↙
                ↪ org.jdesktop.layout.GroupLayout.↙
                ↪ LEADING)
455                 .add(jLabel6)
456                 .add(jLabel9))
457             .addContainerGap(29, Short.MAX_VALUE))
458         );
459         jPanel13Layout.setVerticalGroup(
460             jPanel13Layout.createParallelGroup(org.↙
                ↪ jdesktop.layout.GroupLayout.LEADING)
461             .add(jPanel13Layout.createSequentialGroup()
462                 .add(17, 17, 17)
463                 .add(jPanel13Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
464                     .add(jLabel29)
```

```
465                 .add(jSpinner1, org.jdesktop.layout.↙
                        ↪ GroupLayout.PREFERRED_SIZE, org↙
                        ↪ .jdesktop.layout.GroupLayout.↙
                        ↪ DEFAULT_SIZE, org.jdesktop.↙
                        ↪ layout.GroupLayout.↙
                        ↪ PREFERRED_SIZE))
466             .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
467             .add(jPanel13Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
468                 .add(jLabel30)
469                 .add(jSpinner6, org.jdesktop.layout.↙
                        ↪ GroupLayout.PREFERRED_SIZE, org↙
                        ↪ .jdesktop.layout.GroupLayout.↙
                        ↪ DEFAULT_SIZE, org.jdesktop.↙
                        ↪ layout.GroupLayout.↙
                        ↪ PREFERRED_SIZE))
470             .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
471             .add(jPanel13Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
472                 .add(jLabel33)
473                 .add(jSpinner8, org.jdesktop.layout.↙
                        ↪ GroupLayout.PREFERRED_SIZE, org↙
                        ↪ .jdesktop.layout.GroupLayout.↙
                        ↪ DEFAULT_SIZE, org.jdesktop.↙
                        ↪ layout.GroupLayout.↙
                        ↪ PREFERRED_SIZE)
474                 .add(jLabel6))
475             .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
476             .add(jPanel13Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
477                 .add(jLabel34)
478                 .add(jSpinner9, org.jdesktop.layout.↙
                        ↪ GroupLayout.PREFERRED_SIZE, org↙
                        ↪ .jdesktop.layout.GroupLayout.↙
                        ↪ DEFAULT_SIZE, org.jdesktop.↙
```

```
                    ↪ layout.GroupLayout.↙
                    ↪ PREFERRED_SIZE))
479                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
480                 .add(jPanel13Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
481                   .add(jLabel35)
482                   .add(jSpinner7, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, org↙
                      ↪ .jdesktop.layout.GroupLayout.↙
                      ↪ DEFAULT_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE))
483                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
484                 .add(jPanel13Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
485                   .add(jLabel36)
486                   .add(jSpinner10, org.jdesktop.layout↙
                      ↪ .GroupLayout.PREFERRED_SIZE, ↙
                      ↪ org.jdesktop.layout.GroupLayout↙
                      ↪ .DEFAULT_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE))
487                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
488                 .add(jPanel13Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
489                   .add(jSpinner11, org.jdesktop.layout↙
                      ↪ .GroupLayout.PREFERRED_SIZE, ↙
                      ↪ org.jdesktop.layout.GroupLayout↙
                      ↪ .DEFAULT_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE)
490                   .add(jLabel28))
491                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
```

```
492                     .add(jPanel13Layout.createParallelGroup(↙
                            ↪ org.jdesktop.layout.GroupLayout.↙
                            ↪ BASELINE)
493                       .add(jSpinner2, org.jdesktop.layout.↙
                              ↪ GroupLayout.PREFERRED_SIZE, org↙
                              ↪ .jdesktop.layout.GroupLayout.↙
                              ↪ DEFAULT_SIZE, org.jdesktop.↙
                              ↪ layout.GroupLayout.↙
                              ↪ PREFERRED_SIZE)
494                       .add(jLabel7)
495                       .add(jLabel9))
496                     .addPreferredGap(org.jdesktop.layout.↙
                            ↪ LayoutStyle.RELATED)
497                     .add(jPanel13Layout.createParallelGroup(↙
                            ↪ org.jdesktop.layout.GroupLayout.↙
                            ↪ BASELINE)
498                       .add(jSpinner3, org.jdesktop.layout.↙
                              ↪ GroupLayout.PREFERRED_SIZE, org↙
                              ↪ .jdesktop.layout.GroupLayout.↙
                              ↪ DEFAULT_SIZE, org.jdesktop.↙
                              ↪ layout.GroupLayout.↙
                              ↪ PREFERRED_SIZE)
499                       .add(jLabel8))
500                     .addContainerGap(org.jdesktop.layout.↙
                            ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                            ↪ MAX_VALUE))
501         );
502
503         jTabbedPane2.addTab("Initial␣Prop.", jPanel13);
504
505         org.jdesktop.layout.GroupLayout jPanel3Layout = ↙
              ↪ new org.jdesktop.layout.GroupLayout(jPanel3↙
              ↪ );
506         jPanel3.setLayout(jPanel3Layout);
507         jPanel3Layout.setHorizontalGroup(
508             jPanel3Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
509             .add(org.jdesktop.layout.GroupLayout.↙
                    ↪ TRAILING, jTabbedPane2, org.jdesktop.↙
                    ↪ layout.GroupLayout.DEFAULT_SIZE, 553, ↙
                    ↪ Short.MAX_VALUE)
```

```
510             );
511             jPanel3Layout.setVerticalGroup(
512                 jPanel3Layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING)
513                 .add(org.jdesktop.layout.GroupLayout.↙
                        ↪ TRAILING, jTabbedPane2)
514             );
515
516             org.jdesktop.layout.GroupLayout layout = new org↙
                    ↪ .jdesktop.layout.GroupLayout(getContentPane↙
                    ↪ ());
517             getContentPane().setLayout(layout);
518             layout.setHorizontalGroup(
519                 layout.createParallelGroup(org.jdesktop.↙
                        ↪ layout.GroupLayout.LEADING)
520                 .add(layout.createSequentialGroup()
521                     .addContainerGap()
522                     .add(layout.createParallelGroup(org.↙
                            ↪ jdesktop.layout.GroupLayout.LEADING↙
                            ↪ )
523                         .add(layout.createSequentialGroup()
524                             .add(jPanel2, org.jdesktop.↙
                                    ↪ layout.GroupLayout.↙
                                    ↪ PREFERRED_SIZE, org.↙
                                    ↪ jdesktop.layout.GroupLayout↙
                                    ↪ .DEFAULT_SIZE, org.jdesktop↙
                                    ↪ .layout.GroupLayout.↙
                                    ↪ PREFERRED_SIZE)
525                             .addPreferredGap(org.jdesktop.↙
                                    ↪ layout.LayoutStyle.RELATED)
526                             .add(jPanel3, org.jdesktop.↙
                                    ↪ layout.GroupLayout.↙
                                    ↪ DEFAULT_SIZE, org.jdesktop.↙
                                    ↪ layout.GroupLayout.↙
                                    ↪ DEFAULT_SIZE, Short.↙
                                    ↪ MAX_VALUE))
527                         .add(jPanel1, org.jdesktop.layout.↙
                                ↪ GroupLayout.DEFAULT_SIZE, org.↙
                                ↪ jdesktop.layout.GroupLayout.↙
                                ↪ DEFAULT_SIZE, Short.MAX_VALUE))
528                     .addContainerGap())
```

```
529         );
530         layout.setVerticalGroup(
531             layout.createParallelGroup(org.jdesktop.↙
                ↪ layout.GroupLayout.LEADING)
532             .add(org.jdesktop.layout.GroupLayout.↙
                ↪ TRAILING, layout.createSequentialGroup↙
                ↪ ()
533                 .addContainerGap()
534                 .add(layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING↙
                    ↪ )
535                     .add(jPanel3, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, org.↙
                        ↪ jdesktop.layout.GroupLayout.↙
                        ↪ DEFAULT_SIZE, Short.MAX_VALUE)
536                     .add(jPanel2, org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, org.↙
                        ↪ jdesktop.layout.GroupLayout.↙
                        ↪ DEFAULT_SIZE, Short.MAX_VALUE))
537                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
538                 .add(jPanel1, org.jdesktop.layout.↙
                    ↪ GroupLayout.PREFERRED_SIZE, org.↙
                    ↪ jdesktop.layout.GroupLayout.↙
                    ↪ DEFAULT_SIZE, org.jdesktop.layout.↙
                    ↪ GroupLayout.PREFERRED_SIZE)
539                 .addContainerGap())
540         );
541 
542         pack();
543     }// </editor-fold>//GEN-END:initComponents
544 
545     private void StartSimHandler(java.awt.event.↙
        ↪ ActionEvent evt) {//GEN-FIRST:↙
        ↪ event_StartSimHandler
546         jSpinner1.setEnabled(false);
547         jSpinner6.setEnabled(false);
548         jSpinner8.setEnabled(false);
549         jSpinner9.setEnabled(false);
550         jSpinner7.setEnabled(false);
551         jSpinner10.setEnabled(false);
```

```
552             jSpinner11.setEnabled(false);
553             jSpinner2.setEnabled(false);
554             jSpinner3.setEnabled(false);
555 
556             jSlider1.setEnabled(false);
557             jCheckBox1.setEnabled(false);
558 
559             jButton1.setEnabled(false);
560             jButton2.setEnabled(true);
561             jButton3.setEnabled(true);
562             god.setDelayFactor(jSlider1.getValue());
563             god.setParameters(this, lauchSim.getAgentList(),↙
                    ↪ lauchSim.getMarket(),
564                       lauchSim.getPool(), lauchSim.getLog(), ↙
                              ↪ lauchSim.getRecorder(), lauchSim.↙
                              ↪ getGovernment()          );
565             god.start();
566 
567         }//GEN-LAST:event_StartSimHandler
568 
569         private void PauseResumeHandler(java.awt.event.↙
                ↪ ActionEvent evt) {//GEN-FIRST:↙
                ↪ event_PauseResumeHandler
570             if(evt.getActionCommand().equals("Pause"))
571             {
572                 jButton2.setText("Resume");
573                 synchronized(this){
574                     god.setTimePaused(true);
575                 }
576             }
577             if(evt.getActionCommand().equals("Resume"))
578             {
579                 jButton2.setText("Pause");
580                 synchronized(this){
581                     god.setTimePaused(false);
582                     if(!god.isInterrupted())
583                         god.interrupt();
584                 }
585 
586 //                god.getSimulationClockTimer().enable();
587             }
```

```
588
589     }//GEN-LAST:event_PauseResumeHandler
590
591     private void ReportHandler(java.awt.event.↙
            ↪ ActionEvent evt) {//GEN-FIRST:↙
            ↪ event_ReportHandler
592         // TODO buld simulation report
593         JFrame Report = new Report();
594 //          Thread chart_FREE_MEMORY = new Thread(new ↙
        ↪ ShowCharts(God.getRecorder(), Recorder.FREE_MEMORY,
595 //                  "Free Memory", "Free Memory", "Time", ↙
        ↪ "Memory"));
596 //          chart_FREE_MEMORY.start();
597 //          Thread chart_REAL_TIME_OVERRUNS = new Thread(↙
        ↪ new ShowCharts(God.getRecorder(), Recorder.↙
        ↪ REAL_TIME_OVERRUNS,
598 //                  "Real time overruns", "Real time ↙
        ↪ overruns", "Time", "Overruns"));
599 //          chart_REAL_TIME_OVERRUNS.start();
600
601         //JOptionPane.showMessageDialog(null, "↙
                ↪ Simulation report.");
602         //this.dispose();
603     }//GEN-LAST:event_ReportHandler
604
605     private void TerminateHandler(java.awt.event.↙
            ↪ ActionEvent evt) {//GEN-FIRST:↙
            ↪ event_TerminateHandler
606         jButton2.setEnabled(false);
607         jButton3.setEnabled(false);
608         jButton4.setEnabled(true);
609         synchronized(this){
610                 god.setTimePaused(true);
611             }
612
613
614     }//GEN-LAST:event_TerminateHandler
615
616     private void jSlider1PropertyChange(java.beans.↙
            ↪ PropertyChangeEvent evt) {//GEN-FIRST:↙
            ↪ event_jSlider1PropertyChange
```

```
617
618         }//GEN-LAST:event_jSlider1PropertyChange
619
620         private void jSlider1StateChanged(javax.swing.event.↙
                ↪ ChangeEvent evt) {//GEN-FIRST:↙
                ↪ event_jSlider1StateChanged
621                 god.setDelayFactor(jSlider1.getValue());
622                 jLabel4.setText("" + jSlider1.getValue())↙
                        ↪ ;
623         }//GEN-LAST:event_jSlider1StateChanged
624
625         private void FullSpeedHandler(java.awt.event.↙
                ↪ ItemEvent evt) {//GEN-FIRST:↙
                ↪ event_FullSpeedHandler
626             if(jCheckBox1.isSelected()){
627                  jLabel3.setVisible(true);
628                  jLabel4.setVisible(true);
629                 jSlider1.setPaintLabels(false);
630                 jSlider1.setMaximum(100);
631                 jSlider1.setValue(jSlider1.getValue()*10);
632             }
633             else
634             {
635                 jLabel3.setVisible(false);
636                 jLabel4.setVisible(false);
637                 jSlider1.setMaximum(10);
638                 jSlider1.setMajorTickSpacing(1);
639                 jSlider1.setPaintLabels(true);
640                 if(jSlider1.getValue()>10)jSlider1.setValue↙
                        ↪ (10);
641             }
642         }//GEN-LAST:event_FullSpeedHandler
643
644         private void jTextField1ActionPerformed(java.awt.↙
                ↪ event.ActionEvent evt) {//GEN-FIRST:↙
                ↪ event_jTextField1ActionPerformed
645
646     }//GEN-LAST:event_jTextField1ActionPerformed
647
648
649
```

