



# MULTILINGUAL INVESTIGATION OF THEORY-BASED INTERVENTION FOR PROGRAM COMPREHENSION

Bachelor Thesis  
Paderborn University

Celia García Ledesma  
Supervised by Birte Heinemann

## DECLARATION OF AUTHORSHIP

I hereby declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. According to my knowledge, the content or parts of this thesis have not been presented to any other examination authority and have not been published. I am aware that the respective work can be considered as “fail” in the event of a false declaration. In case of justified suspicion, the thesis in digital form can be examined with the help of “Turnitin”. For the comparison of my work with existing sources  I agree to storage in the institutional repository to enable comparison with future theses submitted /  I do not agree to storage in the repository. Further rights of reproduction and usage, however, are not granted here. In any case, the examination and evaluation of my work has to be carried out individually and independently from the results of the plagiarism detection service.



---

Signature, city, date

## ACKNOWLEDGEMENTS

I would like to thank everyone who helped me during these months. To the professionals in the Computing Education Department. To all participants that took part in the experiment. Thank you Birte for helping me and showing me that working can be also fun. Thanks to my friends Daniel and Miriam for helping me editing the intervention, to my mother Raquel for letting me enjoy this opportunity and to my family, who has sent love everyday along these months. Finally, thank you Pablo for supporting me always even though you do not understand anything about Computer Science.

# Content

---

Figures .....	3
Tables .....	3
Introduction .....	4
Problem Analysis .....	4
Theoretical Background .....	5
The Block Model .....	5
Bloom’s Taxonomy .....	6
Related Work.....	7
Explanation of the Experiment.....	8
Method .....	8
Participants.....	8
Design .....	8
Apparatus and Materials .....	17
Example .....	19
Results of the Experiment .....	20
Conclusions .....	26
Discussion and Follow up Study Ideas.....	27
References.....	30

## Figures

---

Figure 1: The Block Model. ....	6
Figure 2: Welcome Message.....	8
Figure 3: Calibration.....	10
Figure 4: Introduction to Read the Source Code. ....	12
Figure 5: Block Model and Bloom's Taxonomy in the Multiple-Choice Questions.....	13
Figure 6: End of the Experiment. ....	17
Figure 7: Master Workstation.....	17
Figure 8: 20 Equipped Workstations with SMI REDn Scientific Eye Trackers. ....	18
Figure 9: Workstation. ....	18
Figure 10: Logitech Camera with Integrated Microphone. ....	19
Figure 11: SMI REDn Scientific Eye Tracker.....	19

## Tables

---

Table 1: Questionnaire.....	9
Table 2: German Intervention.....	10
Table 3: English Intervention. ....	11
Table 4: Spanish Intervention.....	11
Table 5: List Program and Comprehension Questions.....	15
Table 6: Rectangle Program and Comprehension Questions. ....	16
Table 7: Vehicle Program. ....	20
Table 8: Correct Answers. ....	21
Table 9: Length.....	21
Table 10: Number of Fixations. ....	22
Table 11: Number of Saccades.....	22
Table 12: Programming Expertise. ....	22
Table 13: Java Expertise.....	23
Table 14: Gender.....	23
Table 15: List Heat Maps.....	24
Table 16: Rectangle Heat Maps. ....	25
Table 17: Conclusions. ....	27

# Introduction

---

Eye movement modeling examples (EMME) are demonstrations of a computer-based task by a human model (e.g., a teacher), with the model's eye movements superimposed on the task to guide learners' attention (Tim van Marlen, 2016) along with the model's verbal explanations (Jarodzka, Van Gog, Dorr, Scheiter, & Gerjets, 2013).

This thesis is the continuation of an experiment called "Eye-movement Modeling Examples in Source Code Comprehension: A Classroom Study". This first experiment studies how effective is showing novice programmers how experts read code with a video with the expert's gaze guided by a verbal explanation. Therefore, this thesis studies, using a similar experiment, whether only verbal explanation and visual stimuli without the expert's gaze could be also helpful for the programming novices.

The goal is to find teaching methods and ideas how to help novice programmers, so that they can overcome the obstacle of program code comprehension faster. If it is understood how novices read source code, and what hardships they face during initial learning, better tools and working environments can be designed (Beelders & du Plessis, 2016). In order to achieve this, it might be helpful to use Eye-movement Modeling Examples (EMME) at some point of the beginning of their studies.

## Problem Analysis

When novice programmers read source code for the first time, they usually do not know how to read it since reading program text differs from reading natural text (Busjahn, et al., 2015) (Busjahn, Bednarik, & Schulte, What influences dwell time during source code reading?: analysis of element type and frequency as factors, 2014) and it is also difficult for them to comprehend how the code works. Due to this, is interesting to find how EMME can support reading and comprehending source code.

The experiment that is going to take place is based on the concept of Block - and Relation - reading (BR-reading). BR-Reading is that (Reading) Blocks are basic entities capturing a coherent idea to be understood while reading. Understanding the hole text thus requires understanding the parts (the blocks) and how they are connected (the relation between blocks) (Bednarik, Budde, Heinemann, Schulte, & Vrzakova, 2018). The Block Model is explained carefully in the section "Theoretical Background" below.

In the previous experiment, it is hypothesized that an EMME-based intervention should be effective by guiding attention to important parts of the code and provide ways to integrate elements of the code to a meaningful whole and thereby aid comprehension (Bednarik, Budde, Heinemann, Schulte, & Vrzakova, 2018). However, for motor tasks, the central steps are directly observable, but for cognitive tasks this is not the case. Instead, the model has to be asked to unravel these covert processes by verbalizing them while performing the task (Collins, Brown, & Newman, 1989). In this thesis, it is hypothesized that the verbal explanation is an important part of the process. As the learner is told what to look at, why and how to look at some points of the

source code, he/she should understand the process and should be able to achieve better performance in comprehension (quicker or better understanding).

Nevertheless, EMME with verbal explanations might not be always a helpful resource, since sometimes the given information, both the gaze and the verbal explanation, instead of complementing each other, result not being effective for learning. When verbal explanations are sufficient to guide visual attention, displaying the eye movements would be redundant (Sweller, 2005). In addition, research has shown that redundant information tends to distract students' attention, which can lead to detrimental effects on learning outcomes (Kalyuga, Chandler, & Sweller, 1999). Van Gog presented in other study that the verbal explanation and the eye movements of the model were probably redundant. This was the case because the students could easily infer from the verbalizations alone where to look (Jarodzka, Van Gog, Dorr, Scheiter, & Gerjets, 2013).

In consequence of the ideas presented, the goal of this experiment is to know whether verbal explanation in reading source code foster learning source code reading by novice programmers.

## Theoretical Background

---

In this section, two main references that we used to design the multiple-choice comprehension questions are shortly explained. The multiple-choice questions are defined in the section "Design".

### The Block Model

The Block Model (Schulte, 2008) is an educational model for program understanding that suggests three dimensions and four distinct abstraction levels: firstly atoms/words are read and build the Atom Level, where it is understood the language elements, the operation of the statements and its function. Then, they are joined in semantic units, known as blocks, which are a sequence of statements that function as a sub-goal of the program, this is called the Block Level. After that, relations between blocks are considered and it is understood how they are related to the goals (Relation Level). As the last step the Macrostructure Level, where the overall structure of the program is understood, the algorithm and the final purpose of the program (Bednarik, Budde, Heinemann, Schulte, & Vrzakova, 2018).

The intervention video that is used in the experiment is based in this model as well as in the previous study because of the purpose of comparing both studies after, therefore it was desired to maintain the intervention as similar as possible to the previous one.

<b><u>Macro</u></b> <b><u>struc-</u></b> <b><u>ture</u></b>	(Understanding the) overall structure of the program text	Understanding the „algorithm“ of the program	Understanding the goal/ the purpose of the program (in its context)
<b><u>Rela-</u></b> <b><u>tions</u></b>	References between blocks, e.g.: method calls, object creation, accessing data...	sequence of method calls „object sequence diagrams“	Understanding how subgoals are related to goals, how function is achieved by sub-functions
<b><u>Blocks</u></b>	Regions of Interests' (ROI) that syntactically or semantically build a unit	Operation of a block, a method, or a ROI (as sequence of statements)	Function of a block, maybe seen as sub-goal
<b><u>Atoms</u></b>	Language elements	Operation of a statement	Function of a statement. Goal only understandable in context
	<b><u>Text surface</u></b>	<b><u>Program execution (data flow and control flow)</u></b>	<b><u>Functions (as means or as purpose), goals of the program</u></b>
<b>Duality</b>	<b><u>“Structure”</u></b>		<b><u>“Function”</u></b>

Figure 1: The Block Model.

## Bloom's Taxonomy

Bloom's taxonomy establishes a hierarchy of six levels with an increasing level of student learning. Each level presupposes the training of the student at the preceding levels (Hernán-Losada, Velázquez-Iturbide, & Lázaro, 2019).

- **Level 1 or level of knowledge.** The student can recognize or remember the information without needing any kind of understanding or reasoning about its content.
- **Level 2 or level of comprehension.** The student can understand and explain the meaning of the information received.
- **Level 3 or application level.** The student may select and use data and methods to solve a given task or problem.
- **Level 4 or level of analysis.** The student can distinguish, classify and relate hypotheses and evidences of the given information, as well as to decompose a problem in their parts.
- **Level 5 or synthesis level.** The student can generalize ideas and apply them to solve a new problem.
- **Level 6 or assessment level.** The student can compare, criticize and evaluate methods or solutions to solve a problem or to choose the best one.

To test the level of comprehension that the participants reach, Bloom taxonomy as well as the Block Model are used to design the experiment's multiple-choice questions, since it allows a better overview about how deep the participants comprehend the given source code.

As in “Design” section below, there are three different multiple-choice questions that reach three different levels in the Block Model as in the Bloom's taxonomy from the lowest to the highest to test how well the student understood the program. There is a free text answer as

well, where the participants may explain themselves what they understood, which is in level 2 or level of comprehension.

## Related Work

---

There are two main documents that guide me and are related to the topic treated in this thesis. The first one is the previous study called “Eye-movement Modeling Examples in Source Code Comprehension: A Classroom Study”, that is like this one and is thought to be used to compare the results of this study. The authors created a classroom experiment in which their comprehension strategies were cued by an educational intervention based on theories of program comprehension and accompanied by a visualization of eye-movement strategies of an advanced programmer (Bednarik, Budde, Heinemann, Schulte, & Vrzakova, 2018).

