Master’s Thesis

Debugging strategies of novice and expert programmers using Jeliot 3: a qualitative analysis of verbal and gaze protocol

Maria Kinnunen

10.10.2007

University of Joensuu
Department of Computer Science & Statistics
Master’s thesis

Abstract

Four students in University of Joensuu were studied about debugging performance with Jeliot 3 a Program Visualization tool. Two of the four participants were defined as novice and two as experts in debugging. Background information about programming and Jeliot 3 experience was gathered from the participants before entering the debugging session. Each of the participants also attended a small pre-test in order to gather information about how well they understood the binary trees, program output and if they were able to find errors in a very small program. Both eye-gaze and think-aloud protocol from the debugging sessions were recorded for further analysis. It was found that eye-gaze data helps in interpretation of think-aloud protocol recorded from a debugging session. It was found that gaze-data gives the directions of attention and it is possible to receive the reasons for actions via think-aloud. It was found that the eye-gaze was not consistent with the think-aloud protocol in some of the cases. This way, it was found, that one of the participants had incorrect assumptions about the program when debugging. It was found, that the activities for the novices were more of a bug-driven than for the experts who were firstly interested in understanding the program. It was found that making oneself familiar with the program via code and secondly via visualization helped in understanding the program and location the bugs. Staying strictly to the visualization gave the poorest results and using no visualization at all gave the second worst results in locating the bugs. This suggested that visualization by itself was not a sufficient way of debugging but used along with other representations it was quite efficient. It was noticed that the experts found more bugs than the novices, but both the experts were not equally efficient in debugging.

Key words: debugging strategies, psychology of programming, eye tracking, program visualization, Jeliot 3

Tiivistelmä

Neljä Joensuun yliopiston opiskelijaa osallistui tutkimukseen, jossa tutkittiin suoriutumista virheiden etsintä- ja korjaustehtävässä Jeliot 3 tietokoneohjelmien visualisointiympäristössä. Tutkimuksen neljästä osallistujasta kaksi määriteltiin koodivirheiden etsinnässä ja korjauksessa noviisiksi ja kaksi asiantuntijaksi. Tutkimukseen osallistujilta kerättiin taustatietoa liittyen ohjelmointi- ja Jeliot 3 kokeneisuuteen ennen virheiden etsintä- ja korjausistuntoja. Kukin osallistuja suoritti esitestin, jonka kautta kerättiin tietoa osallistujien ymmärryksestä liittyen binaaripuihin, ohjelmatulosteisiin ja suoriutumisesta virheiden etsinnässä pienestä ohjelmasta. Kunkin osallistujan silmien liikkeet ja ääneen ajattelu nauhoitettiin virheiden etsintä- ja korjausistunnoista analysointia varten. Nauhoitetusta datasta havaittiin, että silmän liikkeet auttoivat tulkitsemaan ääneen ajattelua. Silmän liikkeistä havaittiin selviävän huomion kohde ja toiminnan tarkempien syiden selviävän ääneen ajattelusta. Katseen kohteen ei havaittu aina olevan sisällöllisesti yhteneväinen ääneen ajattelun kanssa. Tämän havainnon ansiosta eräällä tutkimukseen osallistuneista huomattiin olevan erheellisiä tulkintoja ohjelmasta virheiden etsimisen ja korjauksen aikana. Virheiden etsimisen havaittiin ohjaavan noviisien toimintaa kun taas asiantuntijoiden havaittiin ensisijaisesti pyrkivän ohjelman ymmärrykseen. Havaittiin, että tutustuminen ohjelmaan ensisijaisesti ohjelmakoodin ja toissijaisesti visualisoinnin kautta, auttaa ohjelman ymmärtämisessä ja ohjelmavirheiden paikantamisessa. Pitäytyminen ainoastaan visualisoinnissa johti heikoimpiin ja visualisoinnin hyödyntämättä jättäminen johti toiseksi heikoimpiin tuloksiin ohjelmavirheiden paikantamisessa. Visualisoinnin ei todettu yksistään käytettynä olevan riittävä tapa virheiden etsimisessä ja korjauksessa, mutta visualisoinnin merkitys tehostui käytettäessä sitä muiden esitystapojen ohella. Asiantuntijoiden havaittiin löytävän noviiseja enemmän koodivirheitä, mutta heidän ei havaittu olevan keskenään yhtä tehokkaita virheiden etsimisessä ja korjaamisessa.

Avainsanat: virheiden etsintä- ja korjausstrategiat, ohjelmoinnin psykologia, silmänliikejäljitys, ohjelman visualisointi, Jeliot 3

Foreword

I would like to thank all the people who have had the time and energy to read and give valuable feedback on my master's thesis. I truly appreciate the work you have done. In addition I would like to thank my family, friends and mentor who have encouraged and motivated me all along the process.

Contents

1 Introduction

Code debugging is a process by which errors in or malfunctions of the code are eliminated. These errors are called bugs. A person eliminating the bugs is called a debugger. Sometimes a debugger finds the bugs and makes the program work properly and sometimes not. Unfortunately, the debugger may add new bugs to the code is she/he does not understand the consequences of his/her actions to the program.

When a student has no or little experience in programming it could be that he/she will not succeed in debugging either. Poor debugging performance — that is, finding no or small number of bugs or making the code even worse — is considered as an indicator of novice skills. On the contrary, good performance in debugging means that the goal of being an expert has been reached. It should also be kept in mind that being an expert does not mean that there is no room for further development e.g. by deepening or widening the performance skills.

In addition, knowing how to write programs does not guarantee a good performance in debugging that demands an understanding for the program (Winslow, 1996). Furthermore, the skill of being able to debug program codes of others is an expert characteristic, because it requires the ability of identifying what is the purpose of the program code, finding the lines that cause the malfunction, and applying the required knowledge in order to fix the code.

Everybody entering the field of programming is a novice. The education, teaching and learning abilities are playing an important role. Finally, novices are becoming well performing experts that are capable of adapting their knowledge in many kinds of programming and debugging challenges. Learning the skill of debugging, in particular, depends on capability to read and understand programs, choosing a suitable strategy and analysis of the deeds and their justifications (Winslow, 1996). Debugging, too, can be seen as a problem solving skill, as the expertise grows on the basic knowledge and general problem solving skills (Winslow, 1996). To write a flawless simple Java program, a programmer has to know about the semantics and syntax. Consistent use of gained knowledge and skills results into better problem solving skills and ways of applying the knowledge acquired (Winslow, 1996).

When a programmer has more experience and he/she realizes what are the solutions to consider in each task, he/she is able to act in more flexible ways. A programmer is no longer a ”prisoner” of too simplified example programs and dependent of the information they offer. When new and more demanding tasks are faced and the use of more efficient ways of problem solving are applied, expertise grows challenge by challenge (Winslow, 1996).

Research concerning debugging (Gugerty & Olson, 1986) and program comprehension (Koenemann & Robertson, 1991) is not a new research field. One of the freshest approaches is to use visual attention tracking tools to provide a way of following a participant’s eye-gaze while performing comprehension (Bednarik et al., 2006) or debugging (Romero et al., 2003). Using this approach in interpreting, evaluating and analysing debugging strategies with the help of eye-gaze data, the general research questions are as follows:

1) How debugging strategies can be revealed using eye-gaze data during debugging with Jeliot?
2) What kinds of special observations can be noticed on the grounds of comparison of eye-gaze and think-aloud protocol during debugging sessions in Jeliot?
3) What kinds of differences can be found in levels found in think-aloud protocol of novice and expert programmers? 4) What kinds of differences can be found in levels found in think-aloud protocol of good and poor performing debuggers?
5) Are there any regularities that reveal dependence between certain levels of good and poor debugging?

Based on the research questions previously reviewed literature, and results of previous research results, the hypotheses for this study are as follows:

1) It was supposed that experts develop faster, wider and deeper of an understanding about the Java program. If this results is true, the results can be seen as consistent with the characteristics of expertise. When the hypothesized results turn false, it raises a contradiction between the results and the accepted view of expertise. The contradiction should then be discussed in order to find the explanations that resulted in this conflicting situation.

2) Expert programmers were expected to be more efficient debuggers, it means they are supposed to spend less time and find more bugs in given time. This hypothesis is also based on the expertise characteristics (Haapasalo, 2000). An expert should be able to perform more effectively compared to a novice. On the other hand, there is some

evidence arising from the previous studies that does not strongly support this hypothesis (Romero et al., 2003) or suggests, that the relation between good performance in programming and debugging is not so straightforward (Ahmadzadeh & Elliman, 2005).

3) It was assumed that eye-gaze helps in revealing the problem solving process when debugging. If this hypothesis is true, it encourages usage of eye-gaze in future studies, too.

4) Eye-gaze data was also thought to give a deeper view on think-aloud protocol in each of the debugging sessions. If this hypothesis is true, using eye-gaze data side by side think-aloud data could also be considered in following studies.

This work includes analysis of four students that were identified as two novice and two expert debuggers according to the performance in debugging sessions. Background information was gathered about programming experience and experience with Jeliot 3 Program Visualization tool. The participants also attended a pre-test that revealed how well a very small program was understood and if any errors were found in it. The participants were introduced to the Jeliot 3 beforehand. The main data for this study was gathered during each of the debugging sessions. The debugging session consisted of reading the instructions that included a brief description of the program, the current and the correct output. When the participant was ready, he/she would enter the Jeliot 3 and start browsing the code or visualize it if wanted. The participants were asked to think-aloud during the session. Debugging sessions were recorded with think-aloud protocol audio and visual attention video data. A category similar to Bloom's Taxonomy for the think-aloud protocol was designed and applied for the analysis of the data.

Chapter 2 contains the theoretical background of debugging related issues like knowledge, comprehension, learning and problem solving. In addition, Chapter 2 involves theory about relations between eye-gaze and cognition. Jeliot 3 is also introduced. Chapter 3 describes the experiment in concern. Chapter 4 contains results from the four debugging sessions. Chapter 5 involves discussion. Reliability, validity and researcher's view are also included. Chapter 6 includes final conclusions about the work.

2 Theoretical background

2.1 Knowledge and Comprehension

Without a doubt, novice programmers have problems when learning how to program. The most frequent problems are the ones related to the basics of program design — that is, loops and arrays (Garner et al., 2005). When a program does not function properly, a programmer is required to find what is the reason behind the problems and trying to create a flawless program by fixing the bugs. If a programmer has no understanding of the code he/she can not expect to know where the bug lies within the source code. Debugging is based on a comprehension of the code at hand (Wiedenbeck, 1999). As a result, comprehension and debugging are closely related as tasks included in programming. There are several philosophical views for the source of knowledge and for example according to the constructivists, new knowledge should be built on the prior knowledge, understanding and previous experiences (Wilson, 1996). But let us firstly review some of the philosophical views.

Understanding and knowledge are bind together when the mind is existentially considered to work as a whole (Reid, 1986). Philosopher John Locke stated that all the ideas come from experience but all the knowledge doesn't (Hamlyn, 1978). On the other hand, Kant believed that all the knowledge does not come from experience but through experience. In addition, understanding through objective experience must adapt to particular priori principles that are absolute and exist regardless of the empirical study. Aristotle believed scientific knowledge is dependent on first principles that offer a basis for other sciences (Hamlyn, 1978). In Plato's opinion knowledge is a direct awareness of the thing in question and all learning is recollection. Recollection means that the soul has lived many lives and seen a lot (Hamlyn, 1978). The problem is that the soul has forgotten the previous knowledge but it can be recalled when facing it in the present time. As it follows, there is no real new knowledge and the forgotten knowledge is actually untouchable because the reminiscence should occur in the present and should also include knowledge from the past. Knowledge used to perform of a certain task can be old with no need for further experience — that is, a priori or new information obtained from a new experience and combined into existing knowledge and understanding. Plato compared a newborn's mind to an empty wax tablet and Locke into tabula rasa — that is, an empty board being a basis for growing experience (Hamlyn, 1978).

What does it require for a student to acquire knowledge, understanding and programming comprehension? To understand programming a future programmer has to know the basic nature of programs. A program is both syntactical — that is, formed by a text with certain grammar rules — and each sentence has a certain meaning — that is, semantics (Hoc et al., 1990). When debugging, a programmer is facing the task in separating syntactical errors from the semantically ones and learning different ways on how to solve those errors.

Program comprehension can be performed in both a bottom-up and a top-down fashion. According to the bottom-up theory, program comprehension proceeds hierarchically from the patterns of operation seen as higher-order chunks towards the complexity of the whole program while a semantic internal representation is built (Hoc et al., 1990). When applying the means of top-down, the internal representation is built starting from the general function of the program proceeding via hypotheses, testing and altering it until the entire program elements agree with the final improved hypothesis.

2.2 Problem solving and Debugging strategies

Debugging is an example of a problem solving processes. Problem solving includes cognition, knowledge, data processing and learning (Haapasalo, 2000). A problem solving process consists of understanding the given problem, working with the problem, planning the solution for it, putting the solution both into practice and under control, and remembering this process (Haapasalo, 2000).

Generally, problem solving requires a problem and constitutes of a problem space (Haapasalo, 2000). A problem space consists of all the possible stages from the beginning of the task to the conclusion. All the operations between the stages, where novices and experts behave in different ways, are included. It has been found that novices are stuck with individual stages and use a great amount of their resources. This means that all the knowledge and skills in use for analyzing the stages, trying to apply definitions and specific contextual strategies. A novice is unable to see the connections and similarities between definitions and stages, so he is likely to get into cognitive chaos while trying to operate each individual data element as a separate chunk.

An expert sees the whole problem space due to both use of general strategies and the main principles related to the information area the problem is attached. An expert is able to connect conceptual, procedural and general method information in a suitable way. He is also capable of making intentional semantic chunks out of knowledge and definitions. This way he could process a greater amount of information while automation of certain accustomed functions. In addition, an expert has the ability to self control, intuition and to make forward aiming plans in ways of hypothesis and deduction (Haapasalo, 2000).

Learning happens when active selection is made from the possible data that should be learned, the selected data is internalized, interpret and assimilated into the former data structures (Engeström, 1988). Wanted outcome of learning process is holistic high quality knowledge that occurs as clear structures and models. High quality knowledge has high transfer (Engeström, 1988). This means, that the knowledge has a wide range of applicability and it is constructed of principles and concepts suitable for solutions of variety of problems. High quality knowledge also works in real life as the principles are known how to apply in practice and it is not easily forgotten. Knowledge is not of high quality if it does not meet and explain the reality (Engeström, 1988).

A problem solving task such as debugging or writing runnable programs requires both theoretical and former practical knowledge and capacity to apply this knowledge into variety of cases. The ability of applying knowledge into different contexts is called transfer (Rauste-von Wright & von Wright, 1994). Poorly transferable knowledge or skills can only be used in the same context once learned (Engeström, 1988). Correspondingly, when a piece of knowledge or skill is well transferable he/she has the ability to think outside the box and use these in huge amount of contexts and cases (Engeström, 1988). It is found that obtaining more effective results requires learning in similar context to what the skill or knowledge is known to be used in the future. The more varying are the ways of training and learning such as different variations of certain rules, the better the transfer is going to be.

It has been found that neither novice nor expert programmer perform debugging in a random way (Gugerty & Olson, 1986; Romero et al. 2003). Certain regularities in debugging process — strategies — are found.

Debugging strategies found by Gugerty & Olson (1986) are shown in Figure 1. First a participant was having an introduction to the program — that is, he/she red the program description’s paper and viewed the code. In the next step, the participant formed a hypothesis about the possible bugs in the code and a general idea of the program. The participant tested the hypothesis via modifying the code, executing it or finding support from the description’s paper that included the purpose of the program. If testing the hypothesis did not result in finding an error, the participant returned with the paper of the description or went on viewing the code. In Gugerty & Olson study (1986), novice and expert programmers did not act any different but novices needed more time to do the same as the experts. In addition, novice programmers seldom found the bugs in the first try and novices also added new bugs into the program while debugging — that is, testing the hypotheses for the existing ones.

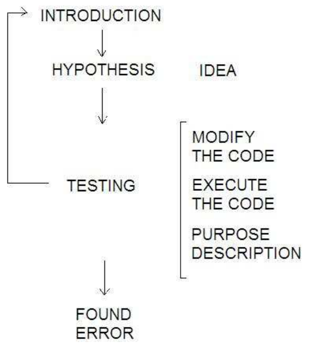


Figure 1: Debugging strategies found by Gugerty & Olson (1986).

Strategies discovered by Romero et al. (2003) are shown in Figure 2. Introduction, where the participant browsed the code, resulted in a situation where the participant found a piece of a ”suspicious code”. At times, the participant reported about an error based on the ”suspicious code” that he/she had been found. If no error was found, the participant would return in browsing the code. The participants also made comparison between the code and the other representations — that is, the visualization and the output of the code. It was also found, that the less experienced programmer spend less time in code browsing and used different representations frequently. The more experienced programmer spend most of the time in code browsing and spotted several suspicious pieces of code that did not lead in finding the real bugs. Both the less and more experienced programmers were as accurate in debugging even if the more experienced programmer was more skilled in translating between different representations, had more programming experience and better verbal skills. Overall, the less and more experienced programmers did not significantly differ in debugging performance.

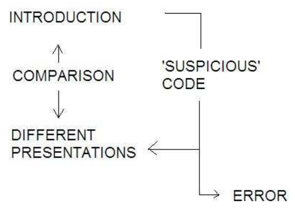


Figure 2: Debugging strategies found by Romero et al. (2003)

The think-aloud method has been used when researching the debugging strategies to reach the thoughts and as a means of bringing the problem solving process in a more reachable form (Romero et al., 2003). Practically, during the debugging process a programmer tells everything that comes into his/her mind while trying to find bugs and make corrections to the code.

The think-aloud method requires interaction between thinking and spoken language (Haapasalo, 2000). This requirement is one of the basics in problem solving. Thinking aloud is also one way of improving metacognitive skills (Haapasalo, 2000). Metacognition is cognition about cognition (Yzerbyt et al, 1998). Furthermore, metacognitions are control strategies that are consciously used in controlling and monitoring information processes and problem solving (Haapasalo, 2000). Metacognitions have an executive force, because they start or reject a particular strategy. Thinking aloud helps in understanding the problem and increases the speed in performance of mental operations. Thinking aloud helps in estimation of steps towards solving the problem and it highlights the final goal. Furthermore, thinking aloud requires the association and finding of new connections that have relevance for the solution.

When trying to find out how the knowledge develops while reading a code, there should be some formal way of interpreting the think-aloud protocol from code viewing session. Bloom's Taxonomy is one of the ways for evaluating how developed knowledge does a programmer have about a program (Buckley & Exton, 2003). This method has six levels with different degrees of difficulty and proficiency for cognitive domain (Xu & Rajlich, 2004). The levels are knowledge based: recall, comprehension, application, analysis, synthesis and evaluation (Buckley & Exton, 2003). On the *recall*-level, the learning and remembering happens in the same context with no demand for greater understanding. In *comprehension*-level a learner should be able to give a linguistic translation for the given material, reorder the material according to different relationships included and make predictions based on the material. In *application*-level the learned knowledge should be able to use also in completely different contexts. In *analysis*-level a learner should be able to classify the data elements of the whole data, see the connections between the elements and the basic principles that hold the elements as a whole. In *synthesis*-level a learner should be capable of combining new information with the former learned. Finally, the level of *evaluation* concerns criteria and standards that are used in an acceptable degree according to the learner.

2.3 Eye-gaze and Cognition

What we see affects our cognitive processes and the actions we take based on these matters. It is found that a newborn is able to recognize and imitate facial gestures seen done by an adult person (Meltzoff & Moore, 1977). A newborn sees the gestures, interprets how the similar gestures could be made with his/her face and via sense of touch turns these interpretations into motoric action (Meltzoff & Moore, 1977). We get information through our eyes and our visual interests are revealed through our eye-gaze direction. The visual information is processed in our brains and the questions caused by the received information most likely gives the next direction for the eye-gaze. An eye fixation or an eye fixation time is closely related to the eye-gaze. In a fixation, eyes move to stabilize the image of an object on the retina (Rayner, 1998). During a fixation, lasting from 200 ms to 300 ms, visual system extracts, decodes and interprets the features of the viewed object.

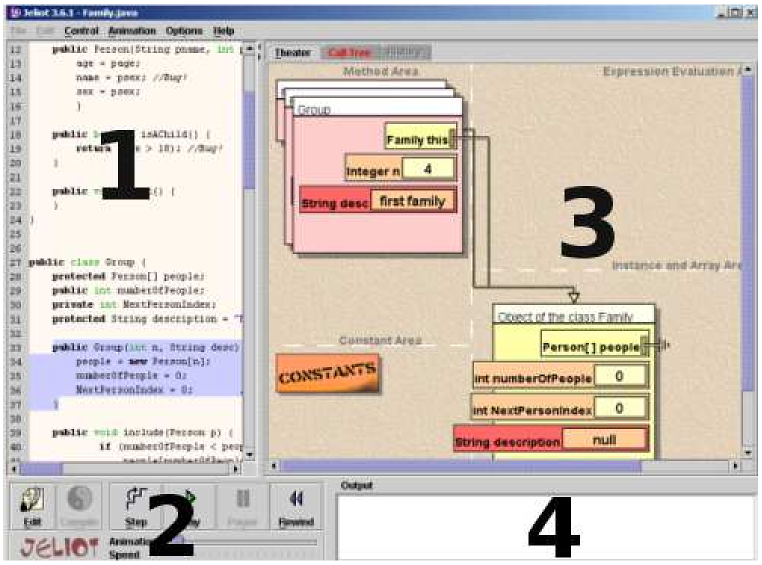
Furthermore, eye fixation is closely connected to the comprehension. A theory based on immediacy and eye-mind assumptions was based on an following idea. When a person was reading, the interpretation about the text was done whilst the eye was fixated (Just & Carpenter, 1976). It was also assumed that fixation of an eye did not end before processing of a word was complete. As a conclusion, fixation time reflected how long it took to extract and interpret the object — that is, the word related information. It has been suggested that the higher the amount of fixations is focused on a certain element, e.g. a code representation in program visualization interface, the more important the element is (Bednarik et al., 2006). It has also been found that fixation duration could become greater due the demand for difficult cognitive processing in order to perform the given task (Goldberg and Kotval, 1999).

Eye-gaze has been used to study how novice programmers could be helped via expert programmers' eye-gaze movements recorded while debugging takes place (Stein & Brennan, 2004). Eye movements have been studied in resolving the ways a less and more experienced programmer differ in viewing a short and complex algorithm written in Pascal (Crosby & Stelovsky, 1990). In addition, the eye fixation connection to the performance of simple cognitive tasks, e.g. sentence verification that are being carried out by operational memory has also been studied (Just & Carpenter, 1976). Program comprehension in the Jeliot 3 visualization system has also been studied with the help of eye-gaze behavior information (Bednarik et al., 2006). The relation with problem solving in visuo-spatial and causal domains involving mental imagery to the effectiveness of Attentive User Interface was also studied with the help of eye-gaze tracking (Yoon & Narayanan, 2004).

It is possible to make conclusions about the cognitive processes behind the given task when the context and order of the visual targets included is known (Bednarik et al., 2006). To make an eye-gaze visible and to revisit these eye-gaze movements it is relevant to use a device designed for this task — that is, an eye tracker (Bednarik et al., 2006). An eye tracker estimates the direction of gaze, usually has accuracy of 1 degree, samples the data at 50-500Hz and must be calibrated for every user individually (Bednarik et al., 2006).

2.4 Jeliot 3

Jeliot 3 is software developed at the University of Joensuu that visualizes program execution and is supposed to aid in specifying feasible program development models (Moreno et al., 2004). In Jeliot, objects and their fields are shown in a notation, similar to UML, of class diagram (Bednarik et al., 2006). Reference semantics of Java language are illustrated via treating references to the objects no different than treating other variables (Bednarik et al., 2006). Variables situated in different scopes are visualized in a separate fashion. ”For example, method frames contain the local variables of the method, objects contain the fields and static variables are separated in their classes” (Bednarik et al., 2006). In addition, no difference is being made between the roles of variables and it is capable of automatic visualization of object oriented programs (Bednarik et al., 2006).



```
    public Person(String pname, int p
        age = page;
        name = pname; //Bug!
        sex = psex;
        }

    public b          isAChild() {
        return       e > 18); //Bug!
    }

    public v          () {
    }
}


public class Group {
    protected Person[] people;
    public int numberOfPeople;
    private int NextPersonIndex;
    protected String description = "

    public Group(int n, String desc)
        people = new Person[n];
        numberOfPeople = 0;
        NextPersonIndex = 0;
    }

    public void include(Person p) {
            if (numberOfPeople < peo
```

Figure 3: User interface of Jeliot 3. (1 = code editor; 2 = control panel; 3 = visualization view; 4 = output console)

User interface of Jeliot 3 is shown in Figure 3. Before visualization can take place the user needs to write or load a program written beforehand and compile it. The code appears in code editor. As it follows, a visualization view opens and shows the program execution via ”play” button of control panel. The control panel includes other options that make it possible to view animation faster or slower, step by step or non-stop, halt the animation to make a closer look of ongoing situation, exit the animation somewhere in the middle or watch the animation from the beginning to the end or look backwards the animation via execution history. The main purpose of the Jeliot, offered for novices, is to increase understanding of the program (Moreno et al., 2004). Jeliot helps a student to see the main ideas among all the complexity and information via call tree visualization that shows previous and current method calls (Bednarik et al., 2006). In history view, mentioned before, a user can go back to the previous execution stages step by step and get a better understanding of the program executed so far (Bednarik et al., 2006). In addition, students using Jeliot 3 gain a more wider vocabulary related to programming structures and concepts (Ben-Bassat Levy et al., 2003).