```
650     // Variables declaration - do not modify//GEN-BEGIN:↙
            ↪ variables
651     private javax.swing.JButton jButton1;
652     private javax.swing.JButton jButton2;
653     private javax.swing.JButton jButton3;
654     private javax.swing.JButton jButton4;
655     private javax.swing.JCheckBox jCheckBox1;
656     private javax.swing.JLabel jLabel1;
657     private javax.swing.JLabel jLabel11;
658     private javax.swing.JLabel jLabel12;
659     private javax.swing.JLabel jLabel2;
660     private javax.swing.JLabel jLabel28;
661     private javax.swing.JLabel jLabel29;
662     private javax.swing.JLabel jLabel3;
663     private javax.swing.JLabel jLabel30;
664     private javax.swing.JLabel jLabel33;
665     private javax.swing.JLabel jLabel34;
666     private javax.swing.JLabel jLabel35;
667     private javax.swing.JLabel jLabel36;
668     private javax.swing.JLabel jLabel4;
669     private javax.swing.JLabel jLabel6;
670     private javax.swing.JLabel jLabel7;
671     private javax.swing.JLabel jLabel8;
672     private javax.swing.JLabel jLabel9;
673     private javax.swing.JPanel jPanel1;
674     private javax.swing.JPanel jPanel10;
675     private javax.swing.JPanel jPanel11;
676     private javax.swing.JPanel jPanel13;
677     private javax.swing.JPanel jPanel2;
678     private javax.swing.JPanel jPanel3;
679     private javax.swing.JPanel jPanel4;
680     private javax.swing.JPanel jPanel8;
681     private javax.swing.JPanel jPanel9;
682     private javax.swing.JScrollPane jScrollPane2;
683     private javax.swing.JScrollPane jScrollPane3;
684     private javax.swing.JScrollPane jScrollPane4;
685     private javax.swing.JScrollPane jScrollPane5;
686     private javax.swing.JSlider jSlider1;
687     private javax.swing.JSpinner jSpinner1;
688     private javax.swing.JSpinner jSpinner10;
689     private javax.swing.JSpinner jSpinner11;
```

```
690     private javax.swing.JSpinner jSpinner2;
691     private javax.swing.JSpinner jSpinner3;
692     private javax.swing.JSpinner jSpinner6;
693     private javax.swing.JSpinner jSpinner7;
694     private javax.swing.JSpinner jSpinner8;
695     private javax.swing.JSpinner jSpinner9;
696     private javax.swing.JTabbedPane jTabbedPane1;
697     private javax.swing.JTabbedPane jTabbedPane2;
698     private javax.swing.JTextArea jTextArea2;
699     private javax.swing.JTextArea jTextArea3;
700     private javax.swing.JTextArea jTextArea4;
701     private javax.swing.JTextArea jTextArea5;
702     private javax.swing.JTextField jTextField1;
703     private javax.swing.JTextField jTextField2;
704     private javax.swing.JTextField jTextField4;
705     private javax.swing.JTextField jTextField6;
706     // End of variables declaration//GEN-END:variables
707
708     public JSlider getjSlider1() {
709         return jSlider1;
710     }
711
712     public JSpinner getjSpinner1() {
713         return jSpinner1;
714     }
715
716     public JSpinner getjSpinner10() {
717         return jSpinner10;
718     }
719
720     public JSpinner getjSpinner6() {
721         return jSpinner6;
722     }
723
724     public JSpinner getjSpinner7() {
725         return jSpinner7;
726     }
727
728     public JSpinner getjSpinner8() {
729         return jSpinner8;
730     }
```

```
731
732     public JSpinner getjSpinner9() {
733         return jSpinner9;
734     }
735
736     public JTextArea getjTextArea2() {
737         return jTextArea2;
738     }
739
740     public JTextArea getjTextArea3() {
741         return jTextArea3;
742     }
743
744     public JTextArea getjTextArea4() {
745         return jTextArea4;
746     }
747
748     public JTextArea getjTextArea5() {
749         return jTextArea5;
750     }
751
752     public JSpinner getjSpinner11() {
753         return jSpinner11;
754     }
755
756     public JTextField getjTextField1() {
757         return jTextField1;
758     }
759
760     public void setjTextField1(JTextField jTextField1) {
761         this.jTextField1 = jTextField1;
762     }
763
764     public JTextField getjTextField2() {
765         return jTextField2;
766     }
767
768     public void setjTextField2(JTextField jTextField2) {
769         this.jTextField2 = jTextField2;
770     }
771
```

```
772     public JTextField getjTextField4() {
773         return jTextField4;
774     }
775 
776     public void setjTextField4(JTextField jTextField4) {
777         this.jTextField4 = jTextField4;
778     }
779 
780     public JTextField getjTextField6() {
781         return jTextField6;
782     }
783 
784     public void setjTextField6(JTextField jTextField6) {
785         this.jTextField6 = jTextField6;
786     }
787 
788 //     public JTextField getjTextField7() {
789 //         return jTextField7;
790 //     }
791 //
792 //     public void setjTextField7(JTextField jTextField7)↙
    ↪  {
793 //         this.jTextField7 = jTextField7;
794 //     }
795 //
796 //     public JTextField getjTextField8() {
797 //         return jTextField8;
798 //     }
799 //
800 //     public void setjTextField8(JTextField jTextField8)↙
    ↪  {
801 //         this.jTextField8 = jTextField8;
802 //     }
803 //
804 //     public JTextField getjTextField9() {
805 //         return jTextField9;
806 //     }
807 //
808 //     public void setjTextField9(JTextField jTextField9)↙
    ↪  {
809 //         this.jTextField9 = jTextField9;
```

```
810 //     }
811
812     public JButton getjButton3()
813     {
814         return jButton3;
815     }
816
817     public JSpinner getjSpinner2() {
818         return jSpinner2;
819     }
820
821     public JSpinner getjSpinner3() {
822         return jSpinner3;
823     }
824
825 }
```

God.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import java.util.ArrayList;
9  import javax.realtime.AbsoluteTime;
10 import javax.realtime.AperiodicParameters;
11 import javax.realtime.AsyncEventHandler;
12 import javax.realtime.Clock;
13 import javax.realtime.PeriodicParameters;
14 import javax.realtime.PriorityParameters;
15 import javax.realtime.PriorityScheduler;
16 import javax.realtime.RealtimeThread;
17 import javax.realtime.RelativeTime;
18 import javax.realtime.ReleaseParameters;
19 import javax.swing.JOptionPane;
20
21
22
23
24 /**
25  *
26  * @author Luis F. Reis Pereira
27  */
28 public class God extends RealtimeThread  {
29
30     private RelativeTime clock;
31     private static RelativeTime simulationClockTimer;
32     private int delayFactor = 1;
33     private DashBoard dashBoard;
34     private boolean TimePaused = false;
35     AsyncEventHandler simulationClockTimerHandler;
36     static private RelativeTime simulationTime = new ↙
          ↪ RelativeTime(0,0);
37     private static ArrayList<Agent> agentList;
38     private Market market;
39     private Pool pool;
```

```
40      private Log log;
41      private boolean END_SIM=false;
42      private static Integer semaphore=0;
43      private static Integer RealTimeOverruns =0;
44      private static Recorder recorder;
45      private static Government government;
46      private static  long impulseLenght;
47      private static AbsoluteTime InitialAbsoluteTime;
48      private static RelativeTime TimePassed;
49
50
51
52
53      public God ()
54      {
55
56
57          clock = Clock.getRealtimeClock().getResolution()↙
                ↪ ;
58          System.out.println("Real-time clock resolution: ↙
                ↪ " + clock.toString());
59
60          setName("God RTT");
61
62           PriorityParameters GodPriorityParameters =
63                  new PriorityParameters(
64                  PriorityScheduler.instance().↙
                        ↪ getMaxPriority());
65         this.setSchedulingParameters(↙
              ↪ GodPriorityParameters);
66      }
67      public void setParameters(DashBoard dashboard, ↙
           ↪ ArrayList<Agent> agentlist, Market market,
68                  Pool pool, Log log, Recorder recorder, ↙
                        ↪ Government government){
69                  this.market = market;
70          this.pool = pool;
71          this.log = log;
72          this.agentList=agentlist;
73          this.dashBoard=dashboard;
```

```
74          this.delayFactor=dashboard.getjSlider1().↙
               ↪ getValue();
75          this.recorder = recorder;
76          this.government = government;
77          this.market = market;
78
79      }
80
81      @Override
82      public void run() {
83
84           ReleaseParameters GodReleaseParameters =
85                  new PeriodicParameters( new RelativeTime↙
                       ↪ (1000, 0));
86           this.setReleaseParameters(GodReleaseParameters)↙
                ↪ ;
87          agentList.add(new Agent(
88                  Double.parseDouble(dashBoard.↙
                       ↪ getjSpinner1().getValue().toString↙
                       ↪ ()),
89                  Double.parseDouble(dashBoard.↙
                       ↪ getjSpinner6().getValue().toString↙
                       ↪ ()),
90                  Double.parseDouble(dashBoard.↙
                       ↪ getjSpinner8().getValue().toString↙
                       ↪ ()),
91                  0.0,
92                  Double.parseDouble(dashBoard.↙
                       ↪ getjSpinner9().getValue().toString↙
                       ↪ ()),
93                  Double.parseDouble(dashBoard.↙
                       ↪ getjSpinner7().getValue().toString↙
                       ↪ ()),
94                  0,
95                  0,
96                  market,
97                  pool, log, dashBoard, this, agentList, ↙
                       ↪ recorder));
98          agentList.add(new Agent(
99                  Double.parseDouble(dashBoard.↙
                       ↪ getjSpinner1().getValue().toString↙
```

```
                ↪ ()),
100             Double.parseDouble(dashBoard.↙
                ↪ getjSpinner6().getValue().toString↙
                ↪ ()),
101             Double.parseDouble(dashBoard.↙
                ↪ getjSpinner8().getValue().toString↙
                ↪ ()),
102             0.0,
103             Double.parseDouble(dashBoard.↙
                ↪ getjSpinner9().getValue().toString↙
                ↪ ()),
104             Double.parseDouble(dashBoard.↙
                ↪ getjSpinner7().getValue().toString↙
                ↪ ()),
105             1,
106             0,
107             market,
108             pool, log, dashBoard, this, agentList, ↙
                ↪ recorder));
109         for(int i = 0;i < agentList.size(); i++)
110         {
111             agentList.get(i).setItSelf(new Thread(↙
                ↪ agentList.get(i), "Thread␣for␣Agent␣↙
                ↪ number␣" + agentList.get(i).getAgentID↙
                ↪ ()));
112             agentList.get(i).getItSelf().start();
113
114         }
115         market.setName("Market␣Thread");
116         market.start();
117         government.setName("Government␣Thread");
118         government.start();
119         simulationClockTimer = new RelativeTime(Long.↙
                ↪ parseLong(""+dashBoard.getjSpinner10().↙
                ↪ getValue())*1000,0);
120         InitialAbsoluteTime = Clock.getRealtimeClock().↙
                ↪ getTime();
121         while(true)
122         {
123
124             this.pool.ResetPool();
```

```
125                 if (semaphore > 0) {
126                     this.setTimePaused(true);
127                     JOptionPane.showMessageDialog(null↙
                            ↪ , "Not enough CPU", "↙
                            ↪ Simulation OVER", JOptionPane↙
                            ↪ .ERROR_MESSAGE);
128                 }
129             synchronized(this) {
130                 GodReleaseParameters =
131                         new PeriodicParameters(
132                         new RelativeTime(1000 / ↙
                                ↪ delayFactor, 0));
133
134                 if (!TimePaused) {
135
136                     simulationTime=simulationTime.add(↙
                            ↪ Math.round(1000/delayFactor), ↙
                            ↪ 0);
137                     impulseLenght = Math.round(1000/↙
                            ↪ delayFactor);
138                     this.setReleaseParameters(↙
                            ↪ GodReleaseParameters);
139                     RealtimeThread.waitForNextPeriod();
140                     TimePassed =
141                             Clock.getRealtimeClock().↙
                                    ↪ getTime().subtract(↙
                                    ↪ InitialAbsoluteTime);
142                     synchronized(Lock.lock){
143                         if(Agent.isSIM_STOP()){
144                             Agent.setSIM_STOP(false);
145                             Government.setSIM_STOP(false↙
                                    ↪ );
146                             Lock.lock.notifyAll();
147                         }
148                     }
149                     synchronized(Lock.timelock){
150                         Lock.timelock.notifyAll();
151                     }
152                     processAgentsLife();
153                     processAgentsStatistics();
154                     processMarketStatistics();
```

```
155                     processGovernmentStatistics();
156                     processSystemStatistics();
157                     if(END_SIM)
158                     {
159                         dashBoard.getjButton3().doClick↙
                               ↪ ();
160                         market.setEND_SIM(true);
161                         government.setEND_SIM(true);
162                     }
163                 } else {
164                     this.setReleaseParameters(new ↙
                           ↪ AperiodicParameters());
165                     this.setReleaseParameters(↙
                           ↪ GodReleaseParameters);
166                     Agent.setSIM_STOP(true);
167                     Government.setSIM_STOP(true);
168                     try {
169                         while (TimePaused) {
170                             this.wait();
171                         }
172 
173                     } catch (InterruptedException ex) {
174 //                      Logger.getLogger(God.class.getName↙
    ↪ ()).log(Level.SEVERE, null, ex);
175                     }
176 
177                 }
178 
179                 dashBoard.getjTextArea5().append("God␣is↙
                       ↪ ␣Alive:␣" + God.getSimulationTime()
180                         + "␣delayFactor:␣" + delayFactor↙
                               ↪ + "\n");
181                 dashBoard.getjTextArea5().↙
                       ↪ setCaretPosition(dashBoard.↙
                       ↪ getjTextArea5().getDocument().↙
                       ↪ getLength());
182             }
183             if(simulationTime.compareTo(↙
                   ↪ simulationClockTimer) > 0 )
184             {
185                 dashBoard.getjButton3().doClick();
```

```
186                 }
187         }
188
189 }
190
191         public int getDelayFactor() {
192             return delayFactor;
193         }
194
195         public void setDelayFactor(int delayFactor) {
196             this.delayFactor = delayFactor;
197
198         }
199
200         public static RelativeTime getSimulationTime() {
201             return simulationTime;
202         }
203
204
205         public RelativeTime getClock() {
206             return clock;
207         }
208
209         public boolean isTimePaused() {
210             return TimePaused;
211         }
212
213         public void setTimePaused(boolean TimePaused) {
214             this.TimePaused = TimePaused;
215         }
216
217         private void processAgentsLife() {
218             synchronized (Lock.AgentListlock) {
219                 for (Agent agent : agentList) {
220                     if (agent != null && agent.↙
                        ↪ getStoredProduct() >0) {
221                      synchronized(agent)
222                      {
223                          agent.setStoredProduct(agent.↙
                             ↪ getStoredProduct()
```

```
224                                 - Double.parseDouble(↙
                                        ↪ dashBoard.getjSpinner6↙
                                        ↪ ().getValue().toString↙
                                        ↪ ()));
225                         }
226                     }
227                 }
228             }
229         }
230
231         private void processAgentsStatistics() {
232
233             double totalProductInSociety = 0;
234             double totalMoneyInSociety= 0;
235             int totalLiveAgents=0;
236             long sumOfAges = 0;
237             synchronized (Lock.AgentListlock) {
238                 for (Agent agent : agentList) {
239                     if (agent != null && agent.↙
                            ↪ getStoredProduct() >0) {
240                         totalLiveAgents++;
241                         totalProductInSociety = ↙
                                ↪ totalProductInSociety + agent.↙
                                ↪ getStoredProduct();
242                         totalMoneyInSociety = ↙
                                ↪ totalMoneyInSociety + agent.↙
                                ↪ getStoredWealth();
243                         if(agent.getBournTime() != null)
244                         sumOfAges = sumOfAges + ↙
                                ↪ simulationTime.subtract(agent.↙
                                ↪ getBournTime()).getMilliseconds↙
                                ↪ ();
245
246                     }
247
248                 }
249                 //System.out.println("\n");
250             }
251             if(totalLiveAgents <1)
252                 END_SIM = true;
253             if (!END_SIM) {
```

```
254             synchronized (Lock.Recorderlock) {
255
256
257                 recorder.addValue(Recorder.↙
                        ↪ NUMBER_OF_AGENTS, new MetricPoint(↙
                        ↪ simulationTime, totalLiveAgents));
258                 dashBoard.getjTextField1().setText("" + ↙
                        ↪ totalLiveAgents);
259                 recorder.addValue(Recorder.↙
                        ↪ PRODUCT_STORED, new MetricPoint(↙
                        ↪ simulationTime, ↙
                        ↪ totalProductInSociety));
260                 dashBoard.getjTextField2().setText("" + ↙
                        ↪ totalProductInSociety);
261                 //recorder.addValue(Recorder.↙
                        ↪ PRODUCTION_RATE, new MetricPoint(↙
                        ↪ simulationTime, ↙
                        ↪ totalProductInSociety/impulseLenght↙
                        ↪ ));
262                 //dashBoard.getjTextField3().setText(""+↙
                        ↪  totalProductInSociety/↙
                        ↪ impulseLenght);
263                 recorder.addValue(Recorder.WEALTH, new ↙
                        ↪ MetricPoint(simulationTime, ↙
                        ↪ totalMoneyInSociety));
264                 dashBoard.getjTextField4().setText("" + ↙
                        ↪ totalMoneyInSociety);
265                 //recorder.addValue(Recorder.↙
                        ↪ WEALTH_GROWTH_RATE, new MetricPoint↙
                        ↪ (simulationTime, ↙
                        ↪ totalMoneyInSociety/impulseLenght))↙
                        ↪ ;
266                 //dashBoard.getjTextField5().setText(""+↙
                        ↪  totalMoneyInSociety/impulseLenght)↙
                        ↪ ;
267                 recorder.addValue(Recorder.TOTAL_BIRTHS,↙
                        ↪  new MetricPoint(simulationTime, ↙
                        ↪ agentList.size()));
268                 dashBoard.getjTextField6().setText("" + ↙
                        ↪ agentList.size());
```

```
269                 //recorder.addValue(Recorder.BIRTH_RATE,↙
                        ↪  new MetricPoint(simulationTime, ↙
                        ↪ agentList.size()/impulseLenght));
270                 //          dashBoard.getjTextField7().↙
                        ↪ setText(""+ agentList.size()/↙
                        ↪ impulseLenght);
271                 recorder.addValue(Recorder.TOTAL_DEATHS,↙
                        ↪  new MetricPoint(simulationTime, (↙
                        ↪ agentList.size() - totalLiveAgents)↙
                        ↪ ));
272                 //          dashBoard.getjTextField8().↙
                        ↪ setText(""+ (agentList.size()-↙
                        ↪ totalLiveAgents));
273                 //          recorder.addValue(Recorder.↙
                        ↪ DEATH_RATE, new MetricPoint(↙
                        ↪ simulationTime, (agentList.size()-↙
                        ↪ totalLiveAgents)/impulseLenght));
274                 //          daAshBoard.getjTextField8().↙
                        ↪ setText(""+ (agentList.size()-↙
                        ↪ totalLiveAgents)/impulseLenght);
275                 recorder.addValue(Recorder.AGE, new ↙
                        ↪ MetricPoint(simulationTime, ↙
                        ↪ sumOfAges/totalLiveAgents));
276             }
277         }
278
279     }
280
281     private synchronized void processMarketStatistics() ↙
            ↪ {
282         synchronized(Lock.Recorderlock)
283         {
284
285
286             //recorder.addValue(Recorder.POOL_SIZE, new ↙
                    ↪ MetricPoint(simulationTime, pool.↙
                    ↪ getSize()));
287             //dashBoard.getjTextField10().setText(""+ ↙
                    ↪ pool.getSize());
288             //recorder.addValue(Recorder.↙
                    ↪ TRANSACTION_RATE, new MetricPoint(↙
```

```
                    ↪ simulationTime, pool.↙
                    ↪ getTotalTransactions()/impulseLenght));
289                 //dashBoard.getjTextField11().setText(""+ ↙
                    ↪ pool.getTotalTransactions()/↙
                    ↪ impulseLenght);
290                 //recorder.addValue(Recorder.↙
                    ↪ TOTAL_TRANSACTIONS, new MetricPoint(↙
                    ↪ simulationTime, pool.↙
                    ↪ getTotalTransactions()));
291                 //dashBoard.getjTextField12().setText(""+ ↙
                    ↪ pool.getTotalTransactions());
292                 //recorder.addValue(Recorder.↙
                    ↪ MEDIAN_WAIT_TIME, new MetricPoint(↙
                    ↪ simulationTime, market.↙
                    ↪ getMedianWaitTime()));
293                 //dashBoard.getjTextField13().setText(""+ ↙
                    ↪ market.getMedianWaitTime());
294             }
295         }
296
297         private void processGovernmentStatistics() {
298
299
300         }
301
302         private synchronized void processSystemStatistics() ↙
                ↪ {
303 //              recorder.addValue(Recorder.TOTAL_THREADS, new ↙
        ↪ MetricPoint(simulationTime, Thread.activeCount()));
304 //              dashBoard.getjTextField19().setText(""+ Thread↙
        ↪ .activeCount());
305 //              recorder.addValue(Recorder.FREE_MEMORY, new ↙
        ↪ MetricPoint(simulationTime, Runtime.getRuntime().↙
        ↪ freeMemory()));
306 //              dashBoard.getjTextField20().setText(""+ ↙
        ↪ Runtime.getRuntime().freeMemory());
307 //              recorder.addValue(Recorder.REAL_TIME_OVERRUNS,↙
        ↪  new MetricPoint(simulationTime, RealTimeOverruns))↙
        ↪ ;
308 //              dashBoard.getjTextField21().setText(""+ ↙
        ↪ RealTimeOverruns);
```

```
309     }
310     public void incrementSemaphore(){
311         synchronized(Lock.Semaphorelock)
312         {
313             semaphore++;
314         }
315 
316     }
317     public void decrementSemaphore(){
318         synchronized(Lock.Semaphorelock)
319         {
320             semaphore--;
321         }
322 
323     }
324 
325     public static Recorder getRecorder() {
326         return recorder;
327     }
328 
329     public static ArrayList<Agent> getAgentList() {
330         return agentList;
331     }
332     public static RelativeTime getRelativeTime()
333     {
334         return simulationClockTimer;
335     }
336     public static RelativeTime getTimePassed()
337     {
338         return TimePassed;
339     }
340 }
```