The other related work is the Eye Movements in Programming Dataset, which establishes that programming education and practice mainly focus on writing code, while the reading skills are often taken for granted. Reading occurs in debugging, maintenance and the learning of programming languages. It provides the essential basis for comprehension. By analysing behavioural data such as gaze during code reading processes, we explore this essential part of programming (Bednarik, et al., 2018). Related to this dataset, in this thesis it is used some of the stimulus material (rectangle and vehicle source codes) and at first, we wanted to use the same questions, but we found them to be not enough accurate to show if the participant understood properly or not, so we decided to change them, as explained in “Design” section.

As the previous experiment is also related to the EMIP dataset, the limitations of these works are similar. First, there is only one question to find if the participant has understood the program shown. I find it difficult to know if the participant really understood the code with just one question, therefore in this study there will be three comprehension questions plus a free text answer based in the Block Model and in Bloom taxonomy with different levels to know how deeply the participant understood the code. Later in “Design” section these questions and their design are explained.

The source codes used to test the participants’ comprehension in these experiments were obvious since they have the class, methods and variables names related to their function. Therefore, it can be thought that the person did not understand but read carefully the content and he/she was able to answer the question without having understood the code. In this experiment, it is changed to names that do not give any cue to the participants (mystery, calculation, etc), so they must do the effort to understand the code and read it carefully. Even though it is known that any experienced programmer would name a class “Mystery” and a method “calculation”, we thought that it could be more helpful to know if they really comprehend the program and its purpose.



# Explanation of the Experiment

---

## Method

This study is drawing on a previous experimental design called “Distributed Collection of Eye Movement Data in Programming” (Bednarik, et al., 2018), where a large data set was collected as above mentioned. The intervention instructions and the source code of both short programs are changed in the design.

## Participants

The participants which take part in the experiment are recruited from Paderborn University, for the English version and from different universities from Spain, such as Universidad Carlos III de Madrid, Universidad Pública de Navarra, Universidad de Valencia, Universidad de Murcia, among other, for the Spanish version. The demographic data and expertise levels were collected according to the EMIP experiment design. A total of 27 participants took part in the experiment, 14 male and 13 female participants, with a mean age of 23’9 years old.

## Design

The design of the experiment was based in the EMIP dataset (Bednarik, et al., 2018). At first, the whole intervention is explained by the person in charge. Each of the participants will be seated in each workstation prepared for this purpose. Then, the intervention will take place. First, the students will see a welcome message, as shown in Figure 2, and after that, the intervention will begin. Second, they must answer some demographic and expertise questions, which are defined in Table 1.

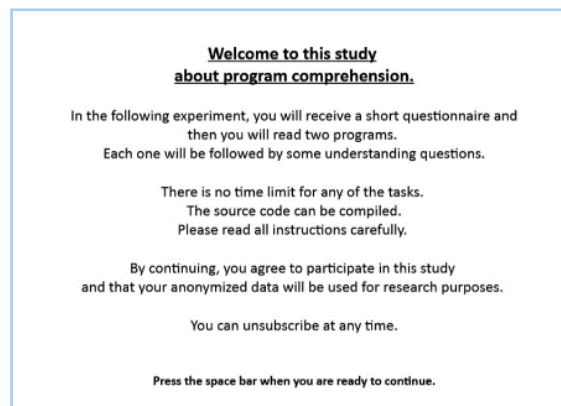


Figure 2: Welcome Message.

Table 1: Questionnaire.

Questionnaire
<ul style="list-style-type: none"> <li>• How old are you?</li> <li>• What is your gender? <ul style="list-style-type: none"> <li>○ Female</li> <li>○ Male</li> <li>○ Other</li> </ul> </li> <li>• What is your mothertonge / native language? <ul style="list-style-type: none"> <li>○ Español</li> <li>○ English</li> <li>○ Deutsch</li> <li>○ Free text answer if it is other</li> </ul> </li> <li>• What is your English level? <i>(so, if they do not understand the code, we will know why)</i> <ul style="list-style-type: none"> <li>○ Low</li> <li>○ Medium</li> <li>○ High</li> </ul> </li> <li>• What is your programming experience? <ul style="list-style-type: none"> <li>○ None</li> <li>○ Low</li> <li>○ Medium</li> <li>○ High</li> </ul> </li> <li>• What is your Java experience? <ul style="list-style-type: none"> <li>○ None</li> <li>○ Low</li> <li>○ Medium</li> <li>○ High</li> </ul> </li> <li>• How long have you been programming (in years)?</li> <li>• How long have you been programming in Java (in years)?</li> <li>• How often do you use programming languages other than Java? <ul style="list-style-type: none"> <li>○ Never</li> <li>○ Less than 1 hour per month</li> <li>○ Less than 1 hour per week</li> <li>○ Less than 1 hour per day</li> <li>○ More than 1 hour per day</li> </ul> </li> <li>• How often do you program in Java? <ul style="list-style-type: none"> <li>○ Never</li> <li>○ Less than 1 hour per month</li> <li>○ Less than 1 hour per week</li> <li>○ Less than 1 hour per day</li> <li>○ More than 1 hour per day</li> </ul> </li> <li>• What other programming languages do you use? Please rate their expertise (low, medium, high), for example: Python (medium), C (high).</li> <li>• Are you wearing glasses or contact lenses right now? <i>(It is a technical question important for the accuracy).</i> <ul style="list-style-type: none"> <li>○ No</li> <li>○ Glasses</li> <li>○ Contact lenses</li> </ul> </li> <li>• Are you currently wearing mascara or any other eye makeup? <i>(It is a technical question important for the accuracy).</i> <ul style="list-style-type: none"> <li>○ Yes</li> <li>○ No</li> </ul> </li> </ul>

After answering all these demographic questions, the next step is to calibrate the eye tracker as shown in Figure 3. This process ensures that the gaze data is being collected properly.

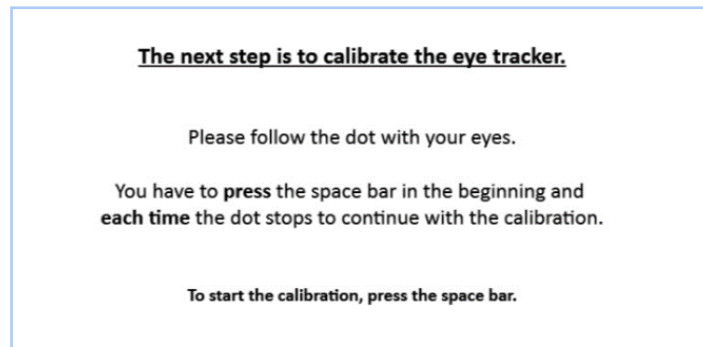


Figure 3: Calibration.

After the calibration, the participants will see a video with the explanation of two different experts, one is an English native speaker, Mr. Careth Davies, and one Spanish native speaker, Mr. José Vicente Álvarez Bravo, who is a professor of Object-Oriented Programming in Universidad de Valladolid (Spain), who will explain the instructions in each language. There is an intervention in English, and other one is in Spanish, for the participants to understand it as best as possible.

The instructions are shown below in German, English and Spanish. The verbal explanation is very similar to the previous used in the gaze intervention, but we upgraded it to a better comprehension, since the gaze it is not shown in this case.

Table 2: German Intervention.

<b>German</b>
<p>Programmieren lernen bedeutet, Programme zu schreiben - das Lesen und Verstehen wird dabei manchmal unterschätzt. Klar ist aber, das man verstehen muss was man da geschrieben hat. Das scheint leicht zu sein, weil man ja ständig liest. Doch es gibt große Unterschiede zu normalem, wir sagen: natürlichsprachlichem Text.</p> <p>Normaler Text ist so geschrieben, dass man ihn möglichst Satz für Satz lesen kann und den Text dann versteht. Der Inhalt ist in einer für das Verstehen sinnvollen Weise aufgeschrieben. Quelltext aber richtet sich ja auch an den Computer, der ihn ausführt - daher ist die dargestellte Reihenfolge und das was in einem Satz oder Absatz - wir sagen hier Block - steht nicht immer direkt zu verstehen.</p> <p>Schauen wir uns das Beispiel genauer an.</p> <p>Wir zeigen euch nun, wie das Lesen bei einem Experten ablaufen kann. Die Bereiche über die wir reden markieren wir mit Rechtecken.</p> <p>Im ersten Moment verschaffen sich viele Experten einen Überblick über den gesamten Programmcode.</p> <p>Nachdem der Code gescannt wurde, geht man den Code blockweise durch. [Stop]</p>

Immer wenn ein Block gelesen wird, wird der Inhalt mit den anderen schon gelesenen Blöcken verknüpft. Wenn wir Experten beim Lesen beobachten, können wir dieses Verknüpfen der Bedeutung verschiedener Blöcke oft sehen.

Experten erkennen auch die wichtigen und schwierigen Blöcke und das Lesen dieser Blöcke dauert länger.

Am Ende hat man dann manchmal noch eine Phase, in der Teile des Programms ein weiteres mal gelesen werden - und eine Phase in der der Gesamtzusammenhang gelesen wird.

Table 3: English Intervention.

<b>English</b>
<p>Learning to program means writing programs, however reading and understanding is sometimes underestimated. What is clear is that you must understand what you wrote. That seems easy because you read all the time. But there are big differences to normal text, or what we call: natural language text.</p> <p>Normal text is written in sentences, and the order of sentences in natural language text is written for reading sentence by sentence, in a way that fosters understanding in a meaningful way. But source code is also directed to the computer that executes it - therefore the order shown is not like we would like to have as humans. What is done in a sentence or paragraph - we say here a block - cannot be understood directly.</p> <p>Let's take a closer look at the example.</p> <p>We will now show you how reading can be done by an expert. We mark the areas we are talking about with rectangles.</p> <p>At first, most experts get an overview of the entire program code. After the code has been scanned, you go through the code block by block. Whenever a block is read, the content is linked to the previous ones. If we observe experts reading program code, we can often see that they link the meaning of different blocks. Experts also recognize the important and difficult blocks and the reading of these blocks takes longer.</p> <p>In the end, there is sometimes a phase in which parts of the program are read again - and a phase in which all the previous steps of reading and understanding are connected to each other.</p>

Table 4: Spanish Intervention.

