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.

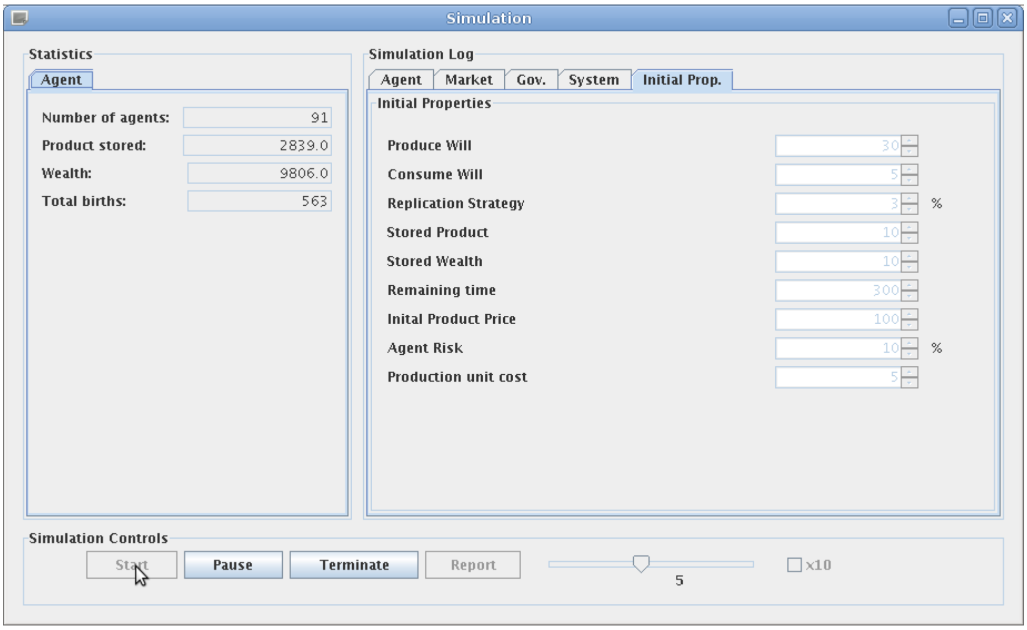
Government is the agent responsible for producing money. It can participate in the market only to buy product. It does not have any other property.

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



Number of agents: 91
Product stored: 2839.0
Wealth: 9806.0
Total births: 563

Produce Will 30
Consume Will 5
Replication Strategy 3 %
Stored Product 10
Stored Wealth 10
Remaining time 300
Inital Product Price 100
Agent Risk 10 %
Production unit cost 5

5

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 society. 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.

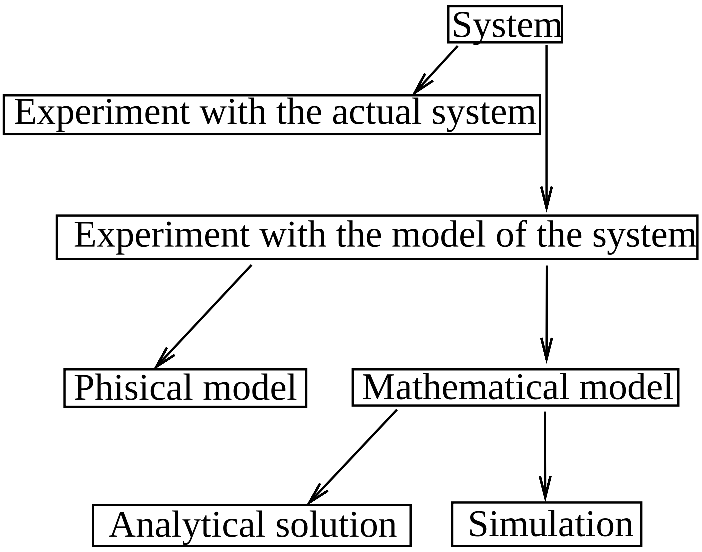

System
Experiment with the actual system
Experiment with the model of the system
Phisical model
Mathematical model
Analytical solution
Simulation

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

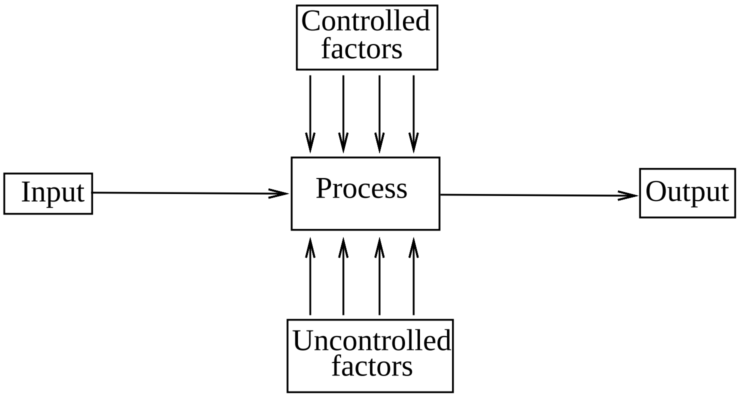


Figure 1.3: General model of process or system

(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:

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

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:

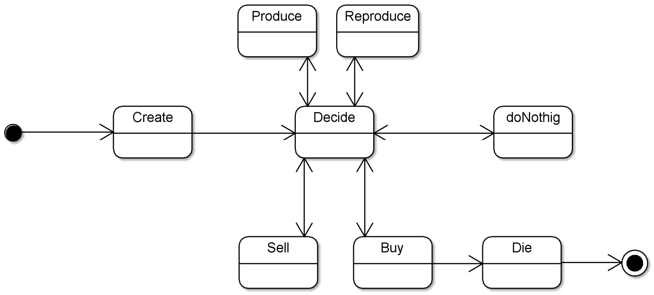


Figure 2.1: State Activity Diagram for the Agent agent

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	+	+	+

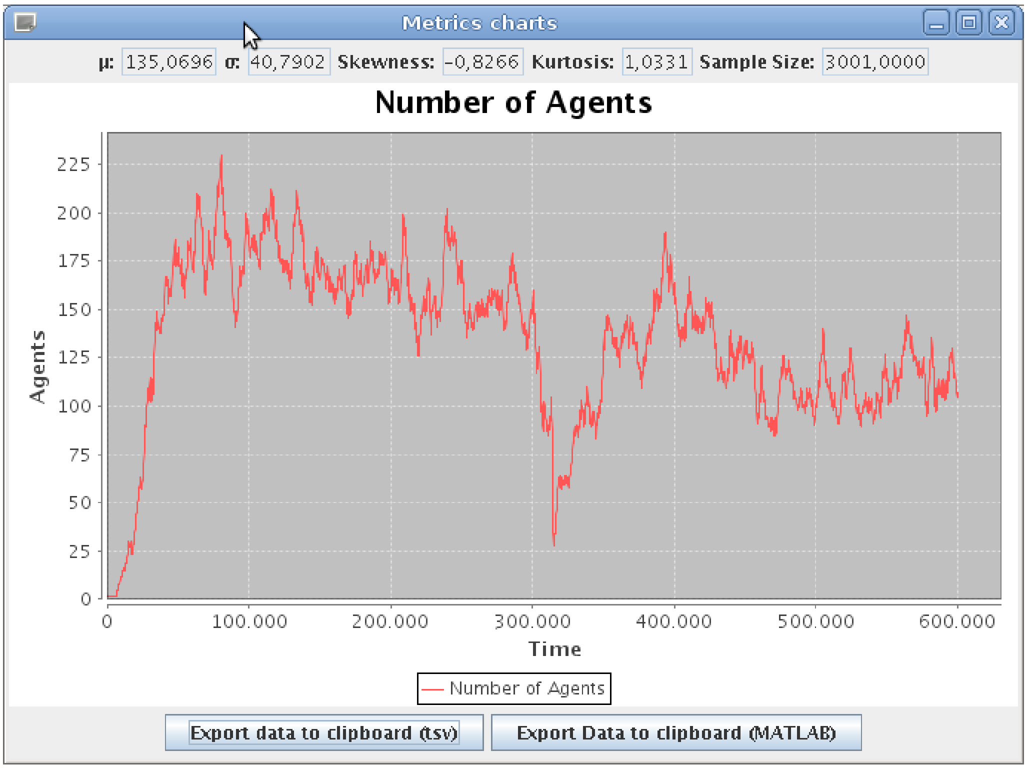
The experiment was repeated two times to have a better estimate on the factors sum of squares.

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

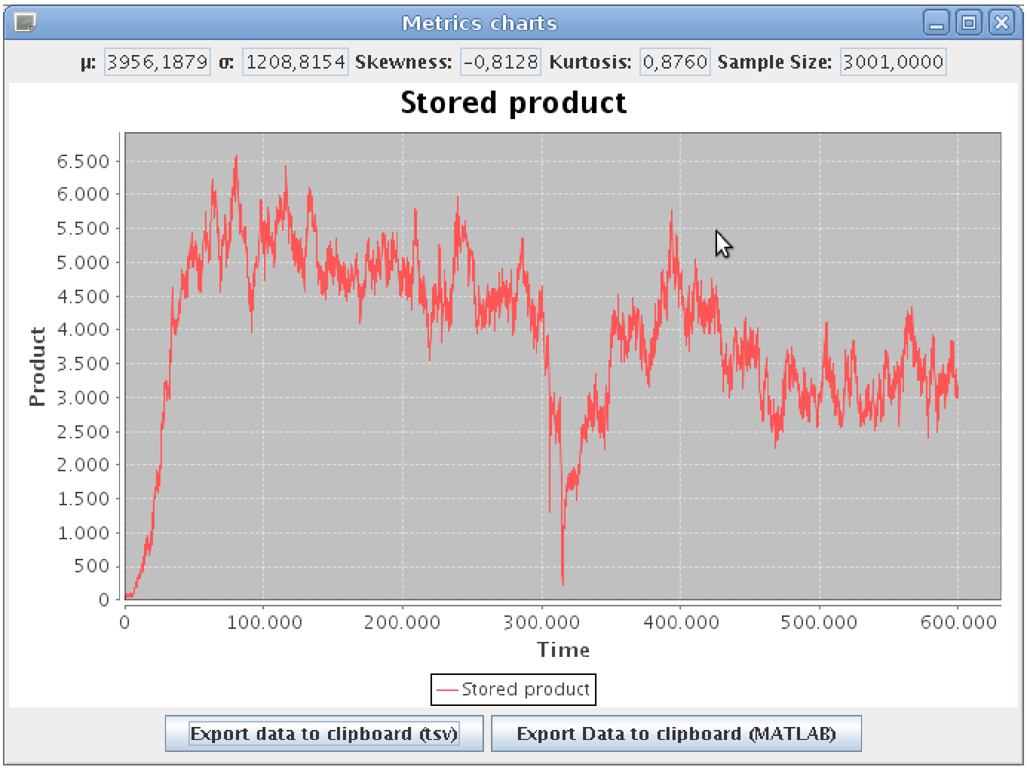


μ: 135,0696 σ: 40,7902 Skewness: -0,8266 Kurtosis: 1,0331 Sample Size: 3001,0000

Figure 3.1: Number of agents time series chart

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



μ: 3956,1879 σ: 1208,8154 Skewness: -0,8128 Kurtosis: 0,8760 Sample Size: 3001,0000

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;

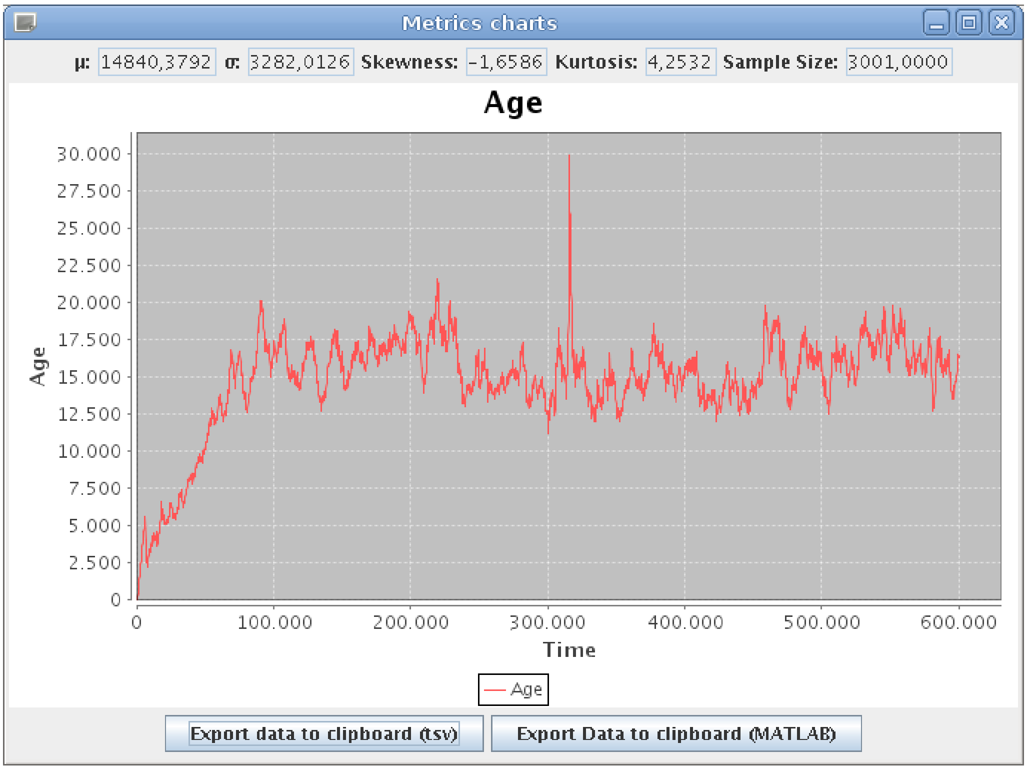
2. Compute the sum of squares for each factor;

3. Compute the total sum of squares;

4. Build the ANOVA table;

5. Determine the significant factors;

6. Estimate the regression model;



μ: 14840,3792 σ: 3282,0126 Skewness: -1,6586 Kurtosis: 4,2532 Sample Size: 3001,0000

