UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Joosep Kaimre

Proficiency and Usage of AI in an Introductory Object-Oriented Programming Course

Master's Thesis (30 ECTS)

Supervisor: Marina Lepp, PhD

Tartu 2024

Proficiency and Usage of AI in an Introductory Object-Oriented Programming Course

Abstract:

In the year 2022, there was a breakthrough in the usage of AI chatbots thanks to the release of ChatGPT. One affected field is computer science and its teaching due to the possibility of generating code snippets with AI chatbots. Due to the newness of the topic, there is a lack of consensus on how students use these tools and how proficient AI chatbots are in solving different tasks. The aim of this master's thesis is to understand how proficient different AI-based chatbots are in solving tasks in the course “Object-Oriented Programming” and how students use these tools. To achieve this, ChatGPT and Copilot were used to solve tests and exams, and their results were compared to students’ results. Additionally, data regarding students’ usage of AI chatbots was gathered with a questionnaire focused on frequency of use, ways of use, and impact on learning. Currently, the free version of ChatGPT-3.5 performs worse than the average student, and Copilot is about on par with the average student. However, they are both still capable of passing the course. The students stated they used AI chatbots mainly for programming tasks, and students with lower grades use them more frequently.

Keywords:

artificial intelligence (AI), object-oriented programming, teaching programming, ChatGPT, Copilot, novice programmers

CERCS:

P175 Informatics, systems theory, S281 Computer-assisted education

Tehisintellekti võimekus ja kasutus algajatele suunatud objektorienteeritud programmeerimise kursusel

Lühikokkuvõte:

Aastal 2022 toimus läbimurre tehisintellektil põhinevate juturobotite kasutuse osas tänu ChatGPT avalikustamisele. Üks tugevalt mõjutatud valdkond on informaatika ja selle õpetamine, arvestades erinevate juturobotite pädevust koodilahendusi genereerida. Teema uudsuse tõttu puudub konsensus ja üldine arusaam, kuidas üliõpilased kasutavad selliseid abivahendeid ja kui hästi suudavad need abivahendid lahendada erinevaid ülesandeid. Magistritöö eesmärk on välja selgitada, kui hästi suudavad tehisarul põhinevad juturobotid lahendada kursusel "Objektorienteeritud programmeerimine" erinevaid ülesandeid ning mismoodi üliõpilased ise neid abivahendeid kasutavad. Selle tegemiseks lahendati ChatGPT ja Copilotiga kursuse kontrolltöö ja eksami ülesanded ning võrreldi nende tulemusi üliõpilaste tulemustega. Lisaks uuriti küsitlusega üliõpilaste tehisintellektil põhinevate juturobotite kasutust, keskendudes sagedusele, viisidele ja mõjule õppimisel. Selgus, et ChatGPT-3.5 tasuta versioon on hetkel veel keskmisest üliõpilasest kehvem ja Copilot on laias laastus sama võimekas kui keskmine üliõpilane. Siiski on mõlemad juturobotid piisavalt võimekad, et saada positiivne hinne. Üliõpilaste sõnul kasutavad nad juturoboteid põhiliselt programmeerimisülesandeid lahendades ning nõrgemad üliõpilased kasutavad neid rohkem kui võimekamad.

Võtmesõnad:

tehisintellekt (TI), objektorienteeritud programmeerimine, programmeerimise õpetamine, ChatGPT, Copilot, algajad programmeerijad

CERCS:

P175 Informaatika, süsteemiteooria, S281 Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

Table of Contents

Introduction

The year 2022 was a breakthrough for Artificial Intelligence. Following the public release of ChatGPT in November of that yesar [1], different AI assistants and chatbots received significant attention [2-3], with fears of AI replacing humans and making some jobs obsolete. One clearly affected field is Computer Science and its education [4].

The breakthrough of AI has sparked an avalanche of research on the topic of AI proficiency in university courses [4-7], its usage as a tool [8-10] and how it affects studying [11-12]. There currently is no consensus on whether AI is more capable than the average student, with some studies indicating that it is outperforming students [5-7], whilst others have found that AI is capable of passing the courses but is worse than students [13-14]. When focusing specifically on courses held on object-oriented programming, there is less research on how AI chatbots compare to students and whether there are specific problems AI makes mistakes in. Additionally, there has not been any significant research on how the AI assistants are capable of parsing input in Estonian and if it significantly impacts their skill level.

Limited research exists on students' perceptions, usage, and attitudes toward AI chatbots and their impact on learning in higher education, particularly in the realm of Computer Science. Some studies indicate that using AI chatbots during the learning process increases self-efficacy and computational skills [11], whilst others have found no statistically significant differences between users and non-users [12]. Supplementary AI materials have also been created [15], with findings suggesting a boost in students' internal motivation when engaging with such resources. In general, students seem to use it more whilst coding and less for explanations and other tasks [12]. Still, there is little research on how AI impacts student performance and skill acquisition.

The main goal of this thesis is to analyse the proficiency of different AI chatbots in an introductory object-oriented programming course and compare them to those of students taking the course. Additionally, students' usage and thoughts regarding AI assistants will be analysed to gauge their perceived positives and negatives. In this thesis, AI assistants, chatbots and AI chatbots are used interchangeably to indicate models based on AI with which students can communicate. To achieve the goal, this thesis will focus on the following research questions:

1. How do different chatbots perform in the course "Object-Oriented Programming" in comparison to students?

2. What are the common mistakes that AI chatbots make in the course "Object-Oriented Programming"?
3. How much and in what ways do students use various artificial intelligence-assisted methods during the course?

The thesis begins by giving an overview of the chatbots used for the analysis of the results. It is followed by a summary of current research on AI chatbots' proficiency in Computer Science courses, their usage by lecturers as a resource and their impact on students and on students' perceptions of them. An overview of the course "Object-Oriented Programming" is given in section 2, followed by the method section, which describes how AI chatbots were evaluated in the course and how student feedback was gathered. Section 4 presents the results, followed by section 5, in which the results are analysed.

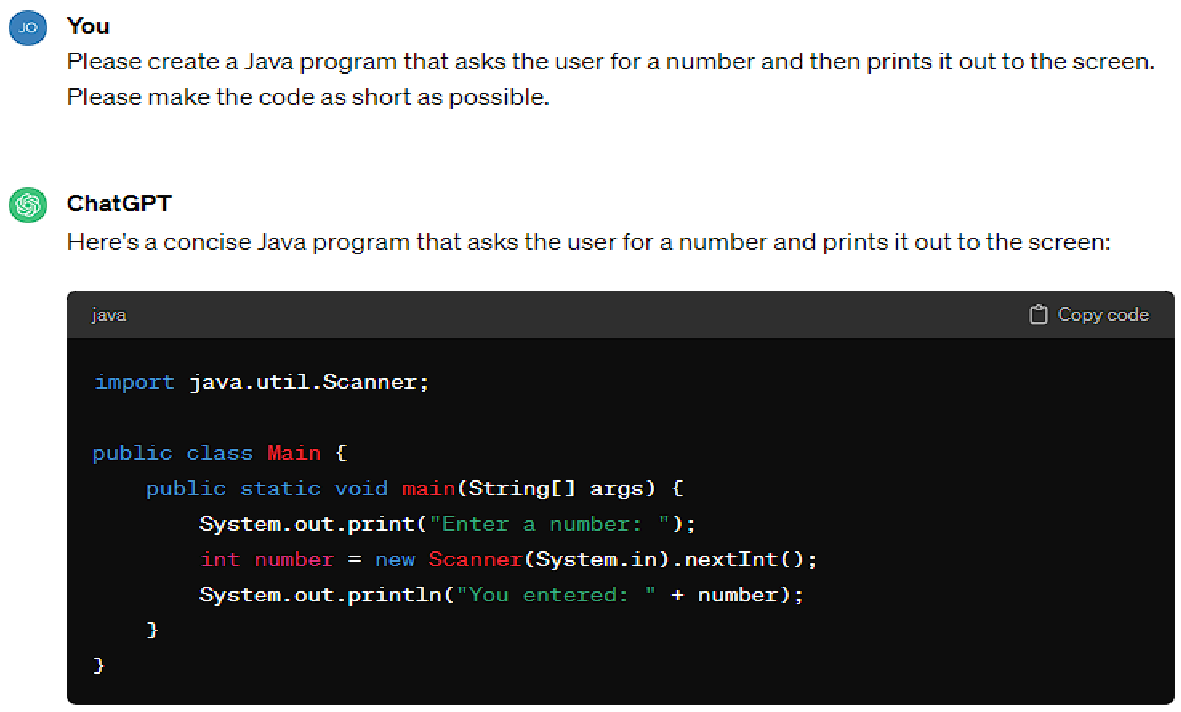
1 Background

This section covers the background of the AI chatbots ChatGPT and Microsoft Copilot that were used for comparison with students. This is followed by a literature overview on AI performance in computer science university courses, its role as a tool for both lecturers and students and the perspectives of students regarding the utilisation of AI chatbots in university courses.

1.1 ChatGPT

ChatGPT (Chat Generative Pre-trained Transformer) is a chatbot based on a Large Language Model (LLM), developed by OpenAI and released to the general public on November 30th, 2022 [1]. After its release, it became one of the fastest-growing websites in history [2], reaching 100 million users in about two months and six months after its release, it is estimated that 58% of the population of the USA knew of it [3]. The chatbot has different versions, with the GPT-3.5 version being free to use and the more complex GPT-4 version being available to paying subscribers.

ChatGPT can be interacted with via a message prompt. The message prompt can be on any topic, and ChatGPT will answer. An example prompt can be seen in Figure 1.



You
Please create a Java program that asks the user for a number and then prints it out to the screen. Please make the code as short as possible.

ChatGPT
Here's a concise Java program that asks the user for a number and prints it out to the screen:

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        System.out.print("Enter a number: ");
        int number = new Scanner(System.in).nextInt();
        System.out.println("You entered: " + number);
    }
}
```

Figure 1. Example of a ChatGPT prompt and answer

As can be seen in Figure 1, it responds to code prompts with a general description and a specially formatted code block. Usually, the code block also contains all the necessary imports, making it possible to directly copy the proposed answer to a file and run it.

1.1.1 Background of ChatGPT

Generative Pre-trained Transformers are a type of NLP (natural language processing) models. The generative indicates [16] the ability to create text based on a prompt, the pre-trained indicates that the model is pre-trained on data and does not continuously update its parameters. Transformers [17] are a type of NLP model that was introduced in 2017 based on a self-attention mechanism with multi-head attention and positional encoding. Self-attention enables the model to weigh the importance of each token in a sequence relative to others. Multi-head attention allows the capture of diverse types of information by applying self-attention in parallel. Data about token order is given by positional encoding. Each token's representation passes through a feed-forward network, with layer normalisation and residual connections aiding training stability. This all together helped to create a model that is state of the art and was able to outperform other models.

1.1.2 Versions of ChatGPT

ChatGPT is based on this model, with the first version released in 2018 [18] and subsequent versions released in 2019 (version 2), 2020 (version 3), and the most recent version released in 2023 (version 4). The parameters of the versions can be seen in Table 1.

Table 1. Data about ChatGPT versions [18]

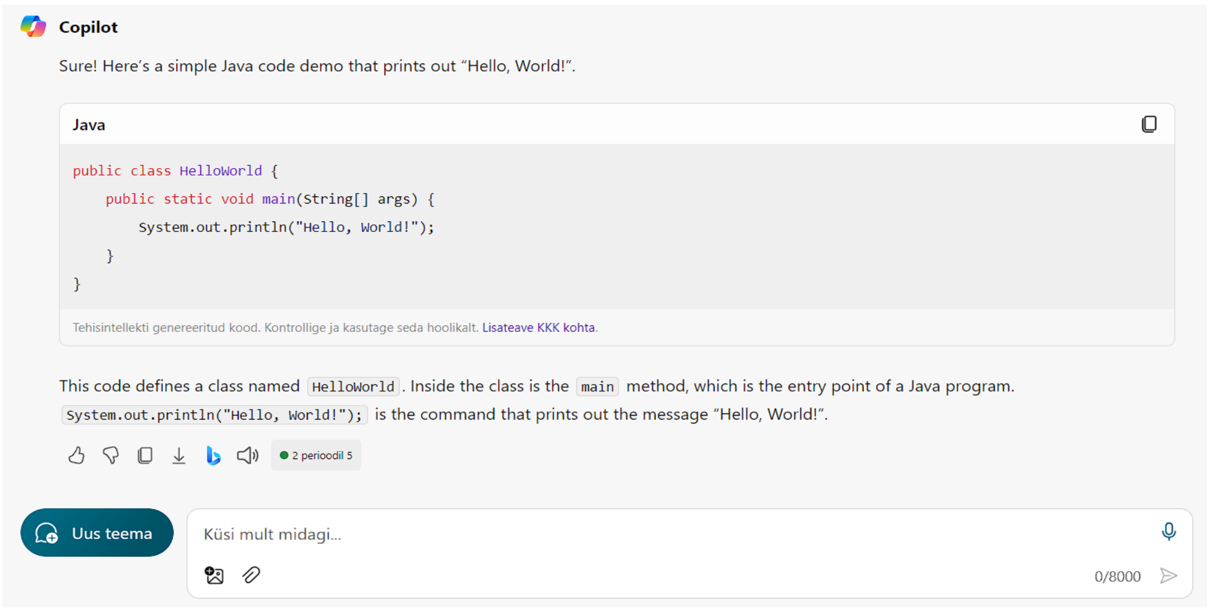
Version	GPT-1	GPT-2	GPT-3	GPT-4
Release date	June 2018	February 2019	May 2020	March 2023
Model parameters	117 million	1.5 billion	175 billion	Unpublished
Model layers	12 layers	48 layers	96 layers	Unpublished
Model parameter dimension	768 dimensions	1600 dimensions	12 888 dimensions	Unpublished
Pre-training data size	5 GB	40 GB	45 TB	Unpublished

As seen in Table 1, each subsequent version was trained on a larger dataset with more input variables and parameters, leading to a more complex and better-performing model. However, the official data about GPT-4 has not been revealed, but the assumption is that the number of parameters, layers, dimensions, and training data sizes have all increased to create a better-performing model. The models were trained on unlabeled data to promote a wider understanding of text for different uses and to not focus only on a singular use [16].

1.2 Microsoft Copilot

Copilot is a chatbot developed by Microsoft which was released for public use on February 7, 2023 [19]. On release, it carried the name Bing Chat, but its name changed following its release. Microsoft Copilot uses Microsoft's Prometheus model, which is built on OpenAI's GPT-4, and additional tweaks and changes have been made to it. Similarly to ChatGPT, it has a free and a paid version, with the paid version being faster [20] and having additional features. As Copilot is built on OpenAI's GPT-4, it is assumed that it has a similar amount of data, parameters, layers and dimensions. However, as Microsoft made additional changes, they are probably not identical. Additionally, as it is based on the same technology, it has the same benefits of having been trained on unlabeled data to gain a wider understanding of different text types and languages.

Copilot can be interacted with via a message prompt, similar to ChatGPT. Figure 2 shows the message prompt with a generated answer.


Copilot

Sure! Here's a simple Java code demo that prints out "Hello, World!".

Java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Tehisintellekti genereeritud kood. Kontrollige ja kasutage seda hoolikalt. Lisateave KKK kohta.

This code defines a class named `HelloWorld`. Inside the class is the `main` method, which is the entry point of a Java program. `System.out.println("Hello, World!");` is the command that prints out the message "Hello, World!".

Figure 2. Example of a Copilot prompt and answer

Overall, the prompt and text fields are similar to ChatGPT, with the user being able to ask about any topic and Copilot generating a response to it. Similarly to ChatGPT, Copilot formats code examples differently from surrounding text. The biggest difference is the fact that Copilot has an input character limit. In Figure 2, it is 8000, but it varies from 2000 to 16000 based on time and current usage.

1.3 Usage of AI in Courses

As the topic of AI assistants is currently relevant, multiple recent studies have come out regarding AI and university courses [5-7, 13-14]. Most of the works have focused on ChatGPT [7, 11-14], however, some have used others, such as GitLab Codex [5-6]. The focus has varied, with some focusing on AI proficiency and ability to pass courses [5-7], some focusing on the usage of AI as a supplementary study tool [21-22], and some focusing on the impact of knowledge acquisition [11-12, 15]. There have been papers published reviewing AI usage in non-English courses [23], however, there have been no papers written covering the subject in the Estonian university system and for courses held in Estonian. This section highlights the papers on performance and proficiency, student usage and supplementary materials.

1.3.1 AI Performance in University Courses

There have been multiple studies that estimate how effective different LLM AI approaches are for solving tasks for different courses in economics [24], law [25], computer science [5-7, 13] and medical studies [26]. However, the main focus of this thesis is on computer science and programming courses.

Multiple studies [5-7] have found that AI tools (Github Codex, ChatGPT) performance places it in the top quartile of class performance in introductory programming courses. AI approaches perform especially well in tasks covering introductory topics but with greater variance in more complex tasks pertaining to data structures and algorithms [6]. This variance in data structures is confirmed by Bordt and Luxburg [13], who found out that ChatGPT-3.5 was able to pass a course on the topic but performed worse than the students on average, whilst ChatGPT-4 performed on par with the students. Shoufan [14] had similar results, with ChatGPT achieving a passable grade but performing worse than students. Richards et al. [7] findings complement these, as they found that ChatGPT was able to pass in undergraduate programmes but failed at the postgraduate level and that it was able to outperform students at introductory topics but was surpassed by students at more complex levels. Contrary to these findings, Savelka et al. [27] found that there was no difference in performance comparing introductory and intermediate results, and in some cases, AI even performed better in intermediate-level assessments. Overall, it appears that AI assistants demonstrate superior performance at the introductory level compared to the expert level. However, the distinction between introductory and intermediate-level tasks and exams is less straightforward. Moreover, it appears that at the introductory level, AI can even surpass students' grades while facing more challenging problems, courses and tasks, the AI's performance may lag behind students, yet still suffice to meet passing standards.

Looking at the variance regarding topics, Joshi et al. [28] have found that AI performed the best in algorithm and data structures, followed by operating systems, machine learning and database management systems. Outside of university courses, AI chatbots have been evaluated against competitive programming challenges such as Leetcode [29]. ChatGPT outperformed the average human acceptance rate for human submission in almost every category and difficulty level (easy, medium, hard) with the sole exception being tasks on bit manipulation. However, these tasks are not as directly comparable to university courses, as individuals may not feel as compelled to perform to the best of their ability, given that the grades from these online tasks do not affect their GPA.

ChatGPT specifically has been evaluated for generating solutions from non-English input (Czech) in information security courses [23]. In the information security courses, ChatGPT was capable of passing all four courses in which its outputs were evaluated, and it usually outperformed students in full-text exams where a text-based answer or solution must be provided. However, the students on average performed better in tasks related to projects, essay writing and completing small snippets of code. All in all, in Czech information security courses the AI outperformed the students' average in one evaluated course, and the students outperformed the AI in the other three courses.

When focusing on AI courses held in Java on object-oriented programming, there is less research, especially on comparing student results to AI assistants. Ouh et al. [30] have found that ChatGPT can solve easier tasks but runs into problems when the tasks get more complex. However, it still offers partial solutions that give students a starting point. The author noted that another issue is that when tasks give some data as UML diagrams or API documentation, then the chatbots are incapable of parsing the whole text input. This problem is not specific to Java or object-oriented tasks, but how AI assistants are capable of parsing non-text input still needs to be taken into consideration. This is supported by Camara et al. [31], who found that ChatGPT is not capable of reliably generating UML outputs and has problems with its syntax. The problems with object-oriented concepts are confirmed by Cipriano and Alves [32], who tried different AI chatbots to generate solutions to object-oriented programming tasks. They found that the AI-generated code often had compilation errors, needed multiple prompts to generate all the necessary classes and functions and did not pass all the unit tests that were used to evaluate task correctness. Overall, it seems that AI assistants can solve easier tasks, but when dealing with more complex and difficult concepts, they have problems. The lack of studies comparing AI assistants to students also makes it hard to gauge whether AI’s problems with the tasks are similar to those of students or if it can outperform them on some tasks and underperform in others.