<b>Spanish</b>
<p>Aprender a programar significa escribir programas - a veces se subestima la lectura y la comprensión.</p> <p>Lo que está claro, sin embargo, es que hay que entender lo que se ha escrito. Eso parece fácil porque leemos continuamente.</p> <p>Pero hay grandes diferencias respecto al texto normal, al que llamamos: texto en lenguaje natural.</p> <p>El texto normal está escrito en oraciones, es decir, se debe leer frase por frase y luego entenderlo, porque el contenido del texto está escrito para entenderlo de una manera significativa.</p> <p>Pero el código fuente también está dirigido al ordenador que lo ejecuta - por lo tanto, el orden en el que se muestra no es el que nos gustaría tener como lectores y lo que se hace en una frase o párrafo -aquí llamado bloque- no siempre se entiende directamente.</p>

Echemos un vistazo más de cerca al ejemplo.

A continuación, le mostraremos cómo puede realizar la lectura un experto. Marcamos las áreas de las que estamos hablando con rectángulos.

Al principio, la mayoría de los expertos obtienen una visión general de todo el código del programa. Una vez escaneado el código, se lee el código bloque a bloque.

Cada vez que se lee un bloque, el contenido se vincula a los otros bloques ya leídos. Si observamos cómo expertos leen el código del programa, a menudo podemos ver que enlazan el significado de diferentes bloques.

Los expertos reconocen también los bloques más importantes y difíciles y dedican más tiempo a la lectura de estos bloques.

Al final, a veces hay una fase en la que se vuelven a leer partes del programa y otra en la que todos los pasos previos de lectura y comprensión se conectan entre sí.

After the instruction video with the verbal explanation in the proper language, the participants will read two short programs List and Rectangle, named Mystery 1 and Mystery 2 respectively, which are shown below in Table 5 and 6, and there is no time restriction to read the code and, after each program, they will be asked three multiple-choice questions and there is one text field where they can explain with their own words what is the program doing or what they understood.

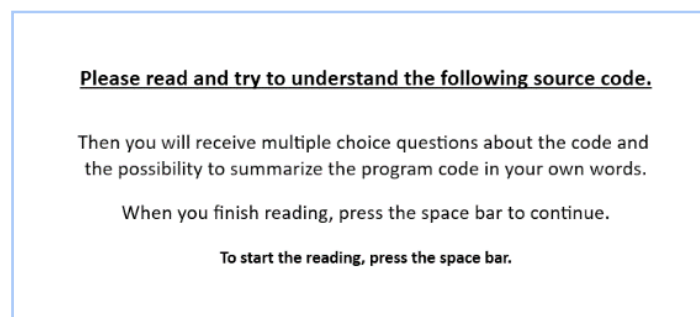


Figure 4: Introduction to Read the Source Code.

The programming language used in the study is Java, since the previous intervention example and source codes were written in Java as well. If we used a different programming language, it would be difficult to compare results after. List program was originally in Python and it comes from this study (Duran, Sorva, & Leite, 2018) but we changed it to Java to maintain the same language in the whole experiment.

In the explanation video the syntax is not highlighted, and we wanted the rest code to be as much similar as the one in the explanation, therefore we wrote them as plain black text. In addition, for experts, reading highlighted syntaxis is helpful but here we want to achieve that the participant reads all parts carefully, so we did not want any part to stand out more than the rest. Moreover, it is possible that some participants are used to program in other languages, so it may be confusing for them to see the code highlighted different.

In a study about the influence of syntax highlighting it is demonstrated that black-and-white code require more fixations than when the syntax is highlighting, which indicates that the comprehension may be easier if syntax highlighting is present. However, the difference is not significant. The study concludes that the learning curve is not impacted by the presentation of learning materials without the use of syntax highlighting (Beelders & du Plessis, 2016).

As it is known, experienced programmers always comment the code, due to it is easier for them to understand the program. In this case, as we wanted the participant to investigate by his/her own the meaning of the program, there are no comments in the given source codes.

The previous intervention only tested the participants comprehension level asking one multiple-choice question. This question, referred to the Rectangle program, gives the goal in the options, as it says “computes the area of rectangles by...” (Bednarik, et al., 2018), then it is hard to know whether the participant understood the code. Therefore, we thought that only one question is not enough because in this case, it reaches the Level of Knowledge or level 1 in Bloom’s taxonomy and the Atom Level in the Block Model, which are the lowest levels and we accomplish them in the first multiple-choice question, as can be seen in Figure 5 below, and we want to achieve higher levels of comprehension to get a better view of the participants understanding.

Consequently, we decided to reach three different comprehension levels, as shown in Figure 5, based in Bloom’s taxonomy and the Block Model, as it becomes more accurate to find the comprehension level the participants achieved.

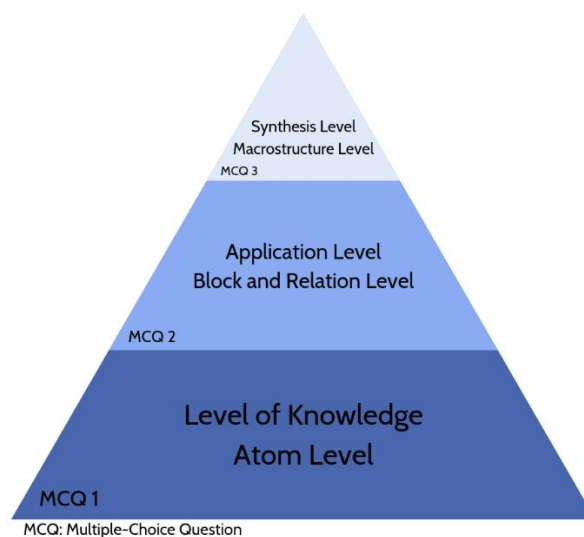


Figure 5: Block Model and Bloom's Taxonomy in the Multiple-Choice Questions.

We wrote three different multiple-choice questions and then added a free text question that reaches the level of comprehension or level 2 in Bloom’s taxonomy, where the participant can explain himself/herself, because we thought that maybe it could be helpful for us to know if the participant is able to demonstrate that he/she understood the program and explain the meaning of the information received with his/her own words.

At first, the questions may seem tricky and difficult to answer but, if the participant understood what the program does, it is so easy to know the correct answer. In the second multiple-choice question in List program, for example, to know the answer the participants just need to think what the program does to find the solution, and they do not really need to remember the code. The same happens with Rectangle program, where the participants just need to know that they are calculating a rectangle area.

In the first question, we wanted to reach the level 1 in Bloom's taxonomy, which is the level of knowledge, where the student can recognize or remember the information without needing any kind of understanding or reasoning about its content. Referring to the Block Model, this question is in the Atom level, since it refers to just the language elements. In this question, we want to know if the participant read the code carefully. The first multiple-choice question in Rectangle program is the most related to the previous experiment, since it is asking about the order of the variables, but we decided to change it to four possible answers because with three combinations the correct answer could be deducted easier.

The second question is defined in the level 3 or application level of Bloom's taxonomy, where the student may select and use data and methods to solve a given task or problem. In this case, we asked the participants to solve a new case of the given problem. This question could also be placed in the level 5 or synthesis level, where the student can apply ideas to solve a new problem. This question also refers to the Block level as well as the Relation level in the Block Model, since it is asking about the operation of a block and to know the answer, it is important to know the references between the blocks. In this question, we want to know if the participants understand what the program does and apply it to a similar problem, then it results as if the participants could use the program in their minds to solve it.

The third question is defined in the level 5 or synthesis level, where the student can generalize ideas. In this last question, we can finally know if the participant understood the whole program and its goal. This question is related to the Macrostructure level in the Block Model, with which we know if the student understands the algorithm of the program and the purpose of it in its context.

About the free text part, we discussed about which order should it have, and we decided to put it as the last one to not change the structure of the experiment, so it doesn't differ too much from the previous one. A further explanation is in "Discussion and Follow up Study Ideas" section.

Table 5: List Program and Comprehension Questions.

List
<pre> public class Mystery1{     public static void mystery1(int[] list){         int var1 = 0;         int count = 0;         int index = 0;         int sent = 9;         int element = list[index];          while(element != sent){             if(element &gt; 0){                 var1 = var1 + element;                 count += 1;             }             index += 1;             element = list[index];         }          if(count &gt; 0){             System.out.println("var1: " + var1 + " count: " + count + "var/count: " + var1/count);         }         else {             System.out.println(-1);         }     }      public static void main (String args[]){         int[] list = {2, -1, 3, 0, 1, 0, -1, 2, 9, 1, 2, 3};         mystery1(list);     } } </pre>
Comprehension Questions for List
<ol style="list-style-type: none"> <li>1. This program: <ol style="list-style-type: none"> <li>a. Prints ("var1: " + var1 + " count: " + count + "var/count: " + var1/count) if count = 0</li> <li><b>b. Prints (-1) if count = 0</b></li> <li>c. Prints ("var1: " + var1 + " count: " + count + "var/count: " + var1/count) if index = 0</li> <li>d. Prints (-1) if index = 0</li> <li>e. I am not sure</li> </ol> </li> <li>2. Imagine there is a new list = {3, -2, 4, 0, -1, 0, 2, -2, 0, 9, 0, 3}. What would System.out.println("var1: " + var1 + " count: " + count + "var/count: " + var1/count); print? <ol style="list-style-type: none"> <li><b>a. var1: 9, count: 3, var/count: 3</b></li> <li>b. var1: 12, count: 3, var/count: 4</li> <li>c. var1: 18, count: 3, var/count: 6</li> <li>d. I am not sure</li> </ol> </li> <li>3. This program: <ol style="list-style-type: none"> <li>a. Sums the numbers in the list and then divides the sum by the total of numbers found until reached a "9".</li> <li>b. Sums the natural numbers until reached a number greater than or equal to "9" and then divides the sum by the quantity of the numbers found.</li> <li><b>c. Sums the natural numbers until reached a "9" and then divides the sum by the quantity of the numbers found.</b></li> <li>d. Sums the numbers in the list and then divides the sum by the total of numbers until reached a number greater than or equal to "9".</li> <li>e. I am not sure.</li> </ol> </li> <li>4. Free text answer.</li> </ol>



Table 6: Rectangle Program and Comprehension Questions.