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

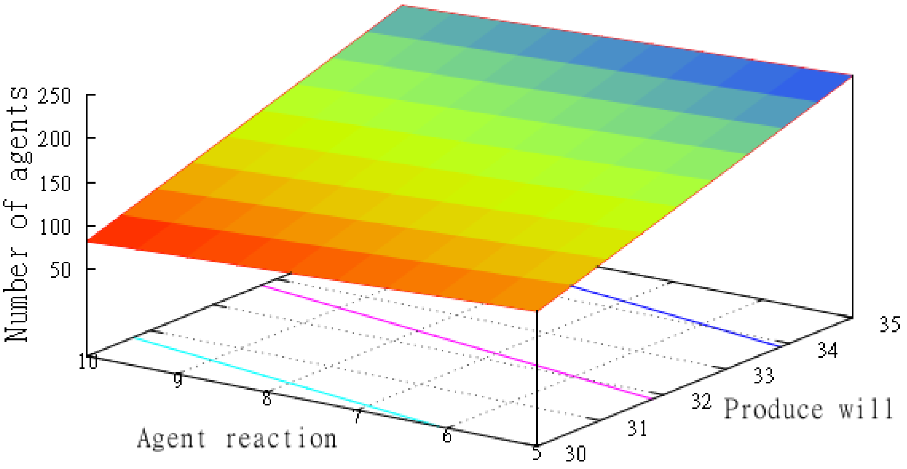


Figure 3.4: Response surface of the Number of agents variable

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

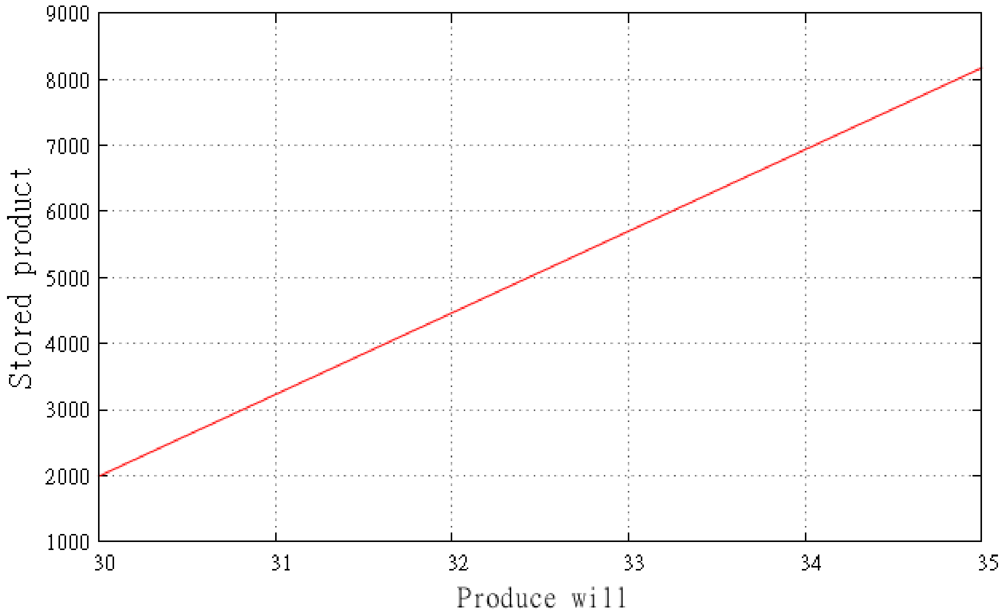


Figure 3.5: Response line of the product stored variable

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

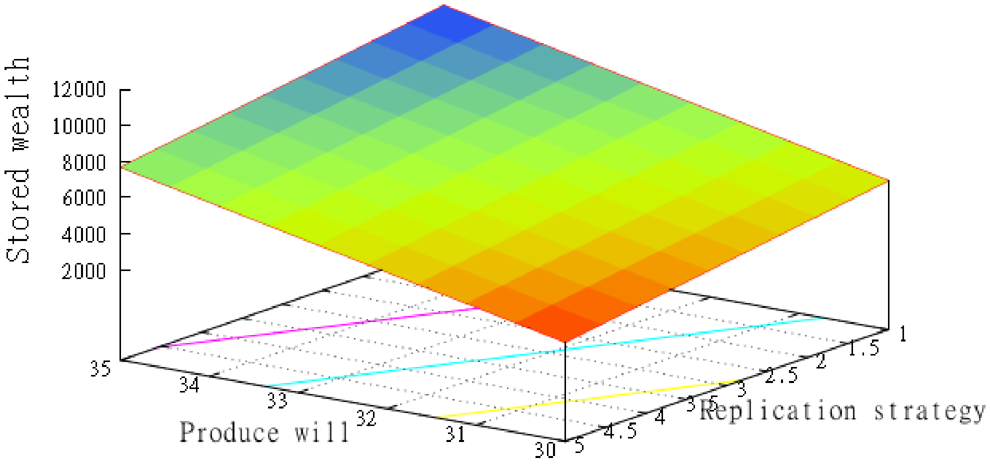


Figure 3.6: Response surface of the Wealth variable

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

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

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

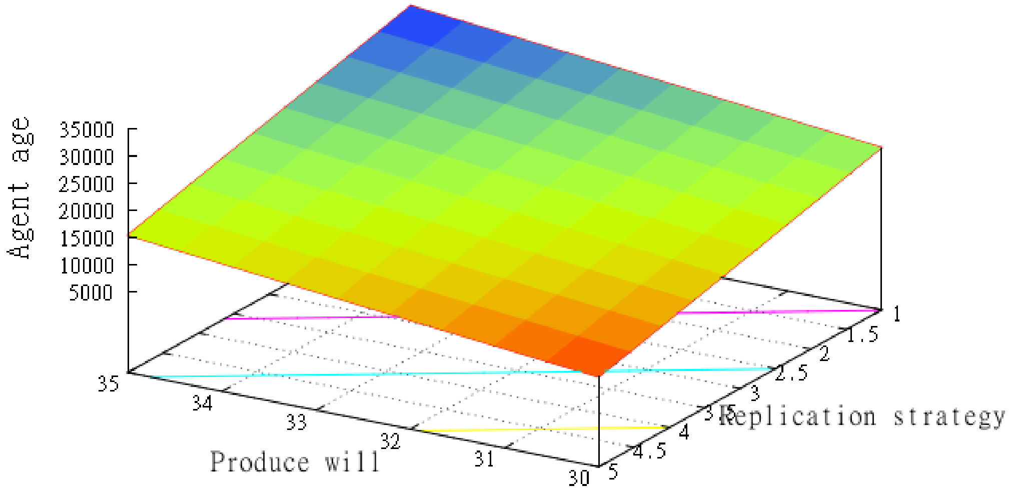


Figure 3.7: Response surface of the Age variable

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			

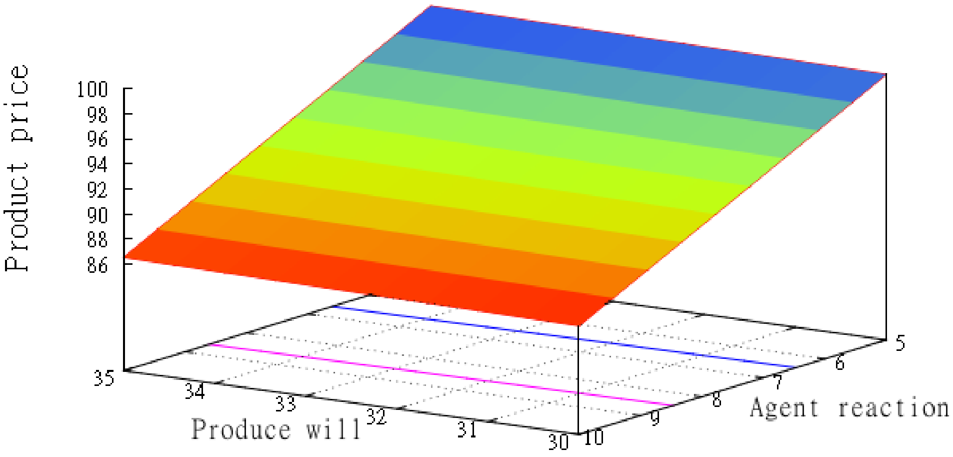


Figure 3.8: Response surface of the Price variable

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

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;


import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.realtime.RelativeTime;
import org.apache.commons.math.MathException;
import org.apache.commons.math.random.JDKRandomGenerator↙
    ↪ ;
import org.apache.commons.math.distribution.↙
    ↪ NormalDistributionImpl;
```

```



/**
 *
 * @author Lu s F. Reis Pereira
 */
public class Agent implements Runnable{
    private static volatile boolean SIM_STOP = false;
    private double produceWill = 0; //Factor controlling the amount of product this agent produces
    private double consumeWill = 1; //Factor controlling the amount of product this agent consumes
                                    //1 = one monetary unit per second
    private double replicationStrategy = 0; //Factor controlling the replication strategy
    private double replicationIntensity = 0; //Factor controlling the replication Intensity
    private double storedProduct = 0; //The amount of stored product


    private double storedWealth = 0; //The amount of wealth the agent
    private int AgentID = 0;
    private long parentAgentID  = 0;
    private long saudade = 0;
    private Market market;
    private final Pool pool;
    private Log log;
    private double lastTransactionPrice = 0;
    private double currentPrice = 0;
    private double productionThreshold = 0.95;
    private double reproductionThreshold = 1.05;

    private static int Decision_DoNothig = 0;
    private static int Decision_Sell = 1;
    private static int Decision_Buy = 2;
    private static int Decision_Produce = 3;
    private static int Decision_Reproduce = 4;
    private DashBoard dashboard;
    private Thread itSelf;
    private God god;
    private RelativeTime lastTime  = new RelativeTime(0,0);
    private ArrayList<Agent> AgentList;
    private double perceptionError=0;
    private JDKRandomGenerator RandomGenerator;
    int seed = AgentID;
    private static volatile int NextID =0;
    private RelativeTime bournTime ;
    private RelativeTime deathTime ;
    private Recorder recorder;

    private static volatile long transactionID = 0;

    public Agent(
            double inheritedProduceWill,
            double inheritedConsumerWill,
            double inheritedReplicationStrategy,
            double inheritedReplicationIntensity,
            double inheritedProduct,
            double inheritedWealth,
            long inheritedparentAgentID,
            int SimSpeedValue,
            Market market,
            Pool pool, Log log, DashBoard dashBoard,
            God god, ArrayList<Agent> AgentList,
            Recorder recorder)
    {
        produceWill = inheritedProduceWill;
        consumeWill = inheritedConsumerWill;
        reproductionThreshold = inheritedReplicationStrategy / 100;
        replicationIntensity = inheritedReplicationIntensity;
        storedProduct = inheritedProduct;
        storedWealth = inheritedWealth;
        parentAgentID = inheritedparentAgentID;
        this.market = market;
        this.pool = pool;
```

```java
        this.log = log;
        this.dashboard = dashBoard;
        this.god=god;
        this.AgentList = AgentList;
        this.recorder = recorder;
        lastTransactionPrice = market.getCurrentPrice();

        addID();

        //connection=log.getConnection();

    }
     private synchronized void addID(){
        synchronized(Lock.AgentIDlock)
        {
            AgentID = NextID;
            NextID++;
        }

    }

    public void run() {

        int decision = 0;
        double decisionIntensity = 0.0;
        double P=0;
        synchronized(dashboard.getjTextArea2()){
                dashboard.getjTextArea2().append(this.itSelf.getName() + "****BORN****\n");
                dashboard.getjTextArea2().setCaretPosition(dashboard.getjTextArea2().getDocument().getLength());
                bournTime = new RelativeTime(God.getSimulationTime());
        }
        Boolean firstrun =true;
        while(storedProduct > 0 )
        {
            if(!firstrun){
             god.incrementSemaphore();
        }

        else
            firstrun =false;


           synchronized(Lock.timelock){
                    try {
                        while(lastTime.equals(God.↙
                           ↪ getSimulationTime()))
                        {Lock.timelock.wait();}
                    } catch (InterruptedException ex↙
                      ↪ ) {
                        Logger.getLogger(Agent.class↙
                           ↪ .getName()).log(Level.↙
                           ↪ SEVERE, null, ex);
                    }

            }
           lastTime.set(God.getSimulationTime());


        if(SIM_STOP){

            synchronized(dashboard.getjTextArea2()){
            dashboard.getjTextArea2().append(this.↙
               ↪ itSelf.getName() + "****PAUSED****\↙
               ↪ n");
            dashboard.getjTextArea2().↙
               ↪ setCaretPosition(dashboard.↙
               ↪ getjTextArea2().getDocument().↙
               ↪ getLength());
            }
            synchronized(Lock.lock){
                    try {
                        while(SIM_STOP){Lock.lock.↙
                           ↪ wait();}
                    } catch (InterruptedException ex↙
                      ↪ ) {
                        Logger.getLogger(Agent.class.getName()).log(Level.SEVERE, null, ex);
                    }

            }
            synchronized(dashboard.getjTextArea2()){
            dashboard.getjTextArea2().append(this.itSelf.getName() + "****UNPAUSED****\n");
            dashboard.getjTextArea2().setCaretPosition(dashboard.getjTextArea2().getDocument().getLength());
            }
        }

        decision = 0;
        decisionIntensity = 0.0;
        // Decider
        //default decision : do Nothig
        decision = Agent.Decision_DoNothig;
        if(currentPrice == 0.0)
        {
            currentPrice = Double.parseDouble(dashboard.getjSpinner11().getValue().toString());
        }
        else
        {
            currentPrice = market.getCurrentPrice();
        }
        lastTransactionPrice = market.getLastTransactionPrice();
        if(lastTransactionPrice == 0.0)
        {
            lastTransactionPrice=currentPrice;
        }
        RandomGenerator= new JDKRandomGenerator();
        RandomGenerator.setSeed(seed++);
```

```
perceptionError=RandomGenerator.nextGaussian();
try {
    P =  currentPrice * (  currentPrice / lastTransactionPrice
              - perceptionError * Double.parseDouble(dashboard.getjSpinner2().getValue().toString())/100);
    P = Math.round(P);

} catch (Exception ex) {
    Logger.getLogger(Agent.class.getName()).log(Level.SEVERE,
             "CurrentPrice: {0} LastTransactionPrice:{1} perceptionError :{2}",
             new Object[]{currentPrice, lastTransactionPrice, perceptionError});
    Logger.getLogger(Agent.class.getName()).log(Level.SEVERE, null, ex);
}
if( P > 0)
{
    decision = Agent.Decision_Sell;
    decisionIntensity = P;
}
else if( P < 0)
{
    if (storedWealth > -P) {
        decision = Agent.Decision_Buy;
        decisionIntensity = P;
    }
    else
        decision = Agent.Decision_DoNothig;
}
else if ( P == 0 )
{
    decision = Agent.Decision_DoNothig;
}
```

```
                if(storedWealth > storedProduct * currentPrice * reproductionThreshold &&
                        storedProduct > produceWill*productionThreshold)
                {
                    Double fertility = RandomGenerator.nextGaussian();
                    //RelativeTime timeFactor = God.getTimePassed().subtract(new RelativeTime(6750, 0));
                    //double fertilityOportunity = Math.abs((bournTime.subtract(God.getSimulationTime()).getMilliseconds() - 6750L) / God.getTimePassed().getMilliseconds());
                    double fertilityOportunity = Math.abs(Math.sin(bournTime.subtract(God.getSimulationTime()).getMilliseconds() / 5000L) );
                    Double NormalVariable = fertility*fertilityOportunity;
//                      double probability = 0;
//                      try {
//                          probability = new NormalDistributionImpl(0.0, 1.0).cumulativeProbability(NormalVariable);
//                          //System.out.println(”prob: ” + probability+ ”fert: ” + fertility + ”fertOpo: ” + fertilityOportunity);
//                      } catch (MathException ex) {
//                          Logger.getLogger(Agent.class.getName()).log(Level.SEVERE, null, ex);
//                      }


                    //if (0.25 > probability || probability > 0.35)
                    if(NormalVariable >0.6)
                    {
                        decision = Agent.Decision_Reproduce;
                decisionIntensity = storedProduct * currentPrice
                    * (reproductionThreshold * 100);
            }
        }
        //produce to sell
        if((storedWealth < storedProduct * currentPrice * productionThreshold &&
                //produce to survive
                storedProduct < produceWill*productionThreshold//)
                //Must have money to produce
                && storedWealth > 0)
                )
        {
            decision = Agent.Decision_Produce;
            decisionIntensity = storedProduct * currentPrice * ( 1 - productionThreshold);
        }

        if( decision == Agent.Decision_Buy)
        {
            // TODO log decision
            synchronized(dashboard.getjTextArea2()){
                dashboard.getjTextArea2().append(this.itSelf.getName() + "****BUY****@" + Math.abs(P) + "\n");
                dashboard.getjTextArea2().setCaretPosition(dashboard.getjTextArea2().getDocument().getLength());
              }
            Transaction transaction = new Transaction();
            transaction.setAgentIndex(this.AgentID);
            transaction.setAmount(1L);
            transaction.setPrice(P);
            transaction.setTYPE(Transaction.BUY);
            synchronized(this){
                transaction.setID(””+transactionID);
                transactionID++;
            }
            transaction.setCreationTime(God.↙
               ↪ getSimulationTime());
            synchronized(pool){pool.setElement(↙
               ↪ transaction);}
        }
        if( decision == Agent.Decision_DoNothig)
        {
            // TODO log decision
            synchronized(dashboard.getjTextArea2()){
                     dashboard.getjTextArea2().↙
                        ↪ append(this.itSelf.getName↙
                        ↪ () + ”****DoNothig****” + ↙
                        ↪ P + ”\n”);
                     dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard↙
                        ↪ .getjTextArea2().↙
                        ↪ getDocument().getLength())↙
                        ↪ ;
                  }
        }
        if( decision == Agent.Decision_Produce)
        {
            // TODO log decision
            this.storedWealth = this.storedWealth -
                    Double.parseDouble(dashboard.↙
                       ↪ getjSpinner3().getValue().↙
                       ↪ toString());
            synchronized(dashboard.getjTextArea2()){
                     dashboard.getjTextArea2().↙
                        ↪ append(this.itSelf.getName↙
                        ↪ () + ”****Produce****\n”);
                     dashboard.getjTextArea2().↙
                        ↪ setCaretPosition(dashboard↙
                        ↪ .getjTextArea2().↙
                        ↪ getDocument().getLength())↙
                        ↪ ;
                  }
```

```

                      storedProduct=storedProduct+produceWill;
                      // Todo Log decision Result
                 }
                if( decision == Agent.Decision_Sell)
                 {
                      // TODO log decision
                    synchronized(dashboard.getjTextArea2()){
                                dashboard.getjTextArea2().append(this.itSelf.getName() + "****SELL****@" + P + "\n");
                                dashboard.getjTextArea2().setCaretPosition(dashboard.getjTextArea2().getDocument().getLength());
                            }
                          // Place ask
                          Transaction transaction = new Transaction();
                          transaction.setAgentIndex(this.AgentID);
                          transaction.setAmount(1L);
                          transaction.setPrice(P);
                          transaction.setTYPE(Transaction.SELL);
                          synchronized(this){
                               transaction.setID(""+transactionID);
                               transactionID++;
                          }
                          transaction.setCreationTime(God.getSimulationTime());
                          synchronized(pool){pool.setElement(transaction);}


                    // Todo Log decision Result
                 }
                 if( decision == Agent.Decision_Reproduce)
```

```
{

    Agent tmpAgent = new Agent(
    Double.parseDouble(dashboard.↙
        ↪ getjSpinner1().getValue().toString↙
        ↪ ()),
    Double.parseDouble(dashboard.↙
        ↪ getjSpinner6().getValue().toString↙
        ↪ ()),
    Double.parseDouble(dashboard.↙
        ↪ getjSpinner8().getValue().toString↙
        ↪ ()),
    0.0,
    Double.parseDouble(dashboard.↙
        ↪ getjSpinner9().getValue().toString↙
        ↪ ()),
    Double.parseDouble(dashboard.↙
        ↪ getjSpinner7().getValue().toString↙
        ↪ ()),
    0,
    0,
    market,
    pool, log, dashboard, god, God.↙
        ↪ getAgentList(), recorder);

    synchronized(Lock.AgentListlock){
        God.getAgentList().add(tmpAgent);
        Thread tmpThread = new Thread(↙
            ↪ tmpAgent, "Thread␣for␣Agent␣↙
            ↪ number␣" +God.getAgentList().↙
            ↪ size());
        tmpAgent.setItSelf(tmpThread);
        tmpThread.start();
    }

    synchronized(this)
    {
        storedProduct = storedProduct - ↙
            ↪ Double.parseDouble(dashboard.↙
            ↪ getjSpinner9().getValue().↙
            ↪ toString());
```