In general, it seems that AI has reached a point where it is able to successfully pass university courses but is not proficient enough yet to outperform the students' average regularly. This could stem from the fact that AI proficiency in different tasks varies greatly. Small code snippet tasks and even exam paragraph questions are easily solved by AI, but larger, complex projects and programs and essays are not yet done well by the AI. For essays, the AI often has problems with correct citations and generating false citations [23]. Hence, whether AI outperforms the students’ average in a course is often dependent on the grading scheme of the course and which tasks make up what percentage of the final grade. An additional factor that determines AI performance is how the descriptions of the tasks are presented, as it has problems with parsing input from UML diagrams and API documentation. AI can also pass courses not in English, but there has not been sufficient research on whether this is exclusive to Czech or is true in other languages as well, especially on tasks in Estonian. There is also a lack of research that focuses specifically on courses given in Java on the topic of object-oriented programming, but in general, it seems to follow the trend seen in other courses, where it is great at easier and smaller tasks, but when the tasks get more complex, the performance falls. The lack of materials on AI compared to students in Java courses also makes it difficult to see where AI performance differs from the students and whether the mistakes are similar.

1.3.2 AI Usage as a Lecturer

AI also offers additional support for students in introductory programming courses. Becker et al. [33] propose that code-generation AI tools can be used by students to get exemplar solutions for tasks, see multiple ways to solve a task and get examples of code written with good style and quality. However, Cipriano and Alves [32] have found that when AI assistants generate code, they often use bad style, indicating that using AI assistant-generated code for better style should be taken cautiously. Becker et al. [33] also mention that AI assistants can be used to get comments and explanations for code that works well. Sarsa et al. [34] have similarly found that LLM can be used to comment on codes to gain a better understanding of it. The models usually comment on every line, however, they often contain small mistakes, particularly regarding being specifically correct about comparisons and other small, easily fixable problems. These AI-based code explainers have also been made into plugins that students can use. GPTutor [21] is an example that uses ChatGPT to provide explanations inside Visual Studio Code. Another code explainer tool was created by Wang [22], who used it as a tutor for students to help learn object-oriented programming. Both of these tools received positive feedback from students, indicating a possibility of integrating AI code explainers into IDEs to help ease programming study. However, as they are still prone to mistakes, communicating the fallibility of these comments is paramount.

Denny et al. [8] compared AI-generated learning materials to student-created resources and the two were evaluated by the students to be of equal value to them. Some researchers have created AI assistants (WorkedGen) [35] specialised in creating step-by-step solution guides with code examples and comments. This assistant was evaluated by students, and the majority of them found it to be useful and would use it again when doing their tasks and homework. This overall indicates a possibility of using AI-generated materials in courses, however, their quality should be assessed and using AI-generated code for style examples should be used cautiously.

In the context of lecturers using AI to ease their work, Kiesler et al. [9] analysed the quality of ChatGPT provided feedback for homework with mistakes. They found that for the majority of the prompts, the AI provided factually incorrect information, which indicates that automating feedback for students with chatbots is not currently viable. These findings are similar to those of Jukiewicz [10], who found that whilst AI and teacher gradings are statistically correlated, they also statistically differ, indicating that AI sometimes makes mistakes when grading students' homework. Overall, this shows that human supervision on grading is still needed and full automation is not currently possible as mistakes, where the teacher gave full marks and the AI gave a 0 are too problematic to make replacing human graders with purely AI ones not feasible.

Sarsa et al. [34] have written about using AI to generate programming tasks for students and evaluated their novelty, the proposed sample solution and automatic tests provided for the programming tasks. About 75% of the AI-generated tasks were novel and sensible, and a matching sample solution was provided. However, only about a quarter of the provided sample solutions passed the tests the AI also proposed. This shows that using AI for tasks is a possibility, however, it is incapable of reliably providing a full set of tasks with solutions and tests, indicating that the generated content needs further human moderation.

Artificial Intelligence offers a range of potentials for educators developing programming courses. AI-generated materials have demonstrated comparable value to those created by students, presenting opportunities for students to explore diverse sample solutions to complement their own solutions. Nevertheless, the quality of AI-generated code varies, cautioning against solely relying on them for teaching good coding practices. Presently, AI assistants remain unreliable for replacing human grading due to significant errors. However, they excel in providing explanations and comments on existing code examples, making them valuable resources for students encountering difficulties with course materials. Additionally, AI can assist in generating ideas and test cases for programming tasks, although their inability to provide a comprehensive set of tasks, solutions, and tests underscores the necessity of human oversight. In essence, AI materials serve as supplementary resources, enriching the learning experience, but their current limitations underscore the continued need for human involvement and supervision.

1.3.3 Students Opinions on Usage of AI in University Courses

Students' perception, usage and thoughts of AI chatbots for higher education in Computer Science is a topic that has limited research, with the majority of existing research being written in the last few years due to their recent breakthrough into public consciousness. Strzelecki [36] has analysed the adoption of AI by students in the context of unified theory of acceptance and use of technology (UTAUT) in a Polish University. He found that the biggest impact on students' usage was habit, performance expectancy and hedonic motivation, indicating that when students see an AI performing to a good standard for their work and form a habit of asking for help from it, it leads to more prevalent usage. These findings are supported by Lai et al. [37], who found that positive experiences and accurate answers help promote the regular use of AI chatbots in students. Sun et al. [12] noted that the students' perceived usefulness, perceived ease of use, and intention to use AI assistants increased after using them, however, their attitude regarding AI remained unchanged. Overall, this indicates that students seem to understand and use AI assistants more after having to use them and learning about their strengths. However, this does not seem to affect their attitude regarding its usage, which is interesting.

Some studies have also analysed the impact of supplementary AI materials on students. Wu et al. [15] found that introducing a supplementary chatbot that helped students engage with the material increased their internal motivation to study and engage with the material. This increase in motivation could be useful for propagating programming learning, however, there are no comparisons as to whether this was specifically due to AI or just that additional supporting study materials increase motivation.

A study conducted in Turkey [38] found that students found the speed of AI chatbots to be their main advantage, with them being always available, speeding up learning about problems, and students even came to perceive the AI as a teacher, which indicates a high level of dependability. However, the perceived negatives were occupational anxiety and incorrect answers. The students seem to mainly use AI helpers specifically when coding [12] and also whilst debugging, but less for other tasks such as task decomposition or theory. This could just indicate that students think that AI assistants are purely for coding and less for getting explanations and comments.

The effects of AI usage on students' skills have also been studied, but the research is limited. Some have found [11] that programming students who used ChatGPT whilst studying improved their computational thinking skills and programming self-efficacy more than students who did not use AI. The subcategories that saw the difference in improvement were creativity, algorithmic thinking and programming itself. However, others [12] have found that there are no statistical differences in results between students who use AI to learn and those who do not. In conclusion, while some studies suggest that incorporating AI into programming education enhances students' computational thinking skills and programming self-efficacy, further research is needed to fully understand the impact of AI usage on student outcomes.