Government.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import java.util.logging.Level;
9  import java.util.logging.Logger;
10 import javax.realtime.PriorityParameters;
11 import javax.realtime.PriorityScheduler;
12 import javax.realtime.RealtimeThread;
13 import javax.realtime.RelativeTime;
14
15 /**
16  *
17  * @author Lus F. Reis Pereira
18  */
19 public class Government extends RealtimeThread {
20
21     private static volatile boolean SIM_STOP = false;
22     private static double StoredWealth = 0.0;
23     private static double StotedProduct = 0.0;
24     private static int GOVERNMENT_BUY = 2;
25     private static int GOVERNMENT_WAIT = 1;
26     private Log log;
27     private God god;
28     private Market market;
29     private DashBoard dashboard;
30     private Thread itSelf;
31     private Pool pool;
32     private RelativeTime lastTime  = new RelativeTime↙
           ↪ (0,0);
33     private Boolean END_SIM = false;
34     private lauchSim LauchSim;
35
36     public Government(Log log, God god, Market market, ↙
           ↪ DashBoard dashboard, Pool pool, lauchSim ↙
           ↪ LauchSim) {
37         this.log=log;
```

```
38          this.god =god;
39          this.market = market;
40          this.dashboard = dashboard;
41          this.pool = pool;
42          this.setSchedulingParameters( new ↙
                ↪ PriorityParameters(PriorityScheduler.↙
                ↪ instance().getNormPriority()));
43          this.LauchSim = LauchSim;
44      }
45
46      @Override
47      public void run() {
48
49          itSelf = this;
50
51          double lastTransactionPrice = 0.0;
52          double targetPrice = Double.parseDouble(↙
                ↪ dashboard.getjSpinner11().getValue().↙
                ↪ toString());
53          while(!END_SIM)
54          {
55
56              synchronized(Lock.timelock){
57                          try {
58                              while(lastTime.equals(God.↙
                                    ↪ getSimulationTime()))
59                              {Lock.timelock.wait();}
60                          } catch (InterruptedException ex↙
                                ↪ ) {
61                              Logger.getLogger(Agent.class↙
                                    ↪ .getName()).log(Level.↙
                                    ↪ SEVERE, null, ex);
62                          }
63
64                  }
65              lastTime.set(God.getSimulationTime());
66
67              if(SIM_STOP){
68
69                  synchronized(dashboard.getjTextArea2()){
```

```
70                  dashboard.getjTextArea2().append(this.↙
                        ↪ itSelf.getName() + "****PAUSED****\↙
                        ↪ n");
71                  dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard.↙
                        ↪ getjTextArea2().getDocument().↙
                        ↪ getLength());
72                  }
73                  synchronized(Lock.lock){
74                          try {
75                              while(SIM_STOP){Lock.lock.↙
                                    ↪ wait();}
76                          } catch (InterruptedException ex↙
                            ↪ ) {
77                              Logger.getLogger(Agent.class↙
                                    ↪ .getName()).log(Level.↙
                                    ↪ SEVERE, null, ex);
78                          }
79
80                  }
81                  synchronized(dashboard.getjTextArea2()){
82                  dashboard.getjTextArea2().append(this.↙
                        ↪ itSelf.getName() + "****UNPAUSED↙
                        ↪ ****\n");
83                  dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard.↙
                        ↪ getjTextArea2().getDocument().↙
                        ↪ getLength());
84                  }
85              }
86              int decision = 0;
87              double decisionIntensity = 0.0;
88              //TODO Government Call decider
89              if(market == null)
90              {
91                  market = LauchSim.getMarket();
92              }
93              if(pool == null)
94              {
95                  pool = LauchSim.getPool();
96              }
```

```
97              if (market != null) {
98                  lastTransactionPrice = market.↙
                        ↪ getCurrentPrice();
99                  if (lastTransactionPrice <= targetPrice)↙
                        ↪  {
100                     decision = Government.GOVERNMENT_BUY↙
                            ↪ ;
101                     synchronized (dashboard.↙
                            ↪ getjTextArea4()) {
102                         dashboard.getjTextArea4().append↙
                                ↪ (this.itSelf.getName() + "↙
                                ↪ ****BUY****@" + ↙
                                ↪ lastTransactionPrice + "\n"↙
                                ↪ );
103                         dashboard.getjTextArea4().↙
                                ↪ setCaretPosition(dashboard.↙
                                ↪ getjTextArea4().getDocument↙
                                ↪ ().getLength());
104                     }
105                     Transaction transaction = new ↙
                            ↪ Transaction();
106                     transaction.setAgentIndex(Integer.↙
                            ↪ MAX_VALUE);
107                     transaction.setAmount(1L);
108                     transaction.setPrice(targetPrice);
109                     transaction.setTYPE(Transaction.BUY)↙
                            ↪ ;
110                     synchronized (this) {
111                         transaction.setID("" + Agent.↙
                                ↪ getTransactionID());
112                         Agent.setTransactionID(Agent.↙
                                ↪ getTransactionID());
113                     }
114                     transaction.setCreationTime(God.↙
                            ↪ getSimulationTime());
115                     synchronized(pool){pool.setElement(↙
                            ↪ transaction);}
116                 } else {
117                     decision = Government.↙
                            ↪ GOVERNMENT_WAIT;
```

```
118                         synchronized (dashboard.↙
                                ↪ getjTextArea4()) {
119                             dashboard.getjTextArea4().append↙
                                    ↪ (this.itSelf.getName() + "↙
                                    ↪ ****WAIT****\n");
120                             dashboard.getjTextArea4().↙
                                    ↪ setCaretPosition(dashboard.↙
                                    ↪ getjTextArea4().getDocument↙
                                    ↪ ().getLength());
121                         }
122                     }
123                 }
124             }
125         }
126         public Thread getItSelf() {
127             return itSelf;
128         }
129 
130         public void setEND_SIM(Boolean END_SIM) {
131             this.END_SIM = END_SIM;
132         }
133 
134         public static void setSIM_STOP(boolean SIM_STOP) {
135             Government.SIM_STOP = SIM_STOP;
136         }
137         public double getStoredWealth()
138         {
139             synchronized(Lock.transactionlock)
140             {
141                 return StoredWealth;
142             }
143         }
144         public double getStoredProduct()
145         {
146             synchronized(Lock.transactionlock)
147             {
148                 return StotedProduct;
149             }
150         }
151         public void setStoredProduct(double product)
152         {
```