```
                }


                synchronized(dashboard.getjTextArea2()){
                        dashboard.getjTextArea2().append(this.itSelf.getName() + "****REPRODUCE**** Created Agent :" + tmpAgent.getAgentID() +"\n");
                        dashboard.getjTextArea2().setCaretPosition(dashboard.getjTextArea2().getDocument().getLength());
                      }
                // TODO reproduce
               // Todo Log decision Result
            }
          god.decrementSemaphore();
        }

        synchronized(Lock.AgentListlock)
        {
            AgentList.set(AgentID, null);
        }
        synchronized(dashboard.getjTextArea2()){
                dashboard.getjTextArea2().append(this.itSelf.getName() + "****DIES***\n");
//                      synchronized(Lock.Recorderlock)
//                      {
//                          deathTime = new RelativeTime(God.getSimulationTime());
//                          recorder.addValue(Recorder.AGE, new MetricPoint(God.getSimulationTime(), deathTime.subtract(bournTime).getMilliseconds()));
//                      }

                synchronized(this)
                {
                    try {
                        finalize();
                    } catch (Throwable ex) {
                        Logger.getLogger(Agent.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }

//                synchronized(Lock.AgentListlock)
//                {
//                    synchronized(Lock.AgentIDlock)
//                    {
//                        AgentList.set(AgentID, null);
//                    }
//
//                }

//                synchronized(AgentList.get(AgentID))
//                {
//                    AgentList.set(AgentID, null);
//
//                }
        }
    }

    @Override
    protected void finalize() throws Throwable
    {
        super.finalize();
        synchronized(Lock.AgentListlock)
        {
            synchronized(Lock.AgentIDlock)
            {
                AgentList.set(AgentID, null);
            }

        }
    }
        public double getConsumeWill() {
        return consumeWill;
```

```java
    }
    public double getProduceWill() {
        return produceWill;
    }
    public double getReplicationIntensity() {
        return replicationIntensity;
    }
    public double getReplicationStrategy() {
        return replicationStrategy;
    }
    public double getStoredProduct() {
        synchronized(Lock.transactionlock)
        {
            return storedProduct;
        }
    }

    public long getAgentID() {
        return AgentID;
    }
    public long getParentAgentID() {
        return parentAgentID;
    }

    public double getStoredWealth() {
        synchronized(Lock.transactionlock)
        {
            return storedWealth;
        }
    }
    public long getSaudade() {
        return saudade;
    }

    public static long getTransactionID() {
        return transactionID;
    }
    public static void setSIM_STOP(boolean SIM_STOP) {
        Agent.SIM_STOP = SIM_STOP;
    }
    public Thread getItSelf() {
            return itSelf;
        }
        public void setItSelf(Thread itSelf) {
            this.itSelf = itSelf;
        }

        public static boolean isSIM_STOP() {
            return SIM_STOP;
        }

        public void setStoredProduct(double storedProduct) {
            synchronized(Lock.transactionlock)
            {
                this.storedProduct = storedProduct;
            }
        }

        public static void setTransactionID(long transactionID) {
            Agent.transactionID = transactionID;
        }

        public void setStoredWealth(double storedWealth) {
            synchronized(Lock.transactionlock)
            {
                this.storedWealth = storedWealth;
            }