Overall, students' adoption of AI chatbots seems to hinge on forming a habit and the chatbot's ability to give correct and good answers. However, it seems that this change in usage does not translate into an attitude change. It also seems that introducing AI-based materials increases the students’ internal motivation for studying, yet it is unclear whether any additional study material could cause this increase. Students valued the speed and constant availability of AI chatbots, even viewing them as teachers, but mentioned occupational anxiety and incorrect answers as negative. They primarily utilised AI assistants for coding and debugging tasks, suggesting a perception that AI is mainly suited for coding assistance rather than broader educational support. The influence of AI usage on students’ abilities is unclear, with some research indicating that using AI benefits the students using it, while others noted no differences. All in all, students seem ready to use AI chatbots and AI-generated materials for study purposes, and it seems that these tools positively impact studying.

2 LTAT.03.003 “Object-Oriented Programming”

Object-oriented programming (OOP) is a widespread paradigm in modern-day programming. It is based on the idea of objects, which are instances of classes that encapsulate data and behaviour related to those data, and these objects interact with each other via interfaces to exchange messages and manipulate data [39]. This paradigm is taught to first-year university students at the University of Tartu in the course LTAT.03.003 “Object-Oriented Programming” using the programming language of Java. The data, tasks, and results from this course were used to compare the proficiency of AI chatbots in programming tasks to that of beginners in programming.

2.1 Object-Oriented Programming

Object-oriented paradigm got its start in the 60s with the programming language Simula [39], however, it reached mainstream popularity in the 90s with languages such as Java, Python and C++ becoming ubiquitous. Their usage remains high to this day, as all three place in the top 4 of the TIOBE index [40], which tracks the popularity of different programming languages. Their rise and stay in popularity can be explained by their ability to ease computer program complexity using abstraction [41], as it helps to select and choose only necessary parts of different models. Capretz [41] also argues that the rise in GUI (graphical user interface) helped facilitate the rise in popularity as they are easier to create when using the object-oriented paradigm. These factors remain important to this day, keeping these languages relevant.

Java programming language was released for public use in 1995 [42] and rapidly gained popularity. Java was based on C++ but with cleaner constructs and a more pure object-oriented approach [41], which helped to maintain its popularity. It is supported to this day, with the most recent long-term support release being Java 21 in 2023 [43], with a newer version of Java 22 being released in 2024. Its sustained popularity can be explained by the prevalence of the object-oriented programming paradigm and it being one of the most common languages to study object-oriented programming in.

2.2 Course

The course “Object-Oriented Programming” is mainly taught to first-year Computer Science students as it is a mandatory course. However, it is chosen as an elective by many other students, making it one of the largest courses at the University of Tartu, with a yearly registration between 270-330 students [44]. As a prerequisite for the "Object-Oriented

Programming" course, students must complete an introductory course in the Python programming language.

As the students are not required to have previous experiences with object-oriented programming or Java, majority of them are complete beginners with the language and topic at the start of the course. As stated on the course website [45], the aim of the course is to give foundational knowledge about the object-oriented paradigm. During the course, the students will learn about encapsulation, inheritance, polymorphism, method overriding, data structures, and event-driven programming. By the end, they are expected to understand and be capable of independently creating, testing and debugging programmes.

2.2.1 Course Structure

The course is built on weekly lectures, homeworks and practice seminars. Each week, the students are expected to watch the weekly lecture videos, take a short quiz on them, solve the homework on the weekly topic and participate in the practice seminar to reinforce their understanding of the topic. During the course, they are expected to take two larger tests and complete two group assignments. At the end of the course, they have to take the exam. The distribution of the assignments, their point values, and the timeline can be seen in Table 2.

Table 2. Course assignments

Assignment	**Week**	**Points**	**Minimum score to pass**
Homework	Weekly	12x0.5	
Practice Seminar	Weekly	12x0.5	
Test 1	Week 7	16	12
Test 2	Week 13	16	
Group assignment 1	Weeks 6-8	5	
Group assignment 2	Weeks 12-14	5	
Group assignment presentation	Week 15	3	
All practice assignments		57	28
Lecture	Weekly	12x1	6
Exam	Week 16	33	15
Total score		**102**	**51**

As seen from Table 2, the final grade of the students is most dependent on the exam and the tests. The points earned in practice seminars are participation points and only require attendance; most homework assignments have automated tests that provide immediate feedback and allow students to submit their assignments until they achieve the maximum score. Group projects are open-ended tasks with specific requirements, but students have the freedom to choose the topic and implementation, resulting in less uniform assessment as the graders also consider the students' level. Therefore, we focus on comparing the results of AI and the students in the exam and tests, as they have clear and consistent grading schemes, enabling more precise comparisons of results. As these two also impact the final grade the most, it gives an insight into how proficient AI is compared to students and whether students could use AI for academic dishonesty, and how much it could actually help them.

2.2.2 Tests

The two tests take place respectively on weeks 7 and 13, and the tests contain the topics covered in the previous weeks. The students are given a description of a program, which includes the name of the classes that they have to create, the methods which the classes have to contain, whether some classes implement an interface or inherit some properties from a superclass and how the main workflow of the program is supposed to look like. To complete the program, the students have 105 minutes, they can use all materials except for communicating with another person or asking an AI chatbot for assistance. A sample test task can be seen in Appendix I. Both tests make up 16 points of the final grade with test 1 having a required minimum score of 12 to pass the course.

The first test mainly focuses on abstract classes, inheritance, polymorphism and interfaces. The second test also contains these topics, however in addition it also covers exceptions and exception handling, streams and more complex data structures. There is an automatic test that checks if all the necessary classes and methods are present and that the class hierarchy is correct, however, it does not check the internal logic of the program. The students themselves have to test and debug their program and be certain that it corresponds to the task description given.

2.2.3 Exam

The exam is taken at the end of the course and makes up 33 points of the final grade. The exam consists of a declaration of honesty, 13 short questions and a long more open-ended question. The exam is taken as a computer test on Moodle, and students can use course materials, look at code examples, and use documentation and Google. However, communicating with other people, using AI assistants, opening IDEs, compiling codes and running them is forbidden. The students have 60 minutes to complete the test. Table 3 shows the topics of the questions and the points distribution of the exam.

Table 3. Topics and points of exam questions

	Topic of Question	**Points**
Q1	Declaration of Honesty	1
Q2	Objects, Classes	2
Q3	Strings, Files, Lists	2
Q4	Interfaces	2
Q5	Interfaces	2
Q6	Subclasses, Superclasses	2
Q7	Subclasses, Superclasses	2
Q8	Abstract classes	2
Q9	Abstract classes	2
Q10	Graphics	2
Q11	Events	2
Q12	Streams	2
Q13	Exception handling	2
Q14	Data structures	2
Q15	Long question	6
Total		**33**

As can be seen from Table 3, the exam covers all the topics learned during the course. To pass the course, a minimum score of 15 points is needed. The first question is always the declaration of honesty and the last question is the longer, open-ended question. Between them are the other questions, however, they are in a randomised order. The other questions consist of multiple-choice, single-choice, fill in the gap and matching. Each question set may contain any of these options. The student cannot move back and forth between the questions, once they have moved on to the next question the previous question becomes unavailable. Examples of the short question can be seen in Appendix II. The longer open-ended question makes the students explain and reason for the answers they provided. The open-ended question has two different types. One consists of filling in the gaps of an existing code snippet where the student has to write all possibilities and give reasons for their answers. The other type consists of an existing code snippet with mistakes where there are multiple statements about what is wrong with the code. The students have to decide if the statement applies to the code snippet and explain their decision. Both versions of the longer task can be seen in Appendix III.

3 Method

This section gives a short overview of how the AI proficiency of the two chatbots was evaluated and how student feedback was gathered for analysis.

3.1 Analysing AI Proficiency

ChatGPT-3.5 and Microsoft Copilot were used for the purpose of testing. ChatGPT-3.5 was chosen as it was free to use, while ChatGPT-4 needed a paid subscription. As this is the more accessible version, it was also assumed that if students were to use AI for the course, then they would choose the free version. Microsoft Copilot was selected as University of Tartu students get free access via their emails to the paid version of the AI chatbot, which makes it more probable that students would use this AI.