```
153             synchronized(Lock.transactionlock)
154             {
155                 StotedProduct = product;
156             }
157         }
158         public void setStoredWealth(double wealth)
159         {
160             synchronized(Lock.transactionlock)
161             {
162                 StoredWealth = wealth;
163             }
164 
165         }
166         public int getAgentIndex()
167         {
168             return Integer.MAX_VALUE;
169         }
170 
171 }
```

lauchSim.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import java.util.ArrayList;
9  import javax.realtime.PriorityParameters;
10 import javax.realtime.PriorityScheduler;
11 import javax.realtime.RealtimeThread;
12
13 /**
14  *
15  * @author Luis Filipe dos Reis Pereira
16  * @email LuisFRPereira@gmail.com
17  *
18  */
19 public class lauchSim {
20
21     God god;
22     DashBoard dashBoard;
23     Log log;
24     Government government;
25     Pool pool;
26     Market market;
27     ArrayList<Agent> agentList;
28     Recorder recorder;
29
30      public static void main(String args[]) {
31         lauchSim lauchSim = new lauchSim();
32
33
34     }
35     public lauchSim()
36     {
37
38         agentList = new ArrayList<Agent>();
39         god = new God();
40         recorder = new Recorder();
```

```
41          dashBoard = new DashBoard(god, this);
42          log = new Log();
43          government = new Government(log, god, market, ↙
                ↪ dashBoard, pool, this);
44          pool = new Pool(dashBoard, government);
45          market = new Market(government, agentList, pool,↙
                ↪  log, dashBoard, god);
46          pool.setMarket(market);
47          god.setParameters(dashBoard, agentList, market, ↙
                ↪ pool, log, recorder, government);
48
49          lauchThreads();
50          // Create Dashboard
51            java.awt.EventQueue.invokeLater(new Runnable()↙
                  ↪  {
52              public void run() {
53                  dashBoard.setVisible(true);
54              }
55          });
56          ThreadGroup tg = Thread.currentThread().↙
                ↪ getThreadGroup();
57          tg.list();
58
59      }
60      private void lauchThreads()
61      {
62          //God is started manualy
63
64          RealtimeThread  governmentThread = new ↙
                ↪ Government(log, god, market, dashBoard, ↙
                ↪ pool, this );
65          governmentThread.setName("Government␣Thread");
66          RealtimeThread  marketThread = new Market(↙
                ↪ government, agentList, pool, log, dashBoard↙
                ↪ , god);
67          marketThread.setName("Market␣Thread");
68
69      }
70
71      public God getGod() {
72          return god;
```

```
73      }
74
75      public ArrayList<Agent> getAgentList() {
76          return agentList;
77      }
78
79      public Log getLog() {
80          return log;
81      }
82
83      public Market getMarket() {
84          return market;
85      }
86
87      public Pool getPool() {
88          return pool;
89      }
90      public lauchSim getlauchSim(){
91          return this;
92      }
93
94      public Recorder getRecorder() {
95          return recorder;
96      }
97
98      public Government getGovernment() {
99          return government;
100     }
101
102 }
```

Lock.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  /**
9   *
10  * @author Luis Filipe dos Reis Pereira
11  * @email LuisFRPereira@gmail.com
12  *
13  */
14
15 public class Lock {
16
17     public static final Object lock = new Object();
18     public static final Object timelock = new Object();
19     public static final Object transactionlock = new ↙
           ↪ Object();
20     public static final Object AgentListlock = new ↙
           ↪ Object();
21     public static final Object AgentIDlock = new Object↙
           ↪ ();
22     public static final Object Semaphorelock = new ↙
           ↪ Object();
23     public static final Object Recorderlock = new Object↙
           ↪ ();
24
25 }
```

Log.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import java.io.BufferedReader;
9  import java.io.File;
10 import java.io.FileNotFoundException;
11 import java.io.FileReader;
12 import java.io.IOException;
13 import java.sql.Connection;
14 import java.sql.DriverManager;
15 import java.sql.SQLException;
16 import java.util.Properties;
17 import java.util.logging.Level;
18 import java.util.logging.Logger;
19 import javax.realtime.PriorityParameters;
20 import javax.realtime.PriorityScheduler;
21 import javax.realtime.RealtimeThread;
22 import javax.realtime.SchedulingParameters;
23 /**
24  *
25  * @author Luis F. Reis Pereira
26  */
27 public class Log extends RealtimeThread {
28     private String databaseURL = "jdbc:derby:memory:↙
        ↪ marketSim";
29     private Properties databaseProperties;
30     static private volatile Connection connection;
31
32     public Log() {
33
34         setName("Log RTT");
35
36         PriorityParameters GodPriorityParameters =
37                 new PriorityParameters(
38                 PriorityScheduler.instance().↙
                    ↪ getNormPriority());
```

```
39
40
41
42
43          this.setSchedulingParameters(↙
                ↪ GodPriorityParameters);
44      }
45
46
47
48      @Override
49      public void run ()
50      {
51
52          databaseProperties = new Properties();
53          databaseProperties.setProperty("createFrom", "/↙
                ↪ home/lfp/BackDerbyEmpty/marketSim");
54          databaseProperties.setProperty("user", "a");
55          databaseProperties.setProperty("password", "a");
56
57          try {
58              Class.forName("org.apache.derby.jdbc.↙
                    ↪ EmbeddedDriver");
59          } catch (ClassNotFoundException ex) {
60              Logger.getLogger(God.class.getName()).log(↙
                    ↪ Level.SEVERE, null, ex);
61          }
62          try {
63              connection = DriverManager.getConnection(↙
                    ↪ databaseURL, databaseProperties);
64
65          } catch (SQLException ex) {
66              Logger.getLogger(Log.class.getName()).log(↙
                    ↪ Level.SEVERE, null, ex);
67          }
68
69      }
70
71      public Connection getConnection() {
72          return connection;
73      }
```

```
74
75
76
77  }
```

Market.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import java.util.ArrayList;
9  import java.util.NoSuchElementException;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12 import javax.realtime.PriorityParameters;
13 import javax.realtime.PriorityScheduler;
14 import javax.realtime.RealtimeThread;
15 import javax.realtime.RelativeTime;
16 import org.apache.commons.math.stat.descriptive.↙
       ↪ DescriptiveStatistics;
17 import java.util.Iterator;
18 import javax.realtime.Clock;
19
20
21
22 /**
23  *
24  * @author Lus F. Reis Pereira
25  */
26 public class Market extends RealtimeThread{
27     private double productPrice = 0;
28     public static final int GOVERNMENT_ACCEPTED = 0;
29     public static final int GOVERNEMNT_DENIED = 1;
30     public static final int TRANSCATION_FULL_SUCCESS = ↙
           ↪ 2;
31     public static final int TRANSACTION_PARTIAL_SUCCESS ↙
           ↪ = 3;
32     public static final int TRANSACTION_FAILD = 4;
33     private ArrayList<Agent> agentList;
34     private Pool pool;
35     private Log log;
36     private boolean paused;
37     private String Buy_ID;
```

```
38      private String Sell_ID;
39      private double currentPrice;
40      private double currentAmount;
41      private DescriptiveStatistics timeTracker;
42      private God god;
43      private RelativeTime lastTime  = new RelativeTime(0,0);
44      private Boolean END_SIM = false;
45      private static Boolean newElement = false;
46      private Iterator<Transaction> iPool = null;
47      private double lastTransactionPrice;
48
49      public Market( Government government,
50              ArrayList<Agent> agentList, Pool pool,Log log, DashBoard dashboard, God god)
51      {
52          this.agentList=agentList;
53          this.pool=pool;
54          this.log=log;
55          this.currentPrice=Double.parseDouble(dashboard.getjSpinner11().getValue().toString());
56          timeTracker = new DescriptiveStatistics();
57          this.god = god;
58          this.setSchedulingParameters(new PriorityParameters(PriorityScheduler.instance().getNormPriority()));
59      }
60      synchronized int doTransaction(Transaction T1, Transaction T2)
61      {
62          if(T1.getCreationTime().compareTo(T1.getCreationTime()) > 0)
63              timeTracker.addValue(T1.getCreationTime().subtract(T2.getCreationTime()).getMilliseconds());
64          else
65              timeTracker.addValue(T2.getCreationTime().subtract(T1.getCreationTime()).getMilliseconds());
66
67          return 0;
```

```
68      }
69      double getMedianWaitTime()
70      {
71          return timeTracker.getPercentile(0.5);
72      }
73
74      @Override
75      public void run() {
76
77          while(!END_SIM)
78          {
79
80              synchronized(Lock.timelock){
81                          try {
82                              while(lastTime.equals(God.↙
                                    ↪ getSimulationTime()))
83                              {Lock.timelock.wait();}
84                          } catch (InterruptedException ex↙
                                ↪ ) {
85                              Logger.getLogger(Agent.class↙
                                    ↪ .getName()).log(Level.↙
                                    ↪ SEVERE, null, ex);
86                          }
87
88                      }
89                  lastTime.set(God.getSimulationTime());
90
91
92
93  //                  logDecision(Buy_ID, Sell_ID, currentPrice,↙
        ↪ currentAmount);
94  //                  updateBuyerValues();
95  //                  updateSellerValues();
96                  // TODO log transaction
97
98
99          }
100     }
101     public boolean isPaused() {
102         return paused;
103     }
```

```
104     public static void fireNewElement()
105     {
106         newElement = true;
107     }
108     public void setPaused(boolean paused) {
109         this.paused = paused;
110     }
111
112     public double getCurrentPrice() {
113         return currentPrice;
114     }
115
116     public void setEND_SIM(Boolean END_SIM) {
117         this.END_SIM = END_SIM;
118     }
119
120     public synchronized void setCurrentPrice(double ↙
            ↪ currentPrice) {
121
122         this.lastTransactionPrice = this.currentPrice;
123         this.currentPrice = currentPrice;
124     }
125
126     public double getLastTransactionPrice() {
127         return lastTransactionPrice;
128     }
129 }
```

MetricPoint.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import javax.realtime.RelativeTime;
9
10 /**
11  *
12  * @author Luis Filipe dos Reis Pereira
13  * @email LuisFRPereira@gmail.com
14  *
15  */
16 public class MetricPoint {
17     private RelativeTime time;
18     private double metric;
19
20     public MetricPoint(RelativeTime time, double metric)↙
       ↪  {
21         this.time = time;
22         this.metric = metric;
23     }
24
25
26     public double getMetric() {
27         return metric;
28     }
29
30     public void setMetric(long metric) {
31         this.metric = metric;
32     }
33
34     public RelativeTime getTime() {
35         return time;
36     }
37
38     public void setTime(RelativeTime time) {
39         this.time = time;
```

```
40          }
41
42  }
```

Pool.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import java.util.HashSet;
9  import java.util.Iterator;
10
11 /**
12  *
13  * @author Luis Filipe dos Reis Pereira
14  * @email LuisFRPereira@gmail.com
15  *
16  */
17 public class Pool {
18
19     private HashSet<Transaction> pool_Buy;
20     private HashSet<Transaction> pool_Sell;
21     private long totalTransactions=0;
22     private Boolean newElement = false;
23     private Transaction elementToTransact = null;
24     private DashBoard dashboard;
25     private volatile Double price =0.0;
26     private boolean doTransaction;
27     private volatile Transaction oldElement;
28     private HashSet<Transaction> removePool;
29     private Government government ;
30     private Market market;
31
32
33
34     public Pool (DashBoard dashboard, Government ↙
           ↪ government)
35     {
36         pool_Buy = new HashSet<Transaction>();
37         pool_Sell = new HashSet<Transaction>();
38         this.dashboard = dashboard;
39         this. government = government;
```

```
40      }
41
42       public synchronized Transaction next (Iterator<↙
            ↪ Transaction> iterator)
43       {
44           return iterator.next();
45       }
46       public synchronized boolean hasNext(Iterator<↙
            ↪ Transaction> iterator)
47       {
48           return iterator.hasNext();
49       }
50       void setElement( Transaction newElement)
51       {
52
53
54           if(newElement.getTYPE() == Transaction.BUY)
55           {
56               synchronized(pool_Sell)
57               {
58                   if(pool_Buy.isEmpty())
59                   {
60                       synchronized(pool_Buy){pool_Buy.add↙
                            ↪ (newElement);}
61                   }
62                   for( Transaction T : pool_Sell)
63                   {
64                       if (Math.abs(T.price) < Math.abs(↙
                            ↪ newElement.price))
65                       {
66                           doTransaction = true;
67                           oldElement = T;
68                           removePool = pool_Sell;
69                           break;
70                       }
71
72                   }
73                   if(!doTransaction)
74                   {
75                       synchronized(pool_Buy){pool_Buy.add(↙
                            ↪ newElement);}
```

```
76                      Market.fireNewElement();
77                  }
78              }
79
80          }
81          if(doTransaction)
82          {
83              transactionDone(oldElement, newElement, removePool);
84              doTransaction =false;
85          }
86          if(newElement.getTYPE() == Transaction.SELL)
87          {
88              synchronized(pool_Buy)
89              {
90                  if(pool_Sell.isEmpty())
91                  {
92                      synchronized(pool_Sell){pool_Sell.add(newElement);}
93                  }
94                  for( Transaction T : pool_Buy)
95                  {
96                      if (Math.abs(T.price) >= Math.abs(newElement.price))
97                      {
98                          doTransaction = true;
99                          oldElement = T;
100                         removePool = pool_Buy;
101                         break;
102                     }
103
104                 }
105                 if(!doTransaction)
106                 {
107                     synchronized(pool_Sell){pool_Sell.add(newElement);}
108                     Market.fireNewElement();
109                 }
110             }
111             if(doTransaction)
112             {
```