        }

        public RelativeTime getBournTime() {
            return bournTime;
        }


    }
```

DashBoard.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * DashBoard.java
 *
 * Created on 5/Out/2010, 14:10:35
 */

package marketsim;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JSlider;
import javax.swing.JSpinner;
import javax.swing.JTextArea;
import javax.swing.JTextField;


/**
 *
 * @author lfp
 */
public class DashBoard extends javax.swing.JFrame {

    God god;
    lauchSim lauchSim;
    /** Creates new form DashBoard */
    public DashBoard(God god, lauchSim lauchSim) {
        this.god = god;
        this.lauchSim=lauchSim;
        initComponents();
        jTabbedPane2.setSelectedIndex(4);
        jSpinner1.setValue(30);
        jSpinner6.setValue(5);
        jSpinner8.setValue(3);
        jSpinner9.setValue(10);
        jSpinner7.setValue(10);
        jSpinner10.setValue(10);
        jSpinner11.setValue(100);
        jSpinner3.setValue(5);
        jSpinner2.setValue(100);


        jLabel3.setVisible(false);
        jLabel4.setText("" + jSlider1.getValue());

//          System.out.println(System.getProperty("java.version"));

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jSlider1 = new javax.swing.JSlider();
        jCheckBox1 = new javax.swing.JCheckBox();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel4 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jTextField6 = new javax.swing.JTextField();
        jPanel3 = new javax.swing.JPanel();
        jTabbedPane2 = new javax.swing.JTabbedPane();
        jPanel8 = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea2 = new javax.swing.JTextArea();
        jPanel10 = new javax.swing.JPanel();
        jScrollPane3 = new javax.swing.JScrollPane();
        jTextArea3 = new javax.swing.JTextArea();
        jPanel11 = new javax.swing.JPanel();
        jScrollPane4 = new javax.swing.JScrollPane();
        jTextArea4 = new javax.swing.JTextArea();
        jPanel9 = new javax.swing.JPanel();
        jScrollPane5 = new javax.swing.JScrollPane();
        jTextArea5 = new javax.swing.JTextArea();
        jPanel13 = new javax.swing.JPanel();
        jLabel29 = new javax.swing.JLabel();
        jLabel30 = new javax.swing.JLabel();
        jLabel33 = new javax.swing.JLabel();
        jLabel34 = new javax.swing.JLabel();
        jLabel35 = new javax.swing.JLabel();
        jLabel36 = new javax.swing.JLabel();
        jSpinner1 = new javax.swing.JSpinner();
        jSpinner6 = new javax.swing.JSpinner();
        jSpinner7 = new javax.swing.JSpinner();
        jSpinner8 = new javax.swing.JSpinner();
        jSpinner9 = new javax.swing.JSpinner();
        jSpinner10 = new javax.swing.JSpinner();
        jSpinner11 = new javax.swing.JSpinner();
        jLabel28 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jSpinner2 = new javax.swing.JSpinner();
        jLabel7 = new javax.swing.JLabel();
        jSpinner3 = new javax.swing.JSpinner();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Simulation");

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Simulation Controls"));

        jButton1.setText("Start");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                StartSimHandler(evt);
            }
        });

        jButton2.setText("Pause");
        jButton2.setEnabled(false);
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                PauseResumeHandler(evt);
            }
        });

        jButton3.setText("Terminate");
        jButton3.setEnabled(false);
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                TerminateHandler(evt);
            }
        });

        jButton4.setText("Report");
        jButton4.setEnabled(false);
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ReportHandler(evt);
            }
        });

        jSlider1.setForeground(new java.awt.Color(255, 0, 0));
        jSlider1.setMajorTickSpacing(1);
        jSlider1.setMaximum(10);
        jSlider1.setMinimum(1);
        jSlider1.setMinorTickSpacing(1);
        jSlider1.setSnapToTicks(true);
        jSlider1.setToolTipText("<html>Simulation Speed<br> <h3>Number of seconds in one second </h3></br></html>");
        jSlider1.setValue(1);
        jSlider1.addChangeListener(new javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                jSlider1StateChanged(evt);
            }
        });
        jSlider1.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
            public void propertyChange(java.beans.PropertyChangeEvent evt) {
                jSlider1PropertyChange(evt);
            }
        });

        jCheckBox1.setText("x10");
        jCheckBox1.setToolTipText("<html><h1>Can crash your system</h1></html>");
        jCheckBox1.setSelectedIcon(new javax.swing.ImageIcon(getClass().getResource("/marketsim/600px-Icon-Warning-Red.svg.png"))); // NOI18N
        jCheckBox1.addItemListener(new java.awt.event.ItemListener() {
            public void itemStateChanged(java.awt.event.ItemEvent evt) {
                FullSpeedHandler(evt);
            }
        });

        jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/marketsim/600px-Icon-Warning-Red.svg.png"))); // NOI18N
        jLabel3.setToolTipText("<html><h1>Can crash your system</h1></html>");

        jLabel4.setText("jLabel4");

        org.jdesktop.layout.GroupLayout jPanel1Layout = new org.jdesktop.layout.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, jPanel1Layout.createSequentialGroup()
                .add(54, 54, 54)
                .add(jButton1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 71, Short.MAX_VALUE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jButton2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 78, Short.MAX_VALUE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jButton3, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 105, Short.MAX_VALUE)
                .addPreferredGap(org.jdesktop.layout.↙
                   ↪ LayoutStyle.RELATED)
                .add(jButton4, org.jdesktop.layout.↙
                   ↪ GroupLayout.DEFAULT_SIZE, 62, Short↙
                   ↪ .MAX_VALUE)
                .add(jPanel1Layout.createParallelGroup(↙
                   ↪ org.jdesktop.layout.GroupLayout.↙
                   ↪ LEADING, false)
                    .add(jPanel1Layout.↙
                       ↪ createSequentialGroup()
                        .add(18, 18, 18)
                        .add(jSlider1, org.jdesktop.↙
                           ↪ layout.GroupLayout.↙
                           ↪ PREFERRED_SIZE, org.↙
                           ↪ jdesktop.layout.GroupLayout↙
                           ↪ .DEFAULT_SIZE, org.jdesktop↙
                           ↪ .layout.GroupLayout.↙
                           ↪ PREFERRED_SIZE)
                        .add(18, 18, 18))
                    .add(org.jdesktop.layout.GroupLayout↙
                       ↪ .TRAILING, jPanel1Layout.↙
                       ↪ createSequentialGroup()
                        .addPreferredGap(org.jdesktop.↙
                           ↪ layout.LayoutStyle.RELATED,↙
                           ↪  org.jdesktop.layout.↙
                           ↪ GroupLayout.DEFAULT_SIZE, ↙
                           ↪ Short.MAX_VALUE)
                        .add(jLabel4)
                        .add(89, 89, 89)))
                .add(jCheckBox1)
                .addPreferredGap(org.jdesktop.layout.↙
                   ↪ LayoutStyle.RELATED)
                .add(jLabel3)
                .add(148, 148, 148))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(org.↙
               ↪ jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.↙
               ↪ TRAILING, jPanel1Layout.↙
               ↪ createSequentialGroup()
            .add(jPanel1Layout.createParallelGroup(↙
             ↪ org.jdesktop.layout.GroupLayout.↙
             ↪ CENTER)
              .add(jButton1)
              .add(jButton2)
              .add(jButton3)
              .add(jButton4)
              .add(jSlider1, org.jdesktop.layout.↙
                ↪ GroupLayout.PREFERRED_SIZE, org↙
                ↪ .jdesktop.layout.GroupLayout.↙
                ↪ DEFAULT_SIZE, org.jdesktop.↙
                ↪ layout.GroupLayout.↙
                ↪ PREFERRED_SIZE)
              .add(jCheckBox1)
              .add(jLabel3))
            .add(21, 21, 21))
          .add(org.jdesktop.layout.GroupLayout.↙
           ↪ TRAILING, jPanel1Layout.↙
           ↪ createSequentialGroup()
            .add(jLabel4)
            .addContainerGap())
        );

        jPanel2.setBorder(javax.swing.BorderFactory.↙
         ↪ createTitledBorder("Statistics"));

        jLabel1.setText("Number␣of␣agents:");

        jLabel2.setText("Product␣stored:");

        jLabel11.setText("Wealth:");

        jLabel12.setText("Total␣births:");

        jTextField1.setEditable(false);
        jTextField1.setHorizontalAlignment(javax.swing.↙
         ↪ JTextField.RIGHT);
        jTextField1.setText("0");
        jTextField1.addActionListener(new java.awt.event↙
         ↪ .ActionListener() {
            public void actionPerformed(java.awt.event.↙
              ↪ ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jTextField2.setEditable(false);
        jTextField2.setHorizontalAlignment(javax.swing.↙
          ↪ JTextField.RIGHT);
        jTextField2.setText(”0”);

        jTextField4.setEditable(false);
        jTextField4.setHorizontalAlignment(javax.swing.↙
          ↪ JTextField.RIGHT);
        jTextField4.setText(”0”);

        jTextField6.setEditable(false);
        jTextField6.setHorizontalAlignment(javax.swing.↙
          ↪ JTextField.RIGHT);
        jTextField6.setText(”0”);

        org.jdesktop.layout.GroupLayout jPanel4Layout = ↙
          ↪ new org.jdesktop.layout.GroupLayout(jPanel4↙
          ↪ );
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(
            jPanel4Layout.createParallelGroup(org.↙
              ↪ jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                .add(jPanel4Layout.createParallelGroup(↙
                  ↪ org.jdesktop.layout.GroupLayout.↙
                  ↪ LEADING)
                    .add(jPanel4Layout.↙
                      ↪ createSequentialGroup()
                        .add(jLabel1)
                        .addPreferredGap(org.jdesktop.↙
                          ↪ layout.LayoutStyle.RELATED)
                        .add(jTextField1, org.jdesktop.↙
                          ↪ layout.GroupLayout.↙
                          ↪ DEFAULT_SIZE, 128, Short.↙
```

```
                           ↪ MAX_VALUE))
                    .add(jPanel4Layout.↙
                       ↪ createSequentialGroup()
                        .add(jLabel2)
                        .add(33, 33, 33)
                        .add(jTextField2, org.jdesktop.↙
                           ↪ layout.GroupLayout.↙
                           ↪ DEFAULT_SIZE, 129, Short.↙
                           ↪ MAX_VALUE))
                    .add(jPanel4Layout.↙
                       ↪ createSequentialGroup()
                        .add(jPanel4Layout.↙
                           ↪ createParallelGroup(org.↙
                           ↪ jdesktop.layout.GroupLayout↙
                           ↪ .LEADING)
                            .add(jLabel11)
                            .add(jLabel12))
                        .add(55, 55, 55)
                        .add(jPanel4Layout.↙
                           ↪ createParallelGroup(org.↙
                           ↪ jdesktop.layout.GroupLayout↙
                           ↪ .LEADING)
                            .add(jTextField6, org.↙
                               ↪ jdesktop.layout.↙
                               ↪ GroupLayout.↙
                               ↪ DEFAULT_SIZE, 130, ↙
                               ↪ Short.MAX_VALUE)
                            .add(jTextField4, org.↙
                               ↪ jdesktop.layout.↙
                               ↪ GroupLayout.↙
                               ↪ DEFAULT_SIZE, 130, ↙
                               ↪ Short.MAX_VALUE)))))
                .addContainerGap())
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(org.↙
               ↪ jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                .add(jPanel4Layout.createParallelGroup(↙
                   ↪ org.jdesktop.layout.GroupLayout.↙
```

```
    ↪ BASELINE)
        .add(jLabel1)
        .add(jTextField1, org.jdesktop.↙
            ↪ layout.GroupLayout.↙
            ↪ PREFERRED_SIZE, org.jdesktop.↙
            ↪ layout.GroupLayout.DEFAULT_SIZE↙
            ↪ , org.jdesktop.layout.↙
            ↪ GroupLayout.PREFERRED_SIZE))
.addPreferredGap(org.jdesktop.layout.↙
    ↪ LayoutStyle.RELATED)
.add(jPanel4Layout.createParallelGroup(↙
    ↪ org.jdesktop.layout.GroupLayout.↙
    ↪ BASELINE)
        .add(jLabel2)
        .add(jTextField2, org.jdesktop.↙
            ↪ layout.GroupLayout.↙
            ↪ PREFERRED_SIZE, org.jdesktop.↙
            ↪ layout.GroupLayout.DEFAULT_SIZE↙
            ↪ , org.jdesktop.layout.↙
            ↪ GroupLayout.PREFERRED_SIZE))
.addPreferredGap(org.jdesktop.layout.↙
    ↪ LayoutStyle.RELATED)
.add(jPanel4Layout.createParallelGroup(↙
    ↪ org.jdesktop.layout.GroupLayout.↙
    ↪ BASELINE)
        .add(jLabel11)
        .add(jTextField4, org.jdesktop.↙
            ↪ layout.GroupLayout.↙
            ↪ PREFERRED_SIZE, org.jdesktop.↙
            ↪ layout.GroupLayout.DEFAULT_SIZE↙
            ↪ , org.jdesktop.layout.↙
            ↪ GroupLayout.PREFERRED_SIZE))
.addPreferredGap(org.jdesktop.layout.↙
    ↪ LayoutStyle.RELATED)
.add(jPanel4Layout.createParallelGroup(↙
    ↪ org.jdesktop.layout.GroupLayout.↙
    ↪ BASELINE)
        .add(jLabel12)
        .add(jTextField6, org.jdesktop.↙
            ↪ layout.GroupLayout.↙
            ↪ PREFERRED_SIZE, org.jdesktop.↙
```

```
                        ↪ layout.GroupLayout.DEFAULT_SIZE↙
                        ↪ , org.jdesktop.layout.↙
                        ↪ GroupLayout.PREFERRED_SIZE))
                    .addContainerGap(218, Short.MAX_VALUE))
        );

        jTabbedPane1.addTab("Agent", jPanel4);

        org.jdesktop.layout.GroupLayout jPanel2Layout = ↙
            ↪ new org.jdesktop.layout.GroupLayout(jPanel2↙
            ↪ );
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(org.↙
                ↪ jdesktop.layout.GroupLayout.LEADING)
            .add(jTabbedPane1)
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(org.↙
                ↪ jdesktop.layout.GroupLayout.LEADING)
            .add(jTabbedPane1, org.jdesktop.layout.↙
                ↪ GroupLayout.DEFAULT_SIZE, 399, Short.↙
                ↪ MAX_VALUE)
        );

        jPanel3.setBorder(javax.swing.BorderFactory.↙
            ↪ createTitledBorder("Simulation␣Log"));

        jTextArea2.setColumns(20);
        jTextArea2.setEditable(false);
        jTextArea2.setRows(5);
        jTextArea2.setAutoscrolls(true);
        jScrollPane2.setViewportView(jTextArea2);

        org.jdesktop.layout.GroupLayout jPanel8Layout = ↙
            ↪ new org.jdesktop.layout.GroupLayout(jPanel8↙
            ↪ );
        jPanel8.setLayout(jPanel8Layout);
        jPanel8Layout.setHorizontalGroup(
            jPanel8Layout.createParallelGroup(org.↙
                ↪ jdesktop.layout.GroupLayout.LEADING)
                .add(jScrollPane2, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 541, Short.↙
                    ↪ MAX_VALUE)
            );
            jPanel8Layout.setVerticalGroup(
                jPanel8Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
                .add(jScrollPane2, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 356, Short.↙
                    ↪ MAX_VALUE)
            );

            jTabbedPane2.addTab(”Agent”, jPanel8);

            jTextArea3.setColumns(20);
            jTextArea3.setEditable(false);
            jTextArea3.setRows(5);
            jScrollPane3.setViewportView(jTextArea3);

            org.jdesktop.layout.GroupLayout jPanel10Layout =↙
                ↪  new org.jdesktop.layout.GroupLayout(↙
                ↪ jPanel10);
            jPanel10.setLayout(jPanel10Layout);
            jPanel10Layout.setHorizontalGroup(
                jPanel10Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
                .add(jScrollPane3, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 541, Short.↙
                    ↪ MAX_VALUE)
            );
            jPanel10Layout.setVerticalGroup(
                jPanel10Layout.createParallelGroup(org.↙
                    ↪ jdesktop.layout.GroupLayout.LEADING)
                .add(jScrollPane3, org.jdesktop.layout.↙
                    ↪ GroupLayout.DEFAULT_SIZE, 356, Short.↙
                    ↪ MAX_VALUE)
            );

            jTabbedPane2.addTab(”Market”, jPanel10);

            jTextArea4.setColumns(20);
```

```
        jTextArea4.setEditable(false);
        jTextArea4.setRows(5);
        jScrollPane4.setViewportView(jTextArea4);

        org.jdesktop.layout.GroupLayout jPanel11Layout = new org.jdesktop.layout.GroupLayout(jPanel11);
        jPanel11.setLayout(jPanel11Layout);
        jPanel11Layout.setHorizontalGroup(
            jPanel11Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jScrollPane4, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 541, Short.MAX_VALUE)
        );
        jPanel11Layout.setVerticalGroup(
            jPanel11Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jScrollPane4, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 356, Short.MAX_VALUE)
        );

        jTabbedPane2.addTab("Gov.", jPanel11);

        jTextArea5.setColumns(20);
        jTextArea5.setEditable(false);
        jTextArea5.setRows(5);
        jScrollPane5.setViewportView(jTextArea5);

        org.jdesktop.layout.GroupLayout jPanel9Layout = new org.jdesktop.layout.GroupLayout(jPanel9);
        jPanel9.setLayout(jPanel9Layout);
        jPanel9Layout.setHorizontalGroup(
            jPanel9Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jScrollPane5, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 541, Short.MAX_VALUE)
        );
        jPanel9Layout.setVerticalGroup(
            jPanel9Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jScrollPane5, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 356, Short.MAX_VALUE)
        );

        jTabbedPane2.addTab("System", jPanel9);

        jPanel13.setBorder(javax.swing.BorderFactory.createTitledBorder("Initial Properties"));

        jLabel29.setText("Produce Will");

        jLabel30.setText("Consume Will");

        jLabel33.setText("Replication Strategy");

        jLabel34.setText("Stored Product");

        jLabel35.setText("Stored Wealth");

        jLabel36.setText("Remaining time");

        jLabel28.setText("Inital Product Price");

        jLabel6.setText("%");

        jLabel7.setText("Agent Reaction");

        jLabel8.setText("Production unit cost");

        jLabel9.setText("%");

        org.jdesktop.layout.GroupLayout jPanel13Layout = new org.jdesktop.layout.GroupLayout(jPanel13);
        jPanel13.setLayout(jPanel13Layout);
        jPanel13Layout.setHorizontalGroup(
jPanel13Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
.add(jPanel13Layout.createSequentialGroup()
    .addContainerGap()
    .add(jPanel13Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(jLabel35)
        .add(jLabel36)
        .add(jLabel33)
        .add(jLabel34)
        .add(jPanel13Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jLabel29)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, jLabel30))
        .add(jLabel28)
        .add(jLabel7)
        .add(jLabel8))
    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 207, Short.MAX_VALUE)
    .add(jPanel13Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING, false)
        .add(org.jdesktop.layout.GroupLayout.TRAILING, jSpinner3)
        .add(org.jdesktop.layout.GroupLayout.TRAILING, jSpinner2)
        .add(org.jdesktop.layout.GroupLayout.TRAILING, jSpinner11)
        .add(org.jdesktop.layout.GroupLayout.TRAILING, jSpinner1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 129, Short.MAX_VALUE)
        .add(org.jdesktop.layout.GroupLayout.TRAILING, jSpinner6, org.
                       ↪ jdesktop.layout.GroupLayout.↙
                       ↪ DEFAULT_SIZE, 129, Short.↙
                       ↪ MAX_VALUE)
                    .add(org.jdesktop.layout.GroupLayout↙
                       ↪ .TRAILING, jSpinner8, org.↙
                       ↪ jdesktop.layout.GroupLayout.↙
                       ↪ DEFAULT_SIZE, 129, Short.↙
                       ↪ MAX_VALUE)
                    .add(org.jdesktop.layout.GroupLayout↙
                       ↪ .TRAILING, jSpinner9, org.↙
                       ↪ jdesktop.layout.GroupLayout.↙
                       ↪ DEFAULT_SIZE, 129, Short.↙
                       ↪ MAX_VALUE)
                    .add(org.jdesktop.layout.GroupLayout↙
                       ↪ .TRAILING, jSpinner7, org.↙
                       ↪ jdesktop.layout.GroupLayout.↙
                       ↪ DEFAULT_SIZE, 129, Short.↙
                       ↪ MAX_VALUE)
                    .add(org.jdesktop.layout.GroupLayout↙
                       ↪ .TRAILING, jSpinner10, org.↙
                       ↪ jdesktop.layout.GroupLayout.↙
                       ↪ DEFAULT_SIZE, 129, Short.↙
                       ↪ MAX_VALUE)))
                .addPreferredGap(org.jdesktop.layout.↙
                   ↪ LayoutStyle.RELATED)
                .add(jPanel13Layout.createParallelGroup(↙
                   ↪ org.jdesktop.layout.GroupLayout.↙
                   ↪ LEADING)
                    .add(jLabel6)
                    .add(jLabel9))
                .addContainerGap(29, Short.MAX_VALUE))
        );
        jPanel13Layout.setVerticalGroup(
            jPanel13Layout.createParallelGroup(org.↙
               ↪ jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel13Layout.createSequentialGroup()
                .add(17, 17, 17)
                .add(jPanel13Layout.createParallelGroup(↙
                   ↪ org.jdesktop.layout.GroupLayout.↙
                   ↪ BASELINE)
                    .add(jLabel29)
            .add(jSpinner1, org.jdesktop.layout.↙
                ↪ GroupLayout.PREFERRED_SIZE, org↙
                ↪ .jdesktop.layout.GroupLayout.↙
                ↪ DEFAULT_SIZE, org.jdesktop.↙
                ↪ layout.GroupLayout.↙
                ↪ PREFERRED_SIZE))
        .addPreferredGap(org.jdesktop.layout.↙
            ↪ LayoutStyle.RELATED)
        .add(jPanel13Layout.createParallelGroup(↙
            ↪ org.jdesktop.layout.GroupLayout.↙
            ↪ BASELINE)
            .add(jLabel30)
            .add(jSpinner6, org.jdesktop.layout.↙
                ↪ GroupLayout.PREFERRED_SIZE, org↙
                ↪ .jdesktop.layout.GroupLayout.↙
                ↪ DEFAULT_SIZE, org.jdesktop.↙
                ↪ layout.GroupLayout.↙
                ↪ PREFERRED_SIZE))
        .addPreferredGap(org.jdesktop.layout.↙
            ↪ LayoutStyle.RELATED)
        .add(jPanel13Layout.createParallelGroup(↙
            ↪ org.jdesktop.layout.GroupLayout.↙
            ↪ BASELINE)
            .add(jLabel33)
            .add(jSpinner8, org.jdesktop.layout.↙
                ↪ GroupLayout.PREFERRED_SIZE, org↙
                ↪ .jdesktop.layout.GroupLayout.↙
                ↪ DEFAULT_SIZE, org.jdesktop.↙
                ↪ layout.GroupLayout.↙
                ↪ PREFERRED_SIZE)
            .add(jLabel6))
        .addPreferredGap(org.jdesktop.layout.↙
            ↪ LayoutStyle.RELATED)
        .add(jPanel13Layout.createParallelGroup(↙
            ↪ org.jdesktop.layout.GroupLayout.↙
            ↪ BASELINE)
            .add(jLabel34)
            .add(jSpinner9, org.jdesktop.layout.↙
                ↪ GroupLayout.PREFERRED_SIZE, org↙
                ↪ .jdesktop.layout.GroupLayout.↙
                ↪ DEFAULT_SIZE, org.jdesktop.↙
```

```
                        ↪ layout.GroupLayout.↙
                        ↪ PREFERRED_SIZE))
                .addPreferredGap(org.jdesktop.layout.↙
                  ↪ LayoutStyle.RELATED)
                .add(jPanel13Layout.createParallelGroup(↙
                  ↪ org.jdesktop.layout.GroupLayout.↙
                  ↪ BASELINE)
                    .add(jLabel35)
                    .add(jSpinner7, org.jdesktop.layout.↙
                        ↪ GroupLayout.PREFERRED_SIZE, org↙
                        ↪ .jdesktop.layout.GroupLayout.↙
                        ↪ DEFAULT_SIZE, org.jdesktop.↙
                        ↪ layout.GroupLayout.↙
                        ↪ PREFERRED_SIZE))
                .addPreferredGap(org.jdesktop.layout.↙
                  ↪ LayoutStyle.RELATED)
                .add(jPanel13Layout.createParallelGroup(↙
                  ↪ org.jdesktop.layout.GroupLayout.↙
                  ↪ BASELINE)
                    .add(jLabel36)
                    .add(jSpinner10, org.jdesktop.layout↙
                        ↪ .GroupLayout.PREFERRED_SIZE, ↙
                        ↪ org.jdesktop.layout.GroupLayout↙
                        ↪ .DEFAULT_SIZE, org.jdesktop.↙
                        ↪ layout.GroupLayout.↙
                        ↪ PREFERRED_SIZE))
                .addPreferredGap(org.jdesktop.layout.↙
                  ↪ LayoutStyle.RELATED)
                .add(jPanel13Layout.createParallelGroup(↙
                  ↪ org.jdesktop.layout.GroupLayout.↙
                  ↪ BASELINE)
                    .add(jSpinner11, org.jdesktop.layout↙
                        ↪ .GroupLayout.PREFERRED_SIZE, ↙
                        ↪ org.jdesktop.layout.GroupLayout↙
                        ↪ .DEFAULT_SIZE, org.jdesktop.↙
                        ↪ layout.GroupLayout.↙
                        ↪ PREFERRED_SIZE)
                    .add(jLabel28))
                .addPreferredGap(org.jdesktop.layout.↙
                  ↪ LayoutStyle.RELATED)
                .add(jPanel13Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(jSpinner2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(jLabel7)
                    .add(jLabel9))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jPanel13Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(jSpinner3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(jLabel8))
                .addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        jTabbedPane2.addTab("Initial␣Prop.", jPanel13);

        org.jdesktop.layout.GroupLayout jPanel3Layout = new org.jdesktop.layout.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, jTabbedPane2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 553, Short.MAX_VALUE)
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, jTabbedPane2)
        );

        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .add(jPanel2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(jPanel3, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .add(jPanel1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jPanel3, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .add(jPanel2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jPanel1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void StartSimHandler(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_StartSimHandler
        jSpinner1.setEnabled(false);
        jSpinner6.setEnabled(false);
        jSpinner8.setEnabled(false);
        jSpinner9.setEnabled(false);
        jSpinner7.setEnabled(false);
        jSpinner10.setEnabled(false);
        jSpinner11.setEnabled(false);
        jSpinner2.setEnabled(false);
        jSpinner3.setEnabled(false);

        jSlider1.setEnabled(false);
        jCheckBox1.setEnabled(false);

        jButton1.setEnabled(false);
        jButton2.setEnabled(true);
        jButton3.setEnabled(true);
        god.setDelayFactor(jSlider1.getValue());
        god.setParameters(this, lauchSim.getAgentList(), lauchSim.getMarket(),
                lauchSim.getPool(), lauchSim.getLog(), lauchSim.getRecorder(), lauchSim.getGovernment()          );
        god.start();

    }//GEN-LAST:event_StartSimHandler

    private void PauseResumeHandler(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_PauseResumeHandler
        if(evt.getActionCommand().equals(”Pause”))
        {
            jButton2.setText(”Resume”);
            synchronized(this){
               god.setTimePaused(true);
            }
        }
        if(evt.getActionCommand().equals(”Resume”))
        {
            jButton2.setText(”Pause”);
            synchronized(this){
                god.setTimePaused(false);
                if(!god.isInterrupted())
                    god.interrupt();
            }

//              god.getSimulationClockTimer().enable();
        }

    }//GEN-LAST:event_PauseResumeHandler

    private void ReportHandler(java.awt.event.↙
       ↪ ActionEvent evt) {//GEN-FIRST:↙
       ↪ event_ReportHandler
        // TODO buld simulation report
        JFrame Report = new Report();
//          Thread chart_FREE_MEMORY = new Thread(new ↙
   ↪ ShowCharts(God.getRecorder(), Recorder.FREE_MEMORY,
//                  "Free Memory", "Free Memory", "Time", ↙
   ↪ "Memory"));
//          chart_FREE_MEMORY.start();
//          Thread chart_REAL_TIME_OVERRUNS = new Thread(↙
   ↪ new ShowCharts(God.getRecorder(), Recorder.↙
   ↪ REAL_TIME_OVERRUNS,
//                  "Real time overruns", "Real time ↙
   ↪ overruns", "Time", "Overruns"));
//          chart_REAL_TIME_OVERRUNS.start();

        //JOptionPane.showMessageDialog(null, "↙
           ↪ Simulation report.");
        //this.dispose();
    }//GEN-LAST:event_ReportHandler

    private void TerminateHandler(java.awt.event.↙
       ↪ ActionEvent evt) {//GEN-FIRST:↙
       ↪ event_TerminateHandler
        jButton2.setEnabled(false);
        jButton3.setEnabled(false);
        jButton4.setEnabled(true);
        synchronized(this){
               god.setTimePaused(true);
            }


    }//GEN-LAST:event_TerminateHandler

    private void jSlider1PropertyChange(java.beans.↙
       ↪ PropertyChangeEvent evt) {//GEN-FIRST:↙
       ↪ event_jSlider1PropertyChange

    }//GEN-LAST:event_jSlider1PropertyChange

    private void jSlider1StateChanged(javax.swing.event.ChangeEvent evt) {//GEN-FIRST:event_jSlider1StateChanged
                god.setDelayFactor(jSlider1.getValue());
                jLabel4.setText("" + jSlider1.getValue());
    }//GEN-LAST:event_jSlider1StateChanged

    private void FullSpeedHandler(java.awt.event.ItemEvent evt) {//GEN-FIRST:event_FullSpeedHandler
        if(jCheckBox1.isSelected()){
             jLabel3.setVisible(true);
             jLabel4.setVisible(true);
            jSlider1.setPaintLabels(false);
            jSlider1.setMaximum(100);
            jSlider1.setValue(jSlider1.getValue()*10);
        }
        else
        {
            jLabel3.setVisible(false);
            jLabel4.setVisible(false);
            jSlider1.setMaximum(10);
            jSlider1.setMajorTickSpacing(1);
            jSlider1.setPaintLabels(true);
            if(jSlider1.getValue()>10)jSlider1.setValue(10);
        }
    }//GEN-LAST:event_FullSpeedHandler

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jTextField1ActionPerformed

}//GEN-LAST:event_jTextField1ActionPerformed