To gauge how well ChatGPT and Copilot are capable of solving tasks in the introductory programming course, they were given the full text of the tests and tasks and the output was graded. ChatGPT was capable of handling the full text, Copilot had a character limit of 4000, which meant that sometimes tasks needed to be split into multiple queries. However, there were no additional modifications made meaning that the AI assistants were given the texts in Estonian, with no translation done. This created the additional aspect of the chatbots understanding text written in Estonian and giving the answers in Estonian.

The tests have a clear standardised grading guide which was used to evaluate the level at which the AI chatbots are capable of solving the tasks. The common mistakes were written down to understand which subjects can cause problems and if there are clearly repeating mistakes the AI chatbots make when solving the tests. As there were multiple variants of each test, the AI assistants were given three different versions to get more data points. As the exam is conducted as a Moodle test with many questions in the question bank, the assistants were given 10 questions from each set to get a more robust and trustworthy average performance for each question topic.

3.2 Student Feedback

Students' usage, thoughts and opinions on AI assistants were gathered with a survey conducted on the course Moodle page. The survey was conducted during the 8th week of the course which is roughly the midpoint. The survey contained six parts, some focusing on general feedback to lectures, seminars, tests, and homeworks. This year a section regarding AI assistants was added. Answering the questionnaire was voluntary, but completing it awarded a bonus point.

233 people answered the questionnaire, which is about 71% of the whole course. The questionnaire was not anonymous in order to award extra points to students and to tie answers about usage with a student's score in test 1. However, complete anonymity was ensured whilst analysing the results of the students' answers.

The AI section (Appendix IV) contained single and multiple-choice, open-ended and Likert scale questions. It began with a single-choice question of whether the student used any AI assistant for this course. Based on the answer the student saw different follow-up questions. Non-users saw a multiple-choice question and an open-ended question where they could answer why they had not used AI assistants. Students who had used them saw questions which gathered data about their experiences with AI assistants. The questionnaire aimed to find out how often, for which tasks and in which ways the students used the assistants. Additionally, we asked students to evaluate how much help they received from the assistants. Students could also write down what they liked and disliked about the AI assistants. The final section consisted of 5 statements in which students had to mark down their agreement or disagreement on a Likert scale. Additionally, a statistical analysis was conducted on the topic of students’ results in test 1 in relation to the frequency with which they used AI assistants. To analyse the results the students were divided into two groups based on the frequency of their usage where one group used AI assistants rarely or never whilst the other group contained the students who used them more frequently. The Shapiro-Wilk test was used to check whether the group results follow normal distribution and based on the results the Mann-Whitney U-test was used to see if there was a statistically significant difference between the two groups.

4 Results

This section covers the results of students and AI score comparison with a focus on the two tests and the exam. This is followed by an analysis of the answers to the student questionnaire and a statistical analysis of the difference in grades for students depending on AI assistant usage.

4.1 AI Proficiency in Tests

ChatGPT and Copilot were given the full texts of the tests and tasks and the output was graded according to the grading guide. There exist multiple versions of each test as there are numerous time slots when the tests and exams are taken. Each slot has a unique set of tasks so the AI assistants were given multiple versions of the tasks and each task was graded to get more insight and data regarding the chatbot's performance in the course. The course is in Estonian, so the input was Estonian and no modification was made to it.

4.1.1 Test 1

Test 1 covers the topics of Java classes, objects, Strings, files, lists, polymorphism, interfaces, and abstract, super- and subclasses. The test gave 16 points to the final grade and the minimum amount of points required to pass is 12 out of 16.

ChatGPT and Copilot were given the problem descriptions of three tests and their output was graded. As Copilot has a character limit on its input, the test description was split into two, whilst ChatGPT was given the test description with no modification. The results of the grading can be seen in Table 4.

On average, ChatGPT was capable of solving the tests to a score of 14.65 points out of 16 and Copilot was capable of solving the tests to a score of 15.85 points out of 16. These results surpass the required minimum score. There were two repeating mistakes in ChatGPT solutions: it never specified the required encoding for files and it defined the logic in superclass abstract methods and did not override it in subclasses. There were no repeating mistakes for Copilot as out of the 3 tests only once did it not specify the encoding and only once it added null values to a *toString* method.

Table 4. ChatGPT and Copilot results in test 1

Test version	T1.1	T1.2	T1.3
ChatGPT mistakes	● Does not use the required encoding for files (-0.25p) ● A method that needs to be abstract is defined (-1p)	● Does not use the required encoding for files (-0.25p) ● A method that needs to be abstract is defined (-1p) ● Uses wrong method names (-0.4p) ● A mistake in application logic (-0.4p)	● Does not use the required encoding for files (-0.25p) ● Does not read data from a file (-0.5p)
ChatGPT points	14.75 p	13.95 p	15.25 p
Copilot mistakes	● Displayed null values in *toString* method (-0.2p)	● Made no mistakes	● Does not use the required encoding for files (-0.25p)
Copilot points	15.8 p	16 p	15.75 p
Number of students	43	113	127
Student average	13.76 p (SD = 4.33)	14.55 p (SD = 2.98)	14.95 p (SD = 2.68)
Student median	15.75 p	15.6 p	15.7 p
ChatGPT average	14.65 p		
Copilot average	15.85 p		
Student average	14.61 p		

This test was taken by 285 students and the average amount of points for this test among students was 14.61 points. Compared to this, both ChatGPT and Copilot performed better as their averages were higher. However the difference between ChatGPT and the students was relatively small, whilst Copilot's difference was larger. Comparing individual tests, ChatGPT outperformed the student average on two of them, however, non-compiling solutions were given an automatic 0 that brought the student average down. Copilot outperformed the students on all tests.

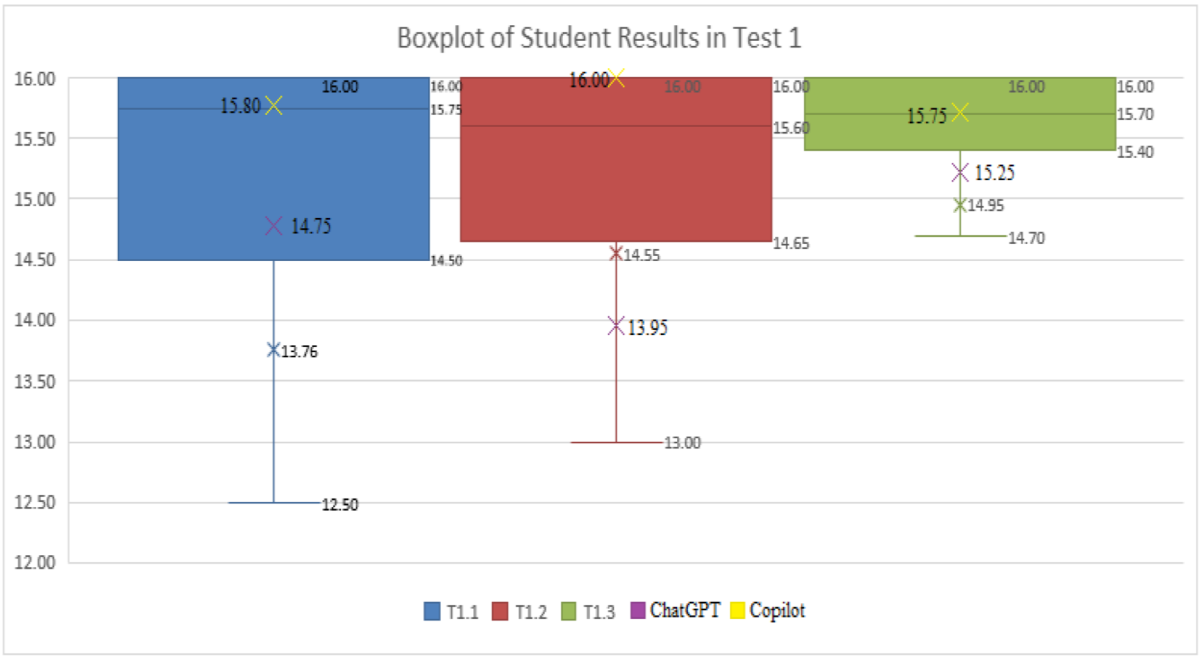


Figure 3. Students' and AI chatbots' results in test 1

Figure 3 depicts the boxplot of students' and AI chatbots' results for test 1. When visualising, the outliers were not included to enhance the readability of the figure as the non-compiling solutions were an automatic zero. The AI chatbots are indicated with coloured crosses and the student average is also shown as a cross on the boxplot. When looking at how the scores would have placed when looking at quartiles, ChatGPT scores were below the bottom quartile for T1.2, T1.3 and barely above it for T1.1. This indicates that ChatGPT does perform worse than the average student on long and complex problem tasks. Copilot fared better as it was above the upper quartile for T1.2 and between the median and upper quartile for T1.1 and T1.3 indicating that it outperforms the average student.

4.1.2 Test 2

Test 2 covers the topics of streams, exception handling, and data structures in addition to the topics already covered in test 1. The test gave 16 points to the final grade and there is no minimum score required to pass.

ChatGPT and Copilot were given the problem descriptions of three tests and their outputs were graded. As Copilot has a character limit on its input, the test description was split into two, whilst ChatGPT was given the test description with no modification. The results of the grading can be seen in Table 5.

Table 5. ChatGPT and Copilot results in test 2

Test version	T2.1	T2.2	T2.3
ChatGPT mistakes	● Methods are not private (-1p) ● The logic for asking user input does not work correctly (-0.5p) ● Does not use user input (-0.7p) ● Multiple mistakes in application logic (-2.2p)	● Methods are not private (-1p) ● The logic for asking user input does not work correctly (-0.5p) ● Sorts in the wrong direction (-0.5p) ● Has not generated some functionalities (-2.5p) ● Does not use user input (-0.7p)	● Methods are not private (-1p) ● Sorts in the wrong direction (-0.5p) ● Small mistake in reading input (-0.5p)
ChatGPT points	11.60 p	10.80 p	14.0 p
Copilot mistakes	● Did not generate *get* and *set* methods (-1p)	● Did not generate some *get* methods (-0.8p) ● Sorts in the wrong direction (-0.5p)	● Did not generate some *get* methods (-0.6p) ● A method was not private (-0.5p)
Copilot Points	15 p	14.7 p	14.9 p
Number of students	114	38	110
Student average	13.1 p (SD = 4.06 p)	13.93 p (SD = 2.71 p)	13.51 p (SD = 2.89 p)
Student median	14.8 p	14.9 p	14.5 p
ChatGPT average	12.13 p		
Copilot average	14.87 p		
Student average	13.39 p		

On average, ChatGPT was capable of solving the tests to a score of 12.13 points out of 16 and Copilot was capable of solving the tests to a score of 14.87 points out of 16. ChatGPT had multiple repeating mistakes, the most prevalent was that methods were public, not private. This requirement was written in the test as the methods should not be callable outside the class, which probably affected the outcome. The tests were different from each other in the sense that two required students to use queues and one required them to use maps. The tests (T2.1, T2.2) that used queues had noticeably more mistakes and had problems with reading user input whilst the version with maps did not have these problems. Also, there was a mistake regarding sorting, the tests required the sorting to be non-decreasing, but it sorted in the other direction.