Rectangle
<pre> public class Mystery2 {     private int x1, y1, x2, y2;     public mystery2(int x1, int y1, int x2, int y2 ) {          this.x1 = x1;         this.y1 = y1;         this.x2 = x2;         this.y2 = y2;      }     public int calculation1 () {         return this.x2 - this.x1;     }     public int calculation2 () {         return this.y2 - this.y1;     }     public double calculation3 () {         return this.calculation1() * this.calculation2();     }     public static void main ( String[] args ) {         Mystery2 m1 = new Mystery2 ( 0, 0, 10, 10 );         System.out.println ( m1.calculation3 ( ) );         Mystery2 m2 = new Mystery2 ( 5, 5, 10, 10 );         System.out.println ( m2.calculation3 ( ) );     } } </pre>
Comprehension Questions for Rectangle
<ol style="list-style-type: none"> <li>1. This program:             <ol style="list-style-type: none"> <li>a. Calculates <math>x1 - x2</math> in calculation1</li> <li>b. Calculates <math>x2 - y2</math> in calculation1</li> <li>c. Calculates <math>x1 - y1</math> in calculation1</li> <li><b>d. Calculates <math>x2 - x1</math> in calculation1</b></li> <li>e. I am not sure</li> </ol> </li>   <li>2. Imagine a Mystery m3 with variables <math>x1 = 2</math>, <math>y1 = 2</math>, <math>x2 = 4</math>, <math>y2 = 4</math>, what would calculation3 return?             <ol style="list-style-type: none"> <li><b>a. 4.0</b></li> <li>b. 2.0</li> <li>c. 8.0</li> <li>d. I am not sure</li> </ol> </li>   <li>3. This program:             <ol style="list-style-type: none"> <li>a. Calculates the area of a Square.</li> <li><b>b. Calculates the area of a Rectangle.</b></li> <li>c. Calculates the area of a Trapezium/Trapezoid.</li> <li>d. I am not sure.</li> </ol> </li>   <li>4. Free text answer.</li> </ol>

At the end, the participants will be asked to wait until the rest of the participants are finished as shown in Figure 6 and then they will leave, and the intervention will be finished.

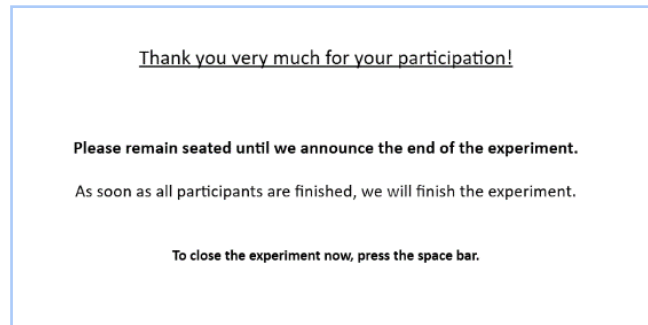


Figure 6: End of the Experiment.

## Apparatus and Materials

The experiment was taken in the PIN-Lab of Paderborn University, located in the Furstenalle building. The explanation about the setup of this laboratory is based in this Master Thesis (Schlichtig, 2018).

The PIN-Lab is a class- and a seminar room of the Computing Education Research chair at Paderborn University. It provides 20 equal workstations equipped with SMI REDn Scientific eye trackers attached to 27-inch LCD screens and Logitech webcams with integrated microphones, as shown in Figure 10. The SMI REDn Scientific eye tracker is attached under the display as shown in Figure 11.

Group experiments can be conducted using another workstation administrated by the educator. This setup enables recording of up to 20 subjects at once and tracking eye movements when participants are working at the workstations, for instance, pupils during class.



Figure 7: Master Workstation.



Figure 8: 20 Equipped Workstations with SMI REDn Scientific Eye Trackers.

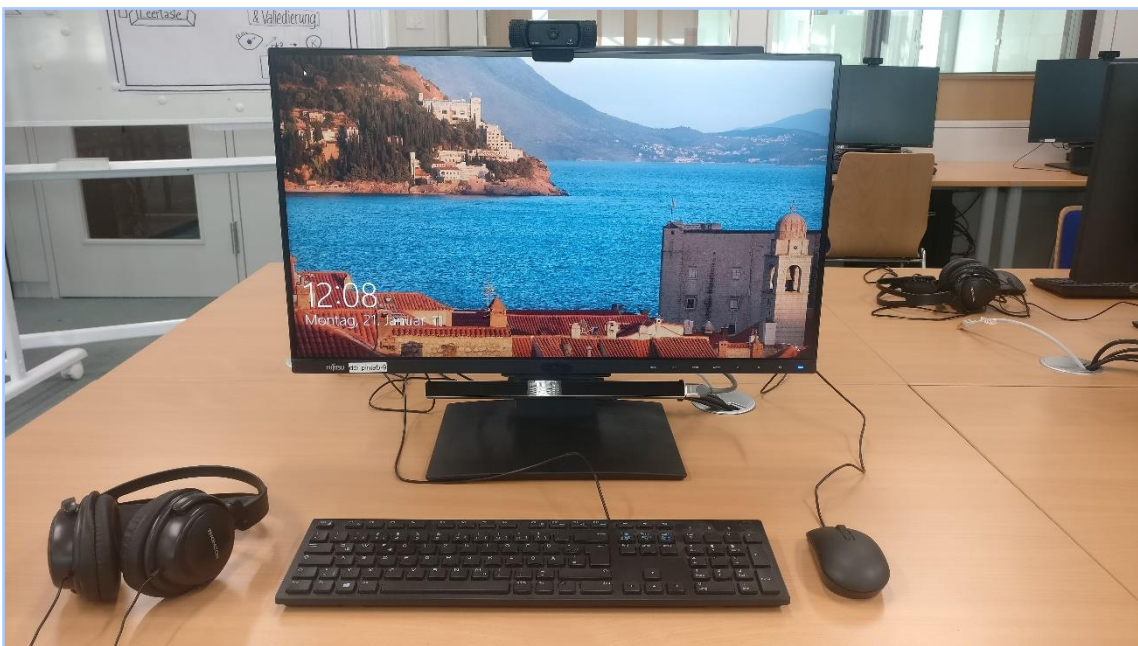


Figure 9: Workstation.





Figure 10: Logitech Camera with Integrated Microphone.



Figure 11: SMI REDn Scientific Eye Tracker.

## Example

In this subsection the example used in the intervention video is shown. This example was taken from the EMIP experiment (Bednarik, et al., 2018). This program was originally used in the previous experiment as the other program apart from the Rectangle program, as we decided to use List program instead, we use Vehicle for the explanation part. We decided not to use Vehicle program since the variables in the code give a lot of information about what the program does and instead of renaming them all, we decided to use another program code. In Table 7 below the code is shown.

Table 7: Vehicle Program.

Vehicle
<pre> public class Vehicle {     String producer, type;     int topSpeed, currentSpeed;      public Vehicle (String p, String t, int tp){         this.producer = p;         this.type = t;         this.topSpeed = tp;         this.currentSpeed = 0;     }      public int accelerate (int kmh) {         if ((this.currentSpeed + kmh) &gt; this.topSpeed){             this.currentSpeed = this.topSpeed;         } else {             this.currentSpeed = this.currentSpeed + kmh;         }         return this.currentSpeed;     }      public static void main ( String args [] ) {         Vehicle v = new Vehicle ("Audi", "A6", 200);         v.accelerate(10);     } } </pre>

## Results of the Experiment

In this section the results of the experiment are explained. First thoughts were to compare the outcomes from this experiment with the ones in the previous study, but due to time constrictions the results are going to be presented with its own conclusions.

As in the “Design” section above described, there are two different versions of the experiment. The List-Rectangle version counted with 14 participants and the Rectangle-List counted with 13 participants. Even though is almost the same amount, the percentages of correct answers of the participants in each version are calculated to make sure that the conclusions are based in the correct results.

In Table 8 below the number of correct answers in general and both versions of the experiment are shown. The first thing noticed is that the program with the best answers is the Rectangle program, which can lead us to think that the List program is more difficult than the Rectangle program, in which the participants haven’t had such difficulties to choose the correct answer like in the other one. It is noticed as well that the worst question answered was the third one in the List program and the best question answered was the second in the Rectangle program.

If we compare the correct answers in the List program, we can notice that the participants did it lightly better in the second version, when it was placed second. Comparing the Rectangle program, it is lightly better in the second version as well. In the Rectangle program, it happens the same, in the second version the participants did it better. We can conclude that the participants did it better in the second version, since they reached more correct answers.

Table 8: Correct Answers.

	Program	Q1	Q2	Q3	TOTAL	%
<b>General</b>	List	16	10	7	33	40'74
	Rectangle	22	24	13	59	72'83
<b>List-Rectangle</b>	List	8	5	2	15	35'71
	Rectangle	12	12	6	30	71'43
<b>Rectangle-List</b>	List	8	5	5	18	46'15
	Rectangle	10	12	7	29	74'36

In Table 9, the mean length is shown. It is noticed that, in general, it took longer to the participants to read the List program, 1' 32" more than to read the Rectangle program. When we compare both versions, we appreciate that the participants in the first version spent less time reading the program than in the second version. In the List program in the second version, the length is much higher if we compare with the first version, and this could be the reason why these participants reached more correct answers (18 in second version versus 15 in first version).

Table 9: Length.

Program	General	List-Rectangle	Rectangle-List
List	3' 17"	2' 22"	4' 15"
Rectangle	1' 44"	1' 37"	1' 52"

In Table 10, the number of fixations is shown. The fixation is the settling of the eye gaze on an object of interest for a minimum period of time (Busjahn, et al., ACM, 2014). The number of fixations is directly related to the effort that the participant put. The interpretation of visual effort variable can be simplified as follows:

- Low fixation count and low time indicate less effort (Sharafi, Soh, Gueheneuc, & Antoniol, 2012) (Sharafi, Marchetto, Susi, Antoniol, & Gueheneuc, 2013).
- High fixation count and more time indicate more effort (Turner, Falcone, Sharif, & Lazar, ACM, 2014) (Sharif & Maletic, 2010).

If we compare in general, the List program counts with a higher number of fixations than the Rectangle program, which indicates the participants needed more effort to understand the program. When comparing the first version and the second, we can appreciate that the second version has a higher number of fixations and the List program counts with almost the double comparing to the first version and to the Rectangle program. This indicates that the participants put more effort in this part, in which the correct answers are higher than in List program in the first version.

Table 10: Number of Fixations.

Program	General	List-Rectangle	Rectangle-List
List	628'19	465'21	803
Rectangle	420	366'36	479'54

The saccades are a quick movement of the eyes from one location to another (Busjahn, et al., ACM, 2014). In Table 11, we appreciate that comparing between both versions, the number of saccades in the second are the around the double than in the first version. I could not find any study explaining what these values mean, but probably has something to do about the comprehension and the number of correct answers as well, since this version has more correct answers.

Table 11: Number of Saccades.