```

```
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel28;
    private javax.swing.JLabel jLabel29;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel30;
    private javax.swing.JLabel jLabel33;
    private javax.swing.JLabel jLabel34;
    private javax.swing.JLabel jLabel35;
    private javax.swing.JLabel jLabel36;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel10;
    private javax.swing.JPanel jPanel11;
    private javax.swing.JPanel jPanel13;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JSlider jSlider1;
    private javax.swing.JSpinner jSpinner1;
    private javax.swing.JSpinner jSpinner10;
    private javax.swing.JSpinner jSpinner11;
    private javax.swing.JSpinner jSpinner2;
    private javax.swing.JSpinner jSpinner3;
    private javax.swing.JSpinner jSpinner6;
    private javax.swing.JSpinner jSpinner7;
    private javax.swing.JSpinner jSpinner8;
    private javax.swing.JSpinner jSpinner9;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTabbedPane jTabbedPane2;
    private javax.swing.JTextArea jTextArea2;
    private javax.swing.JTextArea jTextArea3;
    private javax.swing.JTextArea jTextArea4;
    private javax.swing.JTextArea jTextArea5;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField6;
    // End of variables declaration//GEN-END:variables

    public JSlider getjSlider1() {
        return jSlider1;
    }

    public JSpinner getjSpinner1() {
        return jSpinner1;
    }

    public JSpinner getjSpinner10() {
        return jSpinner10;
    }

    public JSpinner getjSpinner6() {
        return jSpinner6;
    }

    public JSpinner getjSpinner7() {
        return jSpinner7;
    }

    public JSpinner getjSpinner8() {
        return jSpinner8;
    }

    public JSpinner getjSpinner9() {
        return jSpinner9;
    }

    public JTextArea getjTextArea2() {
        return jTextArea2;
    }

    public JTextArea getjTextArea3() {
        return jTextArea3;
    }

    public JTextArea getjTextArea4() {
        return jTextArea4;
    }

    public JTextArea getjTextArea5() {
        return jTextArea5;
    }

    public JSpinner getjSpinner11() {
        return jSpinner11;
    }

    public JTextField getjTextField1() {
        return jTextField1;
    }

    public void setjTextField1(JTextField jTextField1) {
        this.jTextField1 = jTextField1;
    }

    public JTextField getjTextField2() {
        return jTextField2;
    }

    public void setjTextField2(JTextField jTextField2) {
        this.jTextField2 = jTextField2;
    }

    public JTextField getjTextField4() {
        return jTextField4;
    }

    public void setjTextField4(JTextField jTextField4) {
        this.jTextField4 = jTextField4;
    }

    public JTextField getjTextField6() {
        return jTextField6;
    }

    public void setjTextField6(JTextField jTextField6) {
        this.jTextField6 = jTextField6;
    }

//    public JTextField getjTextField7() {
//        return jTextField7;
//    }
//
//    public void setjTextField7(JTextField jTextField7)↙
    ↪ {
//        this.jTextField7 = jTextField7;
//    }
//
//    public JTextField getjTextField8() {
//        return jTextField8;
//    }
//
//    public void setjTextField8(JTextField jTextField8)↙
    ↪ {
//        this.jTextField8 = jTextField8;
//    }
//
//    public JTextField getjTextField9() {
//        return jTextField9;
//    }
//
//    public void setjTextField9(JTextField jTextField9)↙
    ↪ {
//        this.jTextField9 = jTextField9;
//      }

    public JButton getjButton3()
    {
        return jButton3;
    }

    public JSpinner getjSpinner2() {
        return jSpinner2;
    }

    public JSpinner getjSpinner3() {
        return jSpinner3;
    }

}
```

God.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import java.util.ArrayList;
import javax.realtime.AbsoluteTime;
import javax.realtime.AperiodicParameters;
import javax.realtime.AsyncEventHandler;
import javax.realtime.Clock;
import javax.realtime.PeriodicParameters;
import javax.realtime.PriorityParameters;
import javax.realtime.PriorityScheduler;
import javax.realtime.RealtimeThread;
import javax.realtime.RelativeTime;
import javax.realtime.ReleaseParameters;
import javax.swing.JOptionPane;




/**
 *
 * @author Lu s F. Reis Pereira
 */
public class God extends RealtimeThread  {

    private RelativeTime clock;
    private static RelativeTime simulationClockTimer;
    private int delayFactor = 1;
    private DashBoard dashBoard;
    private boolean TimePaused = false;
    AsyncEventHandler simulationClockTimerHandler;
    static private RelativeTime simulationTime = new RelativeTime(0,0);
    private static ArrayList<Agent> agentList;
    private Market market;
    private Pool pool;
    private Log log;
    private boolean END_SIM=false;
    private static Integer semaphore=0;
    private static Integer RealTimeOverruns =0;
    private static Recorder recorder;
    private static Government government;
    private static  long impulseLenght;
    private static AbsoluteTime InitialAbsoluteTime;
    private static RelativeTime TimePassed;




    public God ()
    {


        clock = Clock.getRealtimeClock().getResolution();
        System.out.println("Real-time clock resolution: " + clock.toString());

        setName("God_RTT");

         PriorityParameters GodPriorityParameters =
                new PriorityParameters(
                PriorityScheduler.instance().getMaxPriority());
       this.setSchedulingParameters(GodPriorityParameters);
    }
    public void setParameters(DashBoard dashboard, ArrayList<Agent> agentlist, Market market,
                Pool pool, Log log, Recorder recorder, Government government){
                this.market = market;
        this.pool = pool;
        this.log = log;
        this.agentList=agentlist;
        this.dashBoard=dashboard;
        this.delayFactor=dashboard.getjSlider1().↙
           ↪ getValue();
        this.recorder = recorder;
        this.government = government;
        this.market = market;

    }

    @Override
    public void run() {

         ReleaseParameters GodReleaseParameters =
                new PeriodicParameters( new RelativeTime↙
                   ↪ (1000, 0));
         this.setReleaseParameters(GodReleaseParameters)↙
           ↪ ;
        agentList.add(new Agent(
                Double.parseDouble(dashBoard.↙
                   ↪ getjSpinner1().getValue().toString↙
                   ↪ ()),
                Double.parseDouble(dashBoard.↙
                   ↪ getjSpinner6().getValue().toString↙
                   ↪ ()),
                Double.parseDouble(dashBoard.↙
                   ↪ getjSpinner8().getValue().toString↙
                   ↪ ()),
                0.0,
                Double.parseDouble(dashBoard.↙
                   ↪ getjSpinner9().getValue().toString↙
                   ↪ ()),
                Double.parseDouble(dashBoard.↙
                   ↪ getjSpinner7().getValue().toString↙
                   ↪ ()),
                0,
                0,
                market,
                pool, log, dashBoard, this, agentList, ↙
                   ↪ recorder));
        agentList.add(new Agent(
                Double.parseDouble(dashBoard.↙
                   ↪ getjSpinner1().getValue().toString↙
```

```
                      ↪ ()),
                  Double.parseDouble(dashBoard.↙
                      ↪ getjSpinner6().getValue().toString↙
                      ↪ ()),
                  Double.parseDouble(dashBoard.↙
                      ↪ getjSpinner8().getValue().toString↙
                      ↪ ()),
                  0.0,
                  Double.parseDouble(dashBoard.↙
                      ↪ getjSpinner9().getValue().toString↙
                      ↪ ()),
                  Double.parseDouble(dashBoard.↙
                      ↪ getjSpinner7().getValue().toString↙
                      ↪ ()),
                  1,
                  0,
                  market,
                  pool, log, dashBoard, this, agentList, ↙
                      ↪ recorder));
         for(int i = 0;i < agentList.size(); i++)
        {
             agentList.get(i).setItSelf(new Thread(↙
                 ↪ agentList.get(i), "Thread␣for␣Agent␣↙
                 ↪ number␣" + agentList.get(i).getAgentID↙
                 ↪ ()));
             agentList.get(i).getItSelf().start();

        }
        market.setName("Market␣Thread");
        market.start();
        government.setName("Government␣Thread");
        government.start();
        simulationClockTimer = new RelativeTime(Long.↙
            ↪ parseLong(""+dashBoard.getjSpinner10().↙
            ↪ getValue())*1000,0);
        InitialAbsoluteTime = Clock.getRealtimeClock().↙
            ↪ getTime();
        while(true)
        {

                  this.pool.ResetPool();
```

```
                if (semaphore > 0) {
                    this.setTimePaused(true);
                    JOptionPane.showMessageDialog(null, "Not enough CPU", "Simulation OVER", JOptionPane.ERROR_MESSAGE);
                }
            synchronized(this) {
                GodReleaseParameters =
                        new PeriodicParameters(
                        new RelativeTime(1000 / delayFactor, 0));

                if (!TimePaused) {

                    simulationTime=simulationTime.add(Math.round(1000/delayFactor), 0);
                    impulseLenght = Math.round(1000/delayFactor);
                    this.setReleaseParameters(GodReleaseParameters);
                    RealtimeThread.waitForNextPeriod();
                    TimePassed =
                            Clock.getRealtimeClock().getTime().subtract(InitialAbsoluteTime);
                    synchronized(Lock.lock){
                        if(Agent.isSIM_STOP()){
                            Agent.setSIM_STOP(false);
                            Government.setSIM_STOP(false);
                            Lock.lock.notifyAll();
                        }
                    }
                    synchronized(Lock.timelock){
                        Lock.timelock.notifyAll();
                    }
                    processAgentsLife();
                    processAgentsStatistics();
                    processMarketStatistics();
                    processGovernmentStatistics();
                    processSystemStatistics();
                    if(END_SIM)
                    {
                        dashBoard.getjButton3().doClick();
                        market.setEND_SIM(true);
                        government.setEND_SIM(true);
                    }
                } else {
                    this.setReleaseParameters(new AperiodicParameters());
                    this.setReleaseParameters(GodReleaseParameters);
                    Agent.setSIM_STOP(true);
                    Government.setSIM_STOP(true);
                    try {
                        while (TimePaused) {
                            this.wait();
                        }

                    } catch (InterruptedException ex) {
//                          Logger.getLogger(God.class.getName()).log(Level.SEVERE, null, ex);
                    }

                }

                dashBoard.getjTextArea5().append("God is Alive: " + God.getSimulationTime()
                        + " delayFactor: " + delayFactor + "\n");
                dashBoard.getjTextArea5().setCaretPosition(dashBoard.getjTextArea5().getDocument().getLength());
            }
            if(simulationTime.compareTo(simulationClockTimer) > 0 )
            {
                dashBoard.getjButton3().doClick();
            }
    }

}

    public int getDelayFactor() {
        return delayFactor;
    }

    public void setDelayFactor(int delayFactor) {
        this.delayFactor = delayFactor;

    }

    public static RelativeTime getSimulationTime() {
        return simulationTime;
    }


    public RelativeTime getClock() {
        return clock;
    }

    public boolean isTimePaused() {
        return TimePaused;
    }

    public void setTimePaused(boolean TimePaused) {
        this.TimePaused = TimePaused;
    }

    private void processAgentsLife() {
        synchronized (Lock.AgentListlock) {
            for (Agent agent : agentList) {
                if (agent != null && agent.↙
                   ↪ getStoredProduct() >0) {
                    synchronized(agent)
                    {
                        agent.setStoredProduct(agent.↙
                           ↪ getStoredProduct()
```

```
                            - Double.parseDouble(↙
                               ↪ dashBoard.getjSpinner6↙
                               ↪ ().getValue().toString↙
                               ↪ ()));
                    }
                }
            }
        }
    }

    private void processAgentsStatistics() {

        double totalProductInSociety = 0;
        double totalMoneyInSociety= 0;
        int totalLiveAgents=0;
        long sumOfAges = 0;
        synchronized (Lock.AgentListlock) {
            for (Agent agent : agentList) {
                if (agent != null && agent.↙
                   ↪ getStoredProduct() >0) {
                    totalLiveAgents++;
                    totalProductInSociety = ↙
                       ↪ totalProductInSociety + agent.↙
                       ↪ getStoredProduct();
                    totalMoneyInSociety = ↙
                       ↪ totalMoneyInSociety + agent.↙
                       ↪ getStoredWealth();
                    if(agent.getBournTime() != null)
                    sumOfAges = sumOfAges + ↙
                       ↪ simulationTime.subtract(agent.↙
                       ↪ getBournTime()).getMilliseconds↙
                       ↪ ();

                }

            }
            //System.out.println("\n");
        }
        if(totalLiveAgents <1)
            END_SIM = true;
        if (!END_SIM) {
```

```
        synchronized (Lock.Recorderlock) {


            recorder.addValue(Recorder.↙
              ↪ NUMBER_OF_AGENTS, new MetricPoint(↙
              ↪ simulationTime, totalLiveAgents));
            dashBoard.getjTextField1().setText("" + ↙
              ↪ totalLiveAgents);
            recorder.addValue(Recorder.↙
              ↪ PRODUCT_STORED, new MetricPoint(↙
              ↪ simulationTime, ↙
              ↪ totalProductInSociety));
            dashBoard.getjTextField2().setText("" + ↙
              ↪ totalProductInSociety);
            //recorder.addValue(Recorder.↙
              ↪ PRODUCTION_RATE, new MetricPoint(↙
              ↪ simulationTime, ↙
              ↪ totalProductInSociety/impulseLenght↙
              ↪ ));
            //dashBoard.getjTextField3().setText(""+↙
              ↪  totalProductInSociety/↙
              ↪ impulseLenght);
            recorder.addValue(Recorder.WEALTH, new ↙
              ↪ MetricPoint(simulationTime, ↙
              ↪ totalMoneyInSociety));
            dashBoard.getjTextField4().setText("" + ↙
              ↪ totalMoneyInSociety);
            //recorder.addValue(Recorder.↙
              ↪ WEALTH_GROWTH_RATE, new MetricPoint↙
              ↪ (simulationTime, ↙
              ↪ totalMoneyInSociety/impulseLenght))↙
              ↪ ;
            //dashBoard.getjTextField5().setText(""+↙
              ↪  totalMoneyInSociety/impulseLenght)↙
              ↪ ;
            recorder.addValue(Recorder.TOTAL_BIRTHS,↙
              ↪  new MetricPoint(simulationTime, ↙
              ↪ agentList.size()));
            dashBoard.getjTextField6().setText("" + ↙
              ↪ agentList.size());
```

```
                //recorder.addValue(Recorder.BIRTH_RATE, new MetricPoint(simulationTime, agentList.size()/impulseLenght));
                //          dashBoard.getjTextField7().setText(""+ agentList.size()/impulseLenght);
                recorder.addValue(Recorder.TOTAL_DEATHS, new MetricPoint(simulationTime, (agentList.size() - totalLiveAgents)));
                //          dashBoard.getjTextField8().setText(""+ (agentList.size()-totalLiveAgents));
                //          recorder.addValue(Recorder.DEATH_RATE, new MetricPoint(simulationTime, (agentList.size()-totalLiveAgents)/impulseLenght));
                //          daAshBoard.getjTextField8().setText(""+ (agentList.size()-totalLiveAgents)/impulseLenght);
                recorder.addValue(Recorder.AGE, new MetricPoint(simulationTime, sumOfAges/totalLiveAgents));
            }
        }

    }

    private synchronized void processMarketStatistics() {
        synchronized(Lock.Recorderlock)
        {


            //recorder.addValue(Recorder.POOL_SIZE, new MetricPoint(simulationTime, pool.getSize()));
            //dashBoard.getjTextField10().setText(""+ pool.getSize());
            //recorder.addValue(Recorder.TRANSACTION_RATE, new MetricPoint(
simulationTime, pool.getTotalTransactions()/impulseLenght));
            //dashBoard.getjTextField11().setText(""+ pool.getTotalTransactions()/impulseLenght);
            //recorder.addValue(Recorder.TOTAL_TRANSACTIONS, new MetricPoint(simulationTime, pool.getTotalTransactions()));
            //dashBoard.getjTextField12().setText(""+ pool.getTotalTransactions());
            //recorder.addValue(Recorder.MEDIAN_WAIT_TIME, new MetricPoint(simulationTime, market.getMedianWaitTime()));
            //dashBoard.getjTextField13().setText(""+ market.getMedianWaitTime());
        }
    }

    private void processGovernmentStatistics() {


    }

    private synchronized void processSystemStatistics() {
//            recorder.addValue(Recorder.TOTAL_THREADS, new MetricPoint(simulationTime, Thread.activeCount()));
//            dashBoard.getjTextField19().setText(""+ Thread.activeCount());
//            recorder.addValue(Recorder.FREE_MEMORY, new MetricPoint(simulationTime, Runtime.getRuntime().freeMemory()));
//            dashBoard.getjTextField20().setText(""+ Runtime.getRuntime().freeMemory());
//            recorder.addValue(Recorder.REAL_TIME_OVERRUNS, new MetricPoint(simulationTime, RealTimeOverruns));
//            dashBoard.getjTextField21().setText(""+ RealTimeOverruns);
```

```
    }
    public void incrementSemaphore(){
        synchronized(Lock.Semaphorelock)
        {
            semaphore++;
        }

    }
    public void decrementSemaphore(){
        synchronized(Lock.Semaphorelock)
        {
            semaphore--;
        }