Program comprehension in Jeliot 3 program visualization system has been studied and previous work shows that novice and expert programmers use animation in the same extent (Bednarik et al., 2006). It was also found that the usage of the animation was favoured more among the novices. Novices, too, needed more time to comprehend a feature that was animated at the moment. In addition, switching between different representations was effected by programming experience in a nearly significant way. Novices and experts were found to use different representations in a similar amount. The affective effects of program visualization have been studied with the help of Jeliot, too (Ebel & Ben-Ari, 2006). The results showed that program visualization did have positive effects on the students in means of diminished undisciplined behavior during the lessons.

Jeliot 3 is, by no means, the only program visualization system. There is a various amount of student aid tools developed in different universities. Algorithm Animator and Programming Toolbox or AAPT, for example, was developed to help student in learning how to program in Turbo Pascal and besides the animation it allows program developing all the same (Sanders & Gopal, 1991). JACOT was developed to visualize Java programs (Leroux et al., 2003). In addition, SIAMOA was developed to design, debug and visualize algorithms written in Smalltalk or Pascal (Van de Veire et al., 1998). As it can be concluded, existence of several other tools that were developed to fulfill the similar novice aiding expectations to the Jeliot 3 is apparent.

3 Experiment

The data was gathered from recorded debugging sessions in a laboratory setting. Before the session the participants were given an introductory lesson to Jeliot and its features. All the participants filled a background questionnaire with questions concerning age, sex, glasses, starting year of university education, major, minor, Java programming, other programming and Jeliot experience. Pre-test included three tasks: 1) pointing the right output, 2) drawing a binary tree and 3) debugging of a program with 13 lines of code. Total amount of points was seven: two points from the first and second task and three points from the third task. In the beginning of a session an individual in question was given a description about the program written in Java programming language. The program description constituted of the described problem, description about how the program should work, task description for the participant and both the current and correct output.

The participant was also given instructions verbally by the supervisor in the session. The supervisor asked the participant to read the paper of instructions and start the debugging session when ready. The participant was also asked to think aloud during the session and to report if he/she had found a bug with. The things he/she was asked to give about the found bug were: the line and the class, what caused it and a possible fix to it. The dialogue that the participant had with the supervisor, the monologue given by the participant and the activities seen on the screen in Jeliot 3 environment — with mouse and eye movements — were recorded.

3.1 Participants

Six of the recordings were selected for a closer observation. Unfortunately, two of these recordings were severely damaged and only four of the original recordings were taken into this study. The participants were four students studying in University of Joensuu. Three of them had Computer Science as a major and one of them had major of Mathematics. In table 1 there are the background information and the pre-test results of the four participants on each of the rows. ”Participant” stands for the symbol of each participant, ”Major” for major subject in studies and ”Minor” for minor subject in studies. ”Java exp” stands for Java language programming experience in months and ”#Jprogram” for the amount of written Java programs. ”Other lang” stands for experience of other programming languages in months. ”Jeliot” stands for if the participant knows what Jeliot is (Y) or does not (N). ”prev.” stand for if the participant has previous experience with Jeliot (Y) or not (N). ”#Jeliot” stands for how many times the participant has used the Jeliot before. ”Pre-test” stands for the points gained from the three tasks — that is, giving the righ output, drawing a binary tree and debugging a small program.

Table 1: Background information and pre-test results of the participants.

Participant	*Major*	*Minor*	*Java exp*	*#Jprogram*	*Other lang*	*Jeliot*	*prev.*	*#Jeliot*	*Pre-test*
P1	CS		4	10	1	Y	N		7
P2	MAT	CS	5	100	0	Y	N		3
P3	CS	MAT	0.5	10	24	Y	N	1	7
P4	CS	EDU	18	30	50	Y	Y	50	6

3.2 Settings and Task

One complete program with 113 lines of code was presented to the participants (see Appendix 5). Program included 10 bugs (Table 2) but the number or nature of bugs was not stated for the participants. The remote Tobii ET-1750 (50Hz) eye tracker was used to track the eye movements. It made no contact with the participants and is built inside a TFT panel to make it invisible and mute during the recording. Key strokes, mouse clicks, audio and video were recorded for each session.

The experiment was done in a quiet usability laboratory. The participants were situated in an ordinary office chair near the experimenter faced the 17” TFT display. In addition, an automatic eye-tracking calibration was done — that is, the participant followed one shrinking point at a time situated all around the screen. The total number of points was sixteen and the calibration was done until the highest possible accuracy was reached. In a session a participant was given the instructions, clicking the ”start” – button enabled the use of Jeliot and the experimenter announced when two minutes was left until the end — that is, around 16 in minutes or 960 in seconds.

The bugs within the source code are shown in table 2. 1 point was given from each of the clearly found bugs and 0.5 points were given in case of some uncertainty.

Table 2: Bugs within source code.

Bug	*Correct*
B1 12:psex	pname
B2 age>18	age<18
B3 sum=	sum = sum + people[i].age
B4 58:people.lenght	numberOfPeople
B5 DESC	31:initial value
B6 44:print outside else	inside
B7 50-51:if	if (NextPersonIndex = (numberOfPeople-1)
B8 0->i	79: people[i]
B9 if->while	105: while(i < children.numberOfPeople)
B10 110:print inside the loop	outside

3.3 Method: Protocol of (data) analysis

The specific technique for verbal analyses consists of 1) reducing the protocols, 2) segmenting the protocols, 3) developing or choosing a coding scheme or a formalism, 4) operationalizing evidence for coding, 5) depicting the mapped formalism, 6) seeking pattern and coherence in the depicted data, 7) interpreting the pattern and its validity and 8) repeating the whole process (Chi, 1997). The method relies on the qualitative data, but the analyses are quantified (Chi, 1997). The process of this work towards the qualitative and quantitative analysis started by writing out the dialogues and monologues recorded during each debugging session. Raw data included also corrupted recordings that were disqualified leaving only four usable ones to proceed with.

Reducing the protocols can be made by ”choosing” a subset on the basis of some ”noncontent” criterion e.g. changes in activity or speech (Chi, 1997). None of the files including sessions in textual form was completely coded. There were mouse clicks, pauses, talk belonging to the experimenter, background noises and sections involving technical problems left uncoded.

In segmenting the protocols, the defining cut can lay in ”a proposition, a sentence, an idea, a reasoning chain, a paragraph, an interchange or an episode" e.g. a particular activity (Chi, 1997). During the coding process, the most demanding task was to notice in what a participant was referring to. It may have been the case that there were several defining cuts inside one sentence. In addition, a participant could have started a reasoning chain in one point of the session, view some other aspects in the middle and return in solving the unfinished chain later on. Basis for segmentation can be in semantic features — that is, content of the utterances determines the segment boundaries (Chi, 1997).

After segmenting follows the coding as the codes developed should meet the formalism representing the knowledge (Chi, 1997). In this work, the coding categories were developed in revealing the understanding he/she has throughout the debugging session and the means he/she is trying to solve the given debugging task. Operationalizing evidence for coding is deciding which utterance belongs to which category or code (Chi, 1997). Finding evidence required several times of returning back to the data because new categories or codes were found. It was also becoming evident that mental models and knowledge hiding in the verbal data was not a straightforward task to perform.

Depicting the mapped formalism presents "the data to the audience" and "shows if some patterns can be detected in the depicted data"(Chi, 1997). Tables 3 to 7 with the coding and examples were made to clarify the meaning for their existence. Seeking pattern and coherence in the depicted data in taxonomical categories is easily performed via bar graphs (Chi, 1997). Graphs were drawn from the coded data to give a more illustrative view about the possible problem solving processes included. Special attention was given for blurry situations where a participant was confused e.g. content of monologue differs from the actual seen event in animation. Interpreting the pattern in the depicted data depends on the hypothesis, the research questions and the theoretical orientation (Chi, 1997). Validating and interpretation can be performed confirming it with additional evidence (Chi, 1997). The last stage is to repeat the whole process when recoding is needed e.g. different questions are being asked (Chi, 1997).

3.4 Coding schema

First, the coding was done manually while hearing the think-aloud protocol and following the textual version. Codes were based on suggestions of the mentor of this Master's thesis, made before the coding task was done. This gave a preliminary view of the way the textual data was supposed to code. Follow up of the eye-gaze recordings of the sessions gave support or demand for changing the way of the previous coding. When a part of textual data, that should have been coded, did not fit to the present codes, a new code was defined. Also the similar cases having similar coding in all the various sessions was manually checked up.

Level of utterance

Level of utterance states how profoundly does a participant handle the information seen and interpreted into think-aloud. In other words, if the information processing is superficial or it includes conscious effort to gain a better understanding about the program.

Level of utterance was divided in six categories according to whether the data was 1) just 'pointing' e.g. reading the program code, 2) explaining the meaning or finding hidden meaning e.g. ”a is increased by one”, 3) reasoning ”we move to another person”, 4) asking questions, 5) guessing or 6) validating the execution of the code.

There were difficulties in defining levels of utterance. If he/she was guessing and pointing or explaining the meaning and reasoning. There is a slight differences between explaining the meaning and reasoning, too. Sometimes the sound of insecurity in think-aloud protocol showed it was a case of guessing.

Table 3: Category and levels of utterance.

Category	*Level*	*Example*
Utterance 1	”pointing”	NULL
Utterance 2	explaining the meaning	Nyt se hakkee niitte iät. — It is fetching their ages
Utterance 3	reasoning	Nyt se yhistää niitä ikiä — It is joining the ages together.
Utterance 4	asking questions	Onko childrenillä description. — Does ”children” have a ”description” or Mikäs tuo oli. — What was that?
Utterance 5	guessing	Seuraava tyyppi siinä ryhmässä vissiinki tuo. — Next character in that group is probably that one.

Table 3: Category and levels of utterance.

Category	*Level*	*Example*
Utterance 6	validating	Minä lasken nyt noita ikiä tuosta yhteen. 35 plus 40. Plus 10. Siit tulee 85, 90. — I'm adding those ages in there up. 35 to 40. To 10. It makes 85, 90.

Origin

Origin stands for the representations related to the program that are found in think-aloud protocol. When the origin of think-aloud protocol is found, it is more easy to find out what are the roles or sources of information of different representations for different participants.

Origin was divided in six according to whether the data belongs to 1) code 2) output 3) instructions 4) visualization 5) comparing code with visualization or 6) referring both output and visualization.

Compared to level of utterance origin was easier to define. Nevertheless, levels 5 and 6 are sorts of special cases where it was not possible to divide one sentence into smaller fragments to define e.g. first part is about code and the latter part visualization. Levels 5 and 6 do not exclude the possibility of comparison in other parts of think-aloud protocol. In other cases, the comparison or referring is done in a way the different levels can clearly be separated.

Table 4: Category and levels of origin.

Category	*Level*	*Example*
Origin 1	code	boolean isAChild.
Origin 2	output	Ni joo se tulostaa sitten niitä sieltä. — Oh yes sure. It prints those from there then.
Origin 3	instructions	minkähän takkii se laittaa female sukupuolen siihen. Kuin niin. — Why does it put ”female” gender in there. Why's that?
Origin 4	visualization	Okei. Nyt sinne nimet menee ainaki ihan oikein. — OK. At least, the names are going just rightly in there.

Table 4: Category and levels of origin.

Category	*Level*	*Example*
Origin 5	code <-> visual-ization	Ja se tekee ne sitten tuolla. — And it makes them in there then.
Origin 6	output & visual-ization	Ni joo se tulostaa sitten niitä sieltä. — Oh yes sure. It prints them from there then.

Error

Error is met in a situation when a participant mentions that something is not right e.g. in the code. When error related topics are found in think-aloud session, it helps to interpret in what extent does a participant focus on finding false lines of code also when he/she is not actually finding the real bugs. On the other hand, error related think-aloud protocol may help following the processes towards finding the bugs.

Error was divided in four according to whether it concerned 1) the location of an error, 2) error situation, 3) source of an error or 4) fix to an error.

Error includes a special case where levels of utterance or origin are not coded. That is when a bug is found and the line and class is given. In other cases of level 1 e.g. he/she states a line and class for a bug that is no real, levels of origin and utterance are defined. When level 1 is used in a question, level of origin and utterance are also coded within.

Table 5: Category and levels of error.

Category	*Level*	*Example*
Error 1	location	Rivillä 19. — On the line 19.
Error 2	situation	No tossa nyt ainaki o virhe. Jos se tot- palauttaa tosi tossa. — Well, an error lies there at least. If it returns true in there.
Error 3	source	Tossa, tossa person-luokassa toi lapsijuttu jos se on ikä on yli 18. Nii ei se kyllä lapsi. Lapsi silloin oo. — In there, in that ”person” class by that child thing if it is age is over 18. Not a child then. It’s not a child then.

Table 5: Different categories and levels.

Category	*Level*	*Example*
Error 4	fix	Eli käännetään toisinpäin toi. — So that is turned the other way around.

Gaze

Gaze was divided in two levels according to whether the participant is 1) having a contradiction between what the participant sees and what he/she tells to see or 2) having a contradiction between what a participant sees and what he/she expects to see.

Gaze level 1 is closely connected to the error or location of error that should have been spotted. Gaze level 2 is closely connected to the mental models or ideas about what happens next in animation or what should be written in the code. Unfortunately, the present mental model collides with the seen causing confusion causing further attention or not.

Table 6: Category and levels of gaze.

Category	*Level*	*Example*
Gaze 1	sees <> tells	Se on lapsi. — It's a child.
Gaze 2	sees <> expects	Ei taija. — Maybe it's not.

Plans and comments

Plans and comments include sorts of verbal side notes for the experimenter or for the participant him/herself. Think-aloud in this sense can also be a remark when e.g. going from browsing the code into starting to visualizing it. Plans and comments are also referring to a self aware talk — that is, stopping for a moment for planning what should be done next. Plans and comments can e.g. reveal a participant's opinions about visualization or provide self given explanations for present actions.

Plans and comments was divided in according to whether the participant is 1) explaining what he/she is doing, 2) commenting or 3) planning future action.

Plans and comments involves coding the level of origin, too. Levels of utterance are not coded side by side but level 3 can be seen as a higher level of utterance like explaining the meaning or reasoning. This category can be connected to levels of error e.g. location of an error, too.

Table 7: Category and levels of plans and comments.

Category	*Level*	*Example*
Plans and comments 1	is doing	Jos ihmettelet, et mä oon hiljaa, ni en mä oikeestaan ajattele. Käyn vaan tätä koodia läpi. — If you are wondering about me being silent. Well, I'm not actually thinking. I'm just going through this code.
Plans and comments 2	comment	Onpas hässäkkä. — What a mess.
Plans and comments 3	future actions	Yks vaihtoehto ois ruveta tutkii noita tulosteita ja ruveta sen perusteella selvittää, että mikä siellä on. — One option would be to start taking a closer look of those outputs and based on those start finding out what is the matter in there.

How to interpret the charts

The results include charts about found categories for each of the participants. The categories in the charts are identified with different lines. The X-axis reflects the time in seconds and the Y-axis presents the levels of each of the categories as numbers (Table 8). The numbers on Y-axis have no inherent order — that is, they are merely names. For example 2 can be interpreted as ”explaining the meaning” of utterance, ”output” of origin, ”situation” of error, ”sees<>expects” of gaze or ”comment” of plans and comments.

Summary charts show the proportions of each of the different levels of each of the categories in each of the sessions. The categories are identified with different colors shown in the bars. Now, X-axis presents the names for each of the levels as numbers (Table 8) and the Y-axis stands for the proportions used for different category levels. For example dark blue bar in x=1 and Y=70 shows that level ”code” has representation of 70 percentage of the origin during a debugging session.

Table 8: Y-axis of other than the summary charts. X-axis of the summary charts.

Value	*Utterance*	*Origin*	*Error*	*Gaze*	*Plans and comments*
1	”pointing”	code	location	sees<>tells	is doing
2	explaining the meaning	output	situation	sees<>expects	comment
3	reasoning	instructions	source		future actions
4	asking questions	visualization	fix		
5	guessing	code <-> visualization			
6	validating	output & visualization			

4 Results

Coded data includes both two novices P1 and P2 and experts P3 and P4. Maximum amount of points in pre-test was total 7 (Table 9).

Table 9: Pre-test scores vs. found bugs.

Participant	*Pre-test*	*Bug 1.0*	*Bug 0.5*	*Total Bugs*
P1	7	B1		1
P2	3	B6	B7	1.5
P3	7	B2 B3		2
P4	6	B1 B2 B7	B4	3.5

In the following, the results are presented giving a construct and general view of each of the debugging sessions. The different levels of each coded think-aloud protocols are described and presented as graphs, too.

4.1 Participant P1

General view

The participant P1 reads the instructions and runs the animation. After the first run finishes, the participant runs the animation the second time. The participant P1's origin of think-aloud protocol is mainly on level 4 ”visualization” (Figure 15, Table 10). The participant does not ”ask questions” on level 4 of utterance but is ”guessing” on level 5 a lot (Figure 15, Table 10). There is more of level 2 ”explaining the meaning” and level 3 ”reasoning” compared to level 1 ”pointing” (Figure 15, Table 10). Reporting of an ”error situation” on level 2 that is not related in finding the bug is connected with level 3 ”reasoning” in T(601, 606, 676, 694 and 712) and level 5 ”guessing” in T(188, 236, 260, 612, 669 and 695) of utterance (Figure 4, Figure 5). Reporting of a ”source of an error” on level 3 is connected with level 3 ”reasoning” of utterance in T(580) (Figure 4). There is a lot of gaze related situations during the ”first animation” and the ”second animation” (Figure 4, Figure 5, Figure 15, Table 10). Level 1 ”sees<>tells” of gaze is connected with level 2 ”explaining the meaning” in T(431, 937, 939) and level 3 ”reasoning” in T(412, 930) of utterance (Figure 4, Figure 5, Table 10). Level 2 ”sees<>expects” of gaze is connected with level 1 ”pointing” in T(460, 479) and level 5 ”guessing” in T(669) of utterance (Figure 4, Figure 5). Rest of the gaze level 2 are only given on the level 4 ”visualization” of origin (Figure 4, Table 10) in T(371, 536 and 557).

Introduction

The participant P1 reads the instructions silently for 59 seconds.

Gaze in general

During the ”first animation” (Figure 4), there are some level 1 situations ”sees <> tells”. Rest of the gaze concerns level 2 where the participant ”sees <> expects”.

In T(455), the participant expects program to print names but sees NULL printed in T(460) and in T(479).

When animation is showing calculation of average age the participant expects to see 4 in T(535) but the animation shows different in T(536). In T(556), when calculating the average age the participant expects program to add ages but in T(557) program is seen to act different.

During the ”second animation” (Figure 5), the balance between the amount of levels is opposite to the ”first animation”. There are more level 2 gaze situations where the participant ”tells different from seen”.

In T(667), the participant P1 expects the following stage considers operating with the name but in T(669) program is seen to act different. Situations in T(930), T(937) and T(939) consider the same situation about boolean method isAChild and the return statement ”return (age > 18)”. The participant is explaining that the program is showing that the age in question is not under 18. ”>” is seen as ”<” and ”true” is seen as ”false”.

Gaze about ”age”

This particular session includes an interesting situations where the participant P1 sees > 18 and interprets it as < 18 when testing whether a person is under aged — a child.

1) During the ”first animation” (Figure 4) in T(321) the participant P1 tells that program verifies persons as being children. When the program verifies a person who is over 18

as a child, the participant experiences confusion with the on going situation in T(325).

2) In T(370), the participant P1 expects that a person is not considered as a child — 40 < 18 = false — but in T(371) the animation shows different — 40 > 18 = true.

3) In T(412) a conflict occurs when the participant P1 tells the person is a child when program tells the age is 10 > 18 = false.

4) In T(431) a conflict occurs when the participant P1 tells 5 < 18 when in fact animation shows 5 > 18 = false.

5) During the ”second animation” (Figure 5) starting from T(937) P1 tells, that the program is comparing whether the age is < 18 and the result being false when animation shows 35 > 18 = true.

In that first situation when the person is tested as under aged, the participant has no contradiction between told — ”it checks if they are children...if it's older then..” — and seen — 35 > 18 = false. Anyhow, the participant is confused how this fact affects the following actions in the animation. At the second time the participant is hesitating if the interpretation about what is happening in the animation is correct. At the third time the participant is quite sure that 10 > 18 = false means that a person is proven to be a child. At the fourth time the participant tells fluently 5 > 18 = false as being ”5 is less than 18”. Later during the second run the participant tells whit no doubt quite the opposite as seen. Even if in the beginning the participant has an opportunity to construct an authentic model the outcome is little by little strengthening mental model where the participant sees the expected.

Finding the bug B1

During the ”first animation” (Figure 4) the participant P1 spots that ”name” variables are saved falsely in the structure in T(188). The participant is sure that ”names” are falsely saved in T(606) and assumes that the names were not saved at all in T(612). During the ”second animation” (Figure 5) the participant tells there is something wrong with the ”name” in T(669). The participant finds out that the purposed ”name” for the person was not used in T(676). The participant tells there was again something wrong with the ”names” in (694). The participant reports that the program takes gender twice when a ”name” should have been in process in T(712). The participant locates the error (Figure 5) in the code in T(740) and gives it a fix in T(742). The participant gives error the line number in T(748) and the class in T(752). The first and only bug — a statement where ”psex” should be ”pname” — is found during this particular session in a whole.

Faulty piece of a code

During the ”first animation” (Figure 4) the participant P1 assumes that the number of family members, that is value for variable ”numberOfPeople”, is calculated falsely in T(236).

During the ”first animation” (figure 4) the participant P1 tells there is something wrong when calculating the ”average age” e.g. ages are not added up in T(560). The participant finds the source for the error — ”ages” — are not added up when calculating the ”average age” in T(580). The participant is sure that the ”ages” were not added up in T(601).

Levels of Utterance

Utterances (Figure 4, Figure 5) vary from 1 ”pointing”, 2 explaining, 3 reasoning and 5 guessing. The minority is just reading the code and more effort is given to explaining, reasoning and guessing.

Origin

Origin of think-aloud protocol is mostly on level 4 ”visualization”. The participant P1 refers to level 2 ”output” during the ”first animation” (Figure 4) and level 1 ”code” when finding the bug B1 during the ”second animation” (Figure 5).

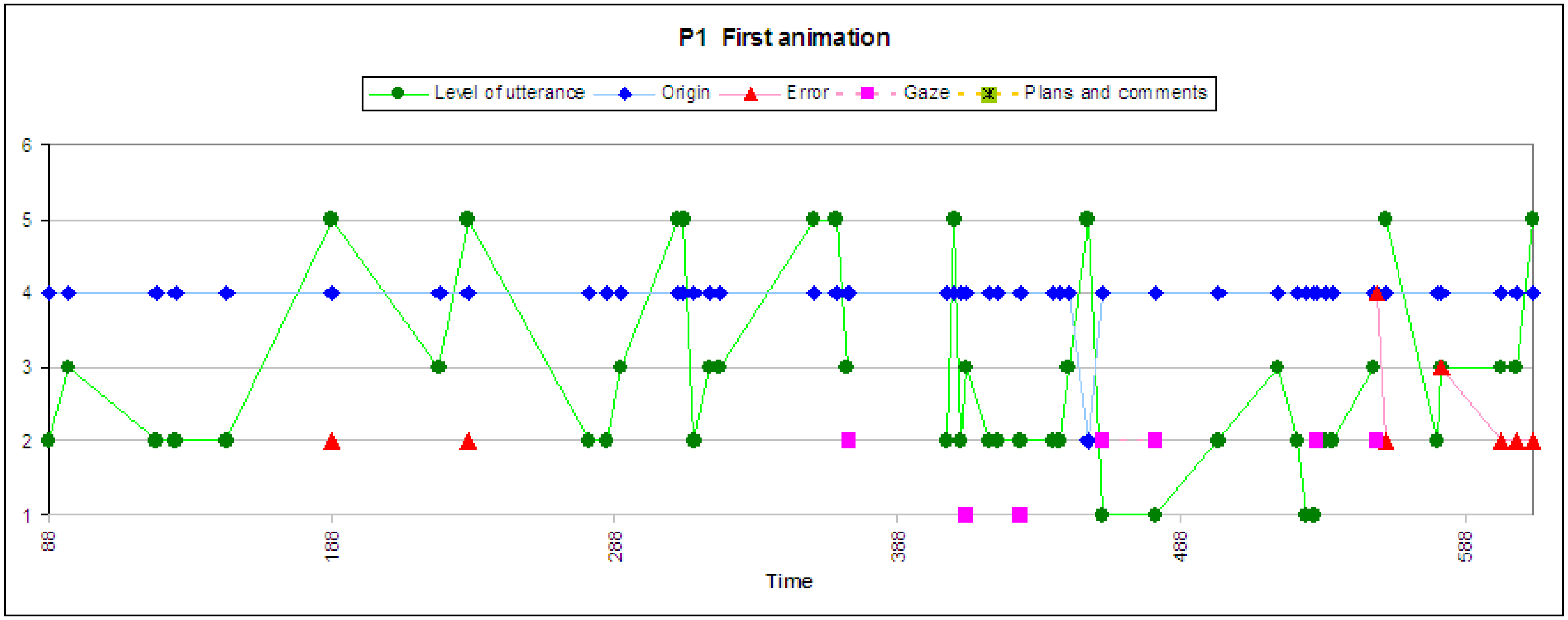


Figure 4: Levels of categories during the ”first animation” performed by P1. See chapter 3.4 and Table 8 for more details in interpreting the charts.