```
113                 if(oldElement != null)
114                 {
115                     transactionDone(oldElement, ↙
                           ↪ newElement, removePool);
116                 }
117                 else
118                 {
119                     synchronized(this){removePool.remove↙
                           ↪ (oldElement);}
120                 }
121 
122                 doTransaction =false;
123             }
124         }
125     }
126     int getSize()
127     {
128         return pool_Buy.size()+pool_Sell.size();
129     }
130     boolean transactionDone(Transaction T1, Transaction ↙
          ↪ T2,  HashSet<Transaction> pool)
131     {
132 
133         try {
134             synchronized (Lock.AgentListlock) {
135                 if ((Math.abs(T1.price) < Math.abs(T2.↙
                       ↪ price))) {
136                     price = Math.abs(T1.price);
137                     if(God.getRelativeTime() == null)
138                     {
139                         //EMPTY
140                     }
141                     else
142                     {     synchronized(Lock.↙
                             ↪ transactionlock){
143                             synchronized(Lock.↙
                                 ↪ Recorderlock)
144                             {
145                                 God.getRecorder().↙
                                     ↪ addValue(Recorder.↙
                                     ↪ PRICE, new ↙
```

```
                                    ↪ MetricPoint(
146                                  God.getTimePassed(),↙
                                        ↪  price));
147                             }
148
149                         }
150
151                     }
152                     synchronized (Lock.transactionlock) {
153                         if (T1.getAgentIndex() == ↙
                            ↪ 2147483647) {
154                             government.setStoredWealth(↙
                                ↪ government.↙
                                ↪ getStoredWealth() + ↙
                                ↪ price);
155                             God.getAgentList().get((int) ↙
                                ↪ T2.getAgentIndex()).↙
                                ↪ setStoredWealth(
156                                     God.getAgentList().↙
                                        ↪ get((int) T2.↙
                                        ↪ getAgentIndex())↙
                                        ↪ .getStoredWealth↙
                                        ↪ () - price);
157
158                             government.setStoredProduct(↙
                                ↪ government.↙
                                ↪ getStoredProduct() - 1);
159                             God.getAgentList().get((int) ↙
                                ↪ T2.getAgentIndex()).↙
                                ↪ setStoredProduct(
160                                     God.getAgentList().↙
                                        ↪ get((int) T2.↙
                                        ↪ getAgentIndex())↙
                                        ↪ .↙
                                        ↪ getStoredProduct↙
                                        ↪ () + 1);
161                         } else if (T2.getAgentIndex() == ↙
                            ↪ 2147483647) {
162                             God.getAgentList().get((int) ↙
                                ↪ T1.getAgentIndex()).↙
                                ↪ setStoredWealth(
```

```
163                                 God.getAgentList().↙
                                        ↪ get((int) T1.↙
                                        ↪ getAgentIndex())↙
                                        ↪ .getStoredWealth↙
                                        ↪ () + price);
164                             government.setStoredWealth(↙
                                    ↪ government.↙
                                    ↪ getStoredWealth() - ↙
                                    ↪ price);
165
166                             God.getAgentList().get((int) ↙
                                    ↪ T1.getAgentIndex()).↙
                                    ↪ setStoredProduct(
167                                 God.getAgentList().↙
                                        ↪ get((int) T1.↙
                                        ↪ getAgentIndex())↙
                                        ↪ .↙
                                        ↪ getStoredProduct↙
                                        ↪ () - 1);
168                             government.setStoredProduct(↙
                                    ↪ government.↙
                                    ↪ getStoredProduct() + 1);
169                         } else {
170                             God.getAgentList().get((int) ↙
                                    ↪ T1.getAgentIndex()).↙
                                    ↪ setStoredWealth(
171                                 God.getAgentList().↙
                                        ↪ get((int) T1.↙
                                        ↪ getAgentIndex())↙
                                        ↪ .getStoredWealth↙
                                        ↪ () + price);
172                             God.getAgentList().get((int) ↙
                                    ↪ T2.getAgentIndex()).↙
                                    ↪ setStoredWealth(
173                                 God.getAgentList().↙
                                        ↪ get((int) T2.↙
                                        ↪ getAgentIndex())↙
                                        ↪ .getStoredWealth↙
                                        ↪ () - price);
174
```

```
175                 God.getAgentList().get((int) ↙
                        ↪ T1.getAgentIndex()).↙
                        ↪ setStoredProduct(
176                         God.getAgentList().↙
                                ↪ get((int) T1.↙
                                ↪ getAgentIndex())↙
                                ↪ .↙
                                ↪ getStoredProduct↙
                                ↪ () - 1);
177                 God.getAgentList().get((int) ↙
                        ↪ T2.getAgentIndex()).↙
                        ↪ setStoredProduct(
178                         God.getAgentList().↙
                                ↪ get((int) T2.↙
                                ↪ getAgentIndex())↙
                                ↪ .↙
                                ↪ getStoredProduct↙
                                ↪ () + 1);
179             }
180             synchronized (dashboard.↙
                    ↪ getjTextArea3()) {
181                 if (T1.getAgentIndex() == ↙
                        ↪ 2147483647) {
182                     dashboard.getjTextArea3()↙
                            ↪ .append("Transaction↙
                            ↪ ␣between␣" + T1.↙
                            ↪ getAgentIndex() + "@↙
                            ↪ " + Math.abs(T1.↙
                            ↪ price)
183                                 + "␣and␣" + T2.↙
                                        ↪ getAgentIndex↙
                                        ↪ () + "@" + ↙
                                        ↪ Math.abs(T2.↙
                                        ↪ price) + "␣↙
                                        ↪ at␣:" + ↙
                                        ↪ price
184                                 + "␣Aa:␣" + ↙
                                        ↪ government.↙
                                        ↪ getStoredWealth↙
                                        ↪ ()
```

```
185                                 + "␣Ab:␣" + God.↙
                                        ↪ getAgentList↙
                                        ↪ ().get((int)↙
                                        ↪  T2.↙
                                        ↪ getAgentIndex↙
                                        ↪ ()).↙
                                        ↪ getStoredWealth↙
                                        ↪ () + "\n");
186                             dashboard.getjTextArea3()↙
                                    ↪ .setCaretPosition(↙
                                    ↪ dashboard.↙
                                    ↪ getjTextArea3().↙
                                    ↪ getDocument().↙
                                    ↪ getLength());
187                         } else if (T2.getAgentIndex()↙
                                ↪  == 2147483647) {
188                             dashboard.getjTextArea3()↙
                                    ↪ .append("Transaction↙
                                    ↪ ␣between␣" + T1.↙
                                    ↪ getAgentIndex() + "@↙
                                    ↪ " + Math.abs(T1.↙
                                    ↪ price)
189                                         + "␣and␣" + T2.↙
                                            ↪ getAgentIndex↙
                                            ↪ () + "@" + ↙
                                            ↪ Math.abs(T2.↙
                                            ↪ price) + "␣↙
                                            ↪ at␣:" + ↙
                                            ↪ price
190                                         + "␣Aa:␣" + God.↙
                                            ↪ getAgentList↙
                                            ↪ ().get((int)↙
                                            ↪  T1.↙
                                            ↪ getAgentIndex↙
                                            ↪ ()).↙
                                            ↪ getStoredWealth↙
                                            ↪ ()
191                                         + "␣Ab:␣" + ↙
                                            ↪ government.↙
                                            ↪ getStoredWealth↙
                                            ↪ () + "\n");
```

```
192                 dashboard.getjTextArea3()↙
                        ↪ .setCaretPosition(↙
                        ↪ dashboard.↙
                        ↪ getjTextArea3().↙
                        ↪ getDocument().↙
                        ↪ getLength());
193                 } else {
194                     dashboard.getjTextArea3()↙
                            ↪ .append("Transaction↙
                            ↪ ␣between␣" + T1.↙
                            ↪ getAgentIndex() + "@↙
                            ↪ " + Math.abs(T1.↙
                            ↪ price)
195                                 + "␣and␣" + T2.↙
                                    ↪ getAgentIndex↙
                                    ↪ () + "@" + ↙
                                    ↪ Math.abs(T2.↙
                                    ↪ price) + "␣↙
                                    ↪ at␣:" + ↙
                                    ↪ price
196                                 + "␣Aa:␣" + God.↙
                                    ↪ getAgentList↙
                                    ↪ ().get((int)↙
                                    ↪  T1.↙
                                    ↪ getAgentIndex↙
                                    ↪ ()).↙
                                    ↪ getStoredWealth↙
                                    ↪ ()
197                                 + "␣Ab:␣" + God.↙
                                    ↪ getAgentList↙
                                    ↪ ().get((int)↙
                                    ↪  T2.↙
                                    ↪ getAgentIndex↙
                                    ↪ ()).↙
                                    ↪ getStoredWealth↙
                                    ↪ () + "\n");
198                     dashboard.getjTextArea3()↙
                            ↪ .setCaretPosition(↙
                            ↪ dashboard.↙
                            ↪ getjTextArea3().↙
                            ↪ getDocument().↙
```

```
                                    ↪ getLength());
199                             }
200                         }
201                     }
202
203
204                 } else {
205                     price = Math.abs(T2.price);
206                     if(God.getRelativeTime() == null)
207                      {
208                          //EMPTY
209                      }
210                      else
211                     {
212                         synchronized(Lock.transactionlock↙
                            ↪ ){
213                          synchronized(Lock.↙
                                ↪ Recorderlock)
214                          {
215                              God.getRecorder().↙
                                    ↪ addValue(Recorder.↙
                                    ↪ PRICE, new ↙
                                    ↪ MetricPoint(
216                                 God.getTimePassed(), ↙
                                        ↪ price));
217                          }
218
219                         }
220                     }
221                     synchronized (Lock.transactionlock) {
222                         if (T1.getAgentIndex() == ↙
                            ↪ 2147483647) {
223                          God.getAgentList().get((int) ↙
                                ↪ T2.getAgentIndex()).↙
                                ↪ setStoredWealth(
224                                     God.getAgentList().↙
                                        ↪ get((int) T2.↙
                                        ↪ getAgentIndex())↙
                                        ↪ .getStoredWealth↙
                                        ↪ () + price);
```

```
225                 government.setStoredWealth(↙
                        ↪ government.↙
                        ↪ getStoredWealth() - ↙
                        ↪ price);
226
227                 government.setStoredProduct(↙
                        ↪ government.↙
                        ↪ getStoredProduct() + 1);
228                 God.getAgentList().get((int) ↙
                        ↪ T2.getAgentIndex()).↙
                        ↪ setStoredProduct(
229                         God.getAgentList().↙
                                ↪ get((int) T2.↙
                                ↪ getAgentIndex())↙
                                ↪ .↙
                                ↪ getStoredProduct↙
                                ↪ () - 1);
230             } else if (T2.getAgentIndex() == ↙
                    ↪ 2147483647) {
231                 government.setStoredWealth(↙
                        ↪ government.↙
                        ↪ getStoredWealth() + ↙
                        ↪ price);
232                 God.getAgentList().get((int) ↙
                        ↪ T1.getAgentIndex()).↙
                        ↪ setStoredWealth(
233                         God.getAgentList().↙
                                ↪ get((int) T1.↙
                                ↪ getAgentIndex())↙
                                ↪ .getStoredWealth↙
                                ↪ () - price);
234
235                 God.getAgentList().get((int) ↙
                        ↪ T1.getAgentIndex()).↙
                        ↪ setStoredProduct(
236                         God.getAgentList().↙
                                ↪ get((int) T1.↙
                                ↪ getAgentIndex())↙
                                ↪ .↙
                                ↪ getStoredProduct↙
                                ↪ () + 1);
```

```
237                 government.setStoredProduct(↙
                        ↪ government.↙
                        ↪ getStoredProduct() - 1);
238             } else {
239                 God.getAgentList().get((int) ↙
                        ↪ T2.getAgentIndex()).↙
                        ↪ setStoredWealth(
240                             God.getAgentList().↙
                                ↪ get((int) T2.↙
                                ↪ getAgentIndex())↙
                                ↪ .getStoredWealth↙
                                ↪ () + price);
241                 God.getAgentList().get((int) ↙
                        ↪ T1.getAgentIndex()).↙
                        ↪ setStoredWealth(
242                             God.getAgentList().↙
                                ↪ get((int) T1.↙
                                ↪ getAgentIndex())↙
                                ↪ .getStoredWealth↙
                                ↪ () - price);
243
244                 God.getAgentList().get((int) ↙
                        ↪ T1.getAgentIndex()).↙
                        ↪ setStoredProduct(
245                             God.getAgentList().↙
                                ↪ get((int) T1.↙
                                ↪ getAgentIndex())↙
                                ↪ .↙
                                ↪ getStoredProduct↙
                                ↪ () + 1);
246                 God.getAgentList().get((int) ↙
                        ↪ T2.getAgentIndex()).↙
                        ↪ setStoredProduct(
247                             God.getAgentList().↙
                                ↪ get((int) T2.↙
                                ↪ getAgentIndex())↙
                                ↪ .↙
                                ↪ getStoredProduct↙
                                ↪ () - 1);
248             }
```

```
249                     synchronized (dashboard.↙
                            ↪ getjTextArea3()) {
250                         if (T1.getAgentIndex() == ↙
                                ↪ 2147483647) {
251                             dashboard.getjTextArea3()↙
                                    ↪ .append("Transaction↙
                                    ↪ ␣between␣" + T1.↙
                                    ↪ getAgentIndex() + "@↙
                                    ↪ " + Math.abs(T1.↙
                                    ↪ price)
252                                         + "␣and␣" + T2.↙
                                            ↪ getAgentIndex↙
                                            ↪ () + "@" + ↙
                                            ↪ Math.abs(T2.↙
                                            ↪ price) + "␣↙
                                            ↪ at␣:" + ↙
                                            ↪ price
253                                         + "␣Aa:␣" + ↙
                                            ↪ government.↙
                                            ↪ getStoredWealth↙
                                            ↪ ()
254                                         + "␣Ab:␣" + God.↙
                                            ↪ getAgentList↙
                                            ↪ ().get((int)↙
                                            ↪  T2.↙
                                            ↪ getAgentIndex↙
                                            ↪ ()).↙
                                            ↪ getStoredWealth↙
                                            ↪ () + "\n");
255                             dashboard.getjTextArea3()↙
                                    ↪ .setCaretPosition(↙
                                    ↪ dashboard.↙
                                    ↪ getjTextArea3().↙
                                    ↪ getDocument().↙
                                    ↪ getLength());
256                         } else if (T2.getAgentIndex()↙
                                ↪  == 2147483647) {
257                             dashboard.getjTextArea3()↙
                                    ↪ .append("Transaction↙
                                    ↪ ␣between␣" + T1.↙
                                    ↪ getAgentIndex() + "@↙
```