    }

    public static Recorder getRecorder() {
        return recorder;
    }

    public static ArrayList<Agent> getAgentList() {
        return agentList;
    }
    public static RelativeTime getRelativeTime()
    {
        return simulationClockTimer;
    }
    public static RelativeTime getTimePassed()
    {
        return TimePassed;
    }
}
```

Government.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.realtime.PriorityParameters;
import javax.realtime.PriorityScheduler;
import javax.realtime.RealtimeThread;
import javax.realtime.RelativeTime;

/**
 *
 * @author Lu s F. Reis Pereira
 */
public class Government extends RealtimeThread {

    private static volatile boolean SIM_STOP = false;
    private static double StoredWealth = 0.0;
    private static double StotedProduct = 0.0;
    private static int GOVERNMENT_BUY = 2;
    private static int GOVERNMENT_WAIT = 1;
    private Log log;
    private God god;
    private Market market;
    private DashBoard dashboard;
    private Thread itSelf;
    private Pool pool;
    private RelativeTime lastTime  = new RelativeTime(0,0);
    private Boolean END_SIM = false;
    private lauchSim LauchSim;

    public Government(Log log, God god, Market market, DashBoard dashboard, Pool pool, lauchSim LauchSim) {
      this.log=log;
```

```
        this.god =god;
        this.market = market;
        this.dashboard = dashboard;
        this.pool = pool;
        this.setSchedulingParameters( new PriorityParameters(PriorityScheduler.instance().getNormPriority()));
        this.LauchSim = LauchSim;
    }

    @Override
    public void run() {

        itSelf = this;

        double lastTransactionPrice = 0.0;
        double targetPrice = Double.parseDouble(dashboard.getjSpinner11().getValue().toString());
        while(!END_SIM)
        {

            synchronized(Lock.timelock){
                        try {
                            while(lastTime.equals(God.getSimulationTime()))
                            {Lock.timelock.wait();}
                        } catch (InterruptedException ex) {
                            Logger.getLogger(Agent.class.getName()).log(Level.SEVERE, null, ex);
                        }

                }
            lastTime.set(God.getSimulationTime());

            if(SIM_STOP){

                synchronized(dashboard.getjTextArea2()){
```

```
            dashboard.getjTextArea2().append(this.itSelf.getName() + "****PAUSED****\n");
            dashboard.getjTextArea2().setCaretPosition(dashboard.getjTextArea2().getDocument().getLength());
            }
            synchronized(Lock.lock){
                    try {
                        while(SIM_STOP){Lock.lock.wait();}
                    } catch (InterruptedException ex) {
                        Logger.getLogger(Agent.class.getName()).log(Level.SEVERE, null, ex);
                    }

            }
            synchronized(dashboard.getjTextArea2()){
            dashboard.getjTextArea2().append(this.itSelf.getName() + "****UNPAUSED****\n");
            dashboard.getjTextArea2().setCaretPosition(dashboard.getjTextArea2().getDocument().getLength());
            }
        }
        int decision = 0;
        double decisionIntensity = 0.0;
        //TODO Government Call decider
        if(market == null)
        {
            market = LauchSim.getMarket();
        }
        if(pool == null)
        {
            pool = LauchSim.getPool();
        }
        if (market != null) {
            lastTransactionPrice = market.↙
                ↪ getCurrentPrice();
            if (lastTransactionPrice <= targetPrice)↙
                ↪ {
                decision = Government.GOVERNMENT_BUY↙
                    ↪ ;
                synchronized (dashboard.↙
                    ↪ getjTextArea4()) {
                    dashboard.getjTextArea4().append↙
                        ↪ (this.itSelf.getName() + ”↙
                        ↪ ****BUY****@” + ↙
                        ↪ lastTransactionPrice + ”\n”↙
                        ↪ );
                    dashboard.getjTextArea4().↙
                        ↪ setCaretPosition(dashboard.↙
                        ↪ getjTextArea4().getDocument↙
                        ↪ ().getLength());
                }
                Transaction transaction = new ↙
                    ↪ Transaction();
                transaction.setAgentIndex(Integer.↙
                    ↪ MAX_VALUE);
                transaction.setAmount(1L);
                transaction.setPrice(targetPrice);
                transaction.setTYPE(Transaction.BUY)↙
                    ↪ ;
                synchronized (this) {
                    transaction.setID(”” + Agent.↙
                        ↪ getTransactionID());
                    Agent.setTransactionID(Agent.↙
                        ↪ getTransactionID());
                }
                transaction.setCreationTime(God.↙
                    ↪ getSimulationTime());
                synchronized(pool){pool.setElement(↙
                    ↪ transaction);}
            } else {
                decision = Government.↙
                    ↪ GOVERNMENT_WAIT;
                    synchronized (dashboard.getjTextArea4()) {
                        dashboard.getjTextArea4().append(this.itSelf.getName() + "****WAIT****\n");
                        dashboard.getjTextArea4().setCaretPosition(dashboard.getjTextArea4().getDocument().getLength());
                    }
                }
            }
        }
    }
    public Thread getItSelf() {
        return itSelf;
    }

    public void setEND_SIM(Boolean END_SIM) {
        this.END_SIM = END_SIM;
    }

    public static void setSIM_STOP(boolean SIM_STOP) {
        Government.SIM_STOP = SIM_STOP;
    }
    public double getStoredWealth()
    {
        synchronized(Lock.transactionlock)
        {
            return StoredWealth;
        }
    }
    public double getStoredProduct()
    {
        synchronized(Lock.transactionlock)
        {
            return StotedProduct;
        }
    }
    public void setStoredProduct(double product)
    {
        synchronized(Lock.transactionlock)
        {
            StotedProduct = product;
        }
    }
    public void setStoredWealth(double wealth)
    {
        synchronized(Lock.transactionlock)
        {
            StoredWealth = wealth;
        }

    }
    public int getAgentIndex()
    {
        return Integer.MAX_VALUE;
    }

}
```

lauchSim.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import java.util.ArrayList;
import javax.realtime.PriorityParameters;
import javax.realtime.PriorityScheduler;
import javax.realtime.RealtimeThread;

/**
 *
 * @author Lu s Filipe dos Reis Pereira
 * @email LuisFRPereira@gmail.com
 *
 */
public class lauchSim {

    God god;
    DashBoard dashBoard;
    Log log;
    Government government;
    Pool pool;
    Market market;
    ArrayList<Agent> agentList;
    Recorder recorder;

     public static void main(String args[]) {
        lauchSim lauchSim = new lauchSim();


    }
    public lauchSim()
    {

        agentList = new ArrayList<Agent>();
        god = new God();
        recorder = new Recorder();
        dashBoard = new DashBoard(god, this);
        log = new Log();
        government = new Government(log, god, market, dashBoard, pool, this);
        pool = new Pool(dashBoard, government);
        market = new Market(government, agentList, pool, log, dashBoard, god);
        pool.setMarket(market);
        god.setParameters(dashBoard, agentList, market, pool, log, recorder, government);

        lauchThreads();
        // Create Dashboard
          java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                dashBoard.setVisible(true);
            }
        });
        ThreadGroup tg = Thread.currentThread().getThreadGroup();
        tg.list();

    }
    private void lauchThreads()
    {
        //God is started manualy

        RealtimeThread  governmentThread = new Government(log, god, market, dashBoard, pool, this );
        governmentThread.setName("Government␣Thread");
        RealtimeThread  marketThread = new Market(government, agentList, pool, log, dashBoard, god);
        marketThread.setName("Market␣Thread");

    }

    public God getGod() {
        return god;
    }

    public ArrayList<Agent> getAgentList() {
        return agentList;
    }

    public Log getLog() {
        return log;
    }

    public Market getMarket() {
        return market;
    }

    public Pool getPool() {
        return pool;
    }
    public lauchSim getlauchSim(){
        return this;
    }

    public Recorder getRecorder() {
        return recorder;
    }

    public Government getGovernment() {
        return government;
    }

}
```

Lock.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

/**
 *
 * @author Lu s Filipe dos Reis Pereira
 * @email LuisFRPereira@gmail.com
 *
 */

public class Lock {

    public static final Object lock = new Object();
    public static final Object timelock = new Object();
    public static final Object transactionlock = new Object();
    public static final Object AgentListlock = new Object();
    public static final Object AgentIDlock = new Object();
    public static final Object Semaphorelock = new Object();
    public static final Object Recorderlock = new Object();

}
```

Log.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.realtime.PriorityParameters;
import javax.realtime.PriorityScheduler;
import javax.realtime.RealtimeThread;
import javax.realtime.SchedulingParameters;
/**
 *
 * @author Lu s F. Reis Pereira
 */
public class Log extends RealtimeThread {
    private String databaseURL = "jdbc:derby:memory:↙
        ↪ marketSim";
    private Properties databaseProperties;
    static private volatile Connection connection;

    public Log() {

        setName("Log RTT");

        PriorityParameters GodPriorityParameters =
                new PriorityParameters(
                PriorityScheduler.instance().↙
                    ↪ getNormPriority());
```

```




            this.setSchedulingParameters(↙
                ↪ GodPriorityParameters);
        }



        @Override
        public void run ()
        {

            databaseProperties = new Properties();
            databaseProperties.setProperty(”createFrom”, ”/↙
                ↪ home/lfp/BackDerbyEmpty/marketSim”);
            databaseProperties.setProperty(”user”, ”a”);
            databaseProperties.setProperty(”password”, ”a”);

            try {
                Class.forName(”org.apache.derby.jdbc.↙
                    ↪ EmbeddedDriver”);
            } catch (ClassNotFoundException ex) {
                Logger.getLogger(God.class.getName()).log(↙
                    ↪ Level.SEVERE, null, ex);
            }
            try {
                connection = DriverManager.getConnection(↙
                    ↪ databaseURL, databaseProperties);

            } catch (SQLException ex) {
                Logger.getLogger(Log.class.getName()).log(↙
                    ↪ Level.SEVERE, null, ex);
            }

        }

        public Connection getConnection() {
            return connection;
        }
```

```



}
```

Market.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import java.util.ArrayList;
import java.util.NoSuchElementException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.realtime.PriorityParameters;
import javax.realtime.PriorityScheduler;
import javax.realtime.RealtimeThread;
import javax.realtime.RelativeTime;
import org.apache.commons.math.stat.descriptive.DescriptiveStatistics;
import java.util.Iterator;
import javax.realtime.Clock;



/**
 *
 * @author Lu s F. Reis Pereira
 */
public class Market extends RealtimeThread{
    private double productPrice = 0;
    public static final int GOVERNMENT_ACCEPTED = 0;
    public static final int GOVERNEMNT_DENIED = 1;
    public static final int TRANSCATION_FULL_SUCCESS = 2;
    public static final int TRANSACTION_PARTIAL_SUCCESS = 3;
    public static final int TRANSACTION_FAILD = 4;
    private ArrayList<Agent> agentList;
    private Pool pool;
    private Log log;
    private boolean paused;
    private String Buy_ID;
    private String Sell_ID;
    private double currentPrice;
    private double currentAmount;
    private DescriptiveStatistics timeTracker;
    private God god;
    private RelativeTime lastTime  = new RelativeTime(0,0);
    private Boolean END_SIM = false;
    private static Boolean newElement = false;
    private Iterator<Transaction> iPool = null;
    private double lastTransactionPrice;

    public Market( Government government,
            ArrayList<Agent> agentList, Pool pool,Log log, DashBoard dashboard, God god)
    {
        this.agentList=agentList;
        this.pool=pool;
        this.log=log;
        this.currentPrice=Double.parseDouble(dashboard.getjSpinner11().getValue().toString());
        timeTracker = new DescriptiveStatistics();
        this.god = god;
        this.setSchedulingParameters(new PriorityParameters(PriorityScheduler.instance().getNormPriority()));
    }
    synchronized int doTransaction(Transaction T1, Transaction T2)
    {
        if(T1.getCreationTime().compareTo(T1.getCreationTime()) > 0)
            timeTracker.addValue(T1.getCreationTime().subtract(T2.getCreationTime()).getMilliseconds());
        else
            timeTracker.addValue(T2.getCreationTime().subtract(T1.getCreationTime()).getMilliseconds());

        return 0;
    }
    double getMedianWaitTime()
    {
        return timeTracker.getPercentile(0.5);
    }

    @Override
    public void run() {

        while(!END_SIM)
        {

           synchronized(Lock.timelock){
                        try {
                            while(lastTime.equals(God.getSimulationTime()))
                            {Lock.timelock.wait();}
                        } catch (InterruptedException ex) {
                            Logger.getLogger(Agent.class.getName()).log(Level.SEVERE, null, ex);
                        }

                }
            lastTime.set(God.getSimulationTime());



//              logDecision(Buy_ID, Sell_ID, currentPrice, currentAmount);
//              updateBuyerValues();
//              updateSellerValues();
            // TODO log transaction


        }
    }
    public boolean isPaused() {
        return paused;
    }
```

```java
    public static void fireNewElement()
    {
        newElement = true;
    }
    public void setPaused(boolean paused) {
        this.paused = paused;
    }

    public double getCurrentPrice() {
        return currentPrice;
    }

    public void setEND_SIM(Boolean END_SIM) {
        this.END_SIM = END_SIM;
    }

    public synchronized void setCurrentPrice(double currentPrice) {

        this.lastTransactionPrice = this.currentPrice;
        this.currentPrice = currentPrice;
    }

    public double getLastTransactionPrice() {
        return lastTransactionPrice;
    }
}
```

MetricPoint.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import javax.realtime.RelativeTime;

/**
 *
 * @author Lu s Filipe dos Reis Pereira
 * @email LuisFRPereira@gmail.com
 *
 */
public class MetricPoint {
    private RelativeTime time;
    private double metric;

    public MetricPoint(RelativeTime time, double metric) {
        this.time = time;
        this.metric = metric;
    }


    public double getMetric() {
        return metric;
    }

    public void setMetric(long metric) {
        this.metric = metric;
    }

    public RelativeTime getTime() {
        return time;
    }

    public void setTime(RelativeTime time) {
        this.time = time;
    }

}
```

Pool.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import java.util.HashSet;
import java.util.Iterator;

/**
 *
 * @author Lu s Filipe dos Reis Pereira
 * @email LuisFRPereira@gmail.com
 *
 */
public class Pool {

    private HashSet<Transaction> pool_Buy;
    private HashSet<Transaction> pool_Sell;
    private long totalTransactions=0;
    private Boolean newElement = false;
    private Transaction elementToTransact = null;
    private DashBoard dashboard;
    private volatile Double price =0.0;
    private boolean doTransaction;
    private volatile Transaction oldElement;
    private HashSet<Transaction> removePool;
    private Government government ;
    private Market market;



    public Pool (DashBoard dashboard, Government government)
    {
         pool_Buy = new HashSet<Transaction>();
         pool_Sell = new HashSet<Transaction>();
         this.dashboard = dashboard;
         this. government = government;
```

```
   }

    public synchronized Transaction next (Iterator<Transaction> iterator)
    {
         return iterator.next();
    }
    public synchronized boolean hasNext(Iterator<Transaction> iterator)
    {
         return iterator.hasNext();
    }
    void setElement( Transaction newElement)
    {


         if(newElement.getTYPE() == Transaction.BUY)
         {
              synchronized(pool_Sell)
              {
                   if(pool_Buy.isEmpty())
                   {
                         synchronized(pool_Buy){pool_Buy.add(newElement);}
                   }
                   for( Transaction T : pool_Sell)
                   {
                        if (Math.abs(T.price) < Math.abs(newElement.price))
                        {
                             doTransaction = true;
                             oldElement = T;
                             removePool = pool_Sell;
                             break;
                        }

                   }
                   if(!doTransaction)
                   {
                        synchronized(pool_Buy){pool_Buy.add(newElement);}
```

```
                    Market.fireNewElement();
                }
            }

        }
        if(doTransaction)
        {
            transactionDone(oldElement, newElement, removePool);
            doTransaction =false;
        }
        if(newElement.getTYPE() == Transaction.SELL)
        {
            synchronized(pool_Buy)
            {
                if(pool_Sell.isEmpty())
                {
                    synchronized(pool_Sell){pool_Sell.add(newElement);}
                }
                for( Transaction T : pool_Buy)
                {
                    if (Math.abs(T.price) >= Math.abs(newElement.price))
                    {
                        doTransaction = true;
                        oldElement = T;
                        removePool = pool_Buy;
                        break;
                    }

                }
                if(!doTransaction)
                {
                    synchronized(pool_Sell){pool_Sell.add(newElement);}
                    Market.fireNewElement();
                }
            }
            if(doTransaction)
            {
             if(oldElement != null)
             {
                 transactionDone(oldElement, ↙
                    ↪ newElement, removePool);
             }
             else
             {
                 synchronized(this){removePool.remove↙
                    ↪ (oldElement);}
             }

             doTransaction =false;
         }
     }
 }
int getSize()
 {
    return pool_Buy.size()+pool_Sell.size();
 }
boolean transactionDone(Transaction T1, Transaction ↙
   ↪ T2, HashSet<Transaction> pool)
{

    try {
        synchronized (Lock.AgentListlock) {
            if ((Math.abs(T1.price) < Math.abs(T2.↙
               ↪ price))) {
                price = Math.abs(T1.price);
                if(God.getRelativeTime() == null)
                 {
                     //EMPTY
                 }
                 else
                {    synchronized(Lock.↙
                   ↪ transactionlock){
                         synchronized(Lock.↙
                            ↪ Recorderlock)
                         {
                             God.getRecorder().↙
                                ↪ addValue(Recorder.↙
                                ↪ PRICE, new ↙
```

```
                                          ↪ MetricPoint(
                                           God.getTimePassed(),↙
                                              ↪  price));
                                    }

                                }

                        }
                        synchronized (Lock.transactionlock) {
                            if (T1.getAgentIndex() == ↙
                                ↪ 2147483647) {
                                  government.setStoredWealth(↙
                                      ↪ government.↙
                                      ↪ getStoredWealth() + ↙
                                      ↪ price);
                                  God.getAgentList().get((int) ↙
                                      ↪ T2.getAgentIndex()).↙
                                      ↪ setStoredWealth(
                                          God.getAgentList().↙
                                              ↪ get((int) T2.↙
                                              ↪ getAgentIndex())↙
                                              ↪ .getStoredWealth↙
                                              ↪ () - price);

                                  government.setStoredProduct(↙
                                      ↪ government.↙
                                      ↪ getStoredProduct() - 1);
                                  God.getAgentList().get((int) ↙
                                      ↪ T2.getAgentIndex()).↙
                                      ↪ setStoredProduct(
                                          God.getAgentList().↙
                                              ↪ get((int) T2.↙
                                              ↪ getAgentIndex())↙
                                              ↪ .↙
                                              ↪ getStoredProduct↙
                                              ↪ () + 1);
                            } else if (T2.getAgentIndex() == ↙
                                ↪ 2147483647) {
                                  God.getAgentList().get((int) ↙
                                      ↪ T1.getAgentIndex()).↙
                                      ↪ setStoredWealth(
                        God.getAgentList().get((int) T1.getAgentIndex()).getStoredWealth() + price);
                government.setStoredWealth(government.getStoredWealth() - price);

                God.getAgentList().get((int) T1.getAgentIndex()).setStoredProduct(
                        God.getAgentList().get((int) T1.getAgentIndex()).getStoredProduct() - 1);
                government.setStoredProduct(government.getStoredProduct() + 1);
            } else {
                God.getAgentList().get((int) T1.getAgentIndex()).setStoredWealth(
                        God.getAgentList().get((int) T1.getAgentIndex()).getStoredWealth() + price);
                God.getAgentList().get((int) T2.getAgentIndex()).setStoredWealth(
                        God.getAgentList().get((int) T2.getAgentIndex()).getStoredWealth() - price);