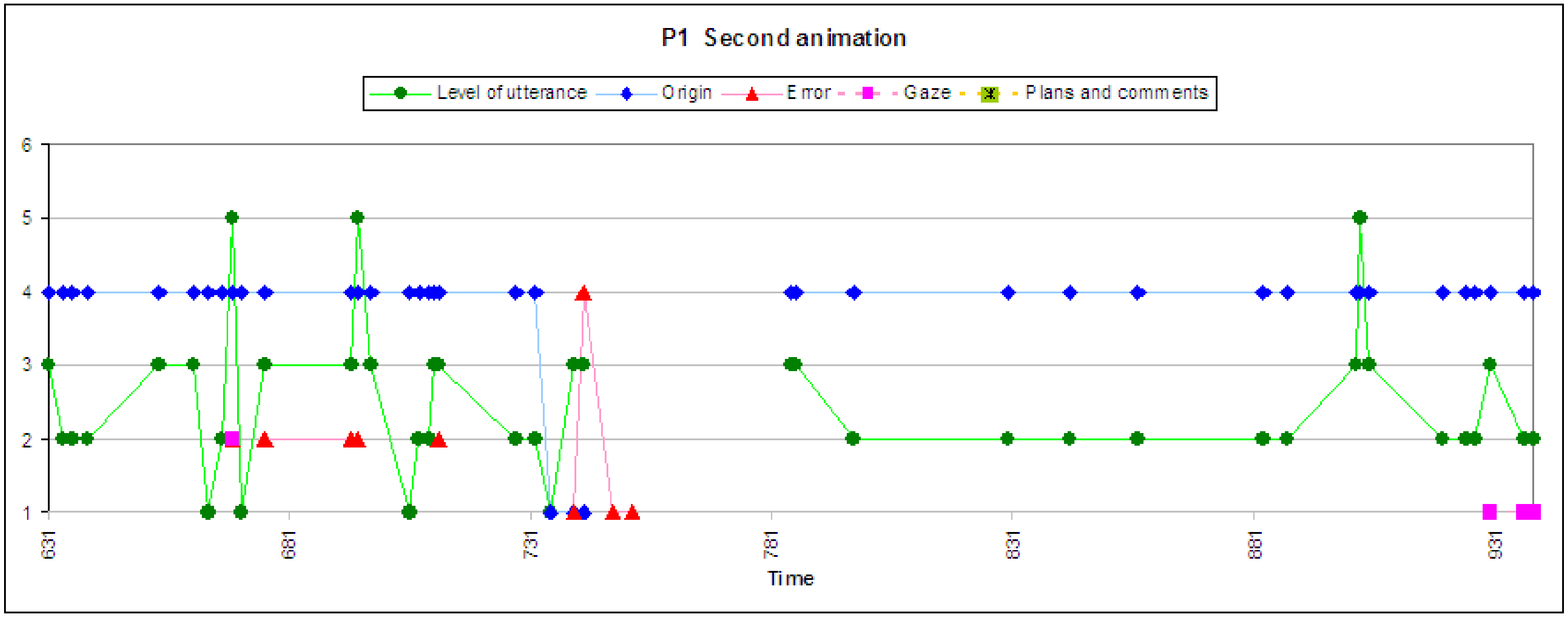


Figure 5: Levels of categories during the ”second animation” performed by participant P1. See chapter 3.4 and Table 8 for more details in interpreting the charts.

4.2 Participant P2

General view

The participant P2 begins with reading the instructions. The participant concentrates on the code and instructions during the rest of the session. The participant P2's origin of think-aloud protocol is mostly on level 1 ”code” and level 3 ”instructions” (Figure 16, Table 10). Level 3 ”instructions” is connected to the level 3 ”reasoning” in T(395, 630, 635, 691), 4 questions in T(397, 677, 919), 5 guessing in T(830, 944), 6 validating in T(906) of utterance (Figure 6). Level 3 ”instructions” of origin is also connected with level 1 ”explaining the deeds” in T(721) and level 2 ”comment” in T(113) of plans and comments (Figure 6). All the levels of utterance are used (Figure 16). The most used levels of utterance are ”reasoning”, ”guessing” and ”asking questions” (Figure 16, Table 10). The most used level of error is ”location” (Figure 16). ”Error locations” on level 1 when not related in finding the bugs are connected with level 1 ”is doing” in T(719, 721) and level 2 ”comment” in T(872) of plans and comments (Figure 16, Table 10). ”Error locations” not related to finding bugs are also connected with level 1 ”pointing” in T(851, 863), level 3 ”reasoning” in T(884), level 4 ”asking questions” in T(847, 917, 919) and level 5 ”guessing” of utterance in T(827, 895, 907) (Figure 6). There is two appearances for gaze during the debugging session (Figure 16, Table 10). Gaze on level 2 ”sees <> expects” in T(354) and T(357) is only connected with level 1 ”code” of origin (Figure 6).

Introduction

The participant P2 makes oneself familiar with the instructions quite a long time — 150 seconds — before starting to operate with the Jeliot. The participant is also concerned if instructions are truly understood.

Getting started

The participant P2 seems to be a little bit of lost in the beginning of code browsing. Generally, the participant starts to read the code from the beginning and tries to find some sense in the code — that is, he/she finds similarities between instructions and the code.

Starting in T(180), the participant P2 tells that boolean isAchild returns age if it is over 18 (Figure 6). Actually it returns true but the participant should have seen the connection between isAchild and over 18 years old — that is, there is something wrong with this part of the program.

Finding the bug B6

Starting in T(387), the participant P2 tells trying to solve from the code why the output seen in instructions is falsely female instead of male (Figure 6). Starting in T(719), the participant tells trying to solve where in the code results the output with such an amount of gender. Starting in T(917) the participant is questioning the causes for the output with female and male. Starting in T(941), the participant suggests that after leaving ”p.print” out on the line 44 genders do not appear in the output anymore. The first bug B6 ”print outside else” is found.

Deduction of average age

Starting in T(467), referring to the average age and the outputs seen in instructions the participant P2 is adding the ages seen in the code up (Figure 6). As it follows, the participant looks outside the code to the instructions. Starting in T(626), the participant starts to deduct how the average age is calculated based on the amount of people — 4 —, ages given to each person and false average age seen in the instructions — 1.25 —. The participant P2 concludes that Abel’s age of 5 is divided with 4. Starting in T(895), the participant reports, that error where Abel’s age of 5 is divided with 4 resulting into 1.25, occurs on the line 111.

Finding a spelling mistake

Starting in T(677), the participant P2 start to question why ”Age” is written starting with an upper-case letter seen as the present output in the instructions and with a lower-case letter in the code (Figure 6). Starting in T(884), the participant reports that the typing error occurs on the line 110.

Finding a ”bug”

Starting in T(827), the participant P2 starts guessing if one error is included on the lines 101 and 102 and suggests replacing some of the code with ”family” (Figure 6). Starting in T(863), the participant states class being ”group”.

Finding the bug B7

Starting in T(1006) the participant P2 is guessing if there is something wrong with the if statement due to ”NextPersonIndex = 0” situated in ”nextPerson” method on the lines 50 and 51 (Figure 6). The second bug B7 — ”if” — during this debugging session is partly found.

An unexpected is seen

There is one situation (Figure 6) where the participant P2 sees different to expected about name ”Cain” in T(354) and T(357).

Levels of Utterance

All the levels of utterance from 1 to 6 are used (Figure 6). The participant P2 is solely using level 6 ”validating” when solving what happens when program is computing the average age in T(467) – T(474) and in T(905) – T(906).

Origin during the session

The origin of talk aloud is mostly referring to the code and to the instructions, at times (Figure 6). The reference to the instructions is made when the participant P2 is concerned in understanding the instructions in T(113). Reference to the output of the instructions is made when P2 is finding reasons in the code to false output seen in instructions for sex in T(395), T(397), T(721), T(830), T(919) and T(944). Another similar situation of false output detection concerns average age in T(630), T(635), T(906). Origin is about instructions when a spelling mistake is found in T(677) and T(691).

Plans and Comments

The participant P2 uses level 1 ”is doing” of Plans and comments (Figure 6) when telling the reason for being silent while browsing the code in T(261) and when starting to take a closer look on one of the methods in T(316). Level 1 ”is doing” is also used when the participant explains starting to detect false output of sexes in T(387), T(719) and T(721). The participant uses level 2 ”comment” when giving a fix to an error in T(843). Level 2 ”comment” is also used when the participant is telling how difficult the program is when trying to find out where the bugs are in a fast way in T(872).

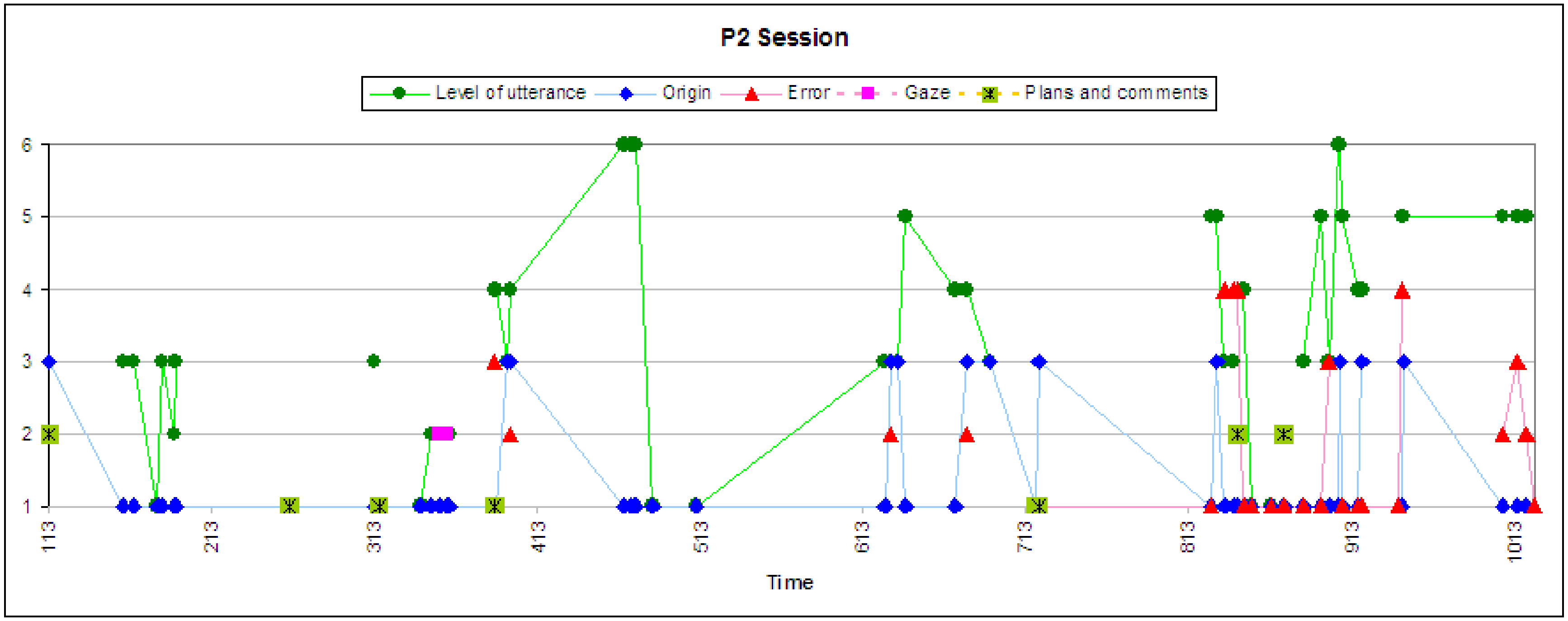


Figure 6: Levels of categories during the debugging session performed by participant P2. See chapter 3.4 and Table 8 for more details in interpreting the charts.

4.3 Participant P3

General view

Debugging session constitutes of reading the instructions, code browsing and getting a more profound understanding of the program with the help of visualization. The participant P3's origin of think-aloud protocol is mainly about code and visualization (Figure 7, Figure 8, Figure 17, Table 10). Level 4 "asking questions" of utterance seems to be the skeleton for the session. The participant asks a lot of questions and finds the answers via level 1 "pointing", 2 "explaining the meaning", 3 "reasoning" or level 5 "guessing" (Figure 17, Table 10). "Location", "situation" and "source" of error are included (Figure 17). The participant talks on levels of error not related in finding the bugs when asking one "question" (Figure 7) about reporting in T(70) and "commenting" (Figure 8) the lack of errors in T(905). Gaze on level 2"sees <> expects" in T(421) refers to level 1 "code" of origin and in T(546 and 882) to level 4 "visualization" of origin (Figure 7, Figure 8, Figure 17, Table 10).

Introduction

During the introduction the participant P3 is concerned and asks more questions compared to the novices. The participant reads the instructions for 110 seconds that is between the times novices used for this task. Debugging session constitutes of the "beginning" and "stepping in animation". The "beginning" includes reading the instructions and browsing the code when "stepping in animation" includes using the possibilities granted by animation and comparing the code to the animation.

From code to visualization

The participant P3 starts browsing the code from the top, finds the meaning for the methods, which methods belong to which class and what happens in the main method. In the code, "description" causes questions starting in T(344) and "people" in T(424) (Figure 7). After this the participant executes the animation in T(480).

Levels of Utterance

In the "beginning" (Figure 7) and "stepping in animation" (Figure 8) the participant P3 is talking mostly on the levels of 2 "explaining", 3 "reasoning" and 4 "asking questions".

Origin

In the ”beginning” (Figure 7) origin concerns mostly level 1 ”code” but the ”questions” before entering Jeliot 3 are about level 3 ”instructions” in T(6) and T(12). During the ”stepping in animation” (Figure 8) the participant P3 is making a lot of comparison between the ”code and visualization” on level 5 or ”output and visualization” on level 6. In addition, there is a lot of change between level 4 ”visualization” and level 1 ”code” (Figure 8).

Gaze

In the ”beginning” in T(421), while browsing the code the participant P3 is confused about ”People” (Figure 7). In ”stepping in animation” in T(546) the participant P3 ”expects to see” the description but sees different (Figure 8). In ”stepping in animation” in T(882), the animation does not do the ”expected”.

Finding the bug B2

The participant P3 finds the first bug B2 that is ”age>18” in T(938) (Figure 8). The participant reports class being ”Person” on the line 19 in T(952).

Finding the bug B3

The second bug B3 is found in T(982) on the line 61 is ”Sum=” that concludes the error in calculating the average age (Figure 8).

Plans and Comments

The participant P3 ”explains” present deeds on level 1 in the ”beginning” (Figure 7) and ”comments” code on level 2. ”Stepping in animation” (Figure 8) involves ”commenting” complexity of the visualization in T(783), ”visualization” in T(800), lack of ”errors” that are being found in T(905) and how the participant has some understanding of the program in T(989).

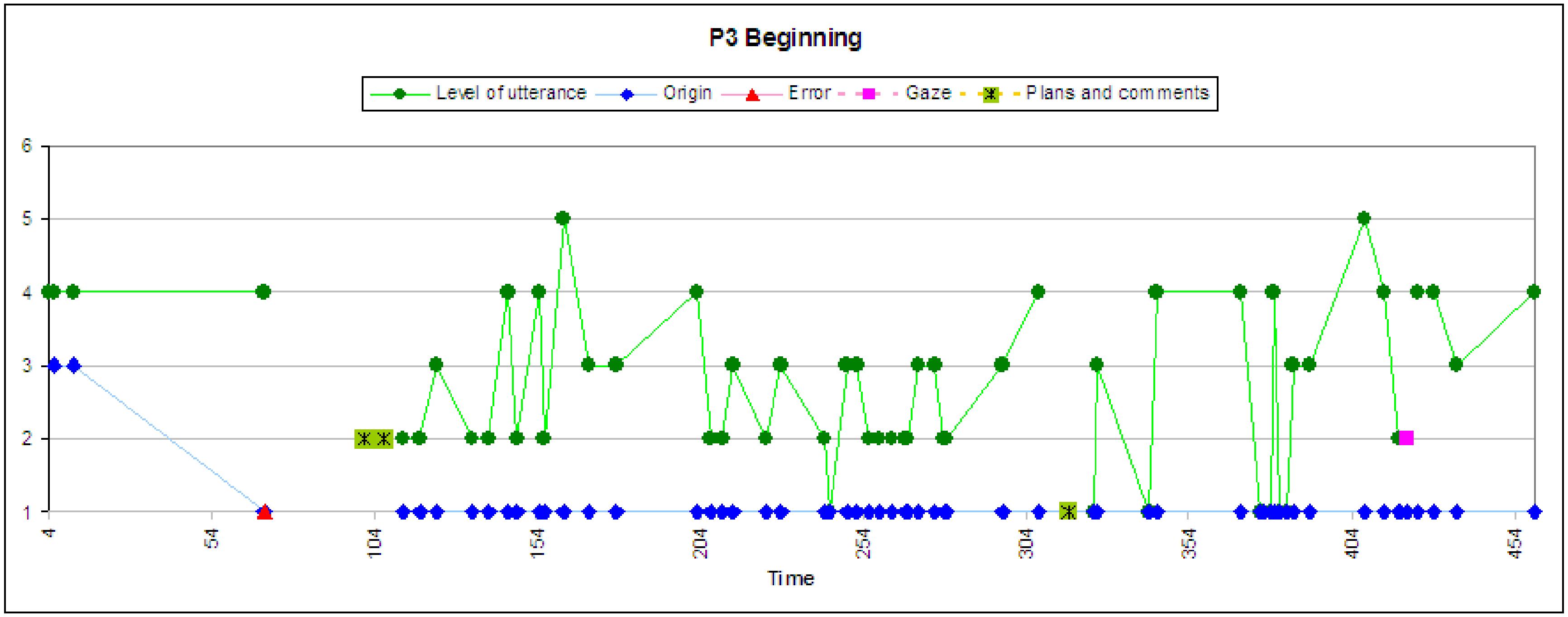


Figure 7: Levels of categories during the ”beginning” of debugging session performed by participant P3. See chapter 3.4 and Table 8 for more details in interpreting the charts.

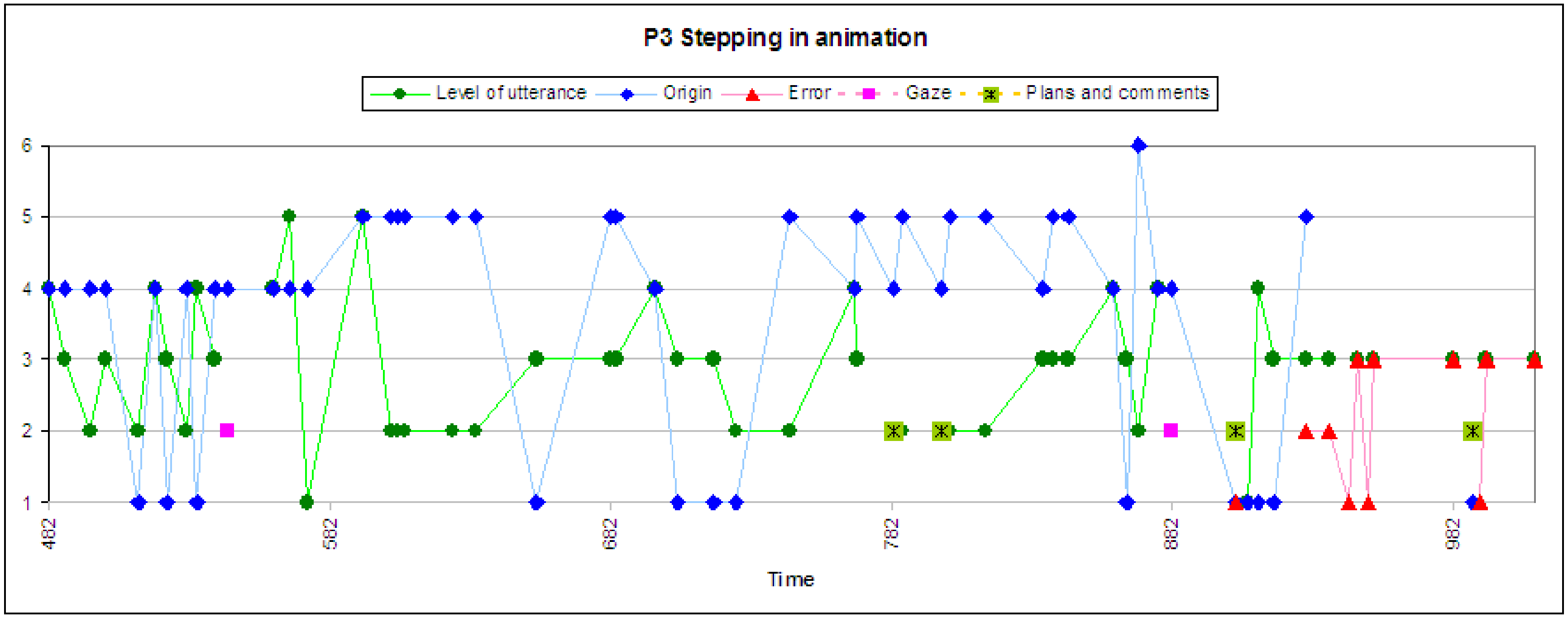


Figure 8: Levels of categories during the ”stepping in animation” performed by participant P3. See chapter 3.4 and Table 8 for more details in interpreting the charts.

4.4 Participant P4

General view

The participant's debugging session consists of six smaller sections: reading the instructions, running the animation for the first time, getting to know the program by reading the code, running the animation for the second time, returning back to reading the code, running the animation for the third time. The participant P4's origin of think-aloud protocol is mainly about code and visualization (Figure 18). The participant uses mainly the level 3 ”reasoning” and 2 ”explaining the meaning” of utterance (Figure 18). The participant is ”guessing” only in T(220) (Figure 11) related to ”code” and in T(747) (Figure 12) related to the ”visualization”. Three sections of session include error related talk (Figure 9, Figure 11, Figure 12). Levels of error that are related to found bugs are situated in ”intermezzo” (Figure 11). ”Error situation” of the ”beginning” in T(126) is connected to the ”instructions” (Figure 9). ”Error situations” in the ”second animation” in T(736, 728 and 755) are connected to ”visualization” and ”reasoning” (Figure 12). There are no signs of gaze (Figure 18).

Introduction

The participant P4 reads the instructions for 107 seconds that is between the times of the two novices.

Incorrect code and Bugs

1) The participant P4 notes that interoutputs are overly done in T(126) (Figure 9).

2) The first bug B1 — ”psex” — is found on the line 12 in class Person in T(222) during the ”intermezzo” (Figure 11).

3) The second bug B2 — ”age>18” — (Figure 11) is found on the line 17 in class Person in T(280).

4) In T(364), the participant notes that there is something wrong on the line 39 in Include method and class Group (Figure 11). The participant P4 notes that maybe it is correct in T(388).

5) The third bug B7 — ”if” — (Figure 11) is found in T(474) on the line 50 in ”If” clause and class ”Group”.

6) The participant notes that ”age” is falsely computed in T(516) (Figure 11).

7) The fourth bug B4 — ”people.lenght” — (Figure 11) is partly found in T(557).

8) During the ”second animation” (Figure 12) the participant P4 notes that ”Descriptions” are not going to the right places in T(738). The participant also notes that calculating the number of children fails in T(755).

Levels of Utterance

The ”beginning” (Figure 9) involves utterances of level 4 ”asking questions” and 3 ”reasoning”. During the ”first animation” (Figure 10) utterances are in 1 ”pointing” and 2 ”explaining the meaning”. ”Intermezzo” (Figure 11) involves levels 1 ”pointing”, 2 ”explaining the meaning”, 3 ”reasoning” and 5 ”guessing”. The ”second animation” (Figure 12) is mostly in level 3 ”reasoning”. The ”intermezzo II” (Figure 13) has levels 1 ”pointing”, 2 ”explaining”, 3 ”reasoning” and 4 ”asking questions”.

Origin

In the ”beginning” (Figure 9) the participant P4 origin of think-aloud protocol is on level 1 ”code” and level 3 ”instructions”. During the ”first animation” (Figure 10) origin is on level 2 ”output” and level 4 ”visualization”. During the ”intermezzo” (Figure 11) where code browsing is done the origins are on level 1 ”code”, 2 ”output” and 3 ”instructions”. During the ”second animation” (Figure 12) the origins are in ”code”, 2 ”output”, 3 ”instructions” and 4 ”visualization’. ”Instructions” on level 3 is due the situation where the participant is comparing output of the ”instructions” (female, male, male, male) in T(674) to the ”output” of the visualization (Eve, Adam, Cain, Abel) in T(680) (Figure 15). During the ”intermezzo II” (Figure 13) when the participant returns to code browsing the origin is solely on level 1 ”code”. During the ”third animation” (Figure 14) the origin is on level 4 ”visualization”.

Plans and Comments

Plans and comments is used through the session. In the ”beginning” (Figure 9) on level 3 the participant P4 is ”planning future actions” in T(118) and T(120) that concerns how the debugging tasks should be started. During the ”first animation” (Figure 10), the participant uses level 1 ”is doing” when he/she is explaining what he/she is doing at the time. In ”first animation”, also, he/she uses level 2 when he/she is ”commenting” about dislike towards animations in general in T(191). The ”intermezzo”

(Figure 11) involves a level 2 ”comment” where the participant tells how introduction to the code takes fifteen minutes in T(587). During the ”second animation” (Figure 12) the participant explains the deeds being done and ”comments” about animation. In T(694) the participant makes a suggestion about improvements to the animation — that is, break points that could be included to Jeliot. The participant also mentions the lack of improvements made in code via fixing the bugs noticed before. In the ”third animation” (Figure 14) the participant P4 explains the deeds being done on level 1 and in level 2 when ”commenting” how visualization takes a lot of time in T(950).