Program	General	List-Rectangle	Rectangle-List
List	526'85	368'71	697'15
Rectangle	362'15	298	431'23

About the programming expertise of the participants, there are four different levels, as seen in Table 12. The most noticeable fact is that the participants with a high level of expertise did not reach the most correct answers, while the participants with a medium level did. Apart from this, the most correct answers in the Rectangle program were acquired by the participants with a low programming expertise level, while in List program were the participants with a medium programming expertise.

In the attached "Demographic Questions" excel file, it can be found the respective tables of each version about programming expertise. There are not shown in this document since the information that they give is not differing much from the one presented in the table below.

Table 12: Programming Expertise.

Expertise	Part.	List			Rectangle			Total	%		
		Q1	Q2	Q3	Q1	Q2	Q3		List	Rect	Total
None	6	1	1	0	5	4	2	13	11'12	61'12	36'11
Low	2	0	1	1	2	2	1	7	33'3	83'34	58'34
Medium	12	10	6	4	11	11	6	48	55'56	77'78	66'67
High	7	5	2	2	4	7	4	24	42'86	71'43	57'14

About the level of java expertise, as shown in Table 13, participants with a high java expertise level as well as participants with a medium level, had the most correct answers. In this case, when we investigate each program, in the Rectangle program the most correct answers were reached by the participants with a high level, while in List program the most correct answers were reached by the participants with a low level. It is possible that in List program, the participants with a high java expertise level, got confused by the multiple-choice questions and could not find the correct answers.

Table 13: Java Expertise.

Expertise	Part.	List			Rectangle			Total	%		
		Q1	Q2	Q3	Q1	Q2	Q3		List	Rect	Total
None	9	3	3	2	8	7	4	27	29'63	70'37	50
Low	3	2	1	1	3	3	0	10	44'45	66'67	55'56
Medium	9	7	4	3	7	8	4	33	25'93	70'37	61'12
High	6	4	2	1	4	6	5	22	38'89	83'34	61'12

As the experiment counts with a similar number of females and males, I found interesting to know whether one gender did better than the other one. Fortunately, the differences between one gender and another are not significant, reaching most correct answers the male participants than the females, but only 3,4% of correct answers more.

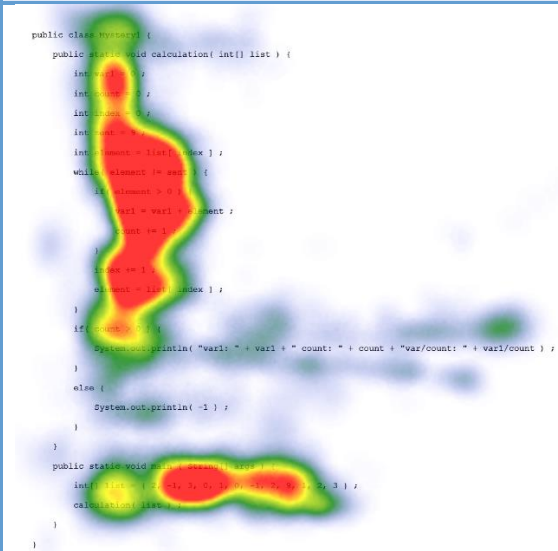

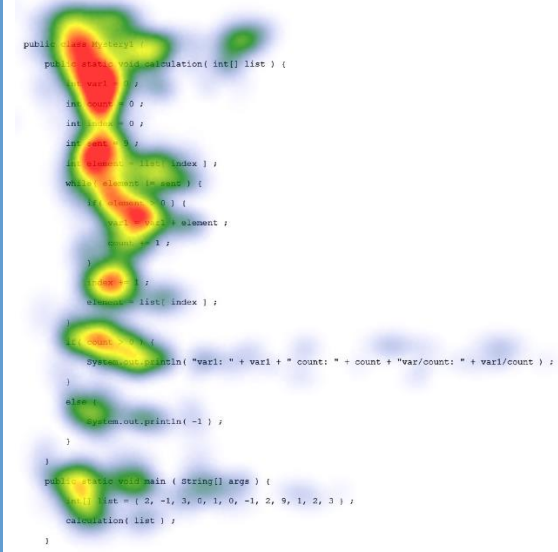
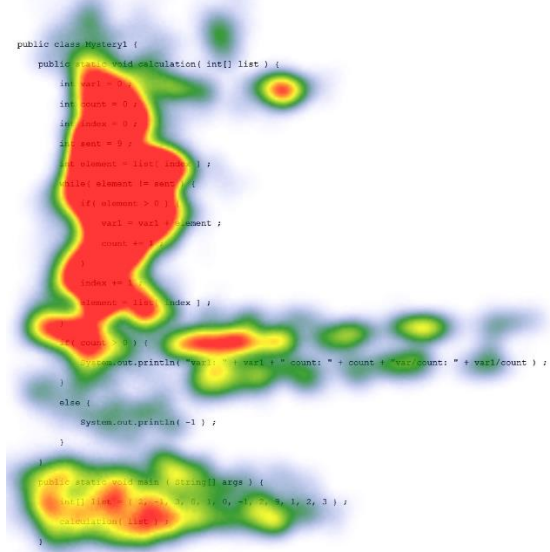
Table 14: Gender.

Expertise	Part.	List			Rectangle			Total	%		
		Q1	Q2	Q3	Q1	Q2	Q3		List	Rect	Total
Female	13	9	3	2	10	12	5	41	35'9	69'23	52'56
Male	14	7	3	5	12	12	8	47	35,7	76,19	55,96

In Table 15 and Table 16, the heat maps of both program codes are shown. The first thing that we notice is the red areas, which are bigger in the List program than in the Rectangle program, which means that the List program needed more attention than the Rectangle program. If we observe the Rectangle – List Spanish heat map, we notice that this part counts with the biggest red area, which explains the high values of the number of fixations above mentioned.



Table 15: List Heat Maps.



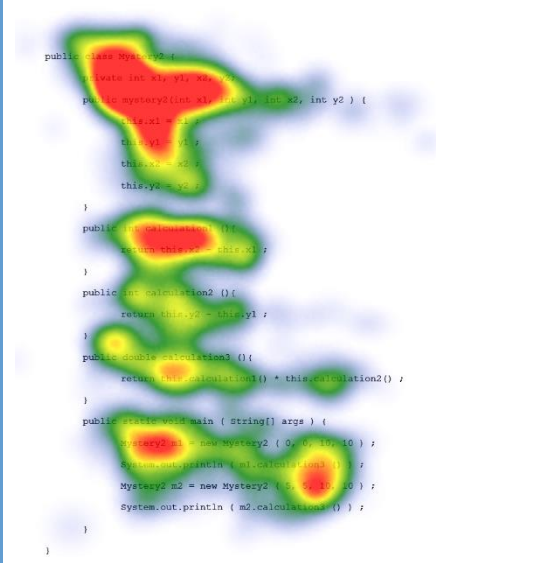
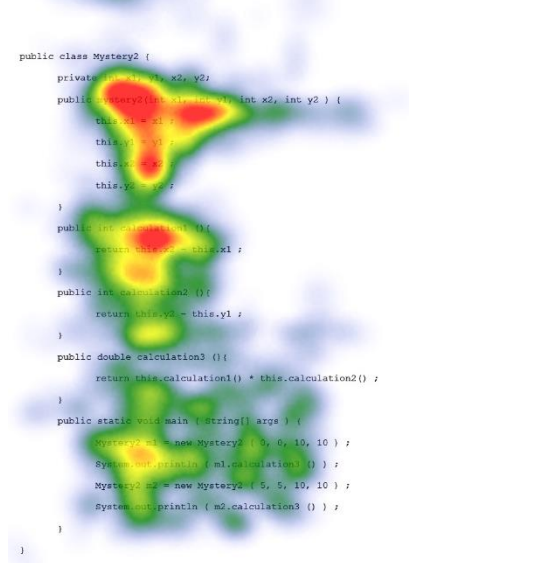
List Heat Maps	
List – Rectangle English	Rectangle – List English
 <pre> public class Mystery1 {     public static void calculation( int[] list ) {         int var1 = 0 ;         int count = 0 ;         int index = 0 ;         int count = 0 ;         int element = list[ index ] ;         while( element != sent ) {             if( element &gt; 0 ) {                 var1 = var1 + element ;                 count = count + 1 ;             }             index = index + 1 ;             element = list[ index ] ;         }         if( count &gt; 0 ) {             System.out.println( "var1: " + var1 + " count: " + count + " var/count: " + var1/count ) ;         }         else {             System.out.println( -1 ) ;         }     } }  public static void main( String[] args ) {     int[] list = { 2, -1, 3, 0, 1, 0, -1, 2, 9, 1, 2, 3 } ;     calculation( list ) ; } </pre>	 <pre> public class Mystery1 {     public static void calculation( int[] list ) {         int var1 = 0 ;         int count = 0 ;         int index = 0 ;         int count = 0 ;         int element = list[ index ] ;         while( element != sent ) {             if( element &gt; 0 ) {                 var1 = var1 + element ;                 count = count + 1 ;             }             index = index + 1 ;             element = list[ index ] ;         }         if( count &gt; 0 ) {             System.out.println( "var1: " + var1 + " count: " + count + " var/count: " + var1/count ) ;         }         else {             System.out.println( -1 ) ;         }     } }  public static void main( String[] args ) {     int[] list = { 2, -1, 3, 0, 1, 0, -1, 2, 9, 1, 2, 3 } ;     calculation( list ) ; } </pre>
List – Rectangle Spanish	Rectangle – List Spanish
 <pre> public class Mystery1 {     public static void calculation( int[] list ) {         int var1 = 0 ;         int count = 0 ;         int index = 0 ;         int count = 0 ;         int element = list[ index ] ;         while( element != sent ) {             if( element &gt; 0 ) {                 var1 = var1 + element ;                 count = count + 1 ;             }             index = index + 1 ;             element = list[ index ] ;         }         if( count &gt; 0 ) {             System.out.println( "var1: " + var1 + " count: " + count + " var/count: " + var1/count ) ;         }         else {             System.out.println( -1 ) ;         }     } }  public static void main( String[] args ) {     int[] list = { 2, -1, 3, 0, 1, 0, -1, 2, 9, 1, 2, 3 } ;     calculation( list ) ; } </pre>	 <pre> public class Mystery1 {     public static void calculation( int[] list ) {         int var1 = 0 ;         int count = 0 ;         int index = 0 ;         int count = 0 ;         int element = list[ index ] ;         while( element != sent ) {             if( element &gt; 0 ) {                 var1 = var1 + element ;                 count = count + 1 ;             }             index = index + 1 ;             element = list[ index ] ;         }         if( count &gt; 0 ) {             System.out.println( "var1: " + var1 + " count: " + count + " var/count: " + var1/count ) ;         }         else {             System.out.println( -1 ) ;         }     } }  public static void main( String[] args ) {     int[] list = { 2, -1, 3, 0, 1, 0, -1, 2, 9, 1, 2, 3 } ;     calculation( list ) ; } </pre>

In Table 16, we observe that the less highlighted parts are in the English version. This means that the participants needed less effort to understand the program, which is possible since all participants in the English version were Computer Science students who had completed their bachelor studies. On the other hand, in the Spanish version, the red areas are bigger since the participants of this version were not students in the field of Computer Science.