```

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

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

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

```
                getLength());
            }
        }
    }


} else {
    price = Math.abs(T2.price);
    if(God.getRelativeTime() == null)
     {
         //EMPTY
     }
     else
    {
        synchronized(Lock.transactionlock){
            synchronized(Lock.Recorderlock)
            {
                God.getRecorder().addValue(Recorder.PRICE, new MetricPoint(
                 God.getTimePassed(), price));
            }

        }
    }
    synchronized (Lock.transactionlock) {
        if (T1.getAgentIndex() == 2147483647) {
            God.getAgentList().get((int) T2.getAgentIndex()).setStoredWealth(
                    God.getAgentList().get((int) T2.getAgentIndex()).getStoredWealth() + price);
```

```
                                government.setStoredWealth(↙
                                    ↪ government.↙
                                    ↪ getStoredWealth() - ↙
                                    ↪ price);

                                government.setStoredProduct(↙
                                    ↪ government.↙
                                    ↪ getStoredProduct() + 1);
                                God.getAgentList().get((int) ↙
                                    ↪ T2.getAgentIndex()).↙
                                    ↪ setStoredProduct(
                                        God.getAgentList().↙
                                            ↪ get((int) T2.↙
                                            ↪ getAgentIndex())↙
                                            ↪ .↙
                                            ↪ getStoredProduct↙
                                            ↪ () - 1);
                            } else if (T2.getAgentIndex() == ↙
                                ↪ 2147483647) {
                                government.setStoredWealth(↙
                                    ↪ government.↙
                                    ↪ getStoredWealth() + ↙
                                    ↪ price);
                                God.getAgentList().get((int) ↙
                                    ↪ T1.getAgentIndex()).↙
                                    ↪ setStoredWealth(
                                        God.getAgentList().↙
                                            ↪ get((int) T1.↙
                                            ↪ getAgentIndex())↙
                                            ↪ .getStoredWealth↙
                                            ↪ () - price);

                                God.getAgentList().get((int) ↙
                                    ↪ T1.getAgentIndex()).↙
                                    ↪ setStoredProduct(
                                        God.getAgentList().↙
                                            ↪ get((int) T1.↙
                                            ↪ getAgentIndex())↙
                                            ↪ .↙
                                            ↪ getStoredProduct↙
                                            ↪ () + 1);
```

```
    government.setStoredProduct(↙
       ↪ government.↙
       ↪ getStoredProduct() - 1);
} else {
    God.getAgentList().get((int) ↙
       ↪ T2.getAgentIndex()).↙
       ↪ setStoredWealth(
            God.getAgentList().↙
               ↪ get((int) T2.↙
               ↪ getAgentIndex())↙
               ↪ .getStoredWealth↙
               ↪ () + price);
    God.getAgentList().get((int) ↙
       ↪ T1.getAgentIndex()).↙
       ↪ setStoredWealth(
            God.getAgentList().↙
               ↪ get((int) T1.↙
               ↪ getAgentIndex())↙
               ↪ .getStoredWealth↙
               ↪ () - price);

    God.getAgentList().get((int) ↙
       ↪ T1.getAgentIndex()).↙
       ↪ setStoredProduct(
            God.getAgentList().↙
               ↪ get((int) T1.↙
               ↪ getAgentIndex())↙
               ↪ .↙
               ↪ getStoredProduct↙
               ↪ () + 1);
    God.getAgentList().get((int) ↙
       ↪ T2.getAgentIndex()).↙
       ↪ setStoredProduct(
            God.getAgentList().↙
               ↪ get((int) T2.↙
               ↪ getAgentIndex())↙
               ↪ .↙
               ↪ getStoredProduct↙
               ↪ () - 1);
}
```

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

                    }
                }

            }
        } catch (NullPointerException e) {
            synchronized (dashboard.getjTextArea3()) {

                                dashboard.getjTextArea3().append("Tried transaction with dead Agent" + T1.getAgentIndex()  + "\n");
                                dashboard.getjTextArea3().setCaretPosition(dashboard.
                                  ↪ getjTextArea3().↙
                                  ↪ getDocument().↙
                                  ↪ getLength());
                               totalTransactions--;
           }
       }
       synchronized(pool){pool.remove(T1);}
       totalTransactions++;



       return true;
   }
   long getTotalTransactions()
    {
       return totalTransactions;
   }

    public void setMarket(Market market) {
        this.market = market;
        market.setCurrentPrice(price);
    }
    public void ResetPool(){
       this.pool_Buy = new HashSet<Transaction>();
       this.pool_Sell = new HashSet<Transaction>();
    }

}
```

Recorder.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;


import java.util.ArrayList;
import java.util.HashMap;

/**
 *
 * @author Lu s Filipe dos Reis Pereira
 * @email LuisFRPereira@gmail.com
 *
 */
public class Recorder {
    public static final Integer NUMBER_OF_AGENTS = 0;
    public static final Integer PRODUCT_STORED=1;
    public static final Integer PRODUCTION_RATE=2;
    public static final Integer WEALTH = 3;
    public static final Integer WEALTH_GROWTH_RATE = 4;
    public static final Integer TOTAL_BIRTHS = 5;
    public static final Integer BIRTH_RATE = 6;
    public static final Integer TOTAL_DEATHS = 7;
    public static final Integer DEATH_RATE = 8;
    public static final Integer POOL_SIZE = 9;
    public static final Integer TRANSACTION_RATE = 10;
    public static final Integer TOTAL_TRANSACTIONS = 11;
    public static final Integer MEDIAN_WAIT_TIME = 12;
    public static final Integer ACTIVE_GOVERNMENT = 13;
    public static final Integer DENYED_TRANSACTIONS = 14;
    public static final Integer SAUDADE = 15;
    public static final Integer GENERATION = 16;
    public static final Integer ACTIVE_TIME = 17;
    public static final Integer TOTAL_THREADS = 18;
    public static final Integer FREE_MEMORY = 19;
    public static final Integer REAL_TIME_OVERRUNS = 20;
    public static final Integer PRICE = 21;
    public static final Integer AGE = 22;

    private HashMap<Integer, ArrayList<MetricPoint> > ↙
       ↪ allMetrics;
    private ArrayList<MetricPoint> series;

    public Recorder(){
        allMetrics = new HashMap<Integer, ArrayList<↙
           ↪ MetricPoint> >();
        series = new ArrayList<MetricPoint>();
}
    public void addValue(Integer type, MetricPoint point)
    {
        series = allMetrics.get(type);
        if (series!=null) {
            series.add(point);
        }
        else{
            allMetrics.put(type, new ArrayList<↙
               ↪ MetricPoint>() );
            series = allMetrics.get(type);
            series.add(point);
        }
        allMetrics.put(type, series);
    }
    public ArrayList<MetricPoint> getList(Integer type)
    {
              return allMetrics.get(type);
    }

}
```

Report.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * Report.java
 *
 * Created on 20/Mar/2011, 11:31:15
 */

package marketsim;

import java.awt.event.ItemEvent;

/**
 *
 * @author lfp
 */
public class Report extends javax.swing.JFrame {
    ShowCharts NUMBER_OF_AGENTS;
    ShowCharts PRODUCT_STORED;
    ShowCharts PRODUCTION_RATE;
    ShowCharts WEALTH;
    ShowCharts WEALTH_GROWTH_RATE;
    ShowCharts TOTAL_BIRTHS;
    ShowCharts BIRTH_RATE;
    ShowCharts TOTAL_DEATHS;
    ShowCharts DEATH_RATE;
    ShowCharts POOL_SIZE;
    ShowCharts TRANSACTION_RATE;
    ShowCharts TOTAL_TRANSACTIONS;
    ShowCharts MEDIAN_WAIT_TIME;
    ShowCharts DENYED_TRANSACTIONS;
    ShowCharts FREE_MEMORY;
    ShowCharts REAL_TIME_OVERRUNS;
    ShowCharts PRICE;
    ShowCharts AGE;

    /** Creates new form Report */
    public Report() {
        initComponents();
        pack();
        setVisible(true);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jCheckBox1 = new javax.swing.JCheckBox();
        jCheckBox2 = new javax.swing.JCheckBox();
        jCheckBox4 = new javax.swing.JCheckBox();
        jCheckBox6 = new javax.swing.JCheckBox();
        jCheckBox8 = new javax.swing.JCheckBox();
        jCheckBox13 = new javax.swing.JCheckBox();
        jCheckBox3 = new javax.swing.JCheckBox();
        jCheckBox5 = new javax.swing.JCheckBox();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jCheckBox1.setText("Number␣of␣Agents");
        jCheckBox1.addItemListener(new java.awt.event.ItemListener() {
          public void itemStateChanged(java.awt.event.ItemEvent evt) {
            Number_of_Agents_handler(evt);
          }
        });

        jCheckBox2.setText("Product␣stored");
        jCheckBox2.addItemListener(new java.awt.event.ItemListener() {
            public void itemStateChanged(java.awt.event.ItemEvent evt) {
                Product_stored_handler(evt);
            }
        });

        jCheckBox4.setText("Wealth");
        jCheckBox4.addItemListener(new java.awt.event.ItemListener() {
            public void itemStateChanged(java.awt.event.ItemEvent evt) {
                Wealth_handler(evt);
            }
        });

        jCheckBox6.setText("Total births");
        jCheckBox6.addItemListener(new java.awt.event.ItemListener() {
            public void itemStateChanged(java.awt.event.ItemEvent evt) {
                Total_births_handler(evt);
            }
        });

        jCheckBox8.setText("Total bankruptcies");
        jCheckBox8.addItemListener(new java.awt.event.ItemListener() {
            public void itemStateChanged(java.awt.event.ItemEvent evt) {
                Total_deaths(evt);
            }
        });

        jCheckBox13.setText("Median wait time");
        jCheckBox13.addItemListener(new java.awt.event.ItemListener() {
            public void itemStateChanged(java.awt.event.ItemEvent evt) {
                Median_wait_time_handler(evt);
            }
        });

        jCheckBox3.setText(”Price”);
        jCheckBox3.addItemListener(new java.awt.event.↙
           ↪ ItemListener() {
            public void itemStateChanged(java.awt.event.↙
               ↪ ItemEvent evt) {
                jCheckBox3ItemStateChanged(evt);
            }
        });

        jCheckBox5.setText(”Age”);
        jCheckBox5.addItemListener(new java.awt.event.↙
           ↪ ItemListener() {
            public void itemStateChanged(java.awt.event.↙
               ↪ ItemEvent evt) {
                AgeEventHandler(evt);
            }
        });
        jCheckBox5.addActionListener(new java.awt.event.↙
           ↪ ActionListener() {
            public void actionPerformed(java.awt.event.↙
               ↪ ActionEvent evt) {
                jCheckBox5ActionPerformed(evt);
            }
        });

        org.jdesktop.layout.GroupLayout layout = new org↙
           ↪ .jdesktop.layout.GroupLayout(getContentPane↙
           ↪ ());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.↙
               ↪ layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.↙
               ↪ TRAILING, layout.createSequentialGroup↙
               ↪ ()
                .addContainerGap(org.jdesktop.layout.↙
                   ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                   ↪ MAX_VALUE)
                .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jCheckBox3)
                    .add(jCheckBox13)
                    .add(jCheckBox6)
                    .add(jCheckBox4)
                    .add(jCheckBox2)
                    .add(jCheckBox1))
                .add(67, 67, 67))
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(jCheckBox5)
                .addContainerGap(159, Short.MAX_VALUE))
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(jCheckBox8)
                .addContainerGap(67, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(jCheckBox1)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jCheckBox2)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jCheckBox4)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jCheckBox6)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jCheckBox8)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                .add(jCheckBox5)
                .addPreferredGap(org.jdesktop.layout.↙
                   ↪ LayoutStyle.RELATED)
                .add(jCheckBox13)
                .addPreferredGap(org.jdesktop.layout.↙
                   ↪ LayoutStyle.RELATED)
                .add(jCheckBox3)
                .addContainerGap(org.jdesktop.layout.↙
                   ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                   ↪ MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void Number_of_Agents_handler(java.awt.event↙
       ↪ .ItemEvent evt) {//GEN-FIRST:↙
       ↪ event_Number_of_Agents_handler

        if(ItemEvent.SELECTED == evt.getStateChange())
        {
            NUMBER_OF_AGENTS = new ShowCharts(God.↙
               ↪ getRecorder(), Recorder.↙
               ↪ NUMBER_OF_AGENTS,
                "Number␣of␣Agents", "Number␣of␣Agents", ↙
                   ↪ "Time", "Agents");
            Thread chart_NUMBER_OF_AGENTS = new Thread(↙
               ↪ NUMBER_OF_AGENTS);
            chart_NUMBER_OF_AGENTS.start();
        }
        else
        {
            NUMBER_OF_AGENTS.dispose();
        }

    }//GEN-LAST:event_Number_of_Agents_handler

    private void Product_stored_handler(java.awt.event.↙
       ↪ ItemEvent evt) {//GEN-FIRST:↙
       ↪ event_Product_stored_handler
        if(ItemEvent.SELECTED == evt.getStateChange())
        {
                PRODUCT_STORED = new ShowCharts(God.↙
                    ↪ getRecorder(), Recorder.PRODUCT_STORED,
                      "Stored␣product", "Stored␣product", "↙
                        ↪ Time", "Product");
                Thread chart_PRODUCT_STORED = new Thread(↙
                    ↪ PRODUCT_STORED);
            chart_PRODUCT_STORED.start();
            }
            else
            {
                PRODUCT_STORED.dispose();
            }
        }//GEN-LAST:event_Product_stored_handler

        private void Wealth_handler(java.awt.event.ItemEvent↙
            ↪  evt) {//GEN-FIRST:event_Wealth_handler
            // TODO add your handling code here:
            if(ItemEvent.SELECTED == evt.getStateChange())
            {
                WEALTH = new ShowCharts(God.getRecorder(), ↙
                    ↪ Recorder.WEALTH,
                      "Wealth", "Wealth", "Time", "Money");
                Thread chart_WEALTH = new Thread(WEALTH);
            chart_WEALTH.start();
            }
            else
            {
                WEALTH.dispose();
            }
        }//GEN-LAST:event_Wealth_handler

        private void Total_births_handler(java.awt.event.↙
            ↪ ItemEvent evt) {//GEN-FIRST:↙
            ↪ event_Total_births_handler
            // TODO add your handling code here:
            if(ItemEvent.SELECTED == evt.getStateChange())
            {
                TOTAL_BIRTHS = new ShowCharts(God.↙
                    ↪ getRecorder(), Recorder.TOTAL_BIRTHS,
                      "Total␣births", "Total␣births", "Time", ↙
                        ↪ "Births" );
            Thread chart_TOTAL_BIRTHS = new Thread(TOTAL_BIRTHS);
        chart_TOTAL_BIRTHS.start();
        }
        else
        {
            TOTAL_BIRTHS.dispose();
        }
    }//GEN-LAST:event_Total_births_handler

    private void Total_deaths(java.awt.event.ItemEvent evt) {//GEN-FIRST:event_Total_deaths
        // TODO add your handling code here:
        if(ItemEvent.SELECTED == evt.getStateChange())
        {
            TOTAL_DEATHS = new ShowCharts(God.getRecorder(), Recorder.TOTAL_DEATHS,
                "Total␣bankruptcies", "Total␣bankruptcies", "Time", "bankruptcies");
            Thread chart_TOTAL_DEATHS = new Thread(TOTAL_DEATHS);
        chart_TOTAL_DEATHS.start();
        }
        else
        {
            TOTAL_DEATHS.dispose();
        }
    }//GEN-LAST:event_Total_deaths

    private void Median_wait_time_handler(java.awt.event.ItemEvent evt) {//GEN-FIRST:event_Median_wait_time_handler
        // TODO add your handling code here:
        if(ItemEvent.SELECTED == evt.getStateChange())
        {
            MEDIAN_WAIT_TIME = new ShowCharts(God.getRecorder(), Recorder.MEDIAN_WAIT_TIME,
                "Median␣wait␣time", "Median␣wait␣time", "Time", "Wait␣Time");
                Thread chart_MEDIAN_WAIT_TIME = new Thread(MEDIAN_WAIT_TIME);
            chart_MEDIAN_WAIT_TIME.start();
            }
            else
            {
                MEDIAN_WAIT_TIME.dispose();
            }
        }//GEN-LAST:event_Median_wait_time_handler

        private void jCheckBox3ItemStateChanged(java.awt.event.ItemEvent evt) {//GEN-FIRST:event_jCheckBox3ItemStateChanged
            if(ItemEvent.SELECTED == evt.getStateChange())
            {
                PRICE = new ShowCharts(God.getRecorder(), Recorder.PRICE,
                    "Price", "Price", "Time", "Price");
                Thread chart_PRICE = new Thread(PRICE);
            chart_PRICE.start();
            }
            else
            {
                PRICE.dispose();
            }
        }//GEN-LAST:event_jCheckBox3ItemStateChanged

        private void AgeEventHandler(java.awt.event.ItemEvent evt) {//GEN-FIRST:event_AgeEventHandler
           if(ItemEvent.SELECTED == evt.getStateChange())
            {
                AGE = new ShowCharts(God.getRecorder(), Recorder.AGE,
                    "Age", "Age", "Time", "Age");
                Thread chart_AGE = new Thread(AGE);
            chart_AGE.start();
            }
            else
            {
                PRICE.dispose();
        }
    }//GEN-LAST:event_AgeEventHandler

    private void jCheckBox5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jCheckBox5ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jCheckBox5ActionPerformed