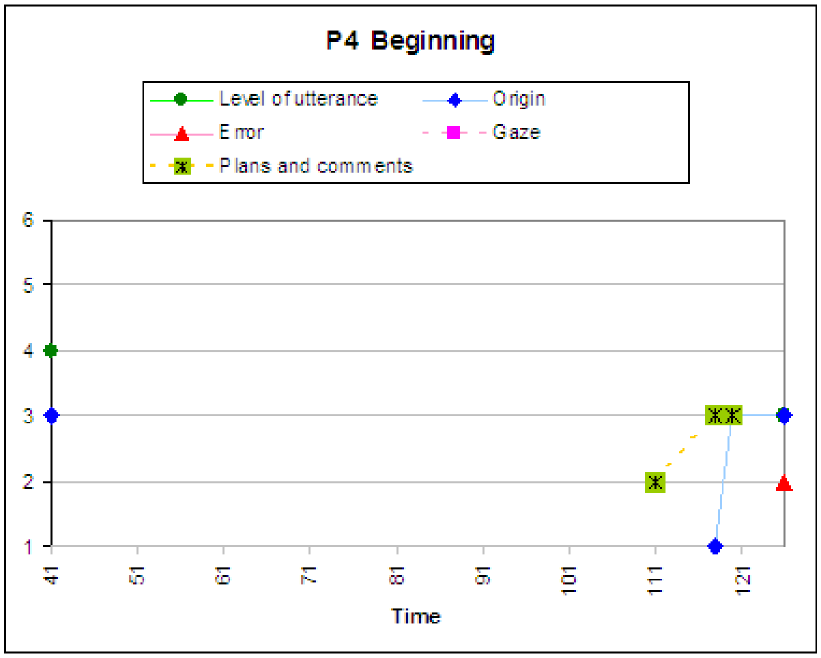


Figure 9: Levels of categories during the ”beginning” performed by participant P4. See chapter 3.4 and Table 8 for more details in interpreting the charts.

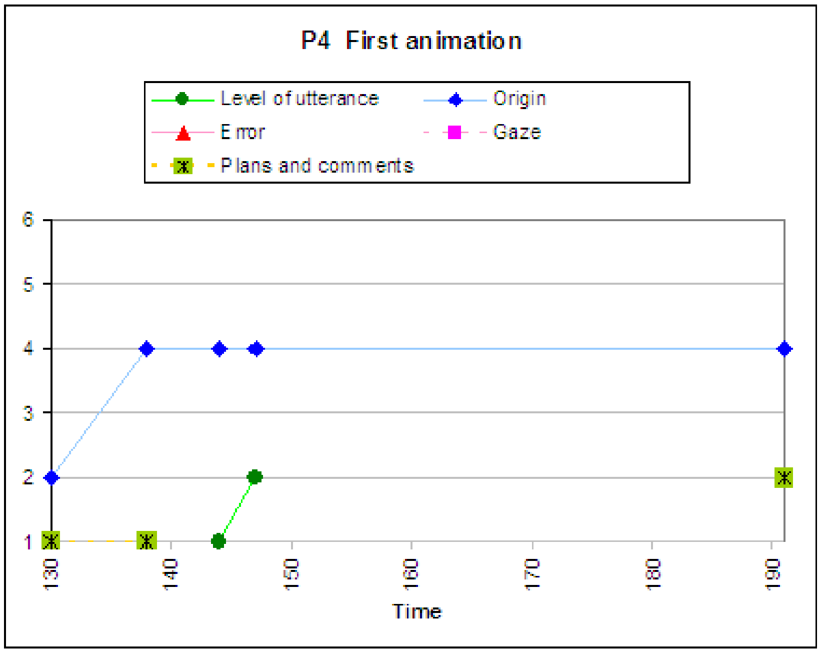


Figure 10: Levels of categories during the ”first animation” performed by participant P4. See chapter 3.4 and Table 8 for more details in interpreting the charts.

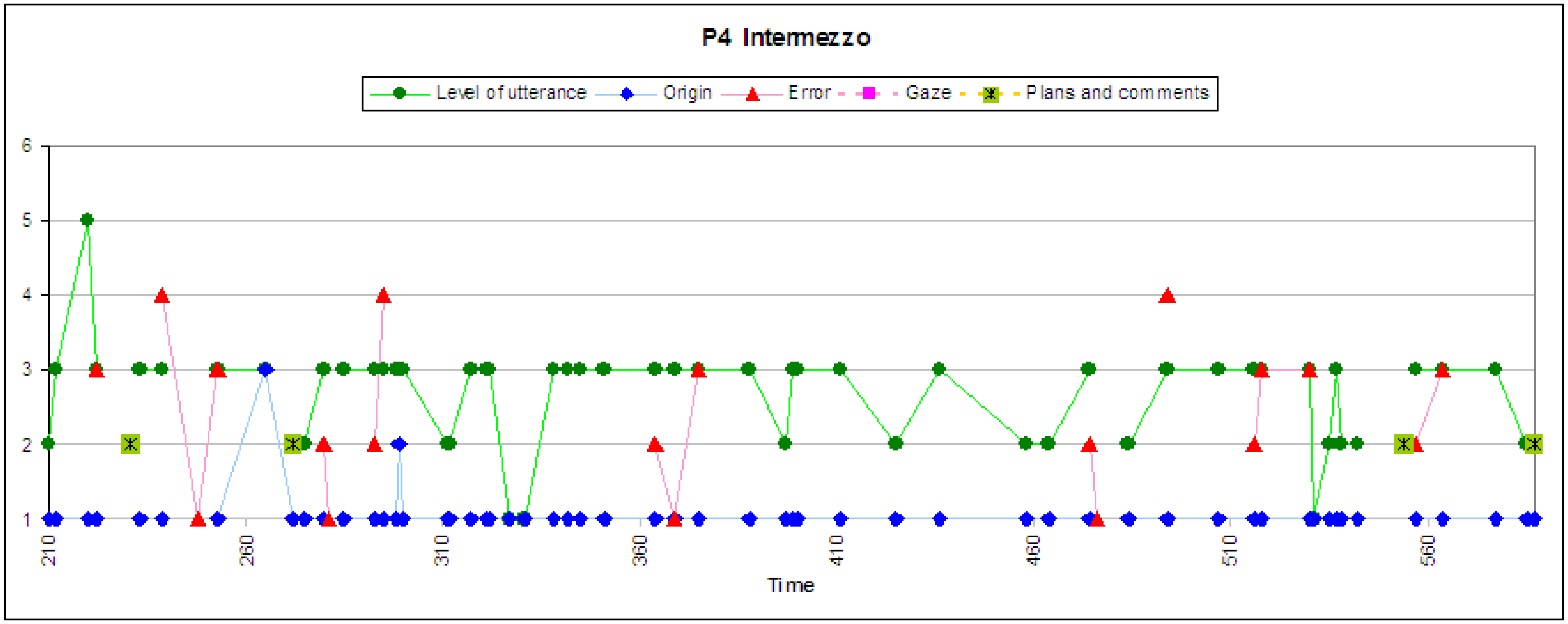


Figure 11: Levels of categories during the "intermezzo" performed by participant P4. See chapter 3.4 and Table 8 for more details in interpreting the charts.

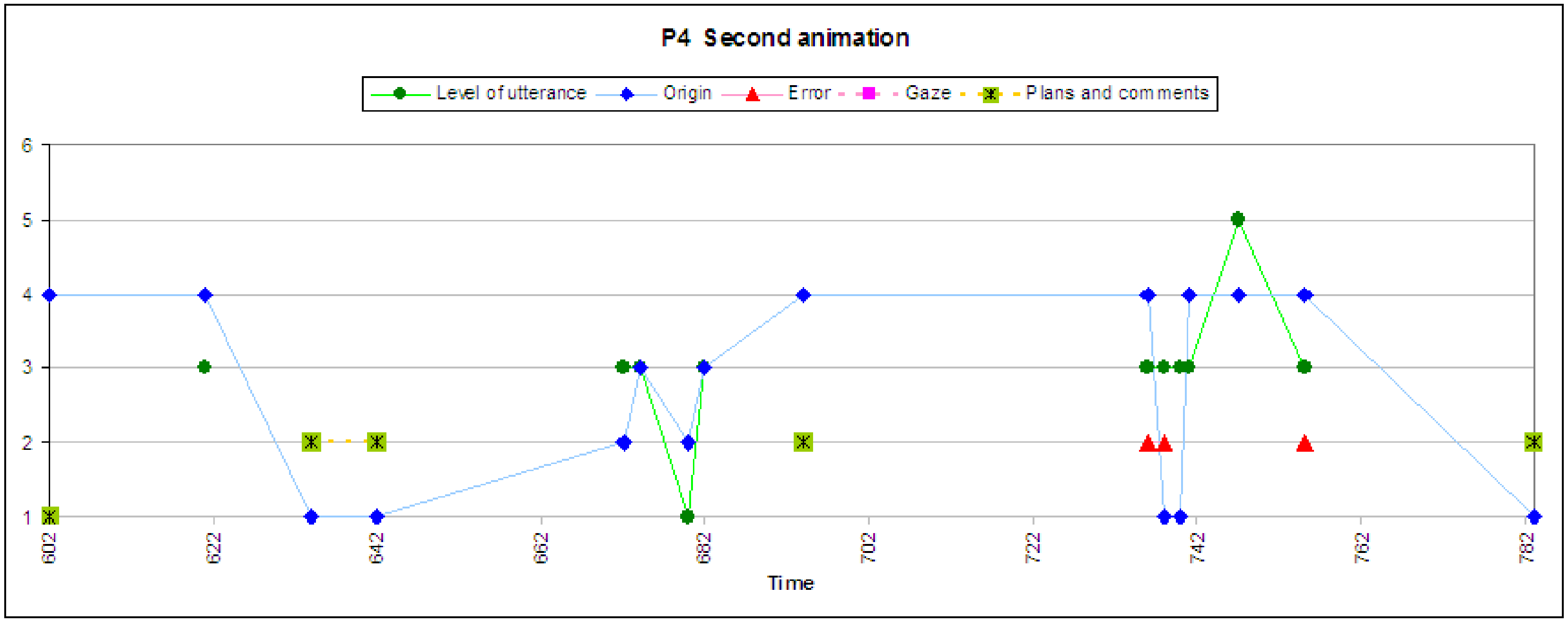


Figure 12: Levels of categories during the ”second animation” performed by participant P4. See chapter 3.4 and Table 8 for more details in interpreting the charts.

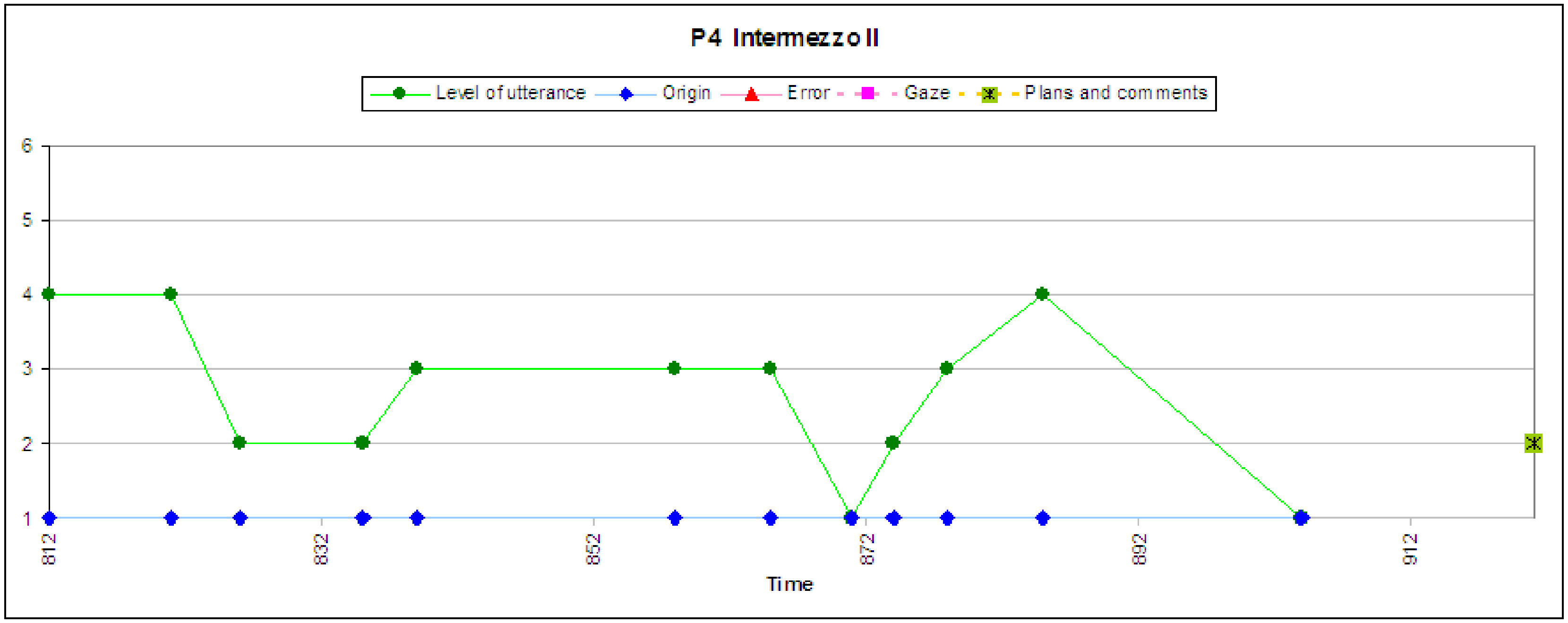


Figure 13: Levels of categories during the ”second intermezzo” performed by participant P4. See chapter 3.4 and Table 8 for more details in interpreting the charts.

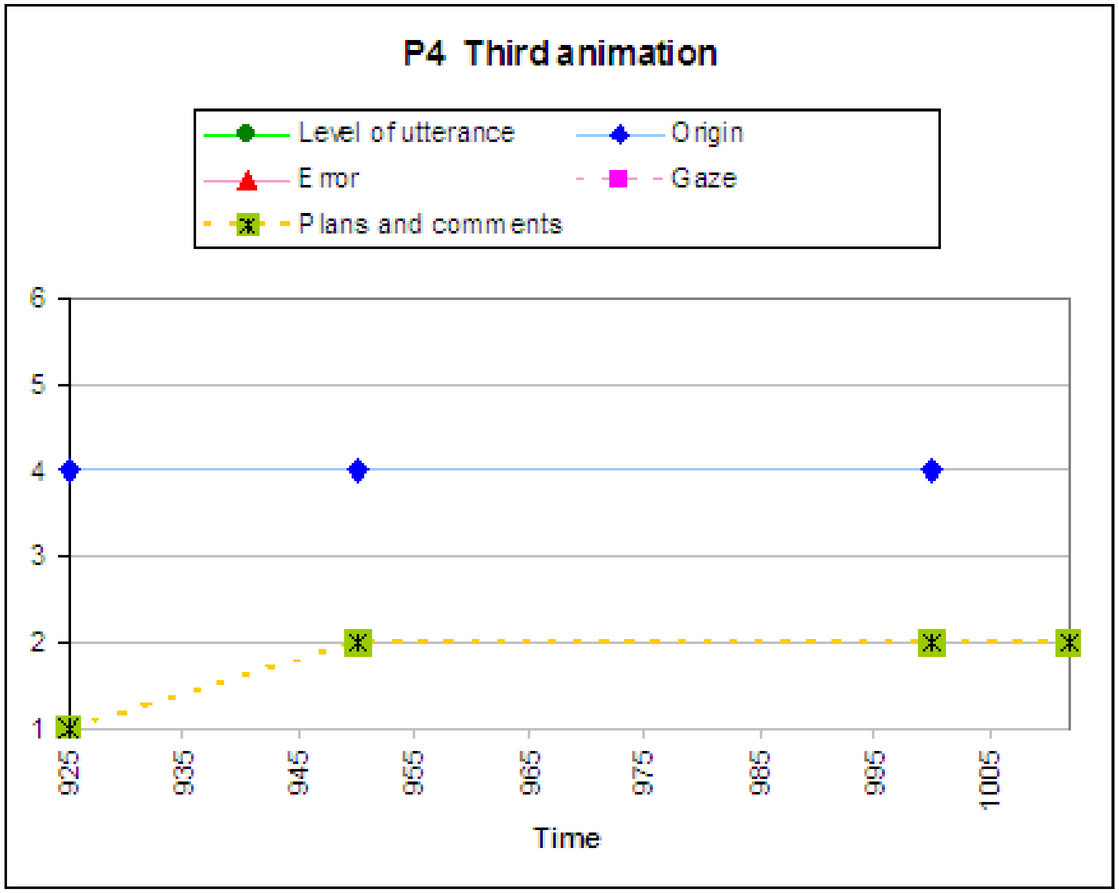


Figure 14: Levels of categories during the ”third animation” performed by participant P4. See chapter 3.4 and Table 8 for more details in interpreting the charts.

4.5 Summary

According to the background information, the participant P1 studies Computer Science as a major, has 4 months of Java experience with 10 written programs and has never used Jeliot (Table 1). In addition, the participant has one month of experience with other programming languages. The participant performs with high scores from the first tasks where the right output should be given, the second task where a binary tree is drawn and the third task of finding the errors in the given code. High scores in pre-test give an impression that the participant P1 understands something about debugging even he/she has little experience on programming. Nevertheless, only one bug is found (Table 9).

The participant P2 studies Mathematics as a major and computer science as a minor (Table 1). The participant has 5 months of Java programming experience with 100 written programs. According to the amount of written programs the participant could be called as an expert programmer but results from the pre-test show some signs of poor debugging skills. The participant performs well with the first task (right output), the second task (binary tree) fails and the third task (debugging) gives one point. Total points are three out of seven, that suggests he/she does not understand much about debugging. It also remains uncertain how demanding those 100 written programs truly were. The total amount of found bugs is 1.5 (Table 9).

The participant P3 studies Computer Science as a major and mathematics as a minor (Table 1). The participant has 0.5 months of Java programming experience with 10 written programs and has used Jeliot once before. This information implies that the participant is a novice programmer in Java but on the other hand the participant has 24 months of experience in other programming languages and gets maximum of seven points from the pre-test. This suggests a good understanding about reading the code and debugging. The total amount of found bugs is 2 (Table 9).

The participant P3 studies Computer Science as a major and mathematics as a minor (Table 1). The participant has 0.5 months of Java programming experience with 10 written programs and has used Jeliot once before. This information implies that the participant is a novice programmer in Java but on the other hand the participant has 24 months of experience in other programming languages and gets maximum of seven points from the pre-test. This suggests a good understanding about reading the code and debugging. The total amount of found bugs is 2 (Table 9).

The participant P4 studies Computer Science as a major and education as a minor (Table 1). The participant has 18 months experience in Java programming with 30 written programs. In addition, the participant has 50 months of experience on other programming languages and 50 times of using Jeliot. The participant P4 seems to be an expert programmer according to this information and debugging is also successful. The participant performs with full points in the first (the right output) and second task (binary tree) but the third task (debugging) gives only two points out of three. Nevertheless, the total sum of points reaches six out of seven. The total amount of found bugs is 3.5 (Table 9).

Some observations can be made about the differences between the participants. The participant P2 is the only to use ”validating” (Table 10). When comparing percentage quantities it becomes clear that ”explaining the meaning” is the most popular within P1 (Figure 15 - Figure 18). P2 gives the least attention for ”explaining the meaning” and P4 uses firstly ”reasoning” and ”explaining the meaning” is used secondly with significantly lesser frequency. ”Reasoning” is also popular among all of the participants and its frequency to ”reasoning” differs radically with P4. In addition, P2 gives the most attention to ”reasoning” and the least for ”explaining the meaning” but the difference between different levels of utterance is not as striking as it is for P4. P2 uses ”guessing”, ”asking questions”, ”validating” and ”pointing” quite evenly over ”explaining the meaning”. P2 uses ”explaining the meaning” clearly lesser than all the other participants do. ”Pointing” is nearly as popular with all of the participants. The participants P1 and P2 use ”guessing” clearly more than P3 and P4 do.

The focus of origin is truly standing out for each of the participants (Figure 15 - Figure 18). P1 favors ”visualization” when other participants favor ”code”. P2 also refers secondly to ”instructions” when P3 and P4 refer secondly to the levels that include visualization. ”Output” is the least mentioned by all of the participants. P2 and P4 are the ones significantly using ”instructions”. Nevertheless, P2 is merely using ”code” besides ”instructions” when P4 uses a variety of representations in addition. P1 and P2 use fewer or in lesser degree of variety of levels of origin when P3 and P4 give the most attention to ”code” and refer to the other levels, too.

P1 and P4 are mostly referring to error ”situations” when P2 prefers ”location” and P3 is evenly referring to ”location” and ”source” of an error (Figure 15 - Figure 18). P1 makes some references to ”location”, ”source” and ”fix”, too. Error related references of P3 concern quite evenly about ”location”, ”source” and ”situation”. P3 is the only not to suggest any fixes to the code. In addition to ”situations” P4 refers to ”source”, ”location” and ”fix”, too.

There are several references about ”sees<>tells” and ”sees<>expects” with P1 (Table 10). The few and only gaze situations for P2 are about ”sees<>expects”. P3 experiences a couple of times confusion between he/she is seeing to what he/she was expecting to see. P3 does not get confused about what he/she sees and is telling at that very moment. P4 does not have situations when he/she would tell different from what he/she is seeing.

Plans and comments are used by P2, P3 and P4 (Figure 15 - Figure 18). P3 and P4 refer to ”comment” the most when P2 is mostly explaining what he/she ”is doing”. P2 is also making comments when P3 and P4 are also referring to ”is doing”. P4 is the only to refer to ”planning future actions”.

Note: See chapter 3.4 and Table 8 for more details in interpreting the charts.

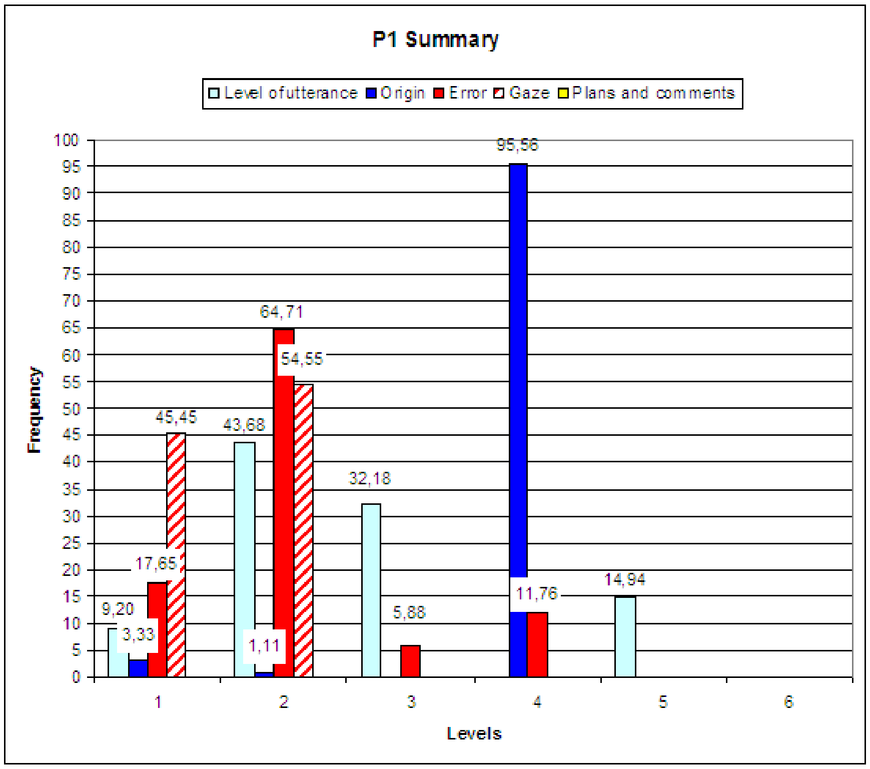


Figure 15: Summarizing the debugging session of participant P1.

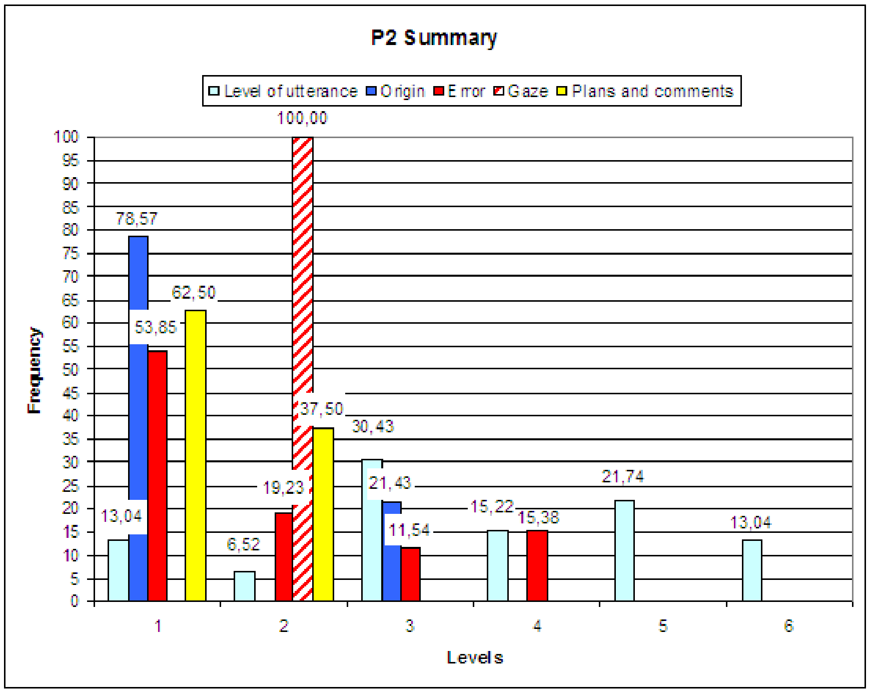


Figure 16: Summarizing the debugging session of participant P2.