The participants from the Rectangle – List version were Engineering students, but the ones from the List – Rectangle version were Business students, which explains that they put more effort to understand this program. If we observe the Table 15 above, we can notice that these participants, the ones that do not know anything about program code, present the smallest heat

map since List program is harder than Rectangle program, where these participants put more effort and reached more correct answers.

Table 16: Rectangle Heat Maps.

Rectangle Heat Maps	
List – Rectangle English	Rectangle – List English
 <pre> public class Mystery2 {     private int x1, y1, x2, y2;     public Mystery2(int x1, int y1, int x2, int y2) {         this.x1 = x1;         this.y1 = y1;         this.x2 = x2;         this.y2 = y2;     }     public int calculation1 () {         return this.x2 - this.x1;     }     public int calculation2 () {         return this.y2 - this.y1;     }     public double calculation3 () {         return this.calculation1() * this.calculation2();     }     public static void main ( String[] args ) {         Mystery2 m1 = new Mystery2 ( 0, 0, 10, 10 );         System.out.println ( m1.calculation1 () );         Mystery2 m2 = new Mystery2 ( 5, 5, 10, 10 );         System.out.println ( m2.calculation3 () );     } }                     </pre>	 <pre> public class Mystery2 {     private int x1, y1, x2, y2;     public Mystery2(int x1, int y1, int x2, int y2) {         this.x1 = x1;         this.y1 = y1;         this.x2 = x2;         this.y2 = y2;     }     public int calculation1 () {         return this.x2 - this.x1;     }     public int calculation2 () {         return this.y2 - this.y1;     }     public double calculation3 () {         return this.calculation1() * this.calculation2();     }     public static void main ( String[] args ) {         Mystery2 m1 = new Mystery2 ( 0, 0, 10, 10 );         System.out.println ( m1.calculation1 () );         Mystery2 m2 = new Mystery2 ( 5, 5, 10, 10 );         System.out.println ( m2.calculation3 () );     } }                     </pre>
List – Rectangle Spanish	Rectangle – List Spanish
 <pre> public class Mystery2 {     private int x1, y1, x2, y2;     public Mystery2(int x1, int y1, int x2, int y2) {         this.x1 = x1;         this.y1 = y1;         this.x2 = x2;         this.y2 = y2;     }     public int calculation1 () {         return this.x2 - this.x1;     }     public int calculation2 () {         return this.y2 - this.y1;     }     public double calculation3 () {         return this.calculation1() * this.calculation2();     }     public static void main ( String[] args ) {         Mystery2 m1 = new Mystery2 ( 0, 0, 10, 10 );         System.out.println ( m1.calculation1 () );         Mystery2 m2 = new Mystery2 ( 5, 5, 10, 10 );         System.out.println ( m2.calculation3 () );     } }                     </pre>	 <pre> public class Mystery2 {     private int x1, y1, x2, y2;     public Mystery2(int x1, int y1, int x2, int y2) {         this.x1 = x1;         this.y1 = y1;         this.x2 = x2;         this.y2 = y2;     }     public int calculation1 () {         return this.x2 - this.x1;     }     public int calculation2 () {         return this.y2 - this.y1;     }     public double calculation3 () {         return this.calculation1() * this.calculation2();     }     public static void main ( String[] args ) {         Mystery2 m1 = new Mystery2 ( 0, 0, 10, 10 );         System.out.println ( m1.calculation1 () );         Mystery2 m2 = new Mystery2 ( 5, 5, 10, 10 );         System.out.println ( m2.calculation3 () );     } }                     </pre>

## Conclusions

---

There are general as well as particular conclusions that are found during this process. The experiment took time to be design in order to acquire the best solution possible but, after taking place and while doing it, I noticed some details that I have not noticed when designing it. First, I think that the codes that we used are not the best desired, because List program is too difficult when it comes to novice programmers, and Rectangle program is maybe too easy for them. I think Rectangle program is more accurate because some of the participants, that did not have any experience programming, understood the purpose of the program. But when it comes to List program, most of the participants did not comprehend it. In my opinion, the programs should have been chosen in a way that none of them are too easy or too difficult for the participant.

The participants that took the experiment are not the best desired, since almost all of them were not novice students. In my opinion, the best participants for this study would have been students that have started the bachelor last semester (October 2018), so they know a little about programming, but they are not experienced programmers yet. As the experiment took part in March, when there were no classes, we did not have that plenty of participants as if it took place another month. Apart from that, we found people who did not have any experience (because their study field is not Computer Science or similar) and some people who had finished their bachelor studies but are not experts yet.

Considering this, in Table 17, the total of correct answers, number of fixations and length are shown. In general, in List program it took longer and more fixations to reach less correct answers. As explained before when investigating the number of fixations, this means that the participants put more effort to understand the program and the low quantity of correct answers indicates two things: the first one is that the program was difficult for them; the second is that it is possible that the questions were not designed properly and it resulted being confusing for the participants. On the other hand, participants in Rectangle program needed less time and less fixations to reach more correct answers if we compare with List, which means that this program is easier for the participants to understand.

If we compare the first program in each version, List in first and Rectangle in second, we notice that the number of fixations is the almost the same, and the gap between the length is the smallest if we compare with the rest. This could mean that independently which program was first, the participants put the same effort to understand the program, while their behaviour changed when they started reading the second program. In the first version, as the Rectangle program is no as hard as the List program, the participants needed less fixations and less time to understand it. On the other hand, in the second version, when they changed to List program, is when they could realize that the program is harder to understand and that is the reason the number of fixations and the length is higher. If we had one more program, we could prove whether the effort that the participants put is the same independently of which program is first or is only coincidence.

Table 17: Conclusions.

	Program	Total	Fixations	Length
<b>General</b>	List	33	628'19	3' 17"
	Rectangle	59	420	1' 44"
<b>List-Rectangle</b>	List	15	465'21	2' 22"
	Rectangle	30	366'36	1' 37"
<b>Rectangle-List</b>	List	18	803	4' 15"
	Rectangle	29	479,54	1' 52"

## Discussion and Follow up Study Ideas

In several parts of this thesis we had a lot of choices to choose and in this section the reasons of the choices taken that do not appear in the rest of the paper are going to be explained here. Most of them were taken in the intervention part, since is the hardest part of the thesis, because we wanted the questions and the process to be as perfect and useful as possible. Although the intervention part is the most complex, there are other questions in the thesis in which we had to decide as well.

Some of the decisions that we made sometimes are not the best solution that we found, but it is because of the limitation of the thesis, since it is a Bachelor Thesis and we had not enough time to include them, but they ended up being some ideas to a future work in this field. Nevertheless, along this paper and in this section, the reasons and the thoughts that we had during these months are explained.

When designing the intervention questions about the codes showed, the first thing that we discussed about was the order of the questions. As mention before in this paper, we added a new free text question. Since we wanted the intervention to be as similar as the one in the EMIP dataset, we thought to place it as the last question. However, I find this fact not as helpful as if it was in the first place because, as being in the last place, the participant could base his/her answers in the free text part in the answer of the previous multiple-choice questions, while being the first part to answer, would really let us know what are their first thoughts about the code. One of the participants wrote "If I had to answer this question just after reading the code (without answering other questions), probably I wouldn't be able to write anything. However, after the last question, is it true that it can be the calculation of some geometric figure.", this quote induces that by answering this question in the last position, the answer may be influenced by the previous questions.

About the fact that first we show the code and then, the multiple-choice questions are asked to the participants, I found that it is more realistic to see the questions and the code next to them, since in real life there are not scenarios where the code cannot be seen when asking questions about it. Normally it is not necessary, and it is useless to remember (or study) the code and then

work on it, because it is common to have to have access to it and see it while working. This could be an interesting position to a next study, due to that it could be more realistic and maybe the level of comprehension could be reached better than we do here, since sometimes a participant maybe does not know the answer not because he does not understand the code, but he/she did not remember it. At the end, we decided to maintain the process as the last intervention, because if not, I would not be able to compare the results obtained in this intervention with the previous ones as thought in first place.

As we used Java language to proceed with the intervention, the complete code was in one class where all parts could be read at the same time. But in Java, normally the main method is in a class by its own, but in this case, it is in the same class as the methods `mystery1` and `mystery2` (List and Rectangle program respectively) in each. We put the main method in the lowest part of the text, but sometimes it is placed just after the construction method, so it is interesting to know if this is maybe distracting for some students that are used to see it in the upper part. Other reason about putting it into the lowest part it is because we took the `mystery1` (List program) from (Duran, Sorva, & Leite, 2018), and we did not want to change it since it was in Python and we already changed it to Java.

Following with the ideas that came up when thinking about the code, I found a possible misunderstanding in the intervention. In the expert's explanation, the hardest part of the code is accidentally the biggest block of the code, which can lead to a quick shortcut that the biggest part is the hardest, and that is not true. This is not explicitly explained in the intervention, so this possible misconception could happen. This fact could appear because when hearing the explanation, the participant is focused on what the expert is saying, and he/she is not reading carefully inside the block highlights but just noticing them. Due to this, maybe it would be helpful to have the real expert's gaze, so the participant focusses on the time aspect (spending more time on the hardest blocks) and not on the block length like in this intervention or make sure to explain carefully this fact in the verbal explanation. In a next study, gaze, pictures instead of video, only voice (just hearing without stimuli) and see the speaker as in YouTube lectures could be interesting to investigate and what the participant is looking while the explanation as well as adding a question about where the hardest part of the code is.

About the level of comprehension that the participants reach, I wonder if we are able to know that the participant understood the programs or not. We supposed that if the participant reached all correct answers, he/she had understood the program. But three of the participants did not reach all correct answers but then, in the free text answer, they explained the programs successfully. Because of this, we should not think that the participant only understood the program if they had all correct answers. This leads me to think again that maybe the questions were not design as clear as we wanted to, since this fact demonstrates that some of the participants got confused by the different options in the multiple-choice questions.