```
                    ↪ " + Math.abs(T1.↙
                    ↪ price)
258                     + "␣and␣" + T2.↙
                            ↪ getAgentIndex↙
                            ↪ () + "@" + ↙
                            ↪ Math.abs(T2.↙
                            ↪ price) + "␣↙
                            ↪ at␣:" + ↙
                            ↪ price
259                     + "␣Aa:␣" + God.↙
                            ↪ getAgentList↙
                            ↪ ().get((int)↙
                            ↪  T1.↙
                            ↪ getAgentIndex↙
                            ↪ ()).↙
                            ↪ getStoredWealth↙
                            ↪ ()
260                     + "␣Ab:␣" + ↙
                            ↪ government.↙
                            ↪ getStoredWealth↙
                            ↪ () + "\n");
261                 dashboard.getjTextArea3()↙
                        ↪ .setCaretPosition(↙
                        ↪ dashboard.↙
                        ↪ getjTextArea3().↙
                        ↪ getDocument().↙
                        ↪ getLength());
262             } else {
263                 dashboard.getjTextArea3()↙
                        ↪ .append("Transaction↙
                        ↪ ␣between␣" + T1.↙
                        ↪ getAgentIndex() + "@↙
                        ↪ " + Math.abs(T1.↙
                        ↪ price)
264                     + "␣and␣" + T2.↙
                            ↪ getAgentIndex↙
                            ↪ () + "@" + ↙
                            ↪ Math.abs(T2.↙
                            ↪ price) + "␣↙
                            ↪ at␣:" + ↙
                            ↪ price
```

```
265                                         + "␣Aa:␣" + God.↙
                                                ↪ getAgentList↙
                                                ↪ ().get((int)↙
                                                ↪  T1.↙
                                                ↪ getAgentIndex↙
                                                ↪ ()).↙
                                                ↪ getStoredWealth↙
                                                ↪ ()
266                                         + "␣Ab:␣" + God.↙
                                                ↪ getAgentList↙
                                                ↪ ().get((int)↙
                                                ↪  T2.↙
                                                ↪ getAgentIndex↙
                                                ↪ ()).↙
                                                ↪ getStoredWealth↙
                                                ↪ () + "\n");
267                                     dashboard.getjTextArea3()↙
                                            ↪ .setCaretPosition(↙
                                            ↪ dashboard.↙
                                            ↪ getjTextArea3().↙
                                            ↪ getDocument().↙
                                            ↪ getLength());
268                                 }
269                             }
270
271                         }
272                     }
273
274             }
275         } catch (NullPointerException e) {
276             synchronized (dashboard.getjTextArea3()) {
277
278                                     dashboard.getjTextArea3()↙
                                            ↪ .append("Tried␣↙
                                            ↪ transaction␣with␣↙
                                            ↪ dead␣Agent" + T1.↙
                                            ↪ getAgentIndex()  + "↙
                                            ↪ \n");
279                                     dashboard.getjTextArea3()↙
                                            ↪ .setCaretPosition(↙
                                            ↪ dashboard.↙
```

```
                                        ↪ getjTextArea3().↙
                                        ↪ getDocument().↙
                                        ↪ getLength());
280                                   totalTransactions--;
281             }
282         }
283         synchronized(pool){pool.remove(T1);}
284         totalTransactions++;
285
286
287
288         return true;
289     }
290     long getTotalTransactions()
291      {
292         return totalTransactions;
293     }
294
295      public void setMarket(Market market) {
296          this.market = market;
297          market.setCurrentPrice(price);
298      }
299      public void ResetPool(){
300         this.pool_Buy = new HashSet<Transaction>();
301         this.pool_Sell = new HashSet<Transaction>();
302      }
303
304 }
```

Recorder.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8
9  import java.util.ArrayList;
10 import java.util.HashMap;
11
12 /**
13  *
14  * @author Luis Filipe dos Reis Pereira
15  * @email LuisFRPereira@gmail.com
16  *
17  */
18 public class Recorder {
19     public static final Integer NUMBER_OF_AGENTS = 0;
20     public static final Integer PRODUCT_STORED=1;
21     public static final Integer PRODUCTION_RATE=2;
22     public static final Integer WEALTH = 3;
23     public static final Integer WEALTH_GROWTH_RATE = 4;
24     public static final Integer TOTAL_BIRTHS = 5;
25     public static final Integer BIRTH_RATE = 6;
26     public static final Integer TOTAL_DEATHS = 7;
27     public static final Integer DEATH_RATE = 8;
28     public static final Integer POOL_SIZE = 9;
29     public static final Integer TRANSACTION_RATE = 10;
30     public static final Integer TOTAL_TRANSACTIONS = 11;
31     public static final Integer MEDIAN_WAIT_TIME = 12;
32     public static final Integer ACTIVE_GOVERNMENT = 13;
33     public static final Integer DENYED_TRANSACTIONS = 14;
34     public static final Integer SAUDADE = 15;
35     public static final Integer GENERATION = 16;
36     public static final Integer ACTIVE_TIME = 17;
37     public static final Integer TOTAL_THREADS = 18;
38     public static final Integer FREE_MEMORY = 19;
39     public static final Integer REAL_TIME_OVERRUNS = 20;
40     public static final Integer PRICE = 21;
```

```
41      public static final Integer AGE = 22;
42
43      private HashMap<Integer, ArrayList<MetricPoint>> ↙
           ↪ allMetrics;
44      private ArrayList<MetricPoint> series;
45
46      public Recorder(){
47          allMetrics = new HashMap<Integer, ArrayList<↙
               ↪ MetricPoint> >();
48          series = new ArrayList<MetricPoint>();
49  }
50      public void addValue(Integer type, MetricPoint point)
51      {
52          series = allMetrics.get(type);
53          if (series!=null) {
54              series.add(point);
55          }
56          else{
57              allMetrics.put(type, new ArrayList<↙
                   ↪ MetricPoint>() );
58              series = allMetrics.get(type);
59              series.add(point);
60          }
61          allMetrics.put(type, series);
62      }
63      public ArrayList<MetricPoint> getList(Integer type)
64      {
65                return allMetrics.get(type);
66      }
67
68  }
```

Report.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  /*
7   * Report.java
8   *
9   * Created on 20/Mar/2011, 11:31:15
10  */
11
12 package marketsim;
13
14 import java.awt.event.ItemEvent;
15
16 /**
17  *
18  * @author lfp
19  */
20 public class Report extends javax.swing.JFrame {
21     ShowCharts NUMBER_OF_AGENTS;
22     ShowCharts PRODUCT_STORED;
23     ShowCharts PRODUCTION_RATE;
24     ShowCharts WEALTH;
25     ShowCharts WEALTH_GROWTH_RATE;
26     ShowCharts TOTAL_BIRTHS;
27     ShowCharts BIRTH_RATE;
28     ShowCharts TOTAL_DEATHS;
29     ShowCharts DEATH_RATE;
30     ShowCharts POOL_SIZE;
31     ShowCharts TRANSACTION_RATE;
32     ShowCharts TOTAL_TRANSACTIONS;
33     ShowCharts MEDIAN_WAIT_TIME;
34     ShowCharts DENYED_TRANSACTIONS;
35     ShowCharts FREE_MEMORY;
36     ShowCharts REAL_TIME_OVERRUNS;
37     ShowCharts PRICE;
38     ShowCharts AGE;
39
40     /** Creates new form Report */
```

```
41      public Report() {
42          initComponents();
43          pack();
44          setVisible(true);
45      }
46
47      /** This method is called from within the ↙
            ↪ constructor to
48       * initialize the form.
49       * WARNING: Do NOT modify this code. The content of ↙
             ↪ this method is
50       * always regenerated by the Form Editor.
51       */
52      @SuppressWarnings("unchecked")
53      // <editor-fold defaultstate="collapsed" desc="↙
            ↪ Generated Code">//GEN-BEGIN:initComponents
54      private void initComponents() {
55
56          jCheckBox1 = new javax.swing.JCheckBox();
57          jCheckBox2 = new javax.swing.JCheckBox();
58          jCheckBox4 = new javax.swing.JCheckBox();
59          jCheckBox6 = new javax.swing.JCheckBox();
60          jCheckBox8 = new javax.swing.JCheckBox();
61          jCheckBox13 = new javax.swing.JCheckBox();
62          jCheckBox3 = new javax.swing.JCheckBox();
63          jCheckBox5 = new javax.swing.JCheckBox();
64
65          setDefaultCloseOperation(javax.swing.↙
                ↪ WindowConstants.EXIT_ON_CLOSE);
66
67          jCheckBox1.setText("Number␣of␣Agents");
68          jCheckBox1.addItemListener(new java.awt.event.↙
                ↪ ItemListener() {
69              public void itemStateChanged(java.awt.event.↙
                    ↪ ItemEvent evt) {
70                  Number_of_Agents_handler(evt);
71              }
72          });
73
74          jCheckBox2.setText("Product␣stored");
```

```
75          jCheckBox2.addItemListener(new java.awt.event.↙
               ↪ ItemListener() {
76              public void itemStateChanged(java.awt.event.↙
                   ↪ ItemEvent evt) {
77                  Product_stored_handler(evt);
78              }
79          });
80
81          jCheckBox4.setText("Wealth");
82          jCheckBox4.addItemListener(new java.awt.event.↙
               ↪ ItemListener() {
83              public void itemStateChanged(java.awt.event.↙
                   ↪ ItemEvent evt) {
84                  Wealth_handler(evt);
85              }
86          });
87
88          jCheckBox6.setText("Total␣births");
89          jCheckBox6.addItemListener(new java.awt.event.↙
               ↪ ItemListener() {
90              public void itemStateChanged(java.awt.event.↙
                   ↪ ItemEvent evt) {
91                  Total_births_handler(evt);
92              }
93          });
94
95          jCheckBox8.setText("Total␣bankruptcies");
96          jCheckBox8.addItemListener(new java.awt.event.↙
               ↪ ItemListener() {
97              public void itemStateChanged(java.awt.event.↙
                   ↪ ItemEvent evt) {
98                  Total_deaths(evt);
99              }
100         });
101
102         jCheckBox13.setText("Median␣wait␣time");
103         jCheckBox13.addItemListener(new java.awt.event.↙
               ↪ ItemListener() {
104             public void itemStateChanged(java.awt.event.↙
                   ↪ ItemEvent evt) {
105                 Median_wait_time_handler(evt);
```

```
106             }
107         });
108 
109         jCheckBox3.setText("Price");
110         jCheckBox3.addItemListener(new java.awt.event.↙
                ↪ ItemListener() {
111             public void itemStateChanged(java.awt.event.↙
                    ↪ ItemEvent evt) {
112                 jCheckBox3ItemStateChanged(evt);
113             }
114         });
115 
116         jCheckBox5.setText("Age");
117         jCheckBox5.addItemListener(new java.awt.event.↙
                ↪ ItemListener() {
118             public void itemStateChanged(java.awt.event.↙
                    ↪ ItemEvent evt) {
119                 AgeEventHandler(evt);
120             }
121         });
122         jCheckBox5.addActionListener(new java.awt.event.↙
                ↪ ActionListener() {
123             public void actionPerformed(java.awt.event.↙
                    ↪ ActionEvent evt) {
124                 jCheckBox5ActionPerformed(evt);
125             }
126         });
127 
128         org.jdesktop.layout.GroupLayout layout = new org↙
                ↪ .jdesktop.layout.GroupLayout(getContentPane↙
                ↪ ());
129         getContentPane().setLayout(layout);
130         layout.setHorizontalGroup(
131             layout.createParallelGroup(org.jdesktop.↙
                    ↪ layout.GroupLayout.LEADING)
132             .add(org.jdesktop.layout.GroupLayout.↙
                    ↪ TRAILING, layout.createSequentialGroup↙
                    ↪ ()
133                 .addContainerGap(org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                        ↪ MAX_VALUE)
```

```
134                 .add(layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING↙
                        ↪ )
135                   .add(jCheckBox3)
136                   .add(jCheckBox13)
137                   .add(jCheckBox6)
138                   .add(jCheckBox4)
139                   .add(jCheckBox2)
140                   .add(jCheckBox1))
141                 .add(67, 67, 67))
142             .add(layout.createSequentialGroup()
143                 .addContainerGap()
144                 .add(jCheckBox5)
145                 .addContainerGap(159, Short.MAX_VALUE))
146             .add(layout.createSequentialGroup()
147                 .addContainerGap()
148                 .add(jCheckBox8)
149                 .addContainerGap(67, Short.MAX_VALUE))
150         );
151         layout.setVerticalGroup(
152             layout.createParallelGroup(org.jdesktop.↙
                    ↪ layout.GroupLayout.LEADING)
153             .add(layout.createSequentialGroup()
154                 .addContainerGap()
155                 .add(jCheckBox1)
156                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
157                 .add(jCheckBox2)
158                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
159                 .add(jCheckBox4)
160                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
161                 .add(jCheckBox6)
162                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
163                 .add(jCheckBox8)
164                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
165                 .add(jCheckBox5)
```