Note: See chapter 3.4 and Table 8 for more details in interpreting the charts.

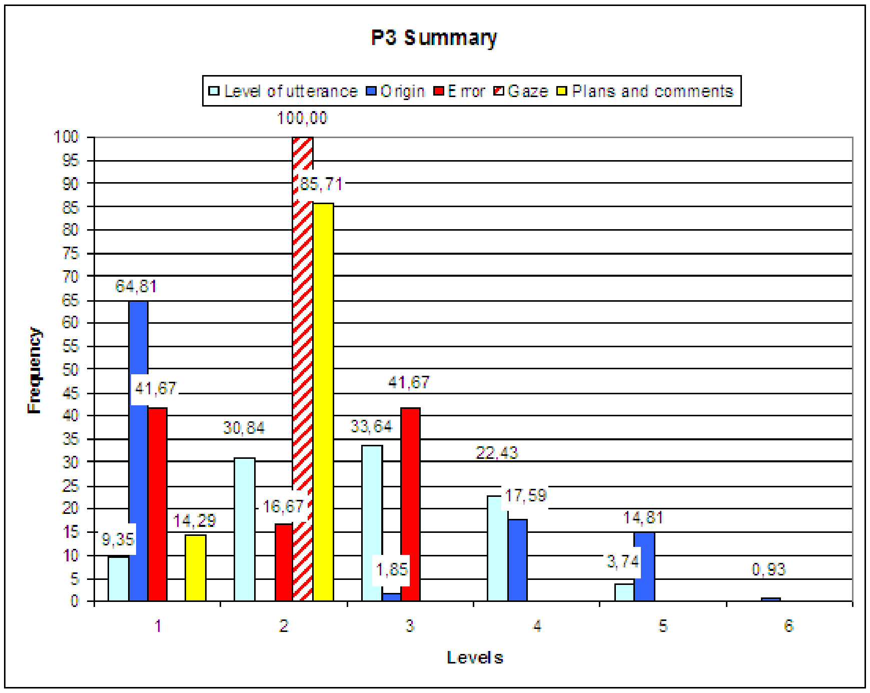


Figure 17: Summarizing the debugging session of participant P3.

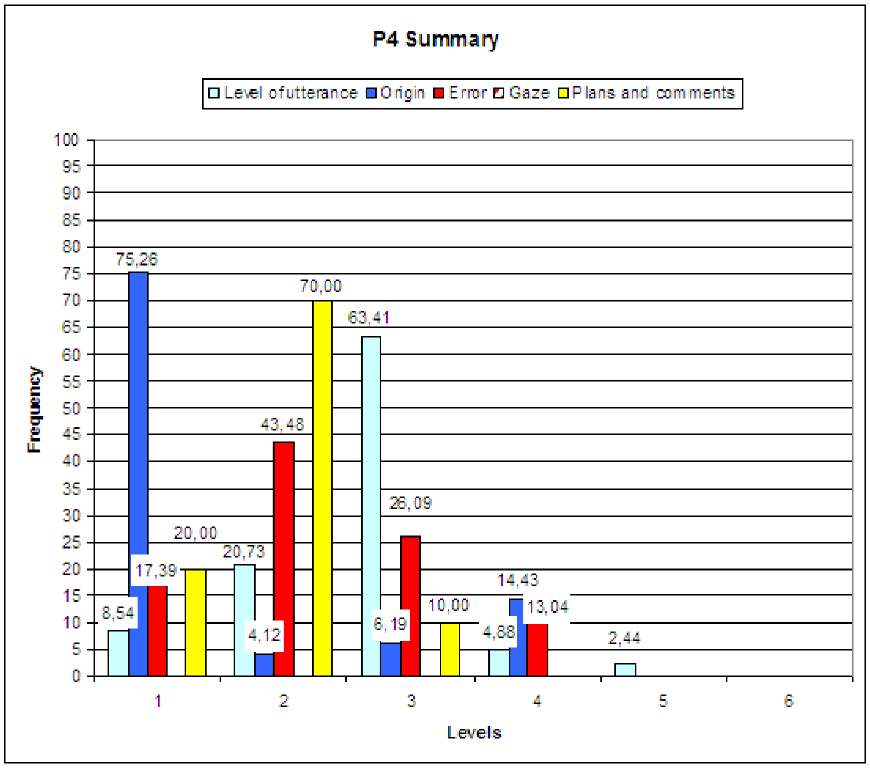


Figure 18: Summarizing the debugging session of participant P4.

Table 10: Total values of the talk aloud levels for each of the participants.

Category	*Total P1*	*Total P2*	*Total P3*	*Total P4*
Utterance				
1 ”pointing”	8	6	10	7
2 explaining the meaning	38	3	33	17
3 reasoning	28	14	36	52
4 asking questions	0	7	24	4
5 guessing	13	10	4	2
6 validating	0	6	0	0
Origin				
1 code	3	44	70	73
2 output	1	0	0	4
3 instructions	0	12	2	6
4 visualization	86	0	19	14
5 code<->visualization	0	0	16	0
6 output & visualization	0	0	1	0
Error				
1 location	3	14	5	4
2 situation	11	5	2	10
3 source	1	3	5	6
4 fix	2	4	0	3
Gaze				
1 sees<>tells	5	0	0	0
2 sees<>expects	6	2	3	0
Plans and comments				
1 is doing	0	5	1	4
2 comment	0	3	6	14
3 planning future actions	0	0	0	2

5 Discussion

Remarks on debugging performances

The participant P1 uses mainly visualization and fails in debugging. According to the background information P1 has the opportunity in successful debugging but the selected strategy does not work. One possible explanation for using merely visualization could be the results of the excitement caused by the whole situation. Other possible explanation could be that the participant is not too familiar with the Jeliot or may have an understanding that browsing the code is not acceptable. In the end, the participant chooses an unexpected way of trying to find the bugs by unfamiliar code visualization and does not get familiar with the code.

The participant P2 spends a lot of time getting to understand the given instructions and looking for hints about bugs by comparing the instructions' paper to the code. As it follows, the participant is likely to see the introduction and the output included as a basis for conclusions. The participant uses mathematical skills in finding the bugs — that is, he/she deducts how the average age should be calculated given the true outcome and the way program actually does it. After comparing the ”output” to the code and when trying to see all the differences and similarities the participant is likely to see the minor details including the typing mistakes, too. The participant has no earlier experience on Jeliot. Working in a strange Jeliot environment might explain why the participant trusts so much in instructions' paper with its written output and does not test the program with animation. It might be the case that the participant P2 does not know how to or is not familiar with visualization environments. It could be assumed that the participant practices debugging normally in a more of a text based environment with the code and output. The only tool for deduction, before, may be the comparison of similarities and differences between the code and the output. In addition, the mathematical deduction seems to fit due to major studies and if the participant does not use animation it has to be ”animated” using the mind.

Unexpectedly, the participant P3 uses the animation after making oneself familiar with the code and, this way, has the opportunity to gain a better understanding about the program when comparing the code to what is happening in the visualization. In the ”beginning” P3 reads the code as what it should be. Later in session the code is seen as it actually is when a clearer understanding of it is reached. The participant P3 executes animation when the information received solely on the code is not enough to provide complete understanding. The participant tries to get a better understanding of the program by comparing the code to the animation. The participant uses pauses and stepping to get the most of the animation. The participant is also thinking aloud more constantly than the novices P1 and P2. There are none of the long mute periods of times that the novices have. The participant P3 has a more of an active and hands on attitude in getting a wider view of the program. Debugging session is mainly getting to know the code by browsing it and getting more familiar with the program via animation. The two bugs are spotted ”accidentally” after knowing the code better. There is only few references related to errors in think-aloud protocol before finding the bugs. Thus, debugging seems to be more of a systematic act — that is, it does not include wild guessing if the bug lies here or there. The participant P3 does not give any suggestions about how to fix the found bugs.

Experience in Java and Jeliot may explain how the participant P4 is so comfortable using Jeliot, has time for self reflection, planning future actions. The participant also makes a comment about Jeliot e.g. how to improve and make debugging faster by involving break points to the code so the animation could start somewhere else than from the beginning. During the session the participant mentions lack of comfort with the animations in general and understanding the animation to demand knowledge from the code. This is shown in the way the participant reads carefully the code, after finding an error gives it a fix and from time to time performs the code with animation to see if there really are any improvements done. As a conclusion, after knowing the code, the participant has no need to see the whole code via animation but only those points of code where the bug may lie. The participant P4 has a very analytical and precise attitude on debugging. The participant is self aware of the things being done. The only quiet longer period of time is during the ”first animation” when the participant mentions that to understand animation is to understand code. The participant is the only one to rewrite code seen in Jeliot 3 code view.

Programming vs. Debugging

When talking about novices and experts in programming or in debugging, the distinction is not always so clear. Results of this work suggest that a student having a lot of programming experience is not necessarily an expert on debugging. It should also be noted, that the number of written programs does not always tell how skilled a programmer is. For instance, the participants P1 and P3 have both written 10 Java programs but

P3 have 24 months of experience in other languages, too. In addition, the participant P2 has written 100 Java programs in 5 months but the participant P4 has written 30 java programs in 18 months. A novice programmer can also succeed in theoretical debugging but fail in practice as the participant P1 did. The participants P3 and P4 both have several months of experience on other programming languages, too, when the other participants are mostly familiar with the Java programming language. The best debugger P4 has the longest and widest experience on programming with Java and other programming languages. Ahmadzadeh & Elliman (2005) finds that most of the good debuggers are good programmers when not even half of the good programmers are good in debugging. The key issue is the knowledge about the code. Because the participants P1 and P2 have the shortest programming experience they are referred as novices and P3 and P4 are seen as experts in programming. The novice programmers prove themselves as novice debuggers and the expert programmers as expert, or at least better, debuggers.

Debugging strategies

It was meant to find how debugging strategies can be revealed using eye-gaze data during debugging in Jeliot (1st Question). When a participant is not speaking, all there is to follow is his or hers eye-gaze. It may give some cues about strategies e.g. the participant is repeatedly looking some part of the code, eye-gaze disappears — that is, instructions are viewed and eye-gaze returns back to the lines of code viewed before. It is difficult to make any certain conclusions about a participant's strategies based on his or hers eye-gaze alone. In addition, in this study, eye-gaze data is analysed, only, when it is not convergent with the think-aloud protocol or the participant is confused about what he/she is seeing at the moment.

Debugging strategies found by Gugerty & Olson (1986) as shown before (2.1) were as follows. Introduction results to hypothesis and idea of the program. A Hypothesis is tested via modifying the code, executing it or finding support from purpose description. If testing does not conclude to found error follows returning to the introduction.

Strategies discovered by Romero et al. (2003) were shown (2.1) in a following nature. Introduction results to ”suspicious code” that is found. If ”suspicious code” is not involving an error then comparison is made between introduction and different presentations of the code.

The first step in this experiment is introduction. Reading the description and seeing the differences between the current and the correct output is a part of the settings. The participants reads the introductory descriptions and are allowed to enter Jeliot 3 environment afterwards. Participants P2, P3 and P4 glance the beginning of the code while reading the instructions before clicking the start-button and entering the Jeliot.

Debugging strategy of the participant P1 is similar to Gugerty's & Olson's (1986) findings. The participant P1 begins with the paper of instructions and tries to get a possible idea of the errors in the program. P1 uses animation, spots areas where something is wrong and returns to rerun the animation once again to get a clearer look of the faulty points seen in visualization.

Debugging strategies of the participants P2 and P3 are similar to findings of Romero et al (2003). The participant P2 starts with the instructions in getting a possible idea of the errors. P2 browses the code and looks for support to his/her hypotheses in the instructions and aims in finding the bugs. The participant P3 starts with the instructions in getting some ideas about the errors. P3 browses the code and later on compares the code to the animation. This way P3 attempts to find some support for the hypotheses and find the bugs.

The participant P4 is using kind of a mixture of both strategies presented (Gugerty & Olson, 1986; Romero et al., 2003). The participant starts with getting familiar with the instructions. This way he/she can get a possible idea of the errors and makes plans how to start the debugging. P4 decides to get started with the animation, but returns briefly in browsing the code. When P4 finds an error, he/she gives a fix to it and rewrites the code. P4 uses the animation to see if the changes, that were made, have fixed a bug. The participant uses the output in the instructions in getting hints about the bugs and compares the outputs in the animation and the instructions.

The selected strategies influence on the debugging performance, too. The novice P1 relies mainly on the visualization and the other novice P2 trusts mainly on the code but uses instruction strongly on the side. The expert P3 relies mainly on the code but uses a lot of visualization and makes comparison between these two representations. The expert P4 relies mainly on the code but uses the visualization, output and the instructions in realizing what is wrong with the program and if the fix for the code gives the wanted results. It is possible that getting to know the program via code and secondly via visualization helps in giving a better understanding about the program and later finding the bugs. Staying strictly to the visualization gives the poorest results for the novice P1 and using no visualization at all gives the second worst results for P2 in finding the bugs. These considerations imply that visualization by itself is not a sufficient tool in debugging but when it is purposefully used along with other representations it is rather efficient.

Eye movement vs. Think-aloud protocol

In the beginning of the research, one of the questions was if some special situations could be found via comparing the gaze-data to the think-aloud protocol(2nd Question). Generally, if a participant is talking while viewing e.g the code, it is possible to find some connection between these two. Think-aloud protocol may provide the reasons for the object of eye-gaze e.g. a participant is looking for answer for faulty output in certain lines of code or asking questions about what is being visualized at the moment. Generally, the participants have the same origin — that is, the code, output, instructions or visualization — in both but in some occasions gaze-data and think-aloud protocol together revealed misconstructions about the seen. These differing situations were coded as two levels of ”gaze”. The participant P1, especially, has a clearly evident misconstruction about the way the program works when it should validate when the person is under aged or not. As a matter of fact, the category gaze with the levels 1 ”sees<>tells” and 2 ”sees<>expects” were due to the problems the participant P1 had.

Level 2 gaze could be seen as being closely connected to the mental models or ideas about what happens next in animation or what should be written in the code. The present mental model is conflicting with the seen and caused confusion. In addition, confusion attracts further attention or the conflict is passed with no particular attention at all. Anyhow, the participant can not be said being ignorant about the fact that under aged are under eighteen or ”< 18”. As stated in the results (Chapter 4.1) participant is going in the right direction with the interpretations but he/she ended up to the opposite direction and stayed there the rest of the debugging session. Maybe the false mental model is so strong that the participant ignores the later correct clues completely. The correct clues have no value because the participant is possibly so sure that the program does not have any problems in this matter. This can also be considered as a learning situation: the first and wrong way is difficult to relearn into the right way later. Furthermore, what you see does not necessarily mean that you interpret the information received from the seen the correct way.

The participants P2 and P3 have some situations of gaze on level 2 ”sees<>expects”. A participant has had some expectations about what is going to happen or what he/she should be seeing next. When the expectations are found wrong, a participant becomes confused about what is the right way he/she should be thinking about the program after all. In situations involving level 1, the participant can be seen as denying clear facts. In situations involving level 2, the previously constructed facts are questioned.

Levels of think-aloud protocol vs. novices and experts

In the beginning of the research a question (3rd), about how the novice and expert programmers differ in think-aloud protocol, was stated. During the research it became clear that the best debuggers are the ones who have the widest and longest experience on programming. The participant P2 makes an exception, but it could be assumed that he/she has not written very demanding 100 programs in 5 months. Consequently, in this study, the experts are experts in programming and in debugging. Finding the answers to the following questions becomes more relevant. There were two research questions (4th and 5th) that concerned the connection between the levels of think-aloud protocol and debugging performance. Novice debuggers P1 and P2 are making more comments related to errors when not finding the real bugs compared to notions that are made by the participants P3 and P4. This suggests that the activities of the novices during the debugging sessions are more of a bug-driven than of the experts, that are firstly concentrating on understanding the program. The participant P4 finds bugs when reading the code and P3 finds the bugs after the code is red and animated. Novices use a lot of guessing in think-aloud. In addition, novice P2 uses guessing mostly related to error and also when reporting the bugs. It is possible, though, that some people speak in a way that sounds like they are guessing when they are actually asking careful questions. The participant P1 thinks-aloud mostly things related to visualization and the other participants related to the code. The participant P3 and P4 talk in several different levels of origin than do the participants P1 and P2. The participant P3 is the one closely comparing visualization to the code and the participant P2 is the one validating manually the things he/she sees the program calculate. The participant P1 has the most problems in seeing correctly what is there in the program when the participant P4 does not have this sorts of problems. The participant P1 does not make any comments when the participant P4 is commenting a lot.

About hypotheses

It was supposed that expert programmers develop faster, wider and deeper of an understanding about the Java program (1st Hypothesis). The time needed in finding the bugs and the total amount of bugs could give some direction of the developing understanding. Novice participants do not significantly differ in debugging performance compared to the expert P3 when the number of found bugs is concerned. Participant P4 is the only expert that clearly finds more bugs than any other participant does. The participant P4 also finds the bugs in a relatively early stage of his/her debugging session compared to the other participants who find their bugs at the last stages of their debugging sessions.

Experts were expected being of a more of an efficient debuggers — that is, spend less time and find more bugs in given time in task of debugging (2nd Hypothesis). Surprisingly, finding bugs does not seem to be extremely easy for experts either. P4 is the only one that seems to locate the bugs quite easy when reading the code. Anyhow, experts find more bugs compared to the novices. Nevertheless, even if participant P3 gets high scores in pre-test, the outcome from the debugging session is only 0.5 bugs better than for the participant P2 who get the lowest pre-test scores. In addition the expert P4 finds the bugs in a relatively early stage of debugging session when the other expert P3 find the bugs near the finish. Both the novices P1 and P2 find their bugs in a later stages of their sessions.

It was assumed that eye-gaze helps in revealing the ways of the problem solving process when debugging (3rd Hypothesis). If the think-aloud protocol is the only source of data, it will be uncertain what is really happening when a participant is quiet or not referring with unequivocal terms to the objects of talk. Unfortunately or not, analysis considering the eye-gaze data is focusing mostly on the situations when eye-gaze was conflicting with the think-aloud protocol. This way it is not possible to make any reliable conclusions about the relations between think-aloud protocol and eye-gaze as a whole. Eye-gaze data was, also, supposed to give a deeper view to think-aloud protocol (4th Hypothesis). If there were no eye-gaze data, it would have been difficult to find out if any of the participants have made any of the misinterpretations they do. Thanks to eye-gaze data, it was possible to find interesting situations where participants do not interpret correctly what they are seeing at the moment.

5.1 Reliability, Validity and Researcher's view

Reliability is met when no or very little errors or randomness is included (Nurmi et al., 2006). Secondly, reliability can be shown by making two frequently done measurements giving equal results (Nurmi et al., 2006). Data was gathered mechanically but writing the think-aloud out, coding and analysis, despite the graphs from coded data, was done manually. Human errors are possible but minimal. Also, the coding between the different sessions and similar situations are multiply checked for human errors.

Ecological validity is met when the measurements meet the real world situations (Nurmi et al., 2006). Think-aloud is supposed to show the inner thoughts during the debugging session. Nevertheless, it is not a question of a real life debugging situation: experimenter is following you, time is limited, the number of bugs is unknown and you want to perform as good as possible because your performance is recorded for future use. These facts may conclude to nervousness and stress. People perform better or poorer under some stress. This may also have an effect on think-aloud — that is, a person is likely being silent or starts to bubble. On the other hand, think-aloud might have an effect on getting a better understanding of the code. Asking questions, wondering, reading or commenting aloud the visualization, instructions, output or the code results to a more of a self conscious perspective.

In addition there are some problems during the sessions of P2 and P3. Interruptions are due to some problems with visual tracking tool and having the right version of Jeliot on use. This may have some effect on the performance during the debugging session. That is, time meant for debugging is reduced by the interruption and the participant's attention and mind is focusing on secondary issues instead of the main ones.

For ethical reasons and making the participants anonymous, the names and sexes are not mentioned. When a study involves qualitative approach, researcher's subjective view shows in some measures in analysis and interpretations. There are several views according to several viewers and appendix includes original textual data from the four think-aloud protocols to make more interpretations if needed. In addition, preconception concerning the four participants' level of expertise are minimized by concentrating firstly on the data received from the sessions and secondly what the pre-test result are.

6 Conclusions

The idea of this work was to study about the debugging process of novice and expert programmers. It was found, that the eye-gaze data helps in interpretation of think-aloud protocol recorded from a debugging session (1st Question). Gaze-data together with think-aloud protocol helps in finding situations where the debugger is denying the seen or has mismatch about the seen and the things he/she is expecting to see (2nd Question). Gaze-data offers the opportunity to see the focus of an attention and it is possible to find the reasons for actions of the debugger via think-aloud. The participants who were seen as novice programmers performed also poorly in debugging and vice versa. This is why answering to the 4th question gives also an answer to the 3rd question. It was found, that the novice debuggers talked about errors also when not reporting a bug (4th Question and 5th Question). In addition, the novice debuggers also made guestions more than the experts did. Only one of the two expert debuggers was clearly developing faster, wider and deeper of an understanding than novices according to the amount of found bugs and the time spent in finding them (1st Hypothesis). The expert debuggers found more bugs than the novices but both the experts were not substantially efficient (2nd Hypothesis). Eye-gaze was helpful when a perticipant did not make any think-aloud. Eye-gaze helped in finding incorrect situations in problem solving processes when debugging (3rd Hypothesis). Conflicting situations between eye-gaze and think-aloud protocol were found thanks to the eye-gaze data (4th Hypothesis). A problem about the work was that there was no standard coding system to rely on. Therefore, future work should consider validating and verification of the current codes and make improvements if necessary. Future interest is, also, studying about the possible motivation aspect of the Jeliot 3 in program comprehension or in problem solving tasks e.g. debugging. In this study, the analysis of the eye-gaze data was limited only to those situations when a participant did not correctly interpret the seen or was not expecting something he/she saw. It could be considered, that the analysis of the eye-gaze data with the think-aloud protocol could be more extensive in future studies. In addition, the reasons that could make a good programmer a poor debugger could be considered to study in the future. The causality of a contradictory situations, where a debugger tells different from the seen, could also be included in future studies.

References

Ahmadzadeh, M., Elliman, D., Higgins C. *An analysis of patterns of debugging among novice computer science students*. Proceedings of the 10th annual SIGCSE conference in Innovation and technology in computer science education. 2005, ACM Press, New York, NY, USA, 84-88.

Bednarik, R., Myller, N., Sutinen, E., Tukiainen, M. (2006). *Analyzing individual differences in program comprehension*. Technology, Instruction, Cognition and Learning (TICL), Old City Publishing, Inc., Tampere, Finland, 3:205-232.

Ben-Bassat Levy, R., Ben-Ari, M., Uronen, P.A. (2003). *The Jeliot 2000 Program Animation System*. Computers & Education, 40(1), 1-15.

Buckley, J., Exton, C. (2003). *Blooms' Taxonomy: A framework for assessing programmers' knowledge of software systems*. Proceedings of 2nd International Conference on Cognitive Informatics, 165-174.

Chi, M. T. H. (1997). *Quantifying qualitative analyses of verbal data: a practical guide*. The Journal of the learning sciences, Lawrence Erlbaum Associates, Inc., 6(3), 271-315.

Crosby, M. and Stelovsky, J. (1990). *How do we read algorithms? A case study*. IEEE Computer, 23(1), 24-35.

Ebel, G., Ben-Ari, M. (2006). *Affective effects of program visualization*. Proceedings of the Second International Computing Education Research Workshop (Canterbury, UK, 2006), ACM Press, New York, NY, USA, 1-5.

Engeström, Y. (1988). *Perustietoa opetuksesta*. Valtion painatuskeskus, Helsinki, Finland.

Garner, S., Haden, P., Robins, A. (2005). *My Program is Correct But it Doesn't Run: A Preliminary Investigation of Novice Programmers Problem*. Proceedings of the 7th Australian conference on Computing education (Newcastle, Australia, 2005), Australian computer Society, Inc, Darlinghurst, Australia, 42:173-180.

Goldberg, J. H., Kotval, X. P. (1999). *Computer interface evaluation using eye movements: methods and constructs*. International Journal of Industrial Ergonomics, 24(6), 631-45.

Gugerty, L., Olson, G. M. (1986). *Debugging by Skilled and Novice Programmers*. Proceedings of the SIGCHI conference on Human factors in computing systems (Boston, Massachusetts, USA, 1986). ACM Press, New York, NY, USA, 171-174.

Haapasalo, L. (2000). *Oppiminen, tieto ja ongelmanratkaisu*. Medusa-Software, Joensuu, Finland.

Hamlyn, D. W. (1978). *Experience and the growth of understanding*. (edt. Peters, R. S.) Routledge & Kegan Paul Ltd, London, England.

Hoc, J.-M., Green, T. R. G., Gilmore, D. J. (1990). *Psychology of programming*. (eds. Gaines, B. R., Monk, A.). ACM Press, London, England.