Copilot had a repeating mistake in the fact that it did not generate some of the required *get* methods for any of the three tests and circumvented using them, other mistakes were more unique. Copilot also had a problem with sorting directions and making a method private just like ChatGPT. However, Copilot made these mistakes only in one variant, not across multiple tests which happened with ChatGPT. Copilot had no noticeable differences between the variants with queues (T2.1, T2.2) and the variant with maps (T2.3).

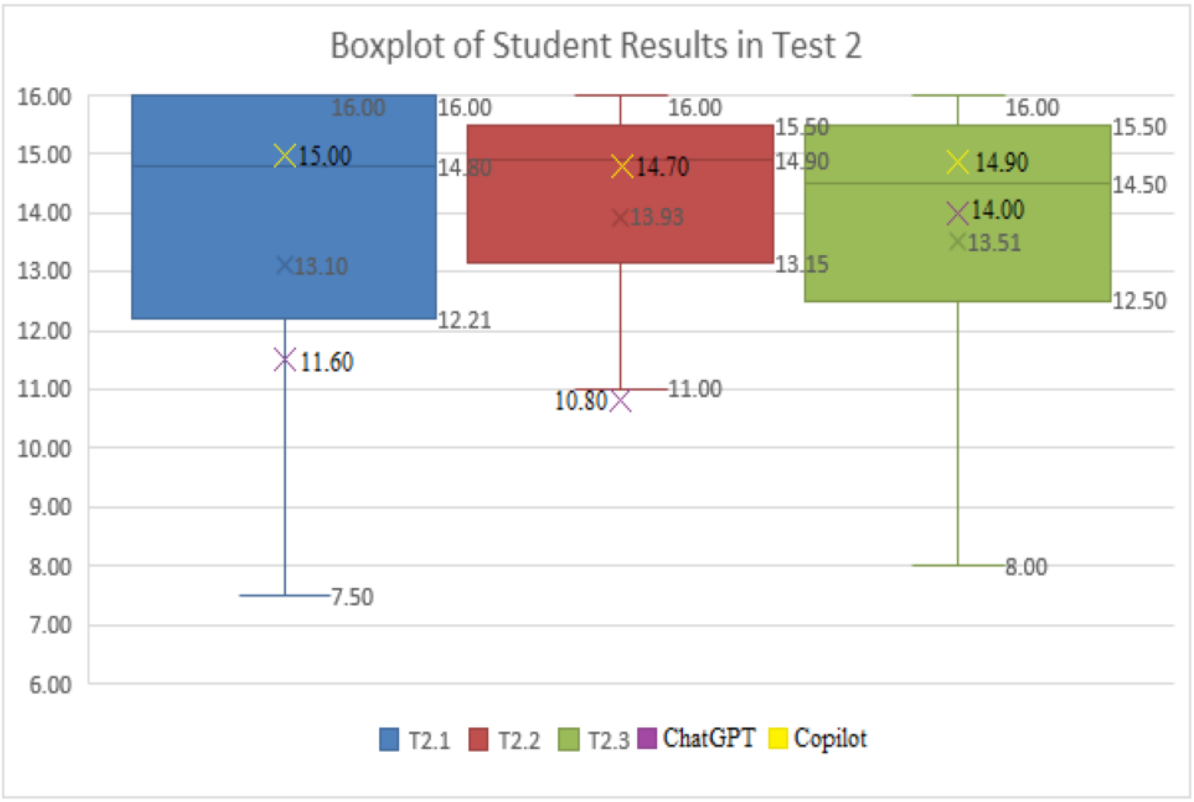


Figure 4. Students' and AI chatbots' results in test 2

The student average for these tests was 13.39 points out of 16. Here, ChatGPT got a lower score and Copilot got a higher score compared to the student. However, when looking at the tests individually, ChatGPT performed better than the students on T2.3, but it was worse at T2.1 and T2.2, whilst Copilot outperformed the students on all tests. Figure 4 depicts students' and AI chatbot' results in test 2. When visualising, the outliers were not included to enhance the readability of the figure as the non-compiling solutions were an automatic zero. The AI chatbots are indicated with coloured crosses and the student average is also shown as a cross on the boxplot. When looking at which quartile the chatbot scores would have placed, ChatGPT was below the bottom quartile for T2.1, T2.2 and above the bottom quartile, but below the median for T2.3. However, Copilot placed above the median for T2.1 and T2.3 and just below it in T2.2. These results reinforce that ChatGPT performs worse than the average student and Copilot outperforms the average student when doing these tests.

4.1.3 Exam

The exam covers all the topics taught in the course. It is taken as a computer quiz on Moodle in which the students need to declare that they will not use prohibited materials, answer 13 short questions and a longer, more difficult question in which students have to reason and explain their answers. The quiz chooses a question from a data bank of questions and each question covers different topics. A more in-depth overview was given in 2.2.3.

To aggregate ChatGPT and Copilot proficiency in the different questions and topics, they were given ten different questions from each block and the results were averaged to get a better overview of its results. The questions presented were the same for both AI assistants. The results can be seen in Tables 6-10.

As seen in Table 6, the AI assistants performed similarly in the questions on the topic of objects and classes. A clear difference in performance can be seen in Q2 on the topic of String, Files and Lists. Both of them had problems with different String methods, but ChatGPT also had problems with *Collections.sort* and list indexes. Their biggest difference which also resulted in the large point difference was the fact that ChatGPT had problems comprehending the String values saved in variables by confusing the values between different variables or making up new values. This problem was not present in Copilot solutions.

Table 6. AI assistant results in exam questions (part 1)

	Topic	AI	Points	Mistakes
Q1	Declaration of Honesty	-	-	-
Q2	Objects, Classes	ChatGPT	***Avg* = 1.9** SD = 0.32 *min* = 0 *max* = 2	● Did not choose answers from the possible answer list given
		Copilot	***Avg* = 1.95** SD = 0.16 *min* = 1 *max* = 2	● Did not follow the order of arguments of a method
Q3	Strings, Files, Lists	ChatGPT	***Avg* = 0.995** SD = 0.74 *min* = 0 *max* = 2	● Comparing substrings indexes ● Problems with uppercase and lowercase comparison ● Thought *Collections.sort* returned, not changed existing list ● Made mistakes with lists related to them being reference-based ● Did not comprehend which String value was saved in the variable
		Copilot	***Avg* = 1.7** SD = 0.63 *min* = 0 *max* = 2	● Problems with uppercase and lowercase comparison ● String contains method had problems

As can be seen in Table 7, Copilot consistently outperformed ChatGPT on the topics of interfaces, subclasses and superclasses as even if they made the same mistakes, Copilot made them less often. Both of them had identical problems regarding class hierarchy: confused the order of searching for a method declaration and had problems with how a superclass's constructor is called in instance creation. Another common mistake was thinking that an abstract class needs to implement an interface's methods. A mistake unique to Copilot was that it tried to define an abstract method in an abstract class without using the keyword abstract in front of the method. ChatGPT had more unique mistakes with it confusing when to use extends and implements, confused when you can use and when you have to use access keywords, abstract and what can be done in classes and what can be done in abstract classes.

Table 7. AI assistant results in exam questions (part 2)

	Topic	AI	Points	Mistakes
Q4	Interfaces	ChatGPT	***Avg* = 1.563** SD = 0.37 *min* = 1 *max* = 2	● Said that abstract class needs to implement interface methods ● An empty method body is still an implementation of a method ● Used extends for interfaces ● Said that interface methods need access keywords ● Used abstract methods in non-abstract classes
		Copilot	***Avg* = 1.847** SD = 0.63 *min* = 1.6 *max* = 2	● Said that abstract class needs to implement interface methods
Q5		ChatGPT	***Avg* = 1.4** SD = 0.78 *min* = 0 *max* = 2	● Used keyword class when methods are not implemented ● Said abstract cannot be used in interfaces ● Said you have to specify the access modifier in an interface ● Sorted in the wrong direction with comparable
		Copilot	***Avg* = 1.82** SD = 0.38 *min* = 1 *max* = 2	● Defined abstract methods without using the keyword abstract in an abstract class
Q6	Class hierarchy	ChatGPT	***Avg* = 1.64** SD = 0.39 *min* = 1 *max* = 2	● Failed to realise a superclass's constructor with no arguments is always called when creating an instance of a subclass ● Failed to realise when a subclass calls a superclass constructor with arguments then the constructor with no arguments is not called
		Copilot	***Avg* = 1.904** SD = 0.22 *min* = 1.33 *max* = 2	
Q7		ChatGPT	***Avg* = 1.733** SD = 0.64 *min* = 0 *max* = 2	● Method declarations are searched for starting from subclasses, not from superclasses
		Copilot	***Avg* = 1.8** SD = 0.63 *min* = 0 *max* = 2	

Table 8. AI assistant results in exam questions (part 3)

	Topic	**AI**	**Points**	**Mistakes**
Q8	Abstract classes	ChatGPT	***Avg* = 1.516** SD = 0.14 *min* = 0.66 *max* = 2	● Said that abstract classes cannot have realised methods ● Did not add the keyword abstract to abstract methods ● Said that abstract classes cannot have abstract subclasses ● Used extends with interfaces ● Did not implement all abstract methods in non-abstract subclass
		Copilot	***Avg* = 1.68** SD = 0.35 *min* = 1 *max* = 2	● Said that an abstract class needs to implement all superclass methods ● Used extends with interfaces ● Said that you can override method only when superclass is abstract
Q9		ChatGPT	***Avg* = 1.663** SD = 0.26 *min* = 1.33 *max* = 2	● Thought that interfaces cannot contain variables ● Thought that abstract classes cannot contain only non-abstract methods
		Copilot	***Avg* = 1.826** SD = 0.29 *min* = 1.33 *max* = 2	● Said that abstract classes cannot have realised methods ● Implemented methods in interfaces ● Thought that interfaces cannot contain variables
Q10	Graphics	-	-	● Could not be analysed (contained pictures in the questions)
Q11	Events	ChatGPT	***Avg* = 1.45** SD = 0.50 *min* = 0.5 *max* = 2	● Made mistakes when String comparison and methods were used ● Confused < and <= ● Sometimes confused different variables
		Copilot	***Avg* = 1.75** SD = 0.35 *min* = 1 *max* = 2	● Made mistakes when String comparison and methods were used ● Confused != and == ● Sometimes confused different variables ● Confusion with list indexes

Table 8 contains AI performance in abstract class and event problems. The graphics question could not be analysed as it contained a picture of a JavaFX program which could not be given as input to the AI assistants. Copilot continued to outperform ChatGPT on these topics. Both continued to have problems with how inheritance, abstract classes and interfaces interact with each other and which methods need to be realised where and which methods do not have to be realised. For events, all of the problems stemmed from logical and string comparison mistakes, not from misunderstanding how events and changes work. Overall, Copilot and ChatGPT made quite similar mistakes and the main difference in points stems from the fact that ChatGPT just made those mistakes more often than Copilot.

Table 9. AI assistant results in exam questions (part 4)

	Topic of Question	AI	Points	Mistakes
Q12	Streams	ChatGPT	***Avg* = 1.799** SD = 0.34 *min* = 1.14 *max* = 2	● Had a problem of not understanding when reading the input file had reached the end of the file ● *readUTF* cannot comprehend input written with the method *writeInt* ● Had a problem understanding how long the input file is
		Copilot	***Avg* = 1.869** SD = 0.29 *min* = 1.14 *max* = 2	● *readUTF* cannot comprehend input written with the method writeInt
Q13	Exception handling	ChatGPT	***Avg* = 1.857** SD = 0.12 *min* = 1.75 *max* = 2	● Did not notice a print statement ● An exception thrown in a catch block is not caught
		Copilot	***Avg* = 1.732** SD = 0.62 *min* = 0 *max* = 2	
Q14	Data structures	ChatGPT	***Avg* = 1** SD = 1.05 *min* = 0 *max* = 2	● Did not understand how Stack data structure works ● Had problems with Queue element removal, and did not understand it worked as FIFO ● Had problems when a set was given the same element multiple times
		Copilot	***Avg* = 1.6** SD = 0.84 *min* = 0 *max* = 2	● Did not understand how Stack data structure works

The AI results on the topic of streams, exception handling and data structures can be seen in Table 9. Overall, there was only one topic in which ChatGPT was capable of outperforming Copilot and it was exception handling, which is interesting as in all other topics Copilot gained better marks. The AI assistants both had problems noticing print statements and just ignored them and thought that if in a catch block an exception is thrown then it is automatically caught which is not true. Another common mistake for them was that they did not understand that data streams cannot use *readUTF* and *writeUTF* for *readInt* and *writeInt* and vice versa, with ChatGPT having additional problems comprehending file lengths. Stack data structure was something that both AI assistants had problems with and it was the only topic where both AIs were always fully wrong, which probably indicates that the training data did not contain enough relevant data about it. ChatGPT had additional problems with Queues and Sets, which is something that was also present in test 2 solutions, indicating that Copilot seems to comprehend different less common data structures better than ChatGPT.