```
166                     .addPreferredGap(org.jdesktop.layout.↙
                            ↪ LayoutStyle.RELATED)
167                     .add(jCheckBox13)
168                     .addPreferredGap(org.jdesktop.layout.↙
                            ↪ LayoutStyle.RELATED)
169                     .add(jCheckBox3)
170                     .addContainerGap(org.jdesktop.layout.↙
                            ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                            ↪ MAX_VALUE))
171             );
172
173             pack();
174         }// </editor-fold>//GEN-END:initComponents
175
176         private void Number_of_Agents_handler(java.awt.event↙
                ↪ .ItemEvent evt) {//GEN-FIRST:↙
                ↪ event_Number_of_Agents_handler
177
178             if(ItemEvent.SELECTED == evt.getStateChange())
179             {
180                 NUMBER_OF_AGENTS = new ShowCharts(God.↙
                        ↪ getRecorder(), Recorder.↙
                        ↪ NUMBER_OF_AGENTS,
181                         "Number of Agents", "Number of Agents", ↙
                            ↪ "Time", "Agents");
182                 Thread chart_NUMBER_OF_AGENTS = new Thread(↙
                        ↪ NUMBER_OF_AGENTS);
183                 chart_NUMBER_OF_AGENTS.start();
184             }
185             else
186             {
187                 NUMBER_OF_AGENTS.dispose();
188             }
189
190         }//GEN-LAST:event_Number_of_Agents_handler
191
192         private void Product_stored_handler(java.awt.event.↙
                ↪ ItemEvent evt) {//GEN-FIRST:↙
                ↪ event_Product_stored_handler
193             if(ItemEvent.SELECTED == evt.getStateChange())
194             {
```

```
195             PRODUCT_STORED = new ShowCharts(God.↙
                    ↪ getRecorder(), Recorder.PRODUCT_STORED,
196                  "Stored product", "Stored product", "↙
                        ↪ Time", "Product");
197             Thread chart_PRODUCT_STORED = new Thread(↙
                    ↪ PRODUCT_STORED);
198         chart_PRODUCT_STORED.start();
199         }
200         else
201         {
202             PRODUCT_STORED.dispose();
203         }
204     }//GEN-LAST:event_Product_stored_handler
205
206     private void Wealth_handler(java.awt.event.ItemEvent↙
        ↪  evt) {//GEN-FIRST:event_Wealth_handler
207         // TODO add your handling code here:
208         if(ItemEvent.SELECTED == evt.getStateChange())
209         {
210             WEALTH = new ShowCharts(God.getRecorder(), ↙
                    ↪ Recorder.WEALTH,
211                  "Wealth", "Wealth", "Time", "Money");
212             Thread chart_WEALTH = new Thread(WEALTH);
213         chart_WEALTH.start();
214         }
215         else
216         {
217             WEALTH.dispose();
218         }
219     }//GEN-LAST:event_Wealth_handler
220
221     private void Total_births_handler(java.awt.event.↙
        ↪ ItemEvent evt) {//GEN-FIRST:↙
        ↪ event_Total_births_handler
222         // TODO add your handling code here:
223         if(ItemEvent.SELECTED == evt.getStateChange())
224         {
225             TOTAL_BIRTHS = new ShowCharts(God.↙
                    ↪ getRecorder(), Recorder.TOTAL_BIRTHS,
226                  "Total births", "Total births", "Time", ↙
                        ↪ "Births" );
```

```
227             Thread chart_TOTAL_BIRTHS = new Thread(↙
                    ↪ TOTAL_BIRTHS);
228         chart_TOTAL_BIRTHS.start();
229         }
230         else
231         {
232             TOTAL_BIRTHS.dispose();
233         }
234     }//GEN-LAST:event_Total_births_handler
235
236     private void Total_deaths(java.awt.event.ItemEvent ↙
        ↪ evt) {//GEN-FIRST:event_Total_deaths
237         // TODO add your handling code here:
238         if(ItemEvent.SELECTED == evt.getStateChange())
239         {
240             TOTAL_DEATHS = new ShowCharts(God.↙
                    ↪ getRecorder(), Recorder.TOTAL_DEATHS,
241                 "Total bankruptcies", "Total ↙
                        ↪ bankruptcies", "Time", "↙
                        ↪ bankruptcies");
242             Thread chart_TOTAL_DEATHS = new Thread(↙
                    ↪ TOTAL_DEATHS);
243         chart_TOTAL_DEATHS.start();
244         }
245         else
246         {
247             TOTAL_DEATHS.dispose();
248         }
249     }//GEN-LAST:event_Total_deaths
250
251     private void Median_wait_time_handler(java.awt.event↙
        ↪ .ItemEvent evt) {//GEN-FIRST:↙
        ↪ event_Median_wait_time_handler
252         // TODO add your handling code here:
253         if(ItemEvent.SELECTED == evt.getStateChange())
254         {
255             MEDIAN_WAIT_TIME = new ShowCharts(God.↙
                    ↪ getRecorder(), Recorder.↙
                    ↪ MEDIAN_WAIT_TIME,
256                 "Median wait time", "Median wait time", ↙
                        ↪ "Time", "Wait Time");
```

```
257             Thread chart_MEDIAN_WAIT_TIME = new Thread(↙
                    ↪ MEDIAN_WAIT_TIME);
258          chart_MEDIAN_WAIT_TIME.start();
259          }
260          else
261          {
262              MEDIAN_WAIT_TIME.dispose();
263          }
264      }//GEN-LAST:event_Median_wait_time_handler
265
266      private void jCheckBox3ItemStateChanged(java.awt.↙
             ↪ event.ItemEvent evt) {//GEN-FIRST:↙
             ↪ event_jCheckBox3ItemStateChanged
267          if(ItemEvent.SELECTED == evt.getStateChange())
268          {
269              PRICE = new ShowCharts(God.getRecorder(), ↙
                     ↪ Recorder.PRICE,
270                  "Price", "Price", "Time", "Price");
271              Thread chart_PRICE = new Thread(PRICE);
272          chart_PRICE.start();
273          }
274          else
275          {
276              PRICE.dispose();
277          }
278      }//GEN-LAST:event_jCheckBox3ItemStateChanged
279
280      private void AgeEventHandler(java.awt.event.↙
             ↪ ItemEvent evt) {//GEN-FIRST:↙
             ↪ event_AgeEventHandler
281          if(ItemEvent.SELECTED == evt.getStateChange())
282          {
283              AGE = new ShowCharts(God.getRecorder(), ↙
                     ↪ Recorder.AGE,
284                  "Age", "Age", "Time", "Age");
285              Thread chart_AGE = new Thread(AGE);
286          chart_AGE.start();
287          }
288          else
289          {
290              PRICE.dispose();
```

```
291             }
292         }//GEN-LAST:event_AgeEventHandler
293 
294         private void jCheckBox5ActionPerformed(java.awt.↙
                ↪ event.ActionEvent evt) {//GEN-FIRST:↙
                ↪ event_jCheckBox5ActionPerformed
295             // TODO add your handling code here:
296         }//GEN-LAST:event_jCheckBox5ActionPerformed
297 
298         /**
299         * @param args the command line arguments
300         */
301         public static void main(String args[]) {
302             java.awt.EventQueue.invokeLater(new Runnable() {
303                 public void run() {
304                     new Report().setVisible(true);
305                 }
306             });
307         }
308 
309         // Variables declaration - do not modify//GEN-BEGIN:↙
                ↪ variables
310         private javax.swing.JCheckBox jCheckBox1;
311         private javax.swing.JCheckBox jCheckBox13;
312         private javax.swing.JCheckBox jCheckBox2;
313         private javax.swing.JCheckBox jCheckBox3;
314         private javax.swing.JCheckBox jCheckBox4;
315         private javax.swing.JCheckBox jCheckBox5;
316         private javax.swing.JCheckBox jCheckBox6;
317         private javax.swing.JCheckBox jCheckBox8;
318         // End of variables declaration//GEN-END:variables
319 
320 }
```

SetInitialProperties.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  /*
7   * SetInitialProperties.java
8   *
9   * Created on 5/Out/2010, 16:27:48
10  */
11
12 package marketsim;
13
14 /**
15  *
16  * @author lfp
17  */
18 public class SetInitialProperties extends javax.swing.↙
      ↪ JFrame {
19
20     private double produceWill = 0; //Factor controlling↙
          ↪  the amount of product this agent produces
21     private double consumeWill = 0; //Factor controlling↙
          ↪  the amount of product this agent consumes
22     private double replicationStrategy = 0; //Factor ↙
          ↪ controlling the replication strategy
23     private double replicationIntensity = 0; //Factor ↙
          ↪ controlling the replication Intensity
24     private double storedProduct = 0; //The amount of ↙
          ↪ stored product
25
26
27     private double storedWealth = 0; //The amount of ↙
          ↪ wealth the agent
28     private long timeLeft =0; //The amount of time the ↙
          ↪ agent will survive
29     private long AgentID = 0;
30     private long parentAgentID  = 0;
31     private long saudade = 0;
32
```

```
33      /** Creates new form SetInitialProperties */
34      public SetInitialProperties() {
35          initComponents();
36      }
37
38      /** This method is called from within the ↙
            ↪ constructor to
39       * initialize the form.
40       * WARNING: Do NOT modify this code. The content of ↙
             ↪ this method is
41       * always regenerated by the Form Editor.
42       */
43      @SuppressWarnings("unchecked")
44      // <editor-fold defaultstate="collapsed" desc="↙
            ↪ Generated Code">//GEN-BEGIN:initComponents
45      private void initComponents() {
46
47          jPanel1 = new javax.swing.JPanel();
48          jLabel1 = new javax.swing.JLabel();
49          jLabel2 = new javax.swing.JLabel();
50          jLabel3 = new javax.swing.JLabel();
51          jLabel4 = new javax.swing.JLabel();
52          jLabel5 = new javax.swing.JLabel();
53          jLabel6 = new javax.swing.JLabel();
54          jSpinner1 = new javax.swing.JSpinner();
55          jSpinner2 = new javax.swing.JSpinner();
56          jSpinner3 = new javax.swing.JSpinner();
57          jSpinner5 = new javax.swing.JSpinner();
58          jSpinner7 = new javax.swing.JSpinner();
59          jSpinner9 = new javax.swing.JSpinner();
60          jLabel7 = new javax.swing.JLabel();
61
62          setDefaultCloseOperation(javax.swing.↙
                ↪ WindowConstants.DISPOSE_ON_CLOSE);
63          setTitle("Inital Properties");
64
65          jPanel1.setBorder(javax.swing.BorderFactory.↙
                ↪ createTitledBorder("Initial Properties"));
66
67          jLabel1.setText("Produce Will");
68
```

```
69          jLabel2.setText("Consume Will");
70
71          jLabel3.setText("Replication Strategy");
72
73          jLabel4.setText("Stored Product");
74
75          jLabel5.setText("Stored Wealth");
76
77          jLabel6.setText("Time Left");
78
79          org.jdesktop.layout.GroupLayout jPanel1Layout = ↙
                ↪ new org.jdesktop.layout.GroupLayout(jPanel1↙
                ↪ );
80          jPanel1.setLayout(jPanel1Layout);
81          jPanel1Layout.setHorizontalGroup(
82               jPanel1Layout.createParallelGroup(org.↙
                     ↪ jdesktop.layout.GroupLayout.LEADING)
83               .add(jPanel1Layout.createSequentialGroup()
84                    .addContainerGap()
85                    .add(jPanel1Layout.createParallelGroup(↙
                           ↪ org.jdesktop.layout.GroupLayout.↙
                           ↪ LEADING)
86                          .add(jLabel5)
87                          .add(jLabel6)
88                          .add(jLabel3)
89                          .add(jLabel4)
90                          .add(jPanel1Layout.↙
                                 ↪ createParallelGroup(org.↙
                                 ↪ jdesktop.layout.GroupLayout.↙
                                 ↪ LEADING)
91                                .add(jLabel1)
92                                .add(org.jdesktop.layout.↙
                                       ↪ GroupLayout.TRAILING, ↙
                                       ↪ jLabel2)))
93                    .addPreferredGap(org.jdesktop.layout.↙
                           ↪ LayoutStyle.RELATED, 18, Short.↙
                           ↪ MAX_VALUE)
94                    .add(jPanel1Layout.createParallelGroup(↙
                           ↪ org.jdesktop.layout.GroupLayout.↙
                           ↪ LEADING)
```

```
95                      .add(jSpinner1, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, ↙
                            ↪ 129, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE)
96                      .add(jSpinner2, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, ↙
                            ↪ 129, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE)
97                      .add(jSpinner5, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, ↙
                            ↪ 129, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE)
98                      .add(jSpinner7, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, ↙
                            ↪ 129, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE)
99                      .add(jSpinner3, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, ↙
                            ↪ 129, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE)
100                     .add(jSpinner9, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, ↙
                            ↪ 129, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE))
101                 .addContainerGap(org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                        ↪ MAX_VALUE))
102         );
103         jPanel1Layout.setVerticalGroup(
104             jPanel1Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
105             .add(jPanel1Layout.createSequentialGroup()
106                 .add(17, 17, 17)
107                 .add(jPanel1Layout.createParallelGroup(↙
                        ↪ org.jdesktop.layout.GroupLayout.↙
                        ↪ BASELINE)
108                     .add(jLabel1)
109                     .add(jSpinner1, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, org↙
                            ↪ .jdesktop.layout.GroupLayout.↙
                            ↪ DEFAULT_SIZE, org.jdesktop.↙
```

```
                    ↪ layout.GroupLayout.↙
                    ↪ PREFERRED_SIZE))
110                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
111                 .add(jPanel1Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
112                   .add(jLabel2)
113                   .add(jSpinner2, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, org↙
                      ↪ .jdesktop.layout.GroupLayout.↙
                      ↪ DEFAULT_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE))
114                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
115                 .add(jPanel1Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
116                   .add(jLabel3)
117                   .add(jSpinner5, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, org↙
                      ↪ .jdesktop.layout.GroupLayout.↙
                      ↪ DEFAULT_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE))
118                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
119                 .add(jPanel1Layout.createParallelGroup(↙
                    ↪ org.jdesktop.layout.GroupLayout.↙
                    ↪ BASELINE)
120                   .add(jLabel4)
121                   .add(jSpinner7, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, org↙
                      ↪ .jdesktop.layout.GroupLayout.↙
                      ↪ DEFAULT_SIZE, org.jdesktop.↙
                      ↪ layout.GroupLayout.↙
                      ↪ PREFERRED_SIZE))
122                 .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED)
```

```
123                 .add(jPanel1Layout.createParallelGroup(↙
                        ↪ org.jdesktop.layout.GroupLayout.↙
                        ↪ BASELINE)
124                     .add(jLabel5)
125                     .add(jSpinner3, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, org↙
                            ↪ .jdesktop.layout.GroupLayout.↙
                            ↪ DEFAULT_SIZE, org.jdesktop.↙
                            ↪ layout.GroupLayout.↙
                            ↪ PREFERRED_SIZE))
126                 .addPreferredGap(org.jdesktop.layout.↙
                        ↪ LayoutStyle.RELATED)
127                 .add(jPanel1Layout.createParallelGroup(↙
                        ↪ org.jdesktop.layout.GroupLayout.↙
                        ↪ BASELINE)
128                     .add(jLabel6)
129                     .add(jSpinner9, org.jdesktop.layout.↙
                            ↪ GroupLayout.PREFERRED_SIZE, org↙
                            ↪ .jdesktop.layout.GroupLayout.↙
                            ↪ DEFAULT_SIZE, org.jdesktop.↙
                            ↪ layout.GroupLayout.↙
                            ↪ PREFERRED_SIZE))
130                 .addContainerGap(org.jdesktop.layout.↙
                        ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                        ↪ MAX_VALUE))
131         );
132
133         jLabel7.setText("Close␣window␣after␣setting␣↙
                ↪ initial␣properties");
134
135         org.jdesktop.layout.GroupLayout layout = new org↙
                ↪ .jdesktop.layout.GroupLayout(getContentPane↙
                ↪ ());
136         getContentPane().setLayout(layout);
137         layout.setHorizontalGroup(
138             layout.createParallelGroup(org.jdesktop.↙
                    ↪ layout.GroupLayout.LEADING)
139             .add(layout.createSequentialGroup()
140                 .addContainerGap()
141                 .add(layout.createParallelGroup(org.↙
                        ↪ jdesktop.layout.GroupLayout.LEADING↙
```

```
                    ↪ )
142                  .add(layout.createSequentialGroup()
143                      .add(jPanel1, org.jdesktop.↙
                             ↪ layout.GroupLayout.↙
                             ↪ DEFAULT_SIZE, org.jdesktop.↙
                             ↪ layout.GroupLayout.↙
                             ↪ DEFAULT_SIZE, Short.↙
                             ↪ MAX_VALUE)
144                      .addContainerGap())
145                  .add(org.jdesktop.layout.GroupLayout↙
                         ↪ .TRAILING, layout.↙
                         ↪ createSequentialGroup()
146                      .add(jLabel7)
147                      .add(25, 25, 25))))
148          );
149          layout.setVerticalGroup(
150              layout.createParallelGroup(org.jdesktop.↙
                     ↪ layout.GroupLayout.LEADING)
151              .add(org.jdesktop.layout.GroupLayout.↙
                     ↪ TRAILING, layout.createSequentialGroup↙
                     ↪ ()
152                  .addContainerGap()
153                  .add(jLabel7)
154                  .addPreferredGap(org.jdesktop.layout.↙
                         ↪ LayoutStyle.RELATED, 18, Short.↙
                         ↪ MAX_VALUE)
155                  .add(jPanel1, org.jdesktop.layout.↙
                         ↪ GroupLayout.PREFERRED_SIZE, org.↙
                         ↪ jdesktop.layout.GroupLayout.↙
                         ↪ DEFAULT_SIZE, org.jdesktop.layout.↙
                         ↪ GroupLayout.PREFERRED_SIZE)
156                  .addContainerGap())
157          );
158
159          pack();
160      }// </editor-fold>//GEN-END:initComponents
161
162
163      // Variables declaration - do not modify//GEN-BEGIN:↙
             ↪ variables
164      private javax.swing.JLabel jLabel1;
```

```
165     private javax.swing.JLabel jLabel2;
166     private javax.swing.JLabel jLabel3;
167     private javax.swing.JLabel jLabel4;
168     private javax.swing.JLabel jLabel5;
169     private javax.swing.JLabel jLabel6;
170     private javax.swing.JLabel jLabel7;
171     private javax.swing.JPanel jPanel1;
172     private javax.swing.JSpinner jSpinner1;
173     private javax.swing.JSpinner jSpinner2;
174     private javax.swing.JSpinner jSpinner3;
175     private javax.swing.JSpinner jSpinner5;
176     private javax.swing.JSpinner jSpinner7;
177     private javax.swing.JSpinner jSpinner9;
178     // End of variables declaration//GEN-END:variables
179
180 }
```