Experiment Center does not give the possibility to change the format inside the questions, and we would like to change the format of the code in them, so it would be easier to distinguish between code parts and normal text parts.

BeGaze does not open different experiments at the same time, so we couldn't compare inside the program the XY experiment in English version with the XY experiment in Spanish version, even though they are the same experiment but different language; this happens also vice versa.

About the voices of the intervention, both interventions, English and Spanish, were recorded by a male voice. It would be interesting to study whether the participants reach better level comprehensions when listening a female voice instead.

## References

---

- Ali, N., Sharafi, Z., Guegeneuc, Y., & Antoniol, G. (IEEE, 2012). An empirical study on requirements traceability using eye tracking. *In Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM).*, 191-200.
- Bednarik, R., Budde, L., Heinemann, B., Schulte, C., & Vrzakova, H. (2018, November 18th). Eye-movement Modeling Examples in Source Code Comprehension: A Classroom Study. *In 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18)*, 22-25. Retrieved from <https://doi.org/10.1145/3279720.3279722>
- Bednarik, R., Busiahn, T., Gibaldi, A., Sharif, B., Bielikova, M., & Tvarozek, J. (2018). The EMIP dataset. *Technical Report*. Retrieved from [http://emipws.org/emip\\_dataset/](http://emipws.org/emip_dataset/)
- Beelders, T., & du Plessis, J.-P. (2016). The influence of syntax highlighting on scanning and reading behaviour for source code. *In Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 5.
- Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, 18, 543-554.
- Busjahn, T., Bednarik, R., & Schulte, C. (2014). What influences dwell time during source code reading?: analysis of element type and frequency as factors. *In Proceedings of the Symposium on Eye Tracking Research and Applications.*, 335-338.
- Busjahn, T., Bednarik, R., Begel, A., Crosby, M., Paterson, J. H., Schulte, C., . . . Tamm, S. (2015). Eye Movements in Code Reading: Relaxing the Linear Order. *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension (ICPC '15)*, 255-265. Retrieved from <https://dl.acm.org/citation.cfm?id=2820282.2820320>
- Busjahn, T., Schulte, C., Sharif, B., Begel, A., Hansen, M., Bednarik, R., . . . Antropova, M. (ACM, 2014). Eye tracking in computing education. *In Proceedings of the 10th Annual Conference on International Computing Education Research*, 3-10.
- Collins, A., & Newman, S. (1989). Cognitive apprenticeship: teaching the craft of reading, writing and mathematics. *In L. B. Resnick (Ed.), Cognition and instruction: Issues and agendas*, 453-494.
- Duran, R., Sorva, J., & Leite, S. (2018). Towards an Analysis of Program Complexity From a Cognitive Perspective. *In Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER '18).*, 21-30. doi:<https://doi.org/10.1145/3230977.3230986>
- Hernán-Losada, I., Velázquez-Iturbide, J., & Lázaro, C. (2019). *Dos herramientas educativas para el aprendizaje de programación: generación de comentarios y creación de objetos.*



- Jarodzka, H., Van Gog, T., Dorr, M., Scheiter, K., & Gerjets, P. (2013). Learning to see: Guiding students' attention via a Model's eye movements fosters learning. *Learning and Instruction, 25*, 62-70. Retrieved from <http://doi.org/10.1016/j.learninstruc.2012.11.004>
- Kalyuga, S., Chandler, P., & Sweller, J. (1999). Managing split-attention and redundancy in multimedia instruction. *Applied Cognitive Psychology, 13*, 351-371.
- Lister, R., Fidge, C., & Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *SIGCSE, 41*(3), 161-165.
- Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology, 29*, 295-341.
- Schlichtig, M. (2018). *Infrastructure Conception for Evaluation of Interactive Tutorials in the Context of a Jupyter Notebook Data Science Course*. M.Sc. Thesis, Universität Paderborn, Paderborn, Germany.
- Schulte, C. (2008). Block Model: An Educational Model of Program Comprehension As a Tool for a Scholarly Approach to Teaching. In *Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08)*, 149-160. Retrieved from <https://doi.org/10.1145/1404520.1404535>
- Sharafi, Z., Marchetto, A., Susi, A., Antoniol, G., & Gueheneuc, Y. (2013, May). An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension. In *Proceedings of the 2013 IEEE 21st International Conference on Program Comprehension (ICPC)*, 33-42.
- Sharafi, Z., Soh, Z., Gueheneuc, Y., & Antoniol, G. (2012, June). Woman and men- Different but equal: On the impact of identifier style on source code reading. In *Proceedings of the 2012 IEEE 20th International Conference on Program Comprehension (ICPC)*, 27-36.
- Sharif, B., & Maletic, J. (2010, June). An eye-tracking study on camel-case and under score identifier styles. In *Proceedings of the 2010 IEEE 18th International Conference on Program Comprehension (ICPC)*, 196-205.
- Shneiderman, B., & Mayer, R. (1979). Syntactic/semantic interactions in programmer behaviour: A model and experimental results. *International Journal of Computer and Information Sciences.*, 8(3), 219-238.
- Soloway, E., & Ehrlich, K. (1984, September). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering, 10*(5), 595-609.
- Sweller, J. (2005). The redundancy principle in multimedia learning. In R. E. Mayer (Ed.), *The Cambridge handbook of multimedia learning.*, 159-167.



- Tim van Marlen, M. v. (2016, December). Showing a model's eye movements in examples does not improve learning of problem-solving tasks. *Computers in Human Behaviour*, 448-459.
- Turner, R., Falcone, M., Sharif, B., & Lazar, A. (ACM, 2014). An eyetracking study assessing the comprehension of C++ and Python source code. *In Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '14)*, 231-234.

# Appendix I: Free Text Answers

## English

### (List-Rectangle) version

Name	Mystery1 (List)	Mystery2 (Rectangle)
P01	Main program calls function by sending a list of integers. In the function the variables are initialized to 0 It calculates the 2 variables var1 and count. It sums all the positive integers in the list. Sum is added to var1 and number of integers added are stored in count. Until a number which is greater than 9 is found, loop gets broken and if the count value is more than 0 it prints var1, count and var1/count or it prints -1.	The main function initializes Multiplication class with the integers x1, y1, x2, y2. Then it calls the function calculation3 which inturn calls calculation1 and calculation2 such that x2-x1 and y2-y1 is calculated respectively. Finally, calculation3 multiplies the results and print the results as double.
P01(1) (excluded)	The program code sums the numbers in the list and divides it until a number greater than 9 is obtained.	The program calculates the area of the square by multiplying its sides which are obtained by the calculation functions. Once the side is known by the calculation function area is calculated by the other functions there by generating the area at the end.
P02	It calculates the sum, counts the elements in the array and also calculates the average.	It gets the border points of a rectangle and can calculate the width (calc1), height (calc2) and size (calc3)
P03 only reached Q1	<b>Add all numbers greater than zero from an array until a 9 is reached. When a 9 is reached, the while loop breaks. Then the program prints the sum of the numbers greater than zero and then the count of all the numbers greater than zero (until the 9 is reached, of course!) and the result of sum/count.</b>	Given 2 points represented in 2D, this program calculates (difference between the x co-ordinates) * (difference between the y co-ordinates)
P01(2)	The program reads the numbers entered in an array and keeps incrementing the value of val1 by adding elements in the array list and this will happen until the value of val1 becomes equal to or greater than 9. And in every iteration the count value gets incremented inside the while loop. And once the value of	In this program we actually calculate the area of a rectangle where its x1, x2 and y1, y2 coordinates are entered. The difference between x coordinates (x2-x1) will give the length and the difference between the y coordinates (y2-y1) will give us the height of the rectangle. And the product of length

	<p>val1 becomes greater than or equal to 9. The control comes out of the while loop and if the count value is greater than zero, it will print the values of 3 variables they are val1, count and val1/count. The program returns -1 if count value is 0.</p>	<p>and height will return the area of the rectangle.</p>
P01(3)	<p>The main method creates a list and calls the method "calculate" with the list as an argument. The Method calculate sums up all positive integers (&gt;0) and keeps track of how many it has counted. Then it prints out the sum, the amount of positive integers and the average of those.</p>	<p>Main Method creates objects that hold 4 integers (x1, x2, y1, y2) Then it prints out the result of the calculation (x2-x1)*(y2-y1). (x2-x1) is the result of calc(1) and (y2-y1) is the result of calc2. Calc3 multiplies these two results. It could happen that you get a negative "surface area".</p>
P04 Reached All	<p><b>This program takes a list of integers and find the sum of the natural numbers (numbers greater than 0) until the number 9 is encountered in the list (9 is excluded from the sum). It also finds the count of the natural numbers until the number 9 is encountered.</b></p>	<p>This program takes the positions of two lines (possibly 2 lines that forms the side of a rectangle) and find the area. For the positions x1, x2, y1, y2, it finds (x2-x1)*(y2-y1)</p>
P02(1)	<p>It takes a list as an input and add the numbers if greater than zero till you find 9 in last you print the count, total, total/count.</p>	<p>Takes two coordinates and finds (x2-x1)*(y2-y1) and prints it</p>
P02(2)	<p>The program prints the variable, count and the sum/count value of the list unless a 9 is found.</p>	<p>The program contains a calculation3 method. This method takes the returned value of calculation1 and calculation2 method as the parameters and then computes their product. Calculation1 and calculation2 methods take two variables as parameters and then returns their difference. The points (x1, y1) and (x2, y2) are the coordinates of two points. Calculation1 and calculation2 calculates (x2-x1) and (y2-y1) respectively. Calculation3 computes (x2-x1)*(y2-y1) which is the magnitude of the vector.</p>