Just, M. A. and Carpenter, P. A. (1976). *Eye fixations and cognitive processes*. Cognitive Psychology, 8, 441-480 .

Koenemann, J., Robertson, S. P. (1991). *Expert Problem Solving Strategies for program Comprehension*. (eds. Robertson, S. P., Olson, G. M., Olson, J. S.), CHI'91 conference proceedings. 125-130.

Leroux, H., Réquilé-Romanczuk, A., Mingins, C. (2003). *JACOT: : a tool to dynamically visualise the execution of concurrent Java programs*. Proceedings of the 2nd international conference on Principles and practice of programming in Java (Kilkenny City, Ireland), Computer Science Press, Inc., New York, NY, USA, 201-206.

Meltzoff, A. N., Moore, M. K. (1977). *Imitation of facial and manual gestures by human neonates*. Science, 198, 75-78.

Moreno, A., Myller, N., Sutinen, E., Ben-Ari, M. (2004). *Visualizing Programs with Jeliot 3*. In Proceedings of Advanced Visual Interfaces, AVI 2004, 373-376.

Nurmi, J.-E., Ahonen, T., Lyytinen, P., Pulkkinen, L., Ruoppila, I. (2006). *Ihmisen psykologinen kehitys*. WSOY, Helsinki, Finland.

Rauste-von Wright, M., von Wright, J. (1994). *Oppiminen ja koulutus*. Wsoy, Juva, Finland.

Rayner, K. (1998). *Eye movements in reading and information processing: 20 years of research*. Psychological Bulletin, 124(3), 372-422.

Reid, L. R. (1986). *Ways of Understanding and Education*. Studies in education (new series) 18, Heinemann Educational Books, Institute of Education, University of London, Heinemann Educational Books Ltd, London, England.

Romero, P., du Boulay, B., Cox, R., Lutz, R. (2003). *Java debugging strategies in multi-representational environments*. (edt. Petre, M.) Psychology of Programming Interest Group 15th Workshop.

Sanders, I., Gopal, H. (1991). *AAPT: algorithm animator and programming toolbox*. ACM SIGCSE Bulletin, ACM Press, New York, NY, USA, 23(4), 41-50.

Stein, R., Brennan, S. E. (2004). *Another person's eye gaze as a cue in solving programming problems*. Proceedings of the 6th international conference on Multimodal interfaces. ACM Press, New York, NY, USA, 9-15.

Van de Veire, F., Szmal, P., Francik, J. (1998). *SIAMOA: a system for visual programming, program visualisation and debugging*. Proceedings of the working conference on Advanced visual interfaces (L'Aquila, Italy). ACM Press, New York, NY, USA, 289-291.

Wiedenbeck, S, Ramalingam, V. (1999). *Novice comprehension of small programs written in the procedural and object-oriented styles*. International Journal of Human Computer Studies, American Press, Inc, Duluth, USA, 51:71-87.

(edt. Wilson, B. G.) (1996). *Constructivist learning environments: case studies in instructional design*. Educational Technology Publications, Englewood Cliffs, New Jersey, USA.

Winslow, L. E. (1996). *Programming pedagogy – A psychological overview*. SIGCSE Bulletin, 28 (3), 17-22.

Xu, S., Rajlich, V. (2004). *Cognitive Process during Program Debugging*. Proceedings of the Third IEEE International Conference on Cognitive Informatics. (ICCI '04). IEEE Computer Society, Washington, DC, USA, 176-182.

Yoon, D., Narayanan, N. H. (2004). *Mental imagery in problem solving: an eye tracking study*. Proceedings of the Third ACM Symposium on Eye Tracking Research & Applications. ACM Press, New York, NY, USA, 77-84 .

(eds. Yzerbyt, V. Y., Lories, G., Dardenne, B.) (1998). *Metacognition: cognitive and social dimensions*. SAGE Publications, London, England.

Appendix 1: Textual data P1

Note: The times are in seconds.

1 xxx outo ääni taustalla xxx
2 - xxx hiljaisuus xxx
31 xxx outo ääni taustalla xxx
32 - xxx hiljaisuus xxx
39 xxx outo ääni taustalla xxx
41 - xxx hiljaisuus xxx
56 xxx hiiren liikutus xxx
58 xxx start-klikkaus hiirellä ja käynnistysääni xxx
59 xxx hiiren napsaus xxx
62 xxx hiiren liikutus xxx
66 xxx hiiren napsaus xxx
67 - xxx hiljaisuus xxx
68 xxx outo ääni taustalla xxx
69 - xxx hiljaisuus xxx
76 mmm
77 xxx outo ääni taustalla xxx
78 - xxx hiljaisuus xxx
81 xxx muu, "Ajattele vaan ääneen." xxx
83 - xxx hiljaisuus xxx
85 mh
87 xxx nielaisu xxx
88 Nyt se luo sen olion.
90 - xxx hiljaisuus xxx
93 Ja.
94 - xxx hiljaisuus xxx
95 Neljällä hengellä.
95 - xxx hiljaisuus xxx
100 mmm
102 - xxx hiljaisuus xxx
114 xxx outo ääni taustalla xxx
115 xxx hiiren liikutus xxx
116 xxx hiiren napsaus xxx
117 xxx hiiren napsaus xxx
118 xxx outo ääni taustalla xxx
119 xxx hiiren napsaus xxx
120 - xxx hiljaisuus xxx
124 mh
125 - xxx hiljaisuus xxx
126 xxx hiiren napsaus xxx Nyt lisää uuden henkilön.
129 xxx hiiren liikutus xxx
132 xxx hiiren napsaus xxx
133 xxx hiiren napsaus xxx Nyt se lisää taas.
136 xxx hiiren napsaus xxx
137 - xxx hiljaisuus xxx
138 xxx hiiren napsaus xxx
139 xxx hiiren napsaus xxx
140 - xxx hiljaisuus xxx
142 xxx nielaisu xxx
143 mmm xxx hiiren napsaus xxx
145 xxx hiiren liikutus xxx
146 - xxx hiljaisuus xxx
151 xxx hiiren napsaus xxx Lisää henkilön.
152 - xxx hiljaisuus xxx
155 xxx outo ääni taustalla xxx
156 - xxx hiljaisuus xxx
157 xxx outo ääni taustalla xxx
158 - xxx hiljaisuus xxx
159 mmm
160 xxx hiiren liikutus xxx
165 xxx hiiren napsaus xxx
168 xxx hiiren napsaus xxx
169 - xxx hiljaisuus xxx
174 mh
176 - xxx hiljaisuus xxx
181 mmm
182 xxx hiiren napsaus xxx
184 mh

185 - xxx hiljaisuus xxx
185 xxx hiiren napsaus xxx
186 xxx muu, "Ajattele vaan ääneen." xxx
188 Ainaki se taitaa nimet tallentaa väärin ihan.
190 xxx vetää henkeä xxx
191 xxx hiiren napsaus xxx
192 phh
195 xxx hiiren napsaus xxx
196 xxx hiiren napsaus xxx
197 xxx hiiren napsaus xxx
198 xxx hiiren liikutus xxx
201 xxx hiiren napsaus xxx
203 xxx hiiren napsaus xxx
205 xxx hiiren liikutus xxx
206 xxx hiiren napsaus xxx
207 xxx hiiren napsaus xxx
208 xxx hiiren napsaus xxx
210 xxx hiiren napsaus xxx
213 mmm
214 - xxx hiljaisuus xxx
214 xxx hiiren liikutus xxx
218 - xxx hiljaisuus xxx
226 Nyt se laskee xxx hiiren napsaus xxx perheen jäsenten xxx hiiren napsaus xxx määrän.
232 xxx hiiren napsaus xxx
233 xxx hiiren napsaus xxx
234 - xxx hiljaisuus xxx
236 Jonka se xxx hiiren napsaus xxx tais laskee väärin.
238 - xxx hiljaisuus xxx
241 xxx hiiren napsaus xxx
242 - xxx hiljaisuus xxx
243 mmm
245 - xxx hiljaisuus xxx
248 xxx epäselvä "No ni." xxx
249 - xxx hiljaisuus xxx
259 xxx hiiren napsaus xxx
260 - xxx hiljaisuus xxx
277 mh
279 Nyt se käy nyt niitä perheen jäseniä läpi.
281 - xxx hiljaisuus xxx
285 mmm Nyt se lisää niitä lapsia sinne.
286 - xxx hiljaisuus xxx
289 Kö
290 Sen xxx hiiren napsaus xxx lapsijoukon.
291 xxx hiiren liikutus xxx
294 xxx nielaisu xxx
295 - xxx hiljaisuus xxx
299 xxx hiiren napsaus xxx
301 xxx hiiren napsaus xxx
302 - xxx hiljaisuus xxx
304 xxx hiiren napsaus xxx
305 - xxx hiljaisuus xxx
310 mmm Nyt se.
312 Lähtee varmaan käymään taas noita läpi.
314 - xxx hiljaisuus xxx
316 Henkilöitä.
317 - xxx hiljaisuus xxx
321 Joo. Nyt se kattoo onko ne lapsia.
322 - xxx hiljaisuus xxx
325 Jos se on totta, että se on vanhempi, niin se
327 - xxx hiljaisuus xxx
329 mmm
330 - xxx hiljaisuus xxx
357 mmm 358 Nyt se on xxx hiiren liikutus xxx lisänny sitä henkilöitten määrää jotenki yhellä.
362 - xxx hiljaisuus xxx
366 xxx hiiren napsaus xxx Kattooko xxx hiiren napsaus xxx xxx hiiren napsaus xxx toise henkilön, onko se lapsi. xxx hiiren napsaus xxx Ei oo.
371 Eiku.
372 - xxx hiljaisuus xxx
381 xxx hiiren napsaus xxx
382 xxx hiiren liikutus xxx
385 xxx hiiren napsaus xxx
386 - xxx hiljaisuus xxx
399 xxx outo ääni taustalla xxx

400 - xxx hiljaisuus xxx
402 xxx hiiren liikutus xxx
405 Nyt se kirjottaa sitä loo..silmukkaa vieläki. Kattoo varmaan. Nyt kattoo vaan sen xxx epäselvä "end.." xxx lapsen. Se on lapsi.
414 xxx hiiren liikutus xxx
415 - xxx hiljaisuus xxx
418 xxx hiiren liikutus xxx
420 Nyt.
423 Pyoräyttäny menee uuvestaan silmukkaan.
425 - xxx hiljaisuus xxx
431 5 on pienempi kuin 18.
434 - xxx hiljaisuus xxx
440 mmm
441 - xxx hiljaisuus xxx
443 Nyt se on.
445 Nyt se lähtee silmukasta ulos.
447 xxx vetää henkeä xxx
448 Nyt se pallauttaa sen lasten. Lapset mitkä se löyti.
451 - xxx hiljaisuus xxx
453 xxx hiiren liikutus xxx
455 Nyt sen pitäs tulostaa kait lapsien nimet.
457 - xxx hiljaisuus xxx
460 NULL
461 xxx hiiren liikutus xxx
464 - xxx hiljaisuus xxx
468 xxx hiiren napsaus xxx
469 xxx hiiren liikutus xxx
470 mmm
471 xxx nielaisu xxx
472 xxx hiiren liikutus xxx
474 xxx hiiren napsaus xxx
475 xxx hiiren liikutus xxx
478 mmm
479 NULL
483 xxx hiiren liikutus xxx
488 xxx nielaisu xxx
489 - xxx hiljaisuus xxx
493 xxx outo ääni taustalla xxx
494 - xxx hiljaisuus xxx
496 mh xxx outo ääni taustalla xxx
497 - xxx hiljaisuus xxx
500 mmm
501 Nyt se ottaa nuo lapsien xxx epselvä "ja" xxx. Tutkii.
504 - xxx hiljaisuus xxx
514 xxx nielaisu xxx
515 - xxx hiljaisuus xxx
517 mh
518 - xxx hiljaisuus xxx
522 Nyt se laskee sitä keskimäärästä ikkee.
524 - xxx hiljaisuus xxx
526 mmm
527 - xxx hiljaisuus xxx
529 Tuolta se hakkee.
531 - xxx hiljaisuus xxx
532 Nolla.
533 - xxx hiljaisuus xxx
535 Neljä. Eiku.
537 xxx hiiren liikutus xxx
539 Nyt se hakkee niitte iät.
541 Silmukassa.
542 - xxx hiljaisuus xxx
545 xxx hiiren liikutus xxx
554 xxx nielaisu xxx
556 Nyt se yhistää niitä ikiä, mutta.
558 - xxx hiljaisuus xxx
560 Eikö se laske niitä yhteen.
563 - xxx hiljaisuus xxx
568 xxx hiiren liikutus xxx mmm
570 - xxx hiljaisuus xxx
574 xxx nielaisu xxx
576 xxx hiiren liikutus xxx
578 Viis jaettuna neljällä.
580 Se ei laskenu yhteen niitä ikiä. Se vaan si- lisäs ne aina. Sit tullee tuo virhe ainakkii.

588 - xxx hiljaisuus xxx
593 xxx hiiren liikutus xxx
594 mmm
596 xxx hiiren liikutus xxx
597 - xxx hiljaisuus xxx
599 xxx hiiren liikutus xxx
601 Joo. Ainakkaa se ei noita xxx hiiren napsaus xxx ikiä laskenu yhteen.
606 Ja nimetki näemmä laitto xxx hiiren napsaus xxx väärin.
610 xxx 7 kertaa hiiren napsaus xxx
612 Se ei nimiä tainnu laittaa etes.
614 xxx hiiren liikutus xxx
628 xxx nielaisu xxx
629 xxx hiiren liikutus xxx
631 Nyt se luo sen uuven perheen.
633 xxx hiiren liikutus xxx
634 Ryhmä.
635 xxx hiiren liikutus xxx
636 Perhe.
637 xxx hiiren liikutus xxx
639 4 henkee.
640 xxx hiiren liikutus xxx
648 - xxx hiljaisuus xxx
654 Nyt se tekee lapsiryhmän.
656 - xxx hiljaisuus xxx
661 Nyt se luo äidin.
663 xxx hiiren liikutus xxx
664 New person.
665 xxx hiiren napsaus xxx
667 Sitte o nimmee.
669 Ei taija.
671 String pname.
672 - xxx hiljaisuus xxx
676 Ei xxx hiiren napsaus xxx hakenu xxx hiiren napsaus xxx tuota.
678 - xxx hiljaisuus xxx
679 xxx hiiren napsaus xxx
680 xxx hiiren napsaus xxx
683 xxx hiiren napsaus xxx
684 xxx hiiren napsaus xxx
685 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
687 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
691 xxx hiiren napsaus xxx
694 Nimeä. xxx hiiren liikutus xxx Ei tainnu. xxx hiiren liikutus xxx
698 xxx hiiren liikutus xxx Nyt se taas tekee uuve henkilön.
701 - xxx hiljaisuus xxx
704 mmm
705 xxx hiiren liikutus xxx
706 String pname.
708 Hakee iän, sukupuolen. Se hakee. Ottaa sukupuolen kahesti vaikka se olis pitänyt ottaa nimi.
714 xxx hiiren napsaus xxx
717 xxx hiiren napsaus xxx
721 xxx hiiren liikutus xxx
722 xxx hiiren napsaus xxx
723 mmm
725 xxx hiiren napsaus xxx
728 Nyt se sijott.
732 Hakee iän.xxx hiiren napsaus xxx
735 pagen
736 - xxx hiljaisuus xxx
739 xxx hiiren liikutus xxx
740 Tuossa on virhe. Nel. Psex sen pitäs olla ää pname.
746 xxx muu, "Eli sano vaa rivi." xxx
748 Ööö. Rivi 14.
751 xxx muu, "Luokka." xxx
752 Luokka xxx hiiren liikutus xxx öö Person.
755 xxx muu, "Joo." xxx
756 xxx hiiren liikutus xxx
758 xxx hiiren napsaus xxx
759 - xxx hiljaisuus xxx
769 xxx hiiren liikutus xxx
771 mmm
772 - xxx hiljaisuus xxx
777 xxx hiiren liikutus xxx

778 - xxx hiljaisuus xxx
782 xxx muu, "Yritä vaan ääneen ajatella." xxx
784 Joo-o.
785 Nyt se taas luo niitä.
786 Nyt se on luonu sen perheen.
788 - xxx hiljaisuus xxx
797 mmm
798 Nyt se lähtee käymään sitä perhettä läpi.
800 - xxx hiljaisuus xxx
809 mh
810 - xxx hiljaisuus xxx
814 xxx nielaisu xxx
815 - xxx hiljaisuus xxx
819 mmm
820 - xxx hiljaisuus xxx
824 xxx outo ääni taustalla xxx
825 - xxx hiljaisuus xxx
830 Se käy nyt täs silmukoita läpi.
832 - xxx hiljaisuus xxx
839 xxx kello xxx
841 xxx muu, "2 minuuttii jäljellä." xxx
842 Joo.
843 Se käy sitä tuolla silmukalla läpi.
845 - xxx hiljaisuus xxx
846 xxx nielaisu xxx
847 - xxx hiljaisuus xxx
854 xxx hiiren liikutus xxx
857 Tutkii xxx epäselvä "" xxx xxx hiiren napsaus xxx perheessä
859 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
860 - xxx hiljaisuus xxx
878 xxx hiiren napsaus xxx
879 xxx hiiren napsaus xxx
880 - xxx hiljaisuus xxx
882 xxx hiiren napsaus xxx
883 Nyt se xxx hiiren napsaus xxx lähtee hakemaan niitä lapsia.
886 - xxx hiljaisuus xxx
888 Neljän kokonen
890 - xxx hiljaisuus xxx
891 xxx hiiren napsaus xxx
892 xxx hiiren napsaus xxx
900 xxx nielaisu xxx
901 mh
902 Nyt se luo. Taulukon tais luoda.
905 Johon se laittaa ni. xxx epäselvä "" xxx niitä perheen lapsia laittaa.
911 - xxx hiljaisuus xxx
920 Silmukka niin pitkään ku on pienempi ku tuo perheen koko.
925 Se menee siihe silmukkaan.
927 Hakee ensimmäisen.
928 - xxx hiljaisuus xxx
930 Eiku tuo. Se haettu ei oo koska se nyt vertaa onko se pienempi ku toi se.
937 18 pienempi tai alle 18. Ei ole.
941 xxx vetää henkeä xxx
942 - xxx hiljaisuus xxx
954 xxx outo ääni taustalla xxx
955 - xxx hiljaisuus xxx
956 - xxx hiljaisuus xxx
958 Ko-
959 xxx kello xxx
960 xxx loppu xxx

Appendix 2: Textual data P2

Note: The times are in seconds.

0 xxx melua xxx
2 xxx muu, "Tossa on ohjeet..sitä." xxx
4 xxx melua xxx
97 xxx vetää henkeä xxx
98 xxx melua xxx
107 xxx muu, "Sie voit sitä koodia katella, ni vaa paina sitä starttia sieltä."
111 - xxx hiljaisuus xxx
113 Laittas vielä tuon kertaalleen, että varmasti on sitten ymmärtäny tuon.
114 xxx muu, "Saa siihe palataki, että ei se." xxx
120 - xxx hiljaisuus xxx
131 xxx outo ääni taustalla xxx
132 - xxx hiljaisuus xxx
146 xxx hiiren liikutus xxx
150 xxx start-klikkaus hiirellä ja käynnistysääni xxx
152 xxx muu, "Tosiaan ajattele vaan ääneen sitten koko ajan. Mitä näät ja mitä mietit." xxx
159 Joo. Vaan ei oikein nyt mä tarkastan nuo tietojäsenet tuosta.
165 Mä meen tohon a- alustusvaiheeseen.
167 xxx kröhöm xxx
168 xxx vetää henkeä xxx
170 - xxx hiljaisuus xxx
172 xxx kröhöm xxx
173 xxx outo ääni taustalla xxx
180 boolean isAChild.
183 Palautt- palauttaa iän jos se on yli 18.
188 - xxx hiljaisuus xxx
190 Print metodi
191 Tulostaa nimen.
192 - xxx hiljaisuus xxx
200 xxx kröhöm xxx
201 - xxx hiljaisuus xxx
216 xxx hiiren napsaus xxx
217 - xxx hiljaisuus xxx
220 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
222 - xxx hiljaisuus xxx
232 xxx outo ääni taustalla xxx
237 xxx vetää henkeä xxx
240 - xxx hiljaisuus xxx
243 xxx hiiren napsaus xxx
244 - xxx hiljaisuus xxx
245 xxx hiiren napsaus xxx
246 - xxx hiljaisuus xxx
247 - xxx hiljaisuus xxx
250 xxx hiiren napsaus xxx
251 - xxx hiljaisuus xxx
253 xxx hiiren napsaus xxx
254 - xxx hiljaisuus xxx
261 Jos ihmettelet, et mä oon hiljaa, ni en mä oikeestaan ajattele. Käyn vaan tätä koodia läpi.
268 xxx muu, "Joo-o." xxx
269 xxx muu, "Kerro vaan, että mitä ajattelet" xxx
272 - xxx hiljaisuus xxx
273 xxx hiiren napsaus xxx
274 - xxx hiljaisuus xxx
288 xxx hiiren napsaus xxx
290 xxx hiiren liikutus xxx
291 - xxx hiljaisuus xxx
313 No tuossa on tuo xxx hiiren napsaus xxx ketä yrittäs ehtii se metodi. xxx hiiren napsaus xxx
316 Katotaanpas sitä xxx hiiren napsaus xxx läpi.
319 - xxx hiljaisuus xxx
329 xxx kröhöm xxx
330 - xxx hiljaisuus xxx
336 xxx hiiren napsaus xxx
337 - xxx hiljaisuus xxx
340 xxx hiiren napsaus xxx
341 - xxx hiljaisuus xxx
342 xxx epäselvä "Cain" xxx
344 - xxx hiljaisuus xxx
346 xxx kröhöm xxx

347 - xxx hiljaisuus xxx
348 Elikkä tuossa on joku Cain ja sit tuolla on Cain.
354 Ootahan.
355 - xxx hiljaisuus xxx
357 Ei. Ei oo.
419 xxx epäselvä "Onha meillä tuossaki Cain." xxx
361 - xxx hiljaisuus xxx
364 xxx muu, "Nyt kyllä haittaa se." xxx
366 Torni, vai?
367 xxx muu, " xxx epäselvä "Hiiri-" xxx käsi tuota." xxx
368 Jaa.
369 - xxx hiljaisuus xxx
370 xxx hiiren napsaus xxx
371 xxx muu, "Yritä vaikka hiirellä vähä" xxx xxx hiiren napsaus xxx xxx muu, "osottaa sitä, että." xxx
374 xxx hiiren liikutus xxx
375 - xxx hiljaisuus xxx
379 xxx muu, "Ja ajattele vaan" xxx epäselvä "rivejä" xxx xxx muu, "ääneen." xxx
381 - xxx hiljaisuus xxx
382 xxx hiiren napsaus xxx
383 mmm
384 xxx hiiren napsaus xxx
385 xxx hiiren napsaus xxx
386 xxx hiiren napsaus xxx
387 Minä yritän selvittää.
388 Että, minkähän takkii tuossa yrittää kovasti xxx epäselvä "child" xxx in the firstFamily olla ni. Pitäs tulostaa miesten, niin minkähän takkii se laittaa female sukupuolen siihen. Kuin niin.
401 xxx hiiren liikutus xxx
402 xxx hiiren napsaus xxx
403 - xxx hiljaisuus xxx
404 xxx vetää henkeä xxx
407 - xxx hiljaisuus xxx
424 xxx hiiren napsaus xxx
425 - xxx hiljaisuus xxx
442 xxx hiiren napsaus xxx
443 xxx muu, "Jos vähä siirryt oikeelle xxx hiiren liikutus xxx, ni ei oikein toinen silmä xxx hiiren napsaus xxx meinaa löytyy. xxx hiiren napsaus xxx No, ni. Se on parempi. xxx
449 - xxx hiljaisuus xxx
450 xxx muu, "Tai no siirry taas vähä oikeelle." xxx
454 xxx muu, xxx epäselvä "Nojaat sen varaan."xxx Joo.xxx
457 - xxx hiljaisuus xxx
467 Minä lasken nyt noita ikiä tuosta yhteen.
470 35 plus 40.
473 Plus 10.
474 Siit tulee 85 90.
478 Joo.
479 - xxx hiljaisuus xxx
482 xxx hiiren liikutus xxx
483 - xxx hiljaisuus xxx
484 firstFamily piste include mum, dad, bigBrother, littleBrother.
489 Joo.
490 - xxx hiljaisuus xxx
491 xxx hiiren napsaus xxx
492 xxx hiiren napsaus xxx
493 xxx muu, "Kierrätkö taas vähä sinne oikeelle."
495 xxx outo ääni taustalla xxx
499 - xxx hiljaisuus xxx
510 xxx hiiren napsaus xxx
511 children piste get discrapson. xxx hiiren napsaus xxx Aoäh.
515 xxx hiiren liikutus xxx
517 xxx hiiren napsaus xxx
518 - xxx hiljaisuus xxx
523 xxx hiiren napsaus xxx
524 xxx hiiren liikutus xxx
525 - xxx hiljaisuus xxx
530 xxx hiiren napsaus xxx
531 - xxx hiljaisuus xxx
535 xxx hiiren napsaus xxx
536 - xxx hiljaisuus xxx
483 xxx hiiren liikutus xxx
544 xxx hiiren napsaus xxx
545 - xxx hiljaisuus xxx
549 xxx hiiren liikutus xxx