ShowCharts.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import java.awt.BorderLayout;
9  import java.awt.FlowLayout;
10 import java.awt.Toolkit;
11 import java.awt.datatransfer.Clipboard;
12 import java.awt.datatransfer.ClipboardOwner;
13 import java.awt.datatransfer.StringSelection;
14 import java.awt.datatransfer.Transferable;
15 import java.awt.event.ActionEvent;
16 import java.awt.event.ActionListener;
17 import java.text.DecimalFormat;
18 import java.util.ArrayList;
19 import javax.swing.JButton;
20 import javax.swing.JFrame;
21 import javax.swing.JLabel;
22 import javax.swing.JPanel;
23 import javax.swing.JTextField;
24 import org.jfree.chart.ChartFactory;
25 import org.jfree.chart.ChartPanel;
26 import org.jfree.chart.JFreeChart;
27 import org.jfree.chart.plot.PlotOrientation;
28 import org.jfree.data.xy.XYSeries;
29 import org.jfree.data.xy.XYSeriesCollection;
30 import org.apache.commons.math.stat.descriptive.↙
    ↪ DescriptiveStatistics;
31
32
33 /**
34  *
35  * @author Luis Filipe dos Reis Pereira
36  * @email LuisFRPereira@gmail.com
37  *
38  */
```

```
39 public class ShowCharts extends JFrame implements ↙
     ↪ Runnable{
40     ArrayList<MetricPoint> pointList;
41     DescriptiveStatistics descriptiveStatistics = null;
42     DecimalFormat precision =null;
43
44
45     public ShowCharts(Recorder recorder, Integer type, ↙
         ↪ String lable,
46             String title, String x, String y) {
47         super("Metrics charts");
48         descriptiveStatistics = new ↙
             ↪ DescriptiveStatistics();
49         XYSeries xySeries = new XYSeries(lable);
50         pointList = recorder.getList(type);
51         if (pointList!=null) {
52             for (MetricPoint p : pointList) {
53                 xySeries.add(p.getTime().getMilliseconds↙
                     ↪ (), p.getMetric());
54                 descriptiveStatistics.addValue(p.↙
                     ↪ getMetric());
55             }
56             XYSeriesCollection dataset = new ↙
                 ↪ XYSeriesCollection();
57             dataset.addSeries(xySeries);
58             JFreeChart chart = ChartFactory.↙
                 ↪ createXYLineChart(title,
59                     x, y, dataset, PlotOrientation.↙
                         ↪ VERTICAL,
60                     rootPaneCheckingEnabled, ↙
                         ↪ rootPaneCheckingEnabled, ↙
                         ↪ rootPaneCheckingEnabled);
61
62             //setContentPane(new ChartPanel(chart));
63             this.add(new ChartPanel(chart), BorderLayout↙
                 ↪ .CENTER);
64
65             JButton exportToExcelButton = new JButton("↙
                 ↪ Export data to clipboard (tsv)");
66             ActionListener exportToExcelHandler = new ↙
                 ↪ exportToExcelHandler(pointList);
```

```
67              exportToExcelButton.addActionListener(↙
                    ↪ exportToExcelHandler);
68
69              JButton exportToMatLabButton = new JButton("↙
                    ↪ Export␣Data␣to␣clipboard␣(MATLAB)");
70              ActionListener exportToMatLabHandler = new ↙
                    ↪ exportToMatLabHandler(pointList);
71              exportToMatLabButton.addActionListener(↙
                    ↪ exportToMatLabHandler);
72              JPanel JPButtons = new JPanel(new FlowLayout↙
                    ↪ (FlowLayout.CENTER));
73              JPButtons.add(exportToExcelButton);
74              JPButtons.add(exportToMatLabButton);
75
76              this.add(JPButtons, BorderLayout.SOUTH);
77              precision = new DecimalFormat("#0.0000");
78              JPanel stats = new JPanel();
79              stats.setLayout(new FlowLayout(FlowLayout.↙
                    ↪ CENTER));
80
81              stats.add(new JLabel("\u03BC:"));
82              JTextField averageField = new JTextField(↙
                    ↪ precision.format(
83                      descriptiveStatistics.getSum()/↙
                            ↪ descriptiveStatistics.getN()));
84              averageField.setEditable(false);
85              stats.add(averageField);
86
87              stats.add(new JLabel("\u03C3:"));
88              JTextField sigmaField = new JTextField(↙
                    ↪ precision.format(descriptiveStatistics.↙
                    ↪ getStandardDeviation()));
89              sigmaField.setEditable(false);
90              stats.add(sigmaField);
91
92              stats.add(new JLabel("Skewness:"));
93              JTextField skewField = new JTextField(↙
                    ↪ precision.format(descriptiveStatistics.↙
                    ↪ getSkewness()));
94              skewField.setEditable(false);
95              stats.add(skewField);
```

```
96
97              stats.add(new JLabel("Kurtosis:"));
98              JTextField kurtField = new JTextField(↙
                    ↪ precision.format(descriptiveStatistics.↙
                    ↪ getKurtosis()));
99              kurtField.setEditable(false);
100             stats.add(kurtField);
101
102             stats.add(new JLabel("Sample␣Size:"));
103             JTextField ssizeField = new JTextField(↙
                    ↪ precision.format(descriptiveStatistics.↙
                    ↪ getN()));
104             ssizeField.setEditable(false);
105             stats.add(ssizeField);
106             this.add(stats, BorderLayout.NORTH);
107
108
109         }
110         else
111         {
112             JButton NoDataButton = new JButton("No␣Data␣↙
                    ↪ Found");
113             ActionListener NoDataActionHandler =new ↙
                    ↪ NoDataButtonHandler(this);
114             NoDataButton.addActionListener(↙
                    ↪ NoDataActionHandler);
115
116             setContentPane(NoDataButton);
117         }
118
119     }
120     public void run()
121     {
122         pack();
123         setVisible(true);
124
125     }
126     private class NoDataButtonHandler implements ↙
            ↪ ActionListener{
127         JFrame frame;
128         public NoDataButtonHandler(JFrame jFrame)
```

```
129         {
130             frame = jFrame;
131         }
132         public void actionPerformed(ActionEvent event)
133         {
134             frame.dispose();
135         }
136     }
137     private class exportToExcelHandler implements ↙
            ↪ ActionListener, ClipboardOwner
138     {
139         ArrayList<MetricPoint> pointList;
140         public exportToExcelHandler(ArrayList<↙
                ↪ MetricPoint> pointList )
141         {
142             this.pointList=pointList;
143         }
144         public void actionPerformed(ActionEvent event)
145         {
146             String tabel = "";
147             for (MetricPoint p : pointList) {
148                 tabel = tabel +
149                         p.getTime().getMilliseconds() + ↙
                            ↪ "\t" + p.getMetric() + "\n"↙
                            ↪ ;
150             }
151             StringSelection stringSelection = new ↙
                    ↪ StringSelection( tabel );
152             Clipboard clipboard = Toolkit.↙
                    ↪ getDefaultToolkit().getSystemClipboard↙
                    ↪ ();
153             clipboard.setContents(stringSelection, this)↙
                    ↪ ;
154         }
155
156         public void lostOwnership(Clipboard clipboard, ↙
                ↪ Transferable contents) {
157             //do nothing
158         }
159     }
```

```
160     private class exportToMatLabHandler implements ↙
            ↪ ActionListener, ClipboardOwner
161     {
162         ArrayList<MetricPoint> pointList;
163 
164         public exportToMatLabHandler(ArrayList<↙
                ↪ MetricPoint> pointList )
165         {
166             this.pointList=pointList;
167         }
168         public void actionPerformed(ActionEvent event)
169         {
170             String tabel = "[";
171             for (MetricPoint p : pointList) {
172                 tabel = tabel +
173                         p.getTime().getMilliseconds() + ↙
                            ↪ "␣" + p.getMetric() + ";";
174             }
175             tabel = tabel + "]";
176             StringSelection stringSelection = new ↙
                    ↪ StringSelection( tabel );
177             Clipboard clipboard = Toolkit.↙
                    ↪ getDefaultToolkit().getSystemClipboard↙
                    ↪ ();
178             clipboard.setContents(stringSelection, this)↙
                    ↪ ;
179         }
180 
181         public void lostOwnership(Clipboard clipboard, ↙
                ↪ Transferable contents) {
182             //do nothing
183         }
184     }
185 
186 }
```

Transaction.java

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package marketsim;
7
8  import javax.realtime.RelativeTime;
9
10 /**
11  *
12  * @author Luis Filipe dos Reis Pereira
13  * @email LuisFRPereira@gmail.com
14  *
15  */
16 class Transaction {
17
18     public static final byte BUY = 0;
19     public static final byte SELL = 1;
20
21     long AgentIndex = -1;
22     String ID = null;
23     Double price = new Double(0.0);
24     Long amount = new Long(0);
25     byte TYPE = Byte.MAX_VALUE;
26     RelativeTime creationTime;
27
28     public long getAgentIndex() {
29         return AgentIndex;
30     }
31
32     public void setAgentIndex(long AgentIndex) {
33         this.AgentIndex = AgentIndex;
34     }
35
36     public String getID() {
37         return ID;
38     }
39
40     public void setID(String ID) {
```

```
41          this.ID = ID;
42      }
43
44      public byte getTYPE() {
45          return TYPE;
46      }
47
48      public void setTYPE(byte TYPE) {
49          this.TYPE = TYPE;
50      }
51
52      public Long getAmount() {
53          return amount;
54      }
55
56      public void setAmount(Long amount) {
57          this.amount = amount;
58      }
59
60      public Double getPrice() {
61          return price;
62      }
63
64      public void setPrice(Double price) {
65          this.price = price;
66      }
67
68      public RelativeTime getCreationTime() {
69          return creationTime;
70      }
71
72      public void setCreationTime(RelativeTime ↙
           ↪ creationTime) {
73          this.creationTime = creationTime;
74      }
75
76  }
```