## Rectangle-List version

Name	Mystery1 (List)	Mystery2 (Rectangle)
P01	<p>The program contains a list and variables like var1, count, index initialized. The main logic says that if the element &gt; 0, then var1+element else print -1 (also the element should not be equal to 9 which is also already initialized). The sum of the numbers greater than 0 and less than 9 are taken and divided by the count.</p>	<p>There are 4 variables x1, y1, x2, y2, and they are initialized using this variable. The program contains 3 functions calculation1(), calculation2() and calculation3(). Calculation1() returns the difference between x2 and x1. calculation2() returns the difference between y2 and y1. calculation3() is returns the product of calculation1() and calculation2() with respect to the parameters passed in the main function to the newMystery function.</p>
P01(1)	<p>A list is sent to function calculation. Variables are assigned and while loop with a condition is given. Till the loop satisfy the condition, given operation are performed. And output is printed.</p>	<p>Initially all variables are initialized. Three functions named calculation1, calculation2 and calculation3 are defined. In the main function the actual values are passed to the function and answer is calculated and returned. Answer for the 1st is 100 and for 2nd is 25.</p>
P02 n/a in Q1 reached the rest	<p><b>This program sums up all non-negative numbers from a list until reading a specified value. It also counts the number of those values that were added to the sum. Then it gives the sum, the counter and sum divided by counter as an output.</b></p>	<p>The first function calculates the difference of two points in x coordinates, the second one the one in y coordinates. Those two functions are then used in the third one to calculate the area of a square with lengths from both functions.</p>
P02(1)	<p>In this static program a list with int elements is created in the main method and afterwards a calculation with the elements in this list is done (all static). The calculation sum up all elements in the list, starting at the beginning with index 0, until an element with the value 9 is reached in the list. Then the while loop ends and if the sum is greater than 0, an output via the command line is produced and it shows the results.</p>	<p>It is a class with three int variables called x1, x2, y1, y2. They are all set in the constructor when creating an object of this kind. The class also has three methods. The first one calculates and returns the following (x2-x1). The second method calculates (y2-y1). And the third one calculates the result of method 1 * the result of method 2, so (x2-x1)*(y2-y1). There is also the main method to start the program. Here, two objects of the class are created. The values of the first object are set (by the constructor) to x1=0, x2=10, y1=0, y2=10. After that, method 3 is called, so (10-0)*(10-0)=100 is calculated. Afterwards, the second object is created. The values of the second object are set (by the constructor) to x1=5, x2=10, y1=5, y2=10. After that,</p>

		method 3 is called, so $(10-5)*(10-5)=25$ is calculated.
<b>P01(2)</b>	We pass a list of integers to the program, it sums up the list elements until encounters a number >9. The total value is divided by the number of elements counted. The result after the division is printed by the main function.	In the program, the object mystery takes 4 arguments. It sends 2 x coordinates to one function calculation1(). It sends 2 Y coordinates to calculation2(). The result of the functions calculation1() and calculation2() are then multiplied. The result is then returned to the main function, where it is sent to the output screen to be printed.
<b>P01(3) Reached All</b>	<b>The program takes a list as input and calculates the sum of all the numbers that are greater than 0 before encountering 9 in the list. It calculates the count of such numbers and the sum of the numbers greater than 0 is divided by the count of such numbers.</b>	The program code takes the value of x1, y1, x2 and y2. Calculate1() calculates the value $x2-x1$ . Calculate2() calculates $y2-y1$ . Calculate3 function returns $(calculate1()*calculate2())$ . For example: if x1, x2, y1 and y2 are considered as coordinated of a rectangle, the calculate3() will return the area as $calculate1()*calculate2()$ .
<b>P02(2)</b>	This program contains a list in which the no which is greater than zero is only taken into account and while the counter goes to 9 its checking the list and after that it display the result.	The program contains the three methods in which the last methods have input from other two methods. This program deals with two objects of class and implement the result.

## Spanish

## List-Rectangle version

Name	Mystery1 (List)	Mystery2 (Rectangle)
P02(2)	The operation of the program goes in blocks, which have different lengths depending on the difficulty they have and the need to explain it. In addition these blocks can be divided into sub-blocks and do not have to be in order.	The functioning of the program was divided into several parts. There were four variables (X1; X2, Y1; Y2) and it asked to do the subtraction $X2-x1$ and the subtraction $Y2-Y1$ ; and then multiply the result of both previous operations.
P03	I haven't finished understanding how the program works, I imagine it describes the work done by a printer.	If you had to answer this question after reading the code (without answering other questions), you probably wouldn't have been able to write anything. However, after the last question, it is true that it could be the calculation of the area of some geometric figure.
P04	I'm not quite sure how the program works. The program may add several numbers, use the variance, the total number of numbers, and finally divide the result.	This program calculates the difference between X1 and X3, subtracting the first from the second. It then calculates the difference between Y1 and Y2 in the same way. The final step is double calculation, which consists of multiplying both results. If we imagine a square, multiplying the distances of the base and height we obtain the area.
P03(1)	The program describes a problem and defines the different possibilities of result. In this way, patterns for statistics can be described.	The program gives the points where the vertices of the square are located and, following the calculations, you can find the area of the figure.

## Rectangle-List version

Name	Mystery1 (List)	Mystery2 (Rectangle)
P01	The program sums until it finds a 9.	Calculates the difference between the positions of the vertices, obtaining the sides, and then multiplies them, obtaining the area.
P01(1) Reach Q1 n/a Q2 not Q3	<b>The program receives a list of numbers and passes them on to a function. This function initializes some parameters, to then go through the list of numbers until it finds a number nine, each number of the list traveled different from nine adds one to the counter variable and another variable increases its value with the sum of the values of the list traveled, at the end the program prints in screen the sum of the numbers, the quantity of numbers of the list traveled and the ratio between the sum of the numbers and the numbers traveled.</b>	The program receives data passed by code and makes the calculations specified in the functions, in the example it receives two different values and displays both calculations. The calculations are the multiplication of the subtractions.
P02	I didn't understand the program. I think that at the beginning I posed five variables, four equal to a number and the last "element" in function of another "index" and later I established hypotheses between the variables but I did not understand the meaning of what I was doing.	As a person who has never used this language, I have understood from the audio that first everything written in a square is focused making a general vision and then it is separated by stripes reading from top to bottom and relating each new strip you read with the ones above. I didn't understand very well the meaning of what was calculated, but in the first stripe it was as a description of the variables. In the following three I did three simple calculations to use them in the fourth strip and thus arrive at the result that we want to obtain.

<p><b>P03 Reached All</b></p>	<p><b>The program starts a loop where in each iteration, each element of an entered list of numbers is analysed. When the number is natural other than "9", its value is added to the variable "var1". When the number is less than "0", it does not carry out any operation. When we find a "9", it leaves the loop and the subsequent numbers are no longer analysed. Finally, it will print the value of the sum, the number of elements found until it leaves the loop, and the division between both values.</b></p>	<p>The program makes the difference between x2 and x1 in calculation1. It then calculates the difference between y2 and y1 in calculation2. Finally, it calculates and prints the product of both differences. In the case of the program shown, it first prints 100 and then 25.</p>
---------------------------------------	---	---



## List-Rectangle version in Spanish

Name	Mystery1 (List)	Mystery2 (Rectangle)
P02(2)	El funcionamiento del programa va por bloques, los cuales tienen distinta extensión dependiendo de la dificultad que tengan y la necesidad de explicarlo. Además estos bloques pueden estar divididos en sub-bloques y no tienen por qué estar en orden.	El funcionamiento del programa se dividía en varias partes. Había cuatro variables ( $X_1; X_2$ , $Y_1; Y_2$ ) y pedía hacer la resta $X_2 - x_1$ y la resta $Y_2 - Y_1$ ; Y después multiplicar el resultado de ambas operaciones anteriores.
P03	No he terminado de entender el funcionamiento del programa, imagino que describe el trabajo realizado por una impresora.	Si tuviese que responder a esta pregunta una vez leído el código (sin responder a otras preguntas), probablemente no hubiese sido capaz de escribir nada. Sin embargo, tras la última pregunta, es cierto que se podría tratar del cálculo del área de alguna figura geométrica.
P04	No estoy muy segura del funcionamiento del programa. Puede que el programa sume varios números, utilice la varianza, el número total de números y finalmente divida el resultado.	Este programa calcula la diferencia entre $X_1$ y $X_3$ , restando el primero al segundo. Después calcula la diferencia entre $Y_1$ e $Y_2$ , de la misma manera. El paso final es double calculation, el cual consiste en multiplicar ambos resultados. Si nos imaginamos un cuadrado, multiplicando las distancias de la base y altura obtenemos el área.
P03(1)	El programa describe un problema y define las diferentes posibilidades de resultado. De esta forma, se pueden describir patrones para estadísticas.	El programa da los puntos en los que se encuentran los vértices del cuadrado y, siguiendo las calculations, puedes hallar el área de la figura.

## Rectangle-List versión in Spanish

Name	Mystery1 (List)	Mystery2 (Rectangle)
P01	El programa va sumando hasta encontrar un 9.	Calcula la diferencia entre las posiciones de los vértices, obteniendo los lados, y a continuación las multiplica, obteniendo el área.
P01(1)	El programa recibe una lista de números y los pasa a una función. Esta función inicializa unos parámetros, para luego recorrer la lista de números hasta encontrar un número nueve, cada número de la lista recorrido diferente de nueve añade uno a la variable contador y otra variable va incrementando su valor con la suma de los valores de la lista recorridos, al final el programa imprime en pantalla la suma de los números, la cantidad de números de la lista recorridos y el ratio entre la suma de los números y los números recorridos.	El programa recibe datos pasados por código y hace los cálculos especificados en las funciones, en el ejemplo recibe dos valores diferentes y saca por pantalla ambos cálculos. Los cálculos son la multiplicación de las restas.
P02	No he entendido el programa. Creo que al principio planteaba cinco variables, cuatro igualadas a un número y la última "element" en función de otra "index" y más adelante establecía hipótesis entre las variables pero no he entendido el significado de lo que estaba haciendo.	Como persona que no ha utilizado este lenguaje nunca, por el audio he entendido que primero se enfoca todo lo escrito en un cuadrado haciendo una visión general y luego se separa por franjas leyendo de arriba abajo y relacionando cada franja nueva que lees con las de arriba. No he entendido muy bien el significado de lo que se calculaba, pero en la primera franja hacía como una descripción de las variables. En las tres siguientes hacía tres cálculos simples para poder utilizarlos en la cuarta franja y así llegar al resultado que se quiere obtener.
P03	El programa inicia un bucle donde en cada iteración, se analiza cada elemento de una lista de números introducida. Cuando el número es natural distinto de "9", se añade su valor a la variable "var1". Cuando el número es menor que "0" no realiza ninguna operación. Cuando nos encontramos con un "9", se sale del bucle y ya no se analizan los números posteriores. Finalmente, imprimirá el valor de la suma, el número de elementos encontrados hasta que se sale del bucle, y la división entre ambos valores.	El programa realiza la diferencia entre $x_2$ y $x_1$ en calculation1. Posteriormente calcula la diferencia entre $y_2$ e $y_1$ en calculation2. Por último, calcula e imprime el producto de ambas diferencias. En el caso del programa mostrado, primero imprime 100 y después 25.