550 - xxx hiljaisuus xxx
552 xxx hiiren napsaus xxx
553 - xxx hiljaisuus xxx
560 xxx hiiren napsaus xxx
561 xxx hiiren napsaus xxx
562 - xxx hiljaisuus xxx
569 xxx hiiren napsaus xxx
570 - xxx hiljaisuus xxx
571 xxx hiiren napsaus xxx
572 - xxx hiljaisuus xxx
583 xxx hiiren napsaus xxx
584 xxx hiiren napsaus xxx
585 - xxx hiljaisuus xxx
603 xxx hiiren liikutus xxx
604 - xxx hiljaisuus xxx
619 xxx hiiren napsaus xxx
620 - xxx hiljaisuus xxx
621 xxx hiiren napsaus xxx
622 xxx hiiren liikutus xxx
623 - xxx hiljaisuus xxx
626 Elikkä, ku ihmisiä oli 4. Ja.
630 Virheellinen tulos sitte keskimääräinen ikä.
635 Keskimääräinen ikä on 1,85.
639 Tarkottaakohan se sitä sit, että se jakaa tuon Abelin iän 4:llä.
642 - xxx hiljaisuus xxx
647 xxx vetää henkeä xxx
649 - xxx hiljaisuus xxx
658 xxx hiiren liikutus xxx
659 - xxx hiljaisuus xxx
660 xxx muu, "Jatka vaan sitä ääneen ajattelua." xxx
661 - xxx hiljaisuus xxx
662 xxx hiiren napsaus xxx
669 xxx hiiren napsaus xxx
670 Minkä ihmeen takii se tuo täällä kun lukee tuo AverageAge xxx hiiren liikutus xxx niin age on pienellä, mut miks se tulostaa sen isolla xxx hiiren liikutus xxx alkukirjaimella.
681 xxx hiiren napsaus xxx
682 - xxx hiljaisuus xxx
684 mmm
685 - xxx hiljaisuus xxx
687 xxx hiiren napsaus xxx
688 xxx muu, "Siinä on kyllä ihan kirjoitusvirhe, että." xxx
691 Nii. Pakko olla.
692 - xxx hiljaisuus xxx
696 xxx hiiren napsaus xxx
698 xxx epäselvä " " xxx
699 xxx muu, "Vois mainita xxx epäselvä "" xxx " tuosta ohjeesta.xxx
703 - xxx hiljaisuus xxx
707 xxx hiiren napsaus xxx
708 - xxx hiljaisuus xxx
718 xxx hiiren napsaus xxx
779 Pitää et-. Koitan ehtii, että mistä se tulostaa tuota xxx hiiren napsaus xxx sukupuolta noin kovasti.
725 xxx hiiren napsaus xxx
726 xxx hiiren napsaus xxx
727 xxx hiiren liikutus xxx
728 xxx vetää henkeä xxx
729 - xxx hiljaisuus xxx
730 xxx hiiren napsaus xxx
731 - xxx hiljaisuus xxx
737 xxx hiiren napsaus xxx
738 - xxx hiljaisuus xxx
746 xxx hiiren napsaus xxx
747 - xxx hiljaisuus xxx
751 xxx hiiren napsaus xxx
752 - xxx hiljaisuus xxx
763 xxx hiiren napsaus xxx
764 - xxx hiljaisuus xxx
769 xxx hiiren napsaus xxx
770 - xxx hiljaisuus xxx
784 xxx hiiren napsaus xxx
785 xxx hiiren napsaus xxx
786 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
787 - xxx hiljaisuus xxx

806 xxx hiiren napsaus xxx
807 xxx hiiren napsaus xxx
808 - xxx hiljaisuus xxx
812 xxx hiiren napsaus xxx
813 - xxx hiljaisuus xxx
815 xxx hiiren liikutus xxx
816 - xxx hiljaisuus xxx
824 xxx hiiren napsaus xxx
825 xxx hiiren napsaus xxx
826 - xxx hiljaisuus xxx
827 Oisikohan yks virhe tuossa, että. Että suo- tulostaa the children in the first family are.
835 Ottas nämä tästä pois, ni nää. children is get discräft.
840 Kirjottas vaan family tilanne.
843 Ei mulla oikein muuta tuohon oo. Oikein muuta kehitellä.
845 xxx muu, "Sano vain rivi" xxx
847 Mitä?
848 xxx muu, "Että sano vaan rivi ja luokka" xxx
851 101 ja 102 on rivit.
853 - xxx hiljaisuus xxx
857 Ja, jaa.
858 - xxx hiljaisuus xxx
860 xxx muu, "Jos saat sitä tosiaan, että xxx epäselvä " " xxx "xxx
863 Sitten tuo group näyttäs olevan tuo luokka.
867 - xxx hiljaisuus xxx
872 Hankala ohjelma, kun ei äkkiseltään pääse jyvälle tästä. Missä ois mikäki virhe.
877 - xxx hiljaisuus xxx
878 xxx hiiren liikutus xxx
880 xxx hiiren napsaus xxx
881 - xxx hiljaisuus xxx
884 Nii ja sit xxx hiiren napsaus xxx tuo kirjotusvirhe tuossa xxx hiiren liikutus xxx niinku rivillä 110. xxx hiiren napsaus xxx Tuo tuo average age kohalla.
891 xxx hiiren napsaus xxx
892 xxx muu, "M-m" xxx
893 - xxx hiljaisuus xxx
894 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
895 Siis tää varmas- rivillä 111 on myös yks virhe. Koska ilmeisesti tuo tämänhetkinen ohjelma ni jakaa 4:llä tuon Abelin iän. 5 jaettuna 4lla on 1,25. Siinä lienee virhe.
909 - xxx hiljaisuus xxx
912 xxx hiiren liikutus xxx
914 Mutta, xxx hiiren napsaus xxx mutta.
917 Mistä. Mistä se tulostaa nuo femalet ja malet tuohon xxx hiiren napsaus xxx alkuun.
922 - xxx hiljaisuus xxx
928 xxx hiiren napsaus xxx
929 xxx hiiren liikutus xxx Sitte.
930 xxx kello xxx
931 xxx muu, "2 minuuttia on vielä jälellä" xxx
933 - xxx hiljaisuus xxx
936 xxx hiiren napsaus xxx
937 - xxx hiljaisuus xxx
938 xxx hiiren liikutus xxx
939 - xxx hiljaisuus xxx
941 Joo, joo. Tuossa rivillä nelkytäneljä vissiin tuo p.print pois, ni se jättäs tuon suku- sukupuolet tulostelematta.
950 xxx hiiren napsaus xxx
951 - xxx hiljaisuus xxx
953 xxx vetää henkeä xxx
954 - xxx hiljaisuus xxx
958 xxx hiiren napsaus xxx
959 - xxx hiljaisuus xxx
965 xxx hiiren napsaus xxx
966 - xxx hiljaisuus xxx
980 xxx hiiren napsaus xxx
981 - xxx hiljaisuus xxx
999 xxx hiiren napsaus xxx
1000 - xxx hiljaisuus xxx
1002 xxx hiiren napsaus xxx
1003 - xxx hiljaisuus xxx
1006 Just tuossa nextPerson-metodissa ni varmaan tuo if-lauseen. Tuo ehto on varmaan jotenki väärin määritelty.
1015 Kun sen laittaa NextPersonIndex = 0 niin se varmaan aiheuttas tuon.
420 No en tiiä, mut joka tapauksessa tuossa näyttäs olevan jotain häikkää kyllä.
1025 Että rivillä 50 ja 51.
1027 - xxx hiljaisuus xxx
1033 xxx hiiren napsaus xxx
1034 xxx hiiren napsaus xxx
1035 - xxx hiljaisuus xxx

1036 xxx hiiren napsaus xxx
1044 xxx hiiren napsaus xxx
1050 xxx kello xxx xxx hiiren napsaus xxx
1051 Aha xxx loppu xxx

Appendix 3: Textual data P3

Note: The times are in seconds.

3 xxx muu, "Elikkä nyt voit alottaa ne." xxx
4 Elikkä pitääks nyt klikata nyt ekaks tästä vai? Vai pitääks mun lukee ekaks toi?
8 xxx muu, "Voit lukee niitä ohjeita ekana."
9 Joo.
10 - xxx hiljaisuus xxx
12 Näitä ei tarvii ajatella vielä ääneen näitä ohjeita? Vai pitääkö?
14 xxx muu, "Ei tarvi." xxx
15 Ok. No sitte.
16 xxx vetää henkeä xxx
17 - xxx hiljaisuus xxx
26 xxx muu, "Se riittää, ku ajattelet sitte äääneen ku alotat."
28 Ok.
29 - xxx hiljaisuus xxx
70 Elikkä siis tää luokka niin siis aina täältä tää. Nimi ja sitte taas toi ja sitte ja mikä rivi joo. xxx muu, "Sen voi sen luokka sisällön xxx epäselvä xxx Joo." xxx
79 Ok. Ok. Ok.
82 xxx outo ääni taustalla xxx
93 xxx Mhhy xxx
94 - xxx hiljaisuus xxx
101 Tuossa.
102 - xxx hiljaisuus xxx
106 xxx Mhhy xxx
107 Katotaan kuinka käy.
110 xxx hiiren liikutus xxx xxx start-klikkaus hiirellä ja käynnistysääni xxx
111 Joo-o.
113 Eli tos on toi xxx hiiren liikutus xxx Person-olio.
117 xxx hiiren napsaus xxx
118 Siellä on ikä ja nimi ja xxx hiiren napsaus xxx sukupuoli.
123 Ja xxx hiiren napsaus xxx tuo palauttaa sitten tiedon siitä xxx hiiren liikutus xxx, onx se xxx hiiren napsaus xxx alaikänen tai isAChildi. xxx hiiren liikutus xxx Ja.
132 xxx hiiren liikutus xxx
134 Toi on se tuo tulostus.
136 xxx hiiren liikutus xxx
137 Joo.
138 xxx hiiren napsaus xxx
139 Sit se tää xxx hiiren napsaus xxx ryhmitysjuttu.
143 - xxx hiljaisuus xxx
145 Mikä täs on. Montako niitä xxx hiiren liikutus xxx on siellä ja.
151 xxx hiiren napsaus xxx
152 Jaa-a
153 - xxx hiljaisuus xxx
155 Mitäs muuta. Ryhmä ja lisätään jotain ja.
161 xxx hiiren napsaus xxx
162 Seuraava tyyppi siinä ryhmässä xxx hiiren napsaus xxx vissiinki tuo.
166 - xxx hiljaisuus xxx
168 xxx hiiren napsaus xxx
169 - xxx hiljaisuus xxx
170 Iän keskiarvo.
172 - xxx hiljaisuus xxx
178 Lasketaan.

180 xxx muu, "xxx epäselvä xxx vanha versio tästä ohjelmasta. "xxx
184 Ai jaa. xxx naurahdus xxx
185 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
187 Jaaha.
187 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
188 xxx muu, "Tää on se uusin. Ei." xxx hiirennapsaus xxx xxx hiiren napsaus xxx xxx hiiren napsaus xxx
191 xxx epäselvä xxx nyt tuo äsken.
192 xxx hiiren napsaus xxx xxx muu, "Ootahan nyt." xxx xxx hiiren napsaus xxx xxx hiiren liikutus xxx
194 Se on nytte se oikee. xxx hiiren liikutus xxx
196 Jaaha. Joo. xxx hiiren liikutus xxx

198 xxx muu, "Jatka vaan." xxx
199 Noh. Mut-. Mmm. Joo. xxx Mhhy xxx
202 xxx hiiren liikutus xxx
203 Mihinkähän xxx hiiren napsaus xxx mä nyt xxx hiirennapsaus xxx jäin. xxx hiiren liikutus xxx Noh. xxx hiiren liikutus xxx Katotaan.
206 xxx hiiren napsaus xxx
207 Toi on toi ryhmä taas ja.
211 Seuraava.

214 Hetkinen xxx hiiren napsaus xxx nää kuuluu niinkun xxx hiiren napsaus xxx kaikki tähän Group-luokkaan.
221 - xxx hiljaisuus xxx
222 Jaaha.
223 - xxx hiljaisuus xxx
224 Sitte perhe.
227 - xxx hiljaisuus xxx
229 Siellä on sitten niitä xxx hiiren napsaus xxx lapsi-xxx hiiren napsaus xxx.
235 Jaa-a.
236 - xxx hiljaisuus xxx
242 Palauttaa jotain.
244 Sit on main.
246 xxx hiiren napsaus xxx
247 xxx hiiren napsaus xxx
248 xxx hiiren napsaus xxx
249 Elikkä ekaks tehään se perhe. Ja.
252 sitten lapset. Ja.
256 Toi on
257 xxx hiiren napsaus xxx
258 - xxx hiljaisuus xxx
259 henkilö.
261 xxx hiiren napsaus xxx
262 xxx hiiren liikutus xxx
263 Tos on xxx hiiren liikutus xxx äiti ja
266 - xxx hiljaisuus xxx
267 isä ja
268 pojat.
269 xxx hiiren napsaus xxx
270 xxx hiiren napsaus xxx
271 Ja sit ne lisätään sinne
272 - xxx hiljaisuus xxx
276 perheeseen. Ja.
277 xxx hiiren liikutus xxx
278 xxx hiiren napsaus xxx
279 Sit lasket-.
280 Katotaan niit xxx hiiren napsaus xxx lapsia.
284 - xxx hiljaisuus xxx
288 Joo.
289 xxx hiiren napsaus xxx
290 xxx hiiren liikutus xxx
291 - xxx hiljaisuus xxx
292 xxx hiiren napsaus xxx
293 xxx Mhhy xxx
295 Sitte.
297 Yks yrittää hakee niitä lapsia, mut sitte ei onnistukaan.
304 - xxx hiljaisuus xxx
306 xxx hiiren napsaus xxx
307 xxx hiiren liikutus xxx
308 Mitäs tossa on.
309 xxx hiiren napsaus xxx
310 xxx hiiren napsaus xxx
311 xxx hiiren liikutus xxx
313 xxx hiiren napsaus xxx
317 Katotaas. xxx hiiren napsaus xxx
321 xxx hiiren napsaus xxx
324 - xxx hiljaisuus xxx
325 Description hiiren napsaus xxx elikkä sille perheelle ei anneta nyt mitään xxx hiiren napsaus xxx nimeä.
338 xxx hiiren napsaus xxx
339 - xxx hiljaisuus xxx
342 Children xxx hiiren napsaus xxx hiiren napsaus xxx description. Mikäs tuo oli.
345 xxx hiiren napsaus xxx
347 xxx hiiren napsaus xxx
349 xxx outo ääni taustalla xxx
350 - xxx hiljaisuus xxx
353 xxx Mhhy xxx
356 xxx hiiren napsaus xxx
357 xxx hiiren napsaus xxx
358 - xxx hiljaisuus xxx
362 xxx Mhhy xxx
364 - xxx hiljaisuus xxx
368 xxx hiiren napsaus xxx
369 xxx hiiren liikutus xxx
370 Mikäs. xxx hiiren liikutus xxx

373 Noh.
374 xxx hiiren napsaus xxx
375 xxx hiiren napsaus xxx
376 firsFamily,
377 children,
379 get description. Mikäs tuo nyt on.
382 firsFamily
383 xxx hiiren liikutus xxx
384 getChildren
385 xxx hiiren napsaus xxx
386 Elikkä se on niinkun tuo. xxx hiiren liikutus xxx
391 Tuo metodi, millä ne haetaan siihen.
398 Duoda duoda.
399 - xxx hiljaisuus xxx
403 xxx hiiren napsaus xxx
408 Tuossa jossain se on nyt määritelty tuo, että neon niitä lapsia.
414 Onkos tää nyt missä.
416 xxx hiiren napsaus xxx
417 xxx hiiren napsaus xxx
418 People on
419 tuolla.
420 xxx hiiren napsaus xxx
421 Eiku.
423 xxx hiiren napsaus xxx
424 Mikä xxx hiiren napsaus xxx ihmeen.
429 people.
430 - xxx hiljaisuus xxx
433 xxx hiiren napsaus xxx
434 - xxx hiljaisuus xxx
435 xxx hiiren napsaus xxx
436 Eihän täällä oo koko ollenkaan.
439 - xxx hiljaisuus xxx
443 Däh.
444 - xxx hiljaisuus xxx
448 xxx Mhhy xxx
449 - xxx hiljaisuus xxx
455 xxx hiiren napsaus xxx
458 Noh. xxx hiiren liikutus xxx xxx hiiren napsaus xxx
460 Mitäs nää nyt xxx hiiren napsaus xxx ees xxx hiiren napsaus xxx on.
463 xxx hiiren liikutus xxx xxx hiiren napsaus xxx
465 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
466 Tuota tuota.
468 xxx Mhhy xxx
469 xxx hiiren napsaus xxx
470 xxx hiiren liikutus xxx xxx hiiren napsaus xxx
473 xxx hiiren napsaus xxx
478 xxx hiiren napsaus xxx
480 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
482 Tapahtuukos tässä yhtään.
484 xxx hiiren napsaus xxx
485 xxx hiiren liikutus xxx
487 xxx hiiren liikutus xxx
488 Jottai.
490 xxx hiiren napsaus xxx
495 xxx hiiren napsaus xxx
497 Siinä on perhe.
499 - xxx hiljaisuus xxx
500 xxx hiiren napsaus xxx
502 Sitten luodaan se.
504 - xxx hiljaisuus xxx
506 xxx hiiren napsaus xxx
508 xxx hiiren napsaus xxx
514 Niin joo se on xxx hiiren napsaus xxx tuolla.
517 - xxx hiljaisuus xxx
519 xxx hiiren napsaus xxx
520 Montako näitä on.
521 xxx hiiren napsaus xxx
522 Niin.
523 xxx hiiren napsaus xxx
524 Tuossa xxx epäselvä "" xxx luodaan.
526 - xxx hiljaisuus xxx
531 xxx epäselvä "" xxx menee sinne ihan ok.

535 Missäs tää nyt voisi olla.
537 xxx hiiren napsaus xxx
540 Nii. Ja sillä on se xxx hiiren liikutus xxx kuvaus siellä tuommonen.
546 Eiku ei.
547 - xxx hiljaisuus xxx
551 xxx hiiren napsaus xxx
557 xxx hiiren napsaus xxx
558 xxx hiiren napsaus xxx
560 xxx hiiren napsaus xxx
561 xxx hiiren napsaus xxx
562 Mites tää nyt oikein on.
564 xxx hiiren liikutus xxx
565 xxx hiiren napsaus xxx
568 Tekeeköhön
569 xxx hiiren napsaus xxx
570 xxx hiiren napsaus xxx
574 people.
575 xxx hiiren napsaus xxx
577 xxx hiiren napsaus xxx
578 xxx hiiren napsaus xxx
581 xxx hiiren napsaus xxx
582 xxx hiiren napsaus xxx
592 xxx Mhhy xxx
594 Kyl siitä vois tuo.
596 xxx hiiren napsaus xxx
601 xxx hiiren napsaus xxx
602 xxx hiiren napsaus xxx
603 xxx hiiren napsaus xxx
604 Noi lapset.
606 xxx hiiren napsaus xxx Ryhmä.
607 xxx hiiren napsaus xxx
609 Sit ne xxx hiiren napsaus xxx henkilöt.
612 xxx hiiren napsaus xxx
614 xxx hiiren napsaus xxx
617 xxx hiiren napsaus xxx
618 xxx hiiren napsaus xxx
620 xxx hiiren napsaus xxx
621 xxx hiiren napsaus xxx
623 Jaaha.
625 xxx hiiren liikutus xxx
626 Elikkä noi ainaki xxx hiiren napsaus xxx menee tuonne.
629 - xxx hiljaisuus xxx
630 xxx hiiren napsaus xxx
633 xxx hiiren napsaus xxx
634 Ja se tekee ne xxx hiiren napsaus xxx sitten tuolla.
638 Jaaha.
639 xxx hiiren napsaus xxx
640 - xxx hiljaisuus xxx
646 Noin.
647 - xxx hiljaisuus xxx
650 xxx hiiren napsaus xxx
652 xxx hiiren napsaus xxx
653 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
655 xxx hiiren napsaus xxx
656 Siellä xxx hiiren napsaus xxx ja sit sama kaikille.
659 xxx hiiren napsaus xxx
660 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
661 xxx hiiren napsaus xxx
662 xxx hiiren napsaus xxx
663 xxx hiiren napsaus xxx
664 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
666 xxx hiiren napsaus xxx
667 xxx hiiren napsaus xxx
668 xxx hiiren napsaus xxx
669 xxx hiiren napsaus xxx
671 xxx hiiren napsaus xxx
672 xxx hiiren napsaus xxx
673 xxx hiiren napsaus xxx
674 - xxx hiljaisuus xxx
677 Mmm.
682 Elikkä nyt siellä on tuo.
684 Ja sit toi pikkuveli kans.

628 - xxx hiljaisuus xxx
633 xxx hiiren napsaus xxx
698 Mitäs se nyt rupes tekemään.
641 xxx hiiren napsaus xxx
642 xxx hiiren napsaus xxx
645 xxx hiiren liikutus xxx
706 Elikkä jos.
648 xxx hiiren napsaus xxx
713 Tuota tuota.
656 - xxx hiljaisuus xxx
719 xxx hiiren napsaus xxx Jos ei oo.
720 xxx hiiren liikutus xxx
724 xxx hiiren napsaus xxx
727 firstFamily get children kun se lähtee.
729 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
732 xxx hiiren napsaus xxx
736 Ööö.tuota tuota
738 - xxx hiljaisuus xxx
743 xxx hiiren napsaus xxx
744 - xxx hiljaisuus xxx
745 xxx hiiren napsaus xxx
746 firstFamily getChildren xxx hiiren napsaus xxx ja xxx hiiren liikutus xxx sit olevinaan katotaan jotain tuommosta vaikka se nyt ois sitte.
755 xxx hiiren napsaus xxx
756 - xxx hiljaisuus xxx
762 xxx hiiren napsaus xxx
763 xxx hiiren napsaus xxx
764 - xxx hiljaisuus xxx
765 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
766 xxx hiiren napsaus xxx
767 - xxx hiljaisuus xxx
768 xxx Mhhy xxx
769 Mitäs se nyt. xxx hiiren napsaus xxx
770 Niin nyt se laittaa ne sinne.
772 xxx hiiren napsaus xxx
773 xxx hiiren napsaus xxx
774 - xxx hiljaisuus xxx
775 xxx hiiren napsaus xxx
776 xxx hiiren napsaus xxx
777 - xxx hiljaisuus xxx
780 Jaaha.
781 xxx hiiren napsaus xxx
783 Onpas xxx hiiren napsaus xxx hässäkkä.
785 xxx hiiren napsaus xxx
786 Nii. xxx hiiren napsaus xxx Se nyt xxx hiirennapsaus xxx laittaa xxx hiiren napsaus xxx niitä nyt tuonne.
790 xxx hiiren napsaus xxx
791 - xxx hiljaisuus xxx
792 xxx hiiren napsaus xxx
793 - xxx hiljaisuus xxx
794 xxx hiiren napsaus xxx
795 - xxx hiljaisuus xxx
797 xxx hiiren napsaus xxx
798 - xxx hiljaisuus xxx
799 xxx hiiren napsaus xxx
800 Ja sitten.
801 xxx hiiren napsaus xxx
802 xxx hiiren napsaus xxx
803 xxx hiiren napsaus xxx Lisää.
805 xxx hiiren napsaus xxx
806 xxx hiiren napsaus xxx
807 xxx hiiren napsaus xxx
808 xxx hiiren napsaus xxx
809 xxx hiiren napsaus xxx
810 xxx hiiren napsaus xxx xxx Mhhy xxx
811 xxx hiiren napsaus xxx
812 xxx hiiren napsaus xxx
813 xxx hiiren napsaus xxx
814 xxx hiiren napsaus xxx
815 xxx hiiren napsaus xxx
816 Ja xxx hiiren napsaus xxx vielä xxx hiiren napsaus xxx yksi.
819 xxx hiiren napsaus xxx
820 xxx hiiren napsaus xxx
821 xxx hiiren napsaus xxx

822 xxx hiiren napsaus xxx
823 xxx hiiren napsaus xxx
824 xxx hiiren napsaus xxx
825 xxx hiiren napsaus xxx
826 xxx hiiren napsaus xxx
827 xxx hiiren napsaus xxx
828 xxx hiiren napsaus xxx
829 xxx hiiren napsaus xxx
830 xxx hiiren napsaus xxx
831 xxx hiiren napsaus xxx
833 xxx hiiren napsaus xxx
835 xxx Mhhy xxx
836 Elikkä nytpä ne on siinä.
840 Nyt se rupee hakeen noitten lapsia.
842 xxx hiiren napsaus xxx
843 - xxx hiljaisuus xxx
844 xxx hiiren napsaus xxx
845 Elikkä ekaks tuo tekee ton ryhmän.
849 xxx hiiren napsaus xxx
850 - xxx hiljaisuus xxx
851 xxx hiiren napsaus xxx
852 - xxx hiljaisuus xxx
855 xxx hiiren napsaus xxx
856 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
857 xxx hiiren napsaus xxx
858 - xxx hiljaisuus xxx
860 xxx hiiren napsaus xxx
861 Sitten mitäs se sitte tekee.
866 Tuolla String namessa oli jotain ihme.
869 xxx epäselvä "" xxx
870 xxx hiiren napsaus xxx Ni joo se tulostaa sitten niitä sieltä.
875 Joo
876 - xxx hiljaisuus xxx
877 Täh.
878 xxx hiiren napsaus xxx
879 - xxx hiljaisuus xxx
880 xxx hiiren napsaus xxx
881 - xxx hiljaisuus xxx
882 Tässä nyt oo xxx hiiren napsaus xxx mitään tolokkua tässä hommassa.
884 xxx hiiren napsaus xxx
885 - xxx hiljaisuus xxx
886 xxx hiiren napsaus xxx
887 - xxx hiljaisuus xxx
889 xxx kello xxx
890 xxx Mhhy xxx
891 xxx hiiren napsaus xxx
892 - xxx hiljaisuus xxx
893 xxx hiiren napsaus xxx
894 xxx muu, "Nii. Sullon enää 2 minuuttia jälellä." xxx
895 Joo-o.
900 xxx hiiren napsaus xxx
901 - xxx hiljaisuus xxx
903 xxx hiiren napsaus xxx
905 Aika xxx hiiren napsaus xxx vähiin kyllä jääpi virheet. En tiiä.
908 xxx hiiren napsaus xxx
909 people isAChild
910 xxx hiiren napsaus xxx
911 xxx hiiren napsaus xxx
912 Tuota. Mistä se nyt oikein xxx hiiren napsaus xxx tarkistaa xxx hiiren napsaus xxx sen, että onko.
918 Ai niin, se tuon iän perusteella. Niinpä xxx hiirennapsaus xxx tietysti.
921 xxx hiiren napsaus xxx
922 xxx hiiren napsaus xxx
923 xxx hiiren napsaus xxx
924 - xxx hiljaisuus xxx
925 xxx hiiren napsaus xxx
926 - xxx hiljaisuus xxx
928 xxx hiiren napsaus xxx
930 Ka siinähän oli xxx hiiren napsaus xxx.
933 xxx hiiren napsaus xxx
934 xxx hiiren napsaus xxx
935 No niin.
936 - xxx hiljaisuus xxx

938 No tossa nyt ainaki o virhe. Jos se tot-palauttaa tosi tossa. Tossa, tossa person-luokassa toi lapsijuttu jos se on ikä on yli 18. Rivillä 19. Nii ei se kyllä lapsi. Lapsi silloin oo.
959 Jos se.
960 xxx hiiren napsaus xxx
962 xxx hiiren liikutus xxx
963 xxx hiiren napsaus xxx
965 Joo.
966 - xxx hiljaisuus xxx
975 xxx hiiren napsaus xxx
976 - xxx hiljaisuus xxx
980 xxx hiiren napsaus xxx
981 - xxx hiljaisuus xxx
982 Nii ja onhan tuossa, jos kattoo suoraan tota keskiarvon laskemistaki itse asiassaki.Ni. Nyt vähä hahmottaa tätä. Ni. Siinä vaan tossa rivillä 61 niin työnnetään vaan tohon yhteen muuttujaan ni se ei käy aina eikä sitä plussata xxx hiiren liikutus xxx siihen ollenkaan, niin se sitten jakaa sen viimesen iän niin si- sillä niitten lukumäärällä, ni siitähän tulee ainaki toi kes- keskimääräsen iän

1010 xxx kello xxx
1011 virhe.
1012 hih
1013 Loppu
1014 xxx muu, "Loppu" xxx
1015 - xxx hiljaisuus xxx

Appendix 4: Textual data P4

Note: The times are in seconds.