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Report().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JCheckBox jCheckBox13;
    private javax.swing.JCheckBox jCheckBox2;
    private javax.swing.JCheckBox jCheckBox3;
    private javax.swing.JCheckBox jCheckBox4;
    private javax.swing.JCheckBox jCheckBox5;
    private javax.swing.JCheckBox jCheckBox6;
    private javax.swing.JCheckBox jCheckBox8;
    // End of variables declaration//GEN-END:variables

}
```

SetInitialProperties.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * SetInitialProperties.java
 *
 * Created on 5/Out/2010, 16:27:48
 */

package marketsim;

/**
 *
 * @author lfp
 */
public class SetInitialProperties extends javax.swing.↙
    ↪ JFrame {

    private double produceWill = 0; //Factor controlling↙
        ↪ the amount of product this agent produces
    private double consumeWill = 0; //Factor controlling↙
        ↪ the amount of product this agent consumes
    private double replicationStrategy = 0; //Factor ↙
        ↪ controlling the replication strategy
    private double replicationIntensity = 0; //Factor ↙
        ↪ controlling the replication Intensity
    private double storedProduct = 0; //The amount of ↙
        ↪ stored product


    private double storedWealth = 0; //The amount of ↙
        ↪ wealth the agent
    private long timeLeft =0; //The amount of time the ↙
        ↪ agent will survive
    private long AgentID = 0;
    private long parentAgentID  = 0;
    private long saudade = 0;

```

```java
    /** Creates new form SetInitialProperties */
    public SetInitialProperties() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jSpinner1 = new javax.swing.JSpinner();
        jSpinner2 = new javax.swing.JSpinner();
        jSpinner3 = new javax.swing.JSpinner();
        jSpinner5 = new javax.swing.JSpinner();
        jSpinner7 = new javax.swing.JSpinner();
        jSpinner9 = new javax.swing.JSpinner();
        jLabel7 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Inital␣Properties");

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Initial␣Properties"));

        jLabel1.setText("Produce␣Will");

```

```
jLabel2.setText("Consume␣Will");

jLabel3.setText("Replication␣Strategy");

jLabel4.setText("Stored␣Product");

jLabel5.setText("Stored␣Wealth");

jLabel6.setText("Time␣Left");

org.jdesktop.layout.GroupLayout jPanel1Layout = ↙
   ↪ new org.jdesktop.layout.GroupLayout(jPanel1↙
   ↪ );
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(org.↙
       ↪ jdesktop.layout.GroupLayout.LEADING)
    .add(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .add(jPanel1Layout.createParallelGroup(↙
           ↪ org.jdesktop.layout.GroupLayout.↙
           ↪ LEADING)
            .add(jLabel5)
            .add(jLabel6)
            .add(jLabel3)
            .add(jLabel4)
            .add(jPanel1Layout.↙
               ↪ createParallelGroup(org.↙
               ↪ jdesktop.layout.GroupLayout.↙
               ↪ LEADING)
                .add(jLabel1)
                .add(org.jdesktop.layout.↙
                   ↪ GroupLayout.TRAILING, ↙
                   ↪ jLabel2)))
        .addPreferredGap(org.jdesktop.layout.↙
           ↪ LayoutStyle.RELATED, 18, Short.↙
           ↪ MAX_VALUE)
        .add(jPanel1Layout.createParallelGroup(↙
           ↪ org.jdesktop.layout.GroupLayout.↙
           ↪ LEADING)
```

```
                    .add(jSpinner1, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, ↙
                      ↪ 129, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE)
                    .add(jSpinner2, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, ↙
                      ↪ 129, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE)
                    .add(jSpinner5, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, ↙
                      ↪ 129, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE)
                    .add(jSpinner7, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, ↙
                      ↪ 129, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE)
                    .add(jSpinner3, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, ↙
                      ↪ 129, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE)
                    .add(jSpinner9, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, ↙
                      ↪ 129, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE))
                .addContainerGap(org.jdesktop.layout.↙
                  ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                  ↪ MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(org.↙
               ↪ jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel1Layout.createSequentialGroup()
                .add(17, 17, 17)
                .add(jPanel1Layout.createParallelGroup(↙
                  ↪ org.jdesktop.layout.GroupLayout.↙
                  ↪ BASELINE)
                    .add(jLabel1)
                    .add(jSpinner1, org.jdesktop.layout.↙
                      ↪ GroupLayout.PREFERRED_SIZE, org↙
                      ↪ .jdesktop.layout.GroupLayout.↙
                      ↪ DEFAULT_SIZE, org.jdesktop.↙
```

```
                          ↪ layout.GroupLayout.↙
                          ↪ PREFERRED_SIZE))
                  .addPreferredGap(org.jdesktop.layout.↙
                      ↪ LayoutStyle.RELATED)
                  .add(jPanel1Layout.createParallelGroup(↙
                      ↪ org.jdesktop.layout.GroupLayout.↙
                      ↪ BASELINE)
                      .add(jLabel2)
                      .add(jSpinner2, org.jdesktop.layout.↙
                          ↪ GroupLayout.PREFERRED_SIZE, org↙
                          ↪ .jdesktop.layout.GroupLayout.↙
                          ↪ DEFAULT_SIZE, org.jdesktop.↙
                          ↪ layout.GroupLayout.↙
                          ↪ PREFERRED_SIZE))
                  .addPreferredGap(org.jdesktop.layout.↙
                      ↪ LayoutStyle.RELATED)
                  .add(jPanel1Layout.createParallelGroup(↙
                      ↪ org.jdesktop.layout.GroupLayout.↙
                      ↪ BASELINE)
                      .add(jLabel3)
                      .add(jSpinner5, org.jdesktop.layout.↙
                          ↪ GroupLayout.PREFERRED_SIZE, org↙
                          ↪ .jdesktop.layout.GroupLayout.↙
                          ↪ DEFAULT_SIZE, org.jdesktop.↙
                          ↪ layout.GroupLayout.↙
                          ↪ PREFERRED_SIZE))
                  .addPreferredGap(org.jdesktop.layout.↙
                      ↪ LayoutStyle.RELATED)
                  .add(jPanel1Layout.createParallelGroup(↙
                      ↪ org.jdesktop.layout.GroupLayout.↙
                      ↪ BASELINE)
                      .add(jLabel4)
                      .add(jSpinner7, org.jdesktop.layout.↙
                          ↪ GroupLayout.PREFERRED_SIZE, org↙
                          ↪ .jdesktop.layout.GroupLayout.↙
                          ↪ DEFAULT_SIZE, org.jdesktop.↙
                          ↪ layout.GroupLayout.↙
                          ↪ PREFERRED_SIZE))
                  .addPreferredGap(org.jdesktop.layout.↙
                      ↪ LayoutStyle.RELATED)
```

```
                .add(jPanel1Layout.createParallelGroup(↙
                   ↪ org.jdesktop.layout.GroupLayout.↙
                   ↪ BASELINE)
                    .add(jLabel5)
                    .add(jSpinner3, org.jdesktop.layout.↙
                       ↪ GroupLayout.PREFERRED_SIZE, org↙
                       ↪ .jdesktop.layout.GroupLayout.↙
                       ↪ DEFAULT_SIZE, org.jdesktop.↙
                       ↪ layout.GroupLayout.↙
                       ↪ PREFERRED_SIZE))
                .addPreferredGap(org.jdesktop.layout.↙
                   ↪ LayoutStyle.RELATED)
                .add(jPanel1Layout.createParallelGroup(↙
                   ↪ org.jdesktop.layout.GroupLayout.↙
                   ↪ BASELINE)
                    .add(jLabel6)
                    .add(jSpinner9, org.jdesktop.layout.↙
                       ↪ GroupLayout.PREFERRED_SIZE, org↙
                       ↪ .jdesktop.layout.GroupLayout.↙
                       ↪ DEFAULT_SIZE, org.jdesktop.↙
                       ↪ layout.GroupLayout.↙
                       ↪ PREFERRED_SIZE))
                .addContainerGap(org.jdesktop.layout.↙
                   ↪ GroupLayout.DEFAULT_SIZE, Short.↙
                   ↪ MAX_VALUE))
        );

        jLabel7.setText("Close␣window␣after␣setting␣↙
           ↪ initial␣properties");

        org.jdesktop.layout.GroupLayout layout = new org↙
           ↪ .jdesktop.layout.GroupLayout(getContentPane↙
           ↪ ());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.↙
               ↪ layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .addContainerGap()
                .add(layout.createParallelGroup(org.↙
                   ↪ jdesktop.layout.GroupLayout.LEADING↙
```

```
                    ↪ )
                    .add(layout.createSequentialGroup()
                        .add(jPanel1, org.jdesktop.↙
                            ↪ layout.GroupLayout.↙
                            ↪ DEFAULT_SIZE, org.jdesktop.↙
                            ↪ layout.GroupLayout.↙
                            ↪ DEFAULT_SIZE, Short.↙
                            ↪ MAX_VALUE)
                        .addContainerGap())
                    .add(org.jdesktop.layout.GroupLayout↙
                        ↪ .TRAILING, layout.↙
                        ↪ createSequentialGroup()
                        .add(jLabel7)
                        .add(25, 25, 25))))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.↙
                ↪ layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.↙
                ↪ TRAILING, layout.createSequentialGroup↙
                ↪ ()
                .addContainerGap()
                .add(jLabel7)
                .addPreferredGap(org.jdesktop.layout.↙
                    ↪ LayoutStyle.RELATED, 18, Short.↙
                    ↪ MAX_VALUE)
                .add(jPanel1, org.jdesktop.layout.↙
                    ↪ GroupLayout.PREFERRED_SIZE, org.↙
                    ↪ jdesktop.layout.GroupLayout.↙
                    ↪ DEFAULT_SIZE, org.jdesktop.layout.↙
                    ↪ GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents


    // Variables declaration - do not modify//GEN-BEGIN:↙
        ↪ variables
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JSpinner jSpinner1;
    private javax.swing.JSpinner jSpinner2;
    private javax.swing.JSpinner jSpinner3;
    private javax.swing.JSpinner jSpinner5;
    private javax.swing.JSpinner jSpinner7;
    private javax.swing.JSpinner jSpinner9;
    // End of variables declaration//GEN-END:variables

}
```

ShowCharts.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.ClipboardOwner;
import java.awt.datatransfer.StringSelection;
import java.awt.datatransfer.Transferable;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.DecimalFormat;
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.apache.commons.math.stat.descriptive.↙
    ↪ DescriptiveStatistics;


/**
 *
 * @author Lu s Filipe dos Reis Pereira
 * @email LuisFRPereira@gmail.com
 *
 */
```

```
public class ShowCharts extends JFrame implements Runnable{
    ArrayList<MetricPoint> pointList;
    DescriptiveStatistics descriptiveStatistics = null;
    DecimalFormat precision =null;


    public ShowCharts(Recorder recorder, Integer type, String lable,
            String title, String x, String y) {
        super("Metrics charts");
        descriptiveStatistics = new DescriptiveStatistics();
        XYSeries xySeries = new XYSeries(lable);
        pointList = recorder.getList(type);
        if (pointList!=null) {
            for (MetricPoint p : pointList) {
                xySeries.add(p.getTime().getMilliseconds(), p.getMetric());
                descriptiveStatistics.addValue(p.getMetric());
            }
            XYSeriesCollection dataset = new XYSeriesCollection();
            dataset.addSeries(xySeries);
            JFreeChart chart = ChartFactory.createXYLineChart(title,
                    x, y, dataset, PlotOrientation.VERTICAL,
                    rootPaneCheckingEnabled, rootPaneCheckingEnabled, rootPaneCheckingEnabled);

            //setContentPane(new ChartPanel(chart));
            this.add(new ChartPanel(chart), BorderLayout.CENTER);

            JButton exportToExcelButton = new JButton("Export data to clipboard (tsv)");
            ActionListener exportToExcelHandler = new exportToExcelHandler(pointList);
        exportToExcelButton.addActionListener(exportToExcelHandler);

        JButton exportToMatLabButton = new JButton("Export Data to clipboard (MATLAB)");
        ActionListener exportToMatLabHandler = new exportToMatLabHandler(pointList);
        exportToMatLabButton.addActionListener(exportToMatLabHandler);
        JPanel JPButtons = new JPanel(new FlowLayout(FlowLayout.CENTER));
        JPButtons.add(exportToExcelButton);
        JPButtons.add(exportToMatLabButton);

        this.add(JPButtons, BorderLayout.SOUTH);
        precision = new DecimalFormat("#0.0000");
        JPanel stats = new JPanel();
        stats.setLayout(new FlowLayout(FlowLayout.CENTER));

        stats.add(new JLabel("\u03BC:"));
        JTextField averageField = new JTextField(precision.format(
                descriptiveStatistics.getSum()/descriptiveStatistics.getN()));
        averageField.setEditable(false);
        stats.add(averageField);

        stats.add(new JLabel("\u03C3:"));
        JTextField sigmaField = new JTextField(precision.format(descriptiveStatistics.getStandardDeviation()));
        sigmaField.setEditable(false);
        stats.add(sigmaField);

        stats.add(new JLabel("Skewness:"));
        JTextField skewField = new JTextField(precision.format(descriptiveStatistics.getSkewness()));
        skewField.setEditable(false);
        stats.add(skewField);
```

```

            stats.add(new JLabel(”Kurtosis:”));
            JTextField kurtField = new JTextField(↙
               ↪ precision.format(descriptiveStatistics.↙
               ↪ getKurtosis()));
            kurtField.setEditable(false);
            stats.add(kurtField);

            stats.add(new JLabel(”Sample␣Size:”));
            JTextField ssizeField = new JTextField(↙
               ↪ precision.format(descriptiveStatistics.↙
               ↪ getN()));
            ssizeField.setEditable(false);
            stats.add(ssizeField);
            this.add(stats, BorderLayout.NORTH);


        }
        else
        {
            JButton NoDataButton = new JButton(”No␣Data␣↙
               ↪ Found”);
            ActionListener NoDataActionHandler =new ↙
               ↪ NoDataButtonHandler(this);
            NoDataButton.addActionListener(↙
               ↪ NoDataActionHandler);

            setContentPane(NoDataButton);
        }

    }
    public void run()
    {
        pack();
        setVisible(true);

    }
    private class NoDataButtonHandler implements ↙
       ↪ ActionListener{
        JFrame frame;
        public NoDataButtonHandler(JFrame jFrame)
        {
            frame = jFrame;
        }
        public void actionPerformed(ActionEvent event)
        {
            frame.dispose();
        }
    }
    private class exportToExcelHandler implements ActionListener, ClipboardOwner
    {
        ArrayList<MetricPoint> pointList;
        public exportToExcelHandler(ArrayList<MetricPoint> pointList )
        {
            this.pointList=pointList;
        }
        public void actionPerformed(ActionEvent event)
        {
            String tabel = "";
            for (MetricPoint p : pointList) {
                tabel = tabel +
                        p.getTime().getMilliseconds() + "\t" + p.getMetric() + "\n";
            }
            StringSelection stringSelection = new StringSelection( tabel );
            Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
            clipboard.setContents(stringSelection, this);
        }

        public void lostOwnership(Clipboard clipboard, Transferable contents) {
            //do nothing
        }
    }
    private class exportToMatLabHandler implements ActionListener, ClipboardOwner
    {
        ArrayList<MetricPoint> pointList;

        public exportToMatLabHandler(ArrayList<MetricPoint> pointList )
        {
            this.pointList=pointList;
        }
        public void actionPerformed(ActionEvent event)
        {
            String tabel = "[";
            for (MetricPoint p : pointList) {
                tabel = tabel +
                        p.getTime().getMilliseconds() + "␣" + p.getMetric() + ";";
            }
            tabel = tabel + "]";
            StringSelection stringSelection = new StringSelection( tabel );
            Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
            clipboard.setContents(stringSelection, this);
        }

        public void lostOwnership(Clipboard clipboard, Transferable contents) {
            //do nothing
        }
    }

}
```

Transaction.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package marketsim;

import javax.realtime.RelativeTime;

/**
 *
 * @author Lu s Filipe dos Reis Pereira
 * @email LuisFRPereira@gmail.com
 *
 */
class Transaction {

    public static final byte BUY = 0;
    public static final byte SELL = 1;

    long AgentIndex = -1;
    String ID = null;
    Double price = new Double(0.0);
    Long amount = new Long(0);
    byte TYPE = Byte.MAX_VALUE;
    RelativeTime creationTime;

    public long getAgentIndex() {
        return AgentIndex;
    }

    public void setAgentIndex(long AgentIndex) {
        this.AgentIndex = AgentIndex;
    }

    public String getID() {
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }

    public byte getTYPE() {
        return TYPE;
    }

    public void setTYPE(byte TYPE) {
        this.TYPE = TYPE;
    }

    public Long getAmount() {
        return amount;
    }

    public void setAmount(Long amount) {
        this.amount = amount;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public RelativeTime getCreationTime() {
        return creationTime;
    }

    public void setCreationTime(RelativeTime creationTime) {
        this.creationTime = creationTime;
    }

}
```