Table 10. AI assistant results in the long exam question

	Topic of Question	**AI**	**Points**	**Mistakes**
Q15	Long question	ChatGPT	***Avg* = 4.3** SD = 1.06 *min* = 3 *max* = 5.5	● Did not add Comparable interface when necessary ● Did not add access modifiers ● Had a problem with String to Integer and Double conversions ● Did not use interfaces ● Thought that method signatures must contain throws NumberFormatException ● Non-static methods cannot be called directly in a static context ● Did not mention creating subclass instances both with subclass and superclass types
		Copilot	***Avg* = 4.65** SD = 0.66 *min* = 3.5 *max* = 5.5	● Did not use interfaces or abstract classes, only class ● Did not mention creating subclass instances both with subclass and superclass types ● Said that protected methods cannot be called from other classes ● Said that a class's main method cannot contain instances of said class ● Did not add access modifiers ● Did not add a call to superclass constructor in subclass

The result of AI in the longer tasks with explaining can be seen in Table 10. The long tasks needed understanding the topics from the whole course, which meant that the mistakes present there were quite wide-ranging and often were similar to mistakes seen in the previous questions, like superclass constructor calls and String-to-number conversion. One quite prevalent problem was that when the task needed filling in the gaps then the answer needed all possible solutions, not just one correct. However, the AI assistants almost never gave multiple possible answers, only just one, for example, only using public whilst other keywords also fit or just using abstract class or the superclass for instance creation. The tasks also contained explaining the proposed solution, which the AI assistants excelled at if their initial answer was correct. Here Copilot also outperformed ChatGPT, which is similar to previous topics and tasks.

Table 11. Comparison of results between ChatGPT, Copilot and students

	ChatGPT	**Copilot**
Points	23.816	27.128
Number of Students	287	
Student Average	26.898 (SD = 3.624)	
Student Median	27.33	

A comparison of students' results and AI assistants' results can be seen in Table 11. As the AI is unable to solve the JavaFX task where some of the input is given as a picture, it is an automatic 0 points. Additionally, one point comes from accepting the declaration of honesty, but as all students must check it to complete the exam and you cannot fail it, we automatically give this point to AI as well for better comparison sake. Copilot has a higher average than the students whilst ChatGPT has a lower average. The students’ and AI chatbots’ results in the exam can be seen as a boxplot in Figure 5.

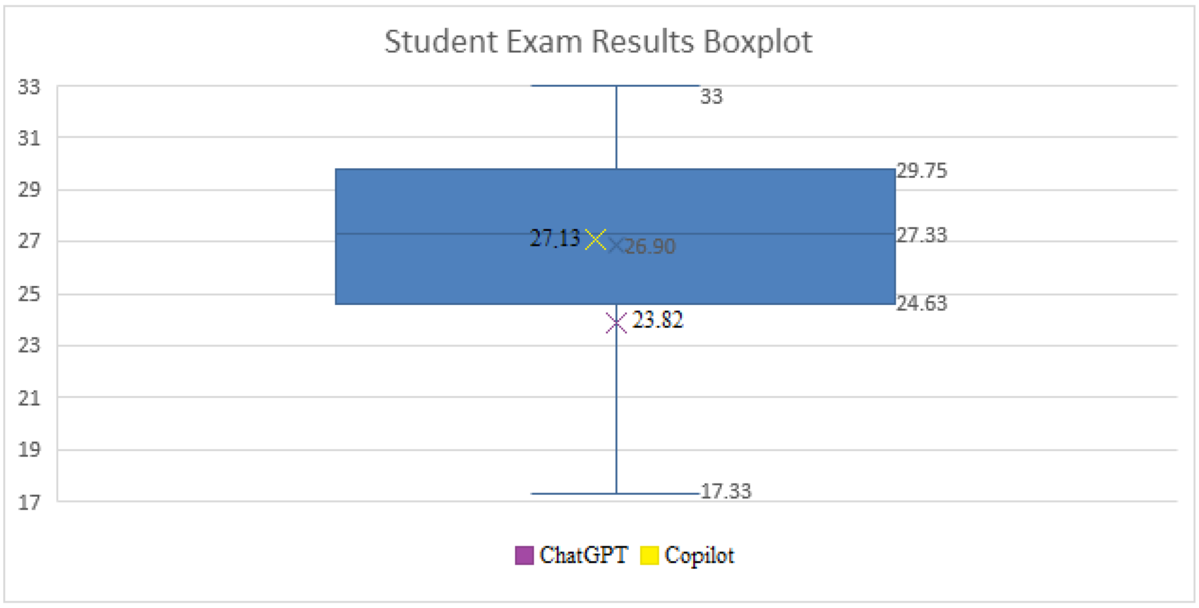


Figure 5. Students' results in exams

Even though Copilot had a higher average than the students, it places below the median indicating that more than half the students are more capable than it. ChatGPT performed even worse, with its average being below the bottom quartile indicating that 75% of students are more capable than it. This overall indicates that half of the students know the topics better than the AI chatbots.

4.2 Student Questionnaire

The first question of the questionnaire focused on whether the students had used AI assistants for this course. The results of the question can be seen in Figure 6.

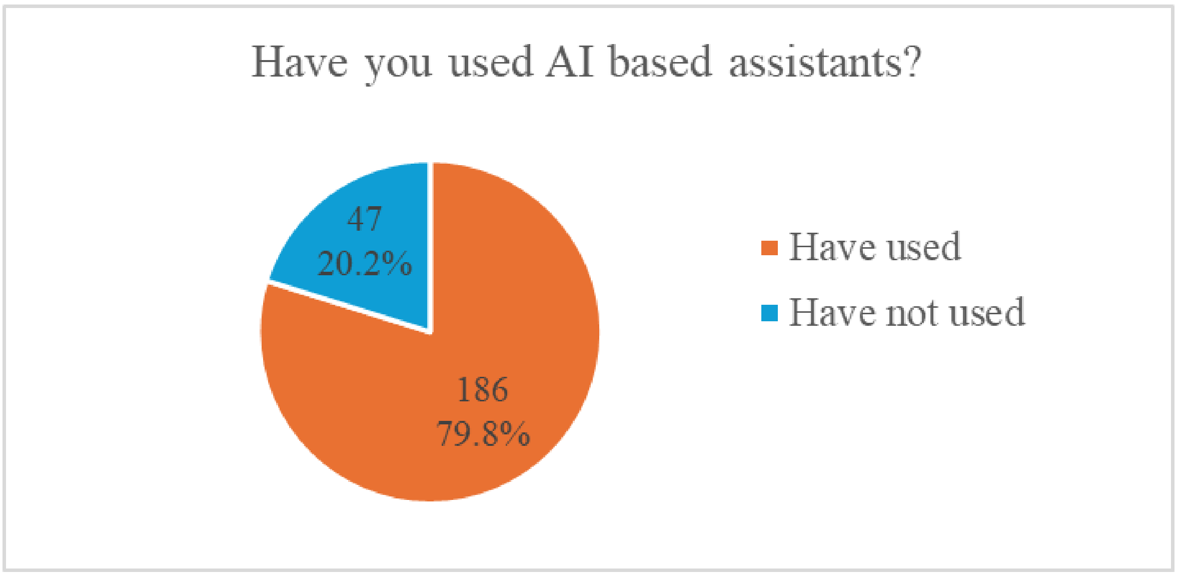


Figure 6. Number of students that have used AI assistants

The results show that 79.8% of the students have used AI-based assistants at least once for this course.

4.2.1 Non-users

The non-users were shown a follow up question to get to know why they have refrained from using AI assistants. It was a multiple-choice question where students could choose different reasons for their lack of use. The answers to the question can be seen in Figure 7.

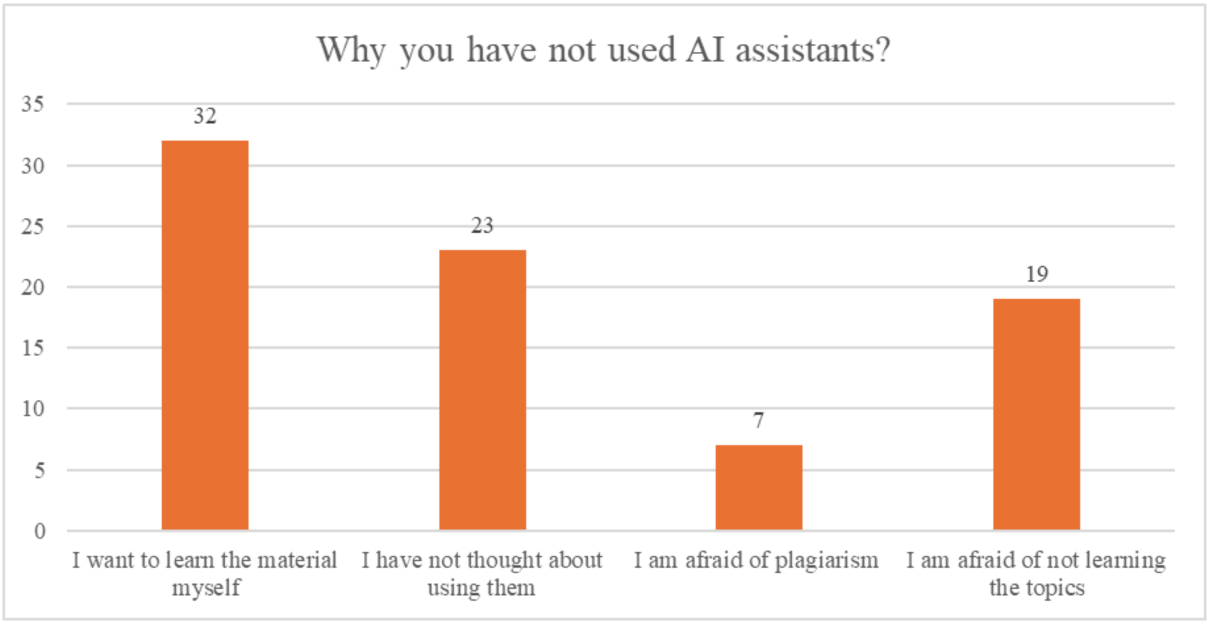


Figure 7. The reasons students have not used AI assistants

Clearly, the most chosen answer is “I want to learn the material myself” with 32 selections, and the least chosen is “I am afraid of plagiarism” with 7.

The students also had an open-ended question where they could give additional reasons why they had not used them. A common reason given was that googling is quicker than using AI assistants and that Java documentation or friends were a more trustworthy helper. One person wrote how they had used AI assistants in a previous course and felt they did not obtain the material and had to relearn it for the exam.

4.2.2 Users

The students who have used AI assistants were shown another set of questions. The first follow-up question was regarding their frequency of use, which is shown in Figure 8.

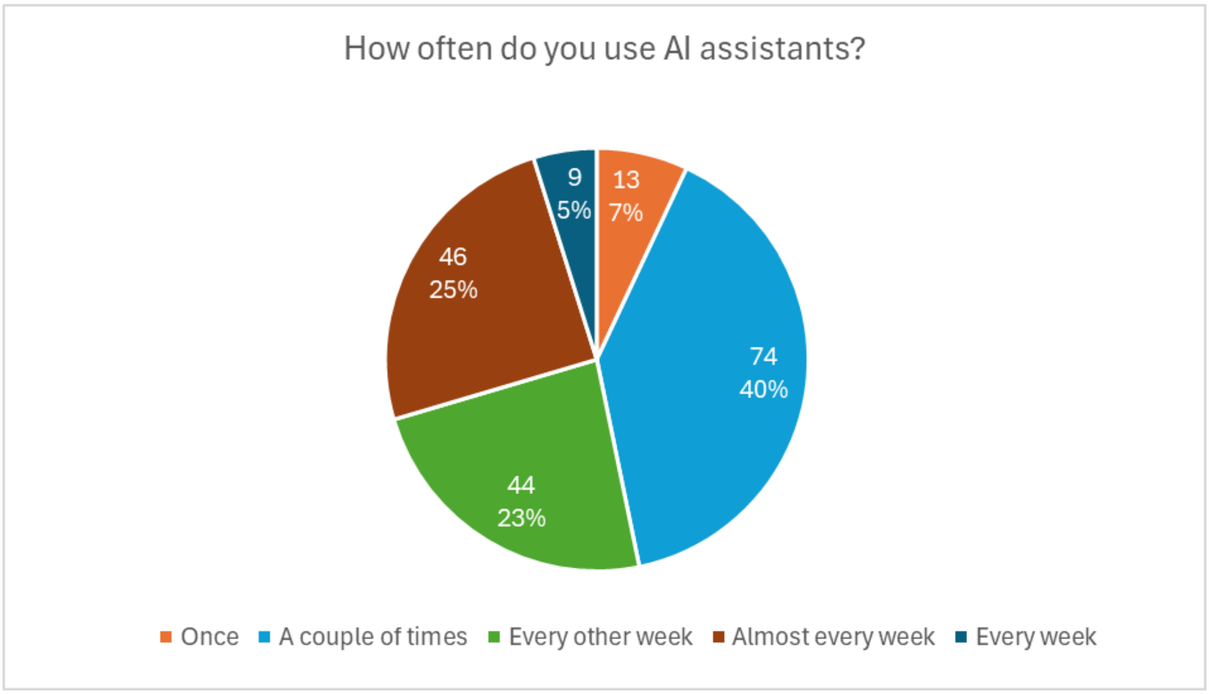


Figure 8. The frequency of using AI assistants

Interestingly the usage frequency was divided almost in half with 47% of the students having used it only once or a couple of times and 53% using it every other week or more. Still, only 5% of the students use it every week indicating that currently, the number of students that rely a lot on different AI assistants is small. When taking into account the students who do not use AI assistants, the numbers drop down to 3.9%.

Figure 9 shows how much the students felt the AI assistants helped them when using them. Overall the feeling was of them helping, with only one student choosing 1 which stands for no help. The most popular answer was 4, a step below always helping. This could indicate that they ran into some problems with AI giving them non-helpful answers or it could come down to not knowing how to ask for their specific problem. Still, only 33 students which is about 18% picked options 1 or 2 which seems to show that the majority of them found AI assistants at least somewhat helpful.

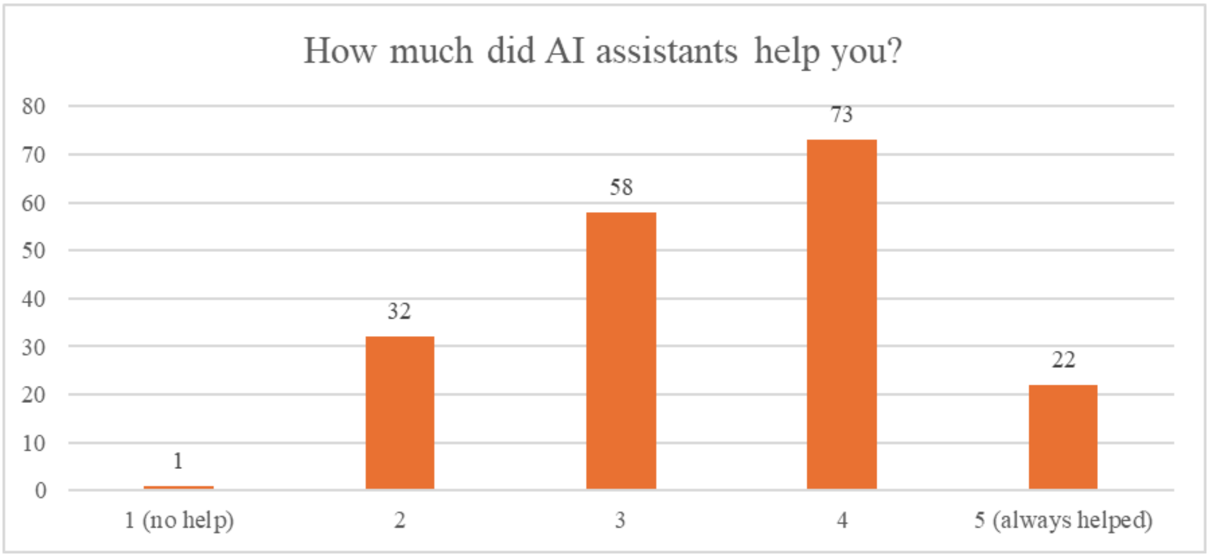


Figure 9. Help received from AI assistants

Figures 10 and 11 contain information about which tasks and how students used AI assistants. Here they could mark multiple answers or just one. Clearly, the most prevalent use for them was when solving homework tasks. As the homework tasks are presented together with the materials and relevant code snippets, the two most popular choices in Figure 10 being solving homework tasks and understanding code examples with 129 choices and 99 choices clearly show that this is where it is most used. This also correlates with Figure 11, where the most popular way of using AI assistants was for finding mistakes in their code and explaining existing code snippets with 156 and 111 students choosing them respectively. Still, students used them during the group assignment and preparing for the test as well with those choices being chosen by 73 and 75 students respectively. Students used AI assistants the least for answering the lecture quizzes with only 14 students choosing that answer. This seems in correlation with the fact that answering theoretical questions was one of the least popular ways to use them with only 37 students choosing it.

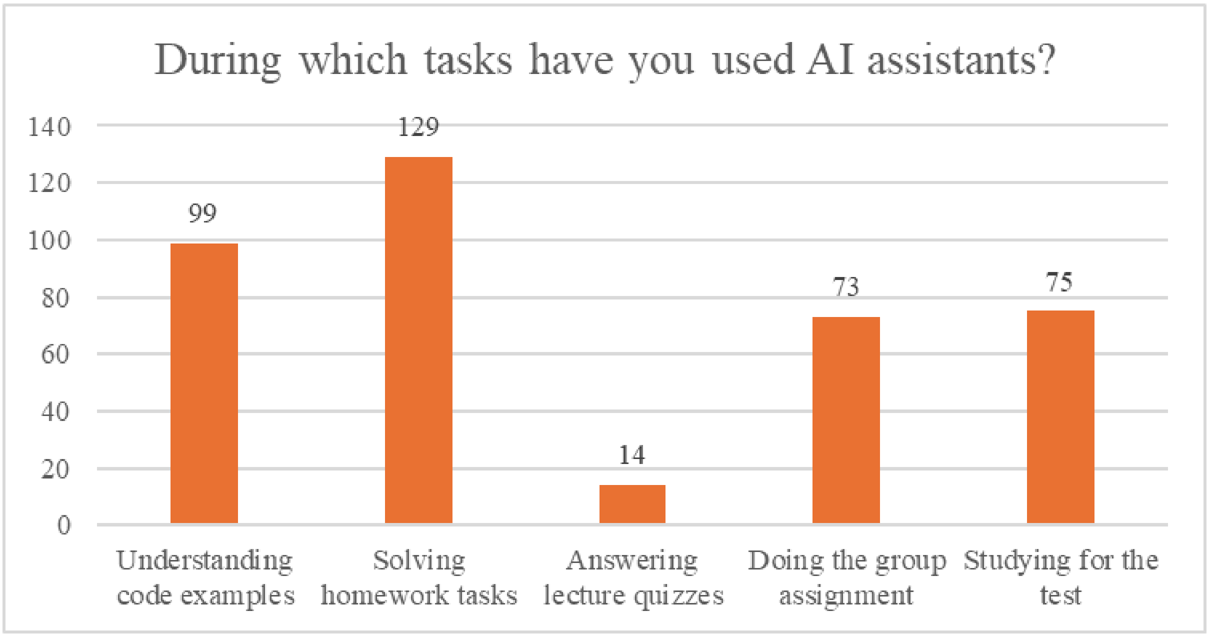


Figure 10. During which tasks students used AI assistants

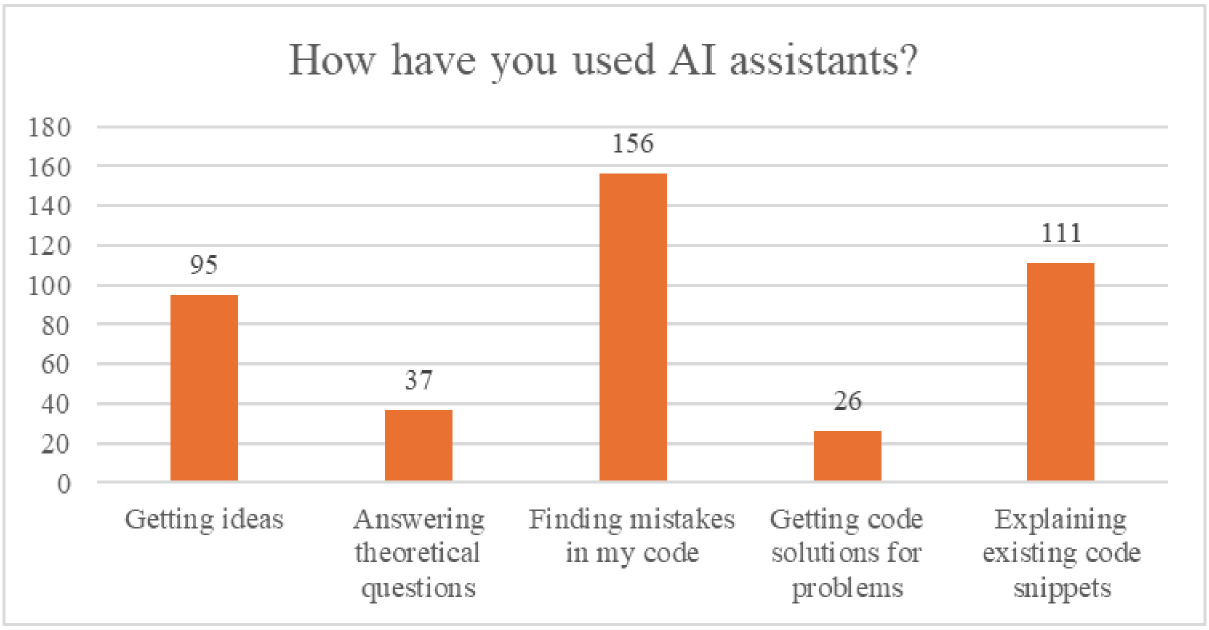


Figure 11. The ways students used AI assistants

Students also had an optional open-ended question where they could mention during which tasks and in which ways they have used AI assistants in this course. The majority of the answers repeated the answers given in previous questions. However, some novel uses were for generating data for either group work assignments or for testing their homework solutions. Another mentioned usage was decomposing a task to understand better where to begin solving it. Maybe the most interesting use was translating Python code to Java code as at the beginning of the course the student had experience with Python but not with Java, indicating an interesting possibility of learning a new programming language using previous experiences in a different programming language.

This section was followed by an open-ended question about what students like about AI assistants. The most prevalent answer was regarding their speed: they instantly replied and lessened the wait that would be asking a question from a teaching assistant and being always available is comfortable. They also mentioned that asking an AI assistant was faster than googling and got a more specific answer. Students also mentioned that it gives good and easy explanations with the possibility of re-asking for another wording if the given explanation is confusing. Another student noted the fact that AI is capable of giving feedback in Estonian is helpful. Multiple students also mentioned the fact that AI assistants are capable of finding mistakes in their code quickly which helps them to discover small errors or typos more quickly.

Students were also asked an open-ended question where they could write about their dislikes regarding AI assistants. The two most prevalent answers were about their mistakes or misunderstandings. The students felt that in some cases the AI assistant made mistakes and they had to ask multiple times to fix the AI answer which ended up taking up more time. Another thing they disliked was the fact that AI did not understand their question or prompt so it answered off-topic. Also when given code snippets the AI often changed it even when only asked to comment on it. Some students mentioned the fact that using them was too easy and comfortable which made them too susceptible to using it instead of trying different solutions themselves. One person mentioned that they were slow and another lamented that the better performing ones are paid. Some students mentioned the fact that the AI-proposed solution was too complex and contained topics and materials that were not covered in the course which made it confusing to use and understand them. One person mentioned a social stigma related to using AI assistants feeling that people perceive them badly for using them. Interestingly, here nobody mentioned the plagiarism and academic integrity aspect.