41 Eli nyt töa- ei o. Niinku ei tarvihe ruveta korjaamaan niitä virheitä, vaan etsiä virheitä, vai?
47 xxx muu, No. No jos haluat. Ihan xxx
49 Joo.
50 xxx muu, miten haluat.xxx
52 xxx hiiren liikutus xxx Mh-h.
54 - xxx hiljaisuus xxx
57 xxx vetää henkeä xxx
60 - xxx hiljaisuus xxx
77 xxx outo ääni taustalla xxx
78 - xxx hiljaisuus xxx
83 xxx outo ääni taustalla xxx
85 - xxx hiljaisuus xxx
90 xxx outo ääni taustalla xxx
91 - xxx hiljaisuus xxx
92 xxx outo ääni taustalla xxx
97 xxx outo ääni taustalla xxx
102 - xxx hiljaisuus xxx
105 Ok. xxx outo ääni taustalla xxx
106 xxx hiiren liikutus xxx
107 xxx start-klikkaus hiirellä ja käynnistysääni xxx
109 xxx vetää henkeä xxx xxx hiiren liikutus xxx xxx hiiren liikutus xxx
111 Eli tuota.
112 xxx vetää henkeä xxx
113 - xxx hiljaisuus xxx
116 Tosta.
117 xxx hiiren liikutus xxx
118 Mistähän tätä lähtis purkamaan.
120 Yks vaihtoehto ois ruveta tutkii noita tulosteita ja ruveta sen xxx hiiren liikutus xxx perusteella selvittää, että mikä siellä on. Kun toi esimerkiks noita xxx hiiren liikutus xxx välitulosteita tekee xxx hiiren liikutus xxx ihan selkeesti xxx hiiren napsaus xxx liikaa. Mut katotaan nyt xxx hiiren liikutus xxx et mitä tää haluaa teknisesti tehä.

135 xxx hiiren napsaus xxx
136 xxx outo ääni taustalla xxx
137 xxx hiiren liikutus xxx
138 Eli ajetaan sitä.
138 xxx hiiren napsaus xxx
140 xxx hiiren napsaus xxx
142 xxx Duoda.
143 xxx hiiren napsaus xxx
144 Family.
147 Sitten xxx outo ääni taustalla xxx siellä on.
151 xxx hiiren napsaus xxx
152 - xxx hiljaisuus xxx
153 xxx hiiren napsaus xxx
154 xxx epäselvä Duupaduup. xxx
155 xxx hiiren napsaus xxx
156 xxx outo ääni taustalla xxx
164 - xxx hiljaisuus xxx
185 xxx outo ääni taustalla xxx
188 xxx hiiren liikutus xxx
189 xxx hiiren napsaus xxx
190 Joo.xxx hiiren napsaus xxx xxx hiiren liikutus xxx

191 Täytyy sanoa, et mä en ikinä ole kovin xxx hiiren napsaus xxx hirveesti noista xxx hiiren napsaus xxx animaatioista öperustanu kun koodia oppinu lukemaan, nii sitä melkein lukee xxx hiiren liikutus xxx mieluummin sitä xxx hiiren liikutus xxx koodia. Siit ei noista animaatioista ei aina saa selvää, et nää tää koodi pitää ensiks ymmärtää, ennen kun noista animaatioista xxx hiiren liikutus xxx saa xxx hiiren liikutus xxx hirveesti mitään.

210 Person-luokka.
212 Josta luodaan 4 oliota eli siihen tallennetaan siis noi
218 - xxx hiljaisuus xxx
220 ilmeisestikin henkilöt.
222 Siellä pitäs olla nimi.
223 xxx vetää henkeä xxx
225 Eli.
226 xxx ähh xxx
230 xxx hiiren napsaus xxx
231 Ok eli tuota. xxx hiiren napsaus xxx xxx hiiren liikutus xxx
233 Tuolla on. xxx hiiren napsaus xxx xxx hiiren liikutus xxx

234 xxx ähh xxx
237 xxx hiiren napsaus xxx
238 xxx ähh xxx
239 Nimeen xxx näppäimistöllä kirjoitusta xxx annetaan tuota xxx näppäimistöllä kirjoitusta xxx toi xxx hiiren liikutus xxx xxx ähh xxx sukupuoli tai.

245 xxx muu, Jos löydät sano xxx
247 Eli
247 xxx muu, rivinumero xxx
248 rivi rivi kakstoista ja luokka Person.
251 xxx muu, Joo ja vika? xxx
253 Vikana oli se, että ni- name-muuttujaan tallennettiinkin öö sukupuoli vaikka sinne olis pitäny tunkee se nimi. xxx hiiren napsaus xxx Sen takia se ainakintulosti tuolta ton femalen tuolla lopussa.
272 Mutta katellaan lisää.

275 isAChild. xxx hiiren napsaus xxx Jos ikä on suurempaa kuin 18. xxx hiiren napsaus xxx Virhe. Person-luokka xxx hiiren liikutus xxx, rivi seittemäntoista xxx hiiren liikutus xxx. Öö.isAChild-metodin nimi viittais siihen xxx hiiren liikutus xxx, että se palauttas truen, jos joku on lapsi ja se on lapsi jos ikä on alle kaheksantoista eikä suurempaa. Eli käännetään toisinpäin xxx hiiren napsaus xxx toi. xxx hiiren liikutus xxx

298 Ja printti xxx hiiren napsaus xxx niin printtaa oikein. Eli sen jälkeen xxx hiiren liikutus xxx sen person-luokka xxx hiiren napsaus xxx periaatteessa pitäs olla kasassa.
307 xxx hiiren napsaus xxx
308 mmm xxx Mhhy XXX
310 xxx hiiren napsaus xxx
311 Family vois.
312 Family extendaa gruupppia Grouppi.
316 xxx hiiren liikutus xxx
317 Groupissa on Person-taulukko.
320 xxx hiiren liikutus xxx
321 Henkilömäärä.
322 Mikä on seuraava henkilö.
325 mmm
327 description
328 Joo
329 xxx hiiren liikutus xxx
330 - xxx hiljaisuus xxx
331 xxx epäselvä xxx People xxx epäselvä xxx Person xxx epäselvä xxx People

337 xxx hiiren napsaus xxx
338 Luodaan tuo.
339 xxx hiiren napsaus xxx
342 Uus taulukko.
345 Missä ei o ketään.
348 Sitten.
351 Lisätään sinne xxx hiiren napsaus xxx xxx hiiren napsaus xxx väkeä.
355 - xxx hiljaisuus xxx
358 xxx ähh xxx xxx hiiren napsaus xxx
361 - xxx hiljaisuus xxx
364 Toi ei kyllä oo oikein. Nyt vaa ei osaa vielä sanoa, että mikä siihen pitäs. Rivillä kolkytyheksän xxx hiiren liikutus xxx on virhe. Ö. Include-metodissa, luokassa Group. Ainakin minun logiikka sanoo, että ei se voi olla toi numberOfPeople xxx hiiren napsaus xxx toi sijotuspaikka xxx hiiren napsaus xxx xxx hiiren napsaus xxx vaan se pitää olla joku muu, mutta sitä pitää kattoo kohta lissee.

385 - xxx hiljaisuus xxx
387 xxx suun maiskahdus xxx xxx outo ääni taustalla xxx
388 Eikun xxx hiiren napsaus xxx hetkinen. Kyllähän se nyt xxx hiiren napsaus xxx periaatteessa xxx outo ääni taustalla xxx kun sitä käytetään noin.

397 Aluks on 0 ja sitte on.
399 Joo, kyl se sittenki toimii. Ei siinä o mitään.
400 Se lisätään. Kyl se laskee sitä xxx hiiren napsaus xxx ihan xxx hiiren napsaus xxx oikein.

408 xxx vetää henkeä xxx xxx hiiren liikutus xxx
411 Mmm. nextPerson xxx hiiren liikutus xxx palauttaa xxx hiiren liikutus xxx seuraavan henkilön.

417 xxx hiiren liikutus xxx.
418 xxx hiiren liikutus xxx
419 mmm xxx hiiren liikutus xxx
420 - xxx hiljaisuus xxx
425 people-taulukosta xxx hiiren liikutus xxx NextPersonIndex plusplus.
432 - xxx hiljaisuus xxx
436 xxx hämm xxx Ja tuossa taas pitäis muistaa, että onko se. xxx epäselvä xxx. Oliko se xxx vetää henkeä xxx plusplus oikeessa välissä. No, sen näkee kohta. Eli toi xxx hiiren napsaus xxx plusplusa xxx hiiren napsaus xxx vaikuttaako se. Kerkiikö se vaikuttaa tohon vai vaikuttaako se vasta seuraavaan ja kummin se pitää olla xxx hiiren liikutus xxx, mut sitä ei vielä muista osaa kattoa.

456 xxx vetää henkeä xxx
458 NextPerson i-xxx epäselvä xxx on pienempää kun numberOfPeople.
463 - xxx hiljaisuus xxx

464 Pienempää tai yhtä suurta, niin sitten nollataan.
471 - xxx hiljaisuus xxx
473 mmm xxx hiiren liikutus xxx
474 Ei. Toi ei ainakaan luultavasti xxx hiiren napsaus xxx xxx hiiren napsaus xxx toimi. Rivi 50 xxx hiiren napsaus xxx if-lause, luokassa Group.
481 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
483 Oöö xxx hiiren liikutus xxx
484 Verrataan xxx hiiren liikutus xxx PersonIndexia xxx hiiren liikutus xxx ja jos se on xxx hiiren liikutus xxx pienempää xxx hiiren liikutus xxx kun mikä tahansa oikeestaan, niin sitten nollataan.
494 Loogisesti ajatellen, nin jos se on suurempaa tai yhtä suurta, nin sit sen pitäs varmaan hypätä takasin nollaan. xxx outo ääni taustalla xxx Kun muutenhan se.
505 mmm
507 siitä ei voi saaha ulos ku sen suurinpiirtein 1. tai 2. tyypin.
513 - xxx hiljaisuus xxx

516 Aa. Ikää se ei osaa laskee oikein, mut sekin vois itse asiassa johtua tosta samasta ongelmasta xxx hiiren napsaus xxx xxx hiiren liikutus xxx ja jollain tapaa.
526 xxx vetää henkeä xxx
528 mmm
530 Nollasta.
531 People length.
532 - xxx hiljaisuus xxx
535 Pienempää kyllä.
537 for-lause kunnossa.
538 Summaan xxx outo ääni taustalla xxx lisätään
599 - xxx hiljaisuus xxx
542 iät.
543 - xxx hiljaisuus xxx

546 xxx outo ääni taustalla xxx xxx hiiren liikutus xxx
548 xxx outo ääni taustalla xxx
550 xxx hiiren liikutus xxx
554 Mutta, niin.
xxx hiiren liikutus xxx

557 Ei se nyt xxx hiiren napsaus xxx ihan toimi xxx hiiren napsaus xxx xxx hiiren liikutus xxx koska tuota.
562 - xxx hiljaisuus xxx
563 aaa xxx hiiren liikutus xxx
564 Siel ei välttämättä ois niin montaa niitä, mutta. Todennäkösesti xxx hiiren liikutus xxx oletus on se äh, että ööy perheen henkilöitten xxx hiiren napsaus xxx määrä on vastaa sitä mitä se onki.
579 Joo.
579 - xxx hiljaisuus xxx
580 xxx vetää henkeä xxx
581 xxx hiiren liikutus xxx
585 Familyn pitäs extendata se sitte.
587 Tätä koodia lukiessa menee suurinpiirtein tää ö viistoista minuuttia, että tän saa xxx muu, Mh xxx ymmärrykseen. Mut, noo.
595 xxx hiiren liikutus xxx
596 xxx hiiren liikutus xxx
600 xxx hiiren liikutus xxx
602 No, xxx hiiren liikutus xxx xxx hiiren napsaus xxx katotaas nyt välillä xxx epäselvä ku ö xxx xxx hiiren napsaus xxx xxx hiiren liikutus xxx haittasko korjaukset vai xxx hiiren napsaus xxx auttoko ne.

610 xxx hiiren liikutus xxx
612 xxx outo ääni taustalla xxx
617 xxx hiiren liikutus xxx
619 xxx outo ääni taustalla xxx
621 Okei. Nyt sinne nimet menee ainaki ihan oikein.
626 - xxx hiljaisuus xxx
631 xxx hiiren liikutus xxx xxx hiiren napsaus xxx
634 Se ei sinänsä oo.
636 - xxx hiljaisuus xxx
638 xxx hiiren napsaus xxx
639 xxx hiiren napsaus xxx
640 xxx hiiren napsaus xxx
641 - xxx hiljaisuus xxx
642 Hirvee hätä.
643 - xxx hiljaisuus xxx
654 xxx outo ääni taustalla xxx
656 - xxx hiljaisuus xxx
669 xxx hiiren napsaus xxx
670 Jaa-a.

671 xxx hiiren liikutus xxx
672 Nyt se tulostelee varmaan noita. Noita, mitkä on virhetulosteissa merkitty female, male, male, male.

680 Eve, Adam, Cain, Abel. Kyllä.
684 - xxx hiljaisuus xxx
689 xxx outo ääni taustalla xxx
691 Duddaduo.
692 - xxx hiljaisuus xxx
694 Joskus Nikolle ehotin, että tässä pitäs iha ehottomasti olla semmone xxx hiiren liikutus xxx semmone breakpoint, johonka pystys xxx muu, M-m xxx asettamaan. Et sen sais niinku nuo tulosteet ö xxx hiiren liikutus xxx skipattas toi animaatio xxx hiiren liikutus xxx siihen asti. xxx hiiren liikutus xxx Koska xxx hiiren liikutus xxx tässäki jos tätä haluaa kattoo näitä xxx hiiren liikutus xxx lopputuloksiaki, ni se vaikka animaation nopeus olis täysillä nii se aika pitkään tota pyörittelee.

778 xxx outo ääni taustalla xxx
723 Dumdum.
725 Joo-o.
726 - xxx hiljaisuus xxx
736 Ok. Eli se ei vielä noita tekstejä tuolta ei saa. Chil- Descriptionit ei mee tonne xxx outo ääni taustalla xxx oikeisiin paikkoihin, mutta la- osaisko se nyt noi lap- lapset sitte laskee.
749 - xxx hiljaisuus xxx
755 Ei sinne päinkään.
747 - xxx hiljaisuus xxx
780 xxx vihellystä xxx
783 Yhtään xxx hiiren liikutus xxx korjausta xxx hiiren liikutus xxx ei vielä saatu aikasiks. xxx hiiren napsaus xxx xxx hiiren liikutus xxx Mikäs siinä.
786 xxx hiiren napsaus xxx
787 xxx hiiren liikutus xxx
788 xxx hiiren liikutus xxx
789 mm-h-h-h-hh
790 xxx hiiren liikutus xxx
792 xxx hiiren napsaus xxx
793 xxx hiiren liikutus xxx
794 - xxx hiljaisuus xxx
796 mmm
797 xxx hiiren liikutus xxx
798 xxx hiiren liikutus xxx
799 xxx hiiren liikutus xxx
800 - xxx hiljaisuus xxx
802 xxx epäselvä xxx
807 Eli eli.
808 - xxx hiljaisuus xxx
811 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
812 Joo se kuttuu childrenistä getDescriptionia, niin miksi se ei sitten anna sitä ulos.
821 Onko childrenillä xxx hiiren liikutus xxx xxx hiiren liikutus xxxdescription.
824 - xxx hiljaisuus xxx
825 xxx hiiren napsaus xxx
826 xxx hiiren napsaus xxx Children-luokka on xxx hiiren liikutus xxx hetkinen.
831 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
832 xxx hiiren liikutus xxx
834 xxx hiiren napsaus xxx xxx hiiren napsaus xxx
835 xxx hiiren napsaus xxx Grouppi.
896 xxx hiiren liikutus xxx
898 xxx hiiren liikutus xxx xxx hiiren napsaus xxx
899 xxx hiiren napsaus xxx Groupin xxx hiiren napsaus xxx descriptioni luodessa xxx hiiren napsaus xxx xxx hiiren liikutus xxx periaatteessa sinne pitäs sijottaa.
847 xxx hiiren napsaus xxx
848 - xxx hiljaisuus xxx
849 xxx hiiren napsaus xxx
850 - xxx hiljaisuus xxx
853 mmm
856 xxx hiiren napsaus xxx
857 xxx hiiren napsaus xxx
858 Eli toi firsFamilyn getChildrenin pitäs.
862 - xxx hiljaisuus xxx
863 xxx hiiren napsaus xxx
864 - xxx hiljaisuus xxx
865 firsFamily oli xxx hiiren napsaus xxx xxx hiiren napsaus xxx family-luokkaa.
869 xxx hiiren liikutus xxx
871 Family. xxx hiiren liikutus xxx
873 xxx hiiren liikutus xxx
874 Tuossa xxx hiiren liikutus xxx.
877 xxx hiiren napsaus xxx
878 xxx hiiren napsaus xxx Periaatteessa sen sinne sen pistää xxx hiiren liikutus xxx, mutta xxx hiiren liikutus xxx mitenkäs xxx hiiren liikutus xxx se ei sitte xxx hiiren liikutus xxx pääse xxx hiiren liikutus xxx

888 xxx kello xxx ulos.
890 xxx muu, 2 minuuttia. xxx
891 Joo.
894 xxx hiiren liikutus xxx
895 - xxx hiljaisuus xxx
898 mmm
900 xxx hiiren napsaus xxx
901 xxx hiiren napsaus xxx
903 xxx hiiren liikutus xxx
904 xxx hiiren napsaus xxx Description xxx hiiren napsaus xxx.
906 - xxx hiljaisuus xxx
908 xxx hiiren napsaus xxx xxx hiiren liikutus xxx xxx hiiren napsaus xxx
910 xxx hiiren napsaus xxx
916 xxx hiiren napsaus xxx xxx hiiren liikutus xxx
918 xxx näppäimistöllä kirjoitusta xxx
921 xxx hiiren liikutus xxx Ei kun en muista.
922 xxx hiiren napsaus xxx
923 xxx hiiren liikutus xxx
924 xxx hiiren napsaus xxx
925 Katotaas.
928 xxx hiiren liikutus xxx
929 xxx hiiren liikutus xxx
950 Täs ajamisessa menee öh niin sanotusti turhaan aikaa koska se. Jos tietää mihinkä kohtaan haluais mennä, niin sit se simulointi vie jonkin verran turhaan aikaa.

967 - xxx hiljaisuus xxx
974 xxx hiiren liikutus xxx xxx hiiren napsaus xxx
976 xxx hiiren napsaus xxx
977 - xxx hiljaisuus xxx
980 xxx hiiren napsaus xxx
981 xxx hiiren liikutus xxx
982 - xxx hiljaisuus xxx
987 xxx hiiren napsaus xxx
988 xxx hiiren napsaus xxx
990 xxx hiiren liikutus xxx
992 xxx hiiren liikutus xxx
993 xxx hiiren napsaus xxx
994 xxx hiiren napsaus xxx
995 xxx hiiren napsaus xxx
996 xxx hiiren napsaus xxx
997 xxx hiiren napsaus xxx
998 xxx hiiren liikutus xxx
1000 xxx vihellystä xxx Voi xxx epäselvä rakentaa vielä xxx
1008 xxx kello xxx
1009 xxx hiiren napsaus xxx
1010 xxx muu, Ok xxx
1011 Joo.
1012 Ei. Ei hirveesti kerenny auttaa.
1014 xxx melua xxx
1019 xxx loppu xxx

Appendix 5: Source program

Note: During the visualization code lines are moved 2 lines below e.g. line 1 becomes line 3. This is due the line 1 is for ”import jeliot.io.*; ” and line 2 is empty.

```
1:  import java.io.*;
2:
3:  public class Person {
4:     public int age;
5:     public String name;
6:     private String sex;
7:
8:     public Person() {}
9:
10:    public Person(String pname, int page, String psex) {
11:        age = page;
12:        name = psex; //Bug!
13:        sex = psex;
14:        }
15:
16:    public boolean isAChild() {
17:        return (age > 18); //Bug!
18:    }
19:
20:    public void print() {
21:        System.out.println(name);
22:    }
23: }
24:
25:  public class Group {
26:     protected Person[] people;
27:     public int numberOfPeople;
28:     private int NextPersonIndex;
29:     protected String description = "NULL";
30:
31:     public Group(int n, String desc) {
32:         people = new Person[n];
33:         numberOfPeople = 0;
34:         NextPersonIndex = 0;        //BUG! no initial value
35:     }
36:
37:     public void include(Person p) {
38:             if (numberOfPeople < people.length) {
39:                 people[numberOfPeople] = p;
40:                 numberOfPeople++;
41:             } else
42:                 System.out.print("Too many people in this group. "
43:                                 + "Couldn’t include ");
44:             p.print();  //Bug!
45:         }
46:
47:     public Person nextPerson() {
48:         Person next_person = people[NextPersonIndex++];
49:
50:         if (NextPersonIndex <= numberOfPeople) //bug!
51:             NextPersonIndex = 0;
52:         return next_person;
53:     }
54:
55:     public double getAverageAge() {
56:         double sum = 0;
57:
58:         for (int i = 0; i < people.length; i++) {
59:             sum = people[i].age; //Bug!
60:         }
61:         return sum / people.length;
62:     }
63:
64:     public String getDescription() {
65:         return description;
66:     }
67: }
```

```

 public class Family extends Group {
    public Family(int n, String desc) {
        super(n, desc);
    }

    public Group getChildren() {
        Group children_group = new Group(numberOfPeople, "children");

        for (int i = 0; i < numberOfPeople; i++) {
            if (people[i].isAChild())
               children_group.include(people[0]); //Bug!
            }
        return children_group;
    }

   public static void main() {
       Family firstFamily = new Family(4, "first family");
       Group children;
       Person next_person = null;
       int i = 0;
       //Do NOT change the next 4 lines
       Person mum = new Person("Eve", 35, "female");
       Person dad = new Person("Adam", 40, "male");
       Person bigBrother = new Person("Cain", 10, "male");
       Person littleBrother = new Person("Abel", 5, "male");

         firstFamily.include(mum);
       firstFamily.include(dad);
       firstFamily.include(bigBrother);
        firstFamily.include(littleBrother);

        children = firstFamily.getChildren();
        System.out.println("The " + children.getDescription() +
              " in the " + firstFamily.getDescription()
              + " are:");

        if (i < children.numberOfPeople) { //BUG!
            next_person = children.nextPerson();
            i++;
        }
    next_person.print(); //Bug!
    System.out.println("Average age is " +
                      firstFamily.getAverageAge());
    }
}
```