The final part of the questionnaire consisted of 5 statements where students had to mark whether they agreed with the statements on a Likert scale where 1 indicated full disagreement, 5 indicated full agreement and 3 indicated neutrality. The results of the first two statements can be seen in Figure 12.

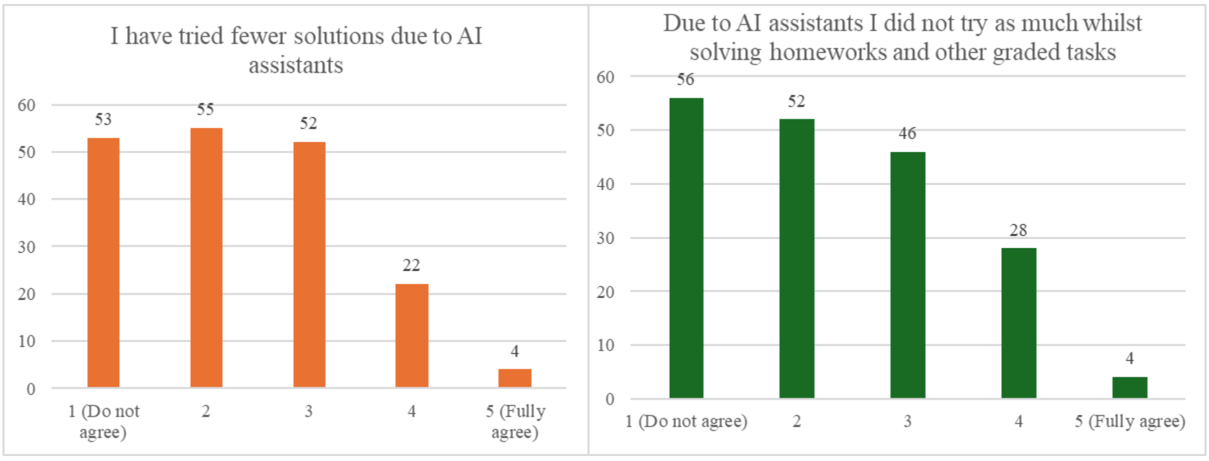


Figure 12. Impact of AI assistants on trying different solutions

The first two statements focused on the impact of AI assistants on the students' experiments with different solutions for tasks. We wanted to see how students agreed with the statements "I have tried fewer solutions due to AI assistants" (statement 1) and "Due to AI assistants I did not try as much whilst solving homeworks and other graded tasks" (statement 2). The majority of students disagreed with both of these statements as for both of these statements 108 students picked either options 1 or 2 which indicate disagreement which is about 58% of the surveyed students. For statement 1, 52 students were neutral with only 26 agreeing with it, making up 28% and 14% respectively. For statement 2 there were 46 neutral students which is about 25% and 32 students who agreed which is 17%.

The third statement was "I did not work through the course materials, but used an AI assistant" and answers regarding it can be seen in Figure 13.

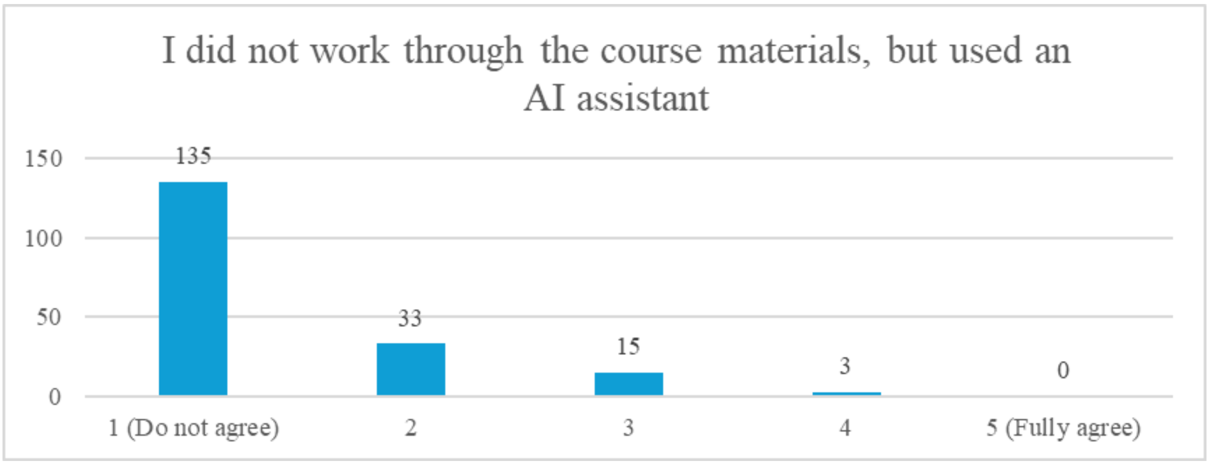


Figure 13. Impact of AI assistants on working through the course materials

The majority of the students strongly disagreed with this statement, with 135 students (73%) choosing option 1 with an additional 33 students (18%) choosing option 2, indicating that there is no problem with students not working through the study materials and relying on AI assistants to learn the materials. This is reinforced by the fact that no students picked the fully agree option and only 3 chose the somewhat agree option.

The fourth statement was "I asked for help less from teaching assistants due to AI assistants" and answers regarding it can be seen in Figure 14.

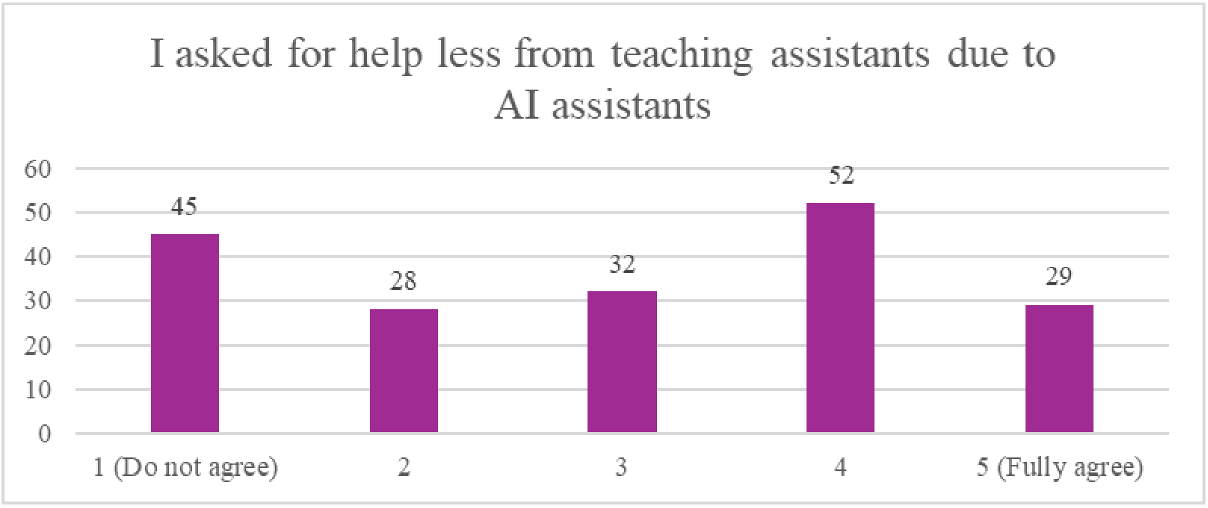


Figure 14. Impact of AI assistants on asking for help from TA

This statement had quite an even distribution of answers, with the most popular choice being 4 with 52 students (28%), followed by 1 with 45 students (24%), with answers 2, 3, and 5 being chosen by 28 students (15%), 32 students (17%) and 29 students (16%). This seems to indicate that there was no clear majority.

The fifth and final statement was "The existence of AI assistants motivated me to solve more homework tasks" and answers to it can be seen in Figure 15.

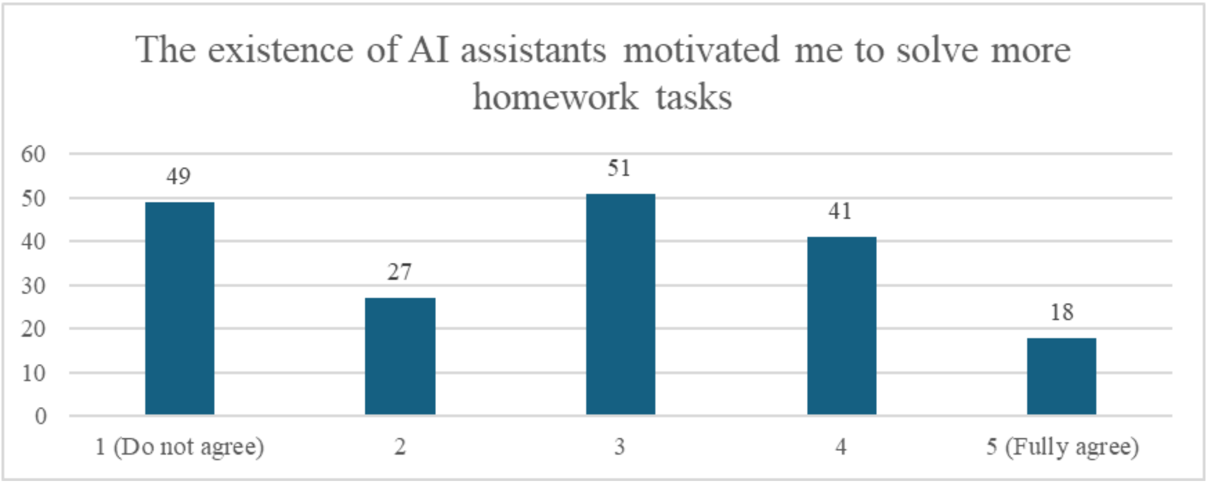


Figure 15. Impact of AI assistants on motivation to solve homework

This statement had quite an even distribution of answers as well, with the most popular answer being the neutral 3 with 51 students (27%) followed by the full disagreement with 49 students (26%) and those who somewhat agree with 41 students choosing (22%).

4.2.3 Usage of AI and Its Correlation With Grades

To analyse the impact of AI assistant usage on grades, we combined the answers to the questions of whether the students used AI assistants and how often they used them to their results in test 1. The results of this can be seen in Table 12.

Table 12. Usage of AI assistant and statistics about students' grades in test 1

AI usage	Never	Once	Couple times	Every other week	Almost every week	Every week
Student Average	15.29	15.42	13.83	13.47	12.75	14.73
Amount of Students	46	13	72	40	45	8
Standard Deviation	0.96	0.88	3.86	3.56	3.87	0.98
Standard Error	0.14	0.24	0.45	0.56	0.58	0.34
Group	I			II		
Group Average	14.5			13.23		
Amount of Students	131 (58.5%)			93 (41.5%)		
Group Standard Deviation	3.01			3.60		

As can be seen from Table 12, the more students used AI assistants during the course, the lower their score for test 1 was. The sole exception was those who used AI assistants every week with them having a higher average than those who used them a couple of times or more. To see whether there was a statistical difference between those who used AI assistants rarely and those who used them regularly, the students were divided into two groups based on their AI usage with one group consisting of students who either never used AI assistants, only used them once or a couple of times. The second group consists of students who use them biweekly or more often. To confirm whether the two group results follow a normal distribution the Shapiro-Wilk test was used. Neither of the groups followed a normal distribution as the results were for group

1 (W = 0.494, $p < 0.0001$) and for group 2 (W = 0.726, $p < 0.0001$). To see if this difference was statistically significant, the Mann-Whitney U-test was used to compare the averages of the two groups. The results were statistically significant (U = 8083.0, $p < 0.0001$), meaning that those who used AI assistants less performed better than those who used them regularly.

5 Discussion

The main goal of this thesis was to analyse the proficiency of different AI chatbots in an introductory object-oriented programming course, to compare their results with students taking the course and to gather students' perceptions and usage regarding AI assistants. This section analyses the results to answer the stated research questions.

5.1 AI Comparison to Students

One of the goals was to analyse how different chatbots perform in the course “Object-Oriented Programming” in comparison to students. To answer this question ChatGPT and Copilot were given the full text of the tests and tasks and the output was graded. The texts were given in Estonian to additionally see whether the AI chatbots are capable of understanding it.

When looking at test 1, then both the AI chatbots outperformed the student average. However, this was due to the fact that non-compiling solutions resulted in an automatic zero which brought the student average down. When looking at quartiles, then ChatGPT was always below the bottom quartile, indicating that 75% of the students are better than it. Copilot fared better with it scoring higher than the median student on all three versions of the test and on one test it scored in the upper quartile. These results seem in line with Bordt and Luxburg's [13] findings who found that ChatGPT-3.5 was able to pass a course whilst performing worse than the students whilst ChatGPT-4 performed on par with students. As Copilot is based on GPT-4 while ChatGPT-3.5 was used for ChatGPT analysis then the results indicate a similarity. These findings do seem to differ from other studies [5-7] that found that different AI tools (ChatGPT, Github Codex) placed above the upper quartile in introductory computer science courses when compared to students. It is unclear what could be the cause, one possible reason is the fact that the tasks were in English when the different AI tools placed above the upper quartile whilst here the tasks were in Estonian. This would be in line with research conducted on ChatGPT capability in Czech information security courses [23] where students outperformed ChatGPT in three of the evaluated courses whilst AI outperformed students in one of the courses, indicating a variance similar to what was present in this course regarding Copilot and ChatGPT.

When looking at test 2, the AI results differed when looking at the student average with ChatGPT scoring worse and Copilot scoring better. Looking at AI placements regarding students, ChatGPT was in the bottom 25% for two of the test variants and in the bottom half for the third test variant. Copilot performed better with it scoring higher than the median for two of the variants and being below the median score for one of the variants. Overall, these results seem to continue the trend seen in test 1 where ChatGPT-3.5 is passing the test but scoring worse than the students whilst Copilot is performing about on par with them. Still, it is noticeable that the AI chatbots performed worse comparatively than in test 1 when looking at their placements compared to students. These findings are in line with some of the previous research [6-7] that found that AI assistants are more capable in introductory topics, but perform worse when more complexity is introduced. This however contradicts other research that found that there was no difference in AI chatbot performance when comparing introductory and intermediate results [27].

Exam results confirm the previous findings with ChatGPT performing worse than the student average while Copilot outperformed it. However, when looking at the median score then both of the AI assistants performed worse than the median student with ChatGPT being below the lower quartile. However, the difference in averages is somewhat the result of the fact that AI assistants were incapable of parsing one question as the data was given as a picture which resulted in an automatic loss of 2 points. These input parsing problems are similar to problems with interpreting UML diagrams [30-31].

An additional interesting facet is AI performing comparatively better on tests than in the exam. The test tasks consist of creating a program with multiple classes and functionalities based on a long textual description whilst the exam questions are shorter with having to decide on smaller code snippets. It would seem more probable that longer texts contain more possible mistakes, but here the AI assistants struggled relatively more with shorter questions. These findings are somewhat similar to previous research [23] which found that students performed better at completing small code snippets than AI chatbots, but AI proficiency in creating larger programmes based on longer textual descriptions is interesting.

All in all, ChatGPT and Copilot are capable of passing a course in an introductory object-oriented programming course in Estonian. However, ChatGPT consistently lagged behind the student average, especially evident in its placing below the lower quartile multiple times in the tests and in the exam. Copilot exhibited a more competitive performance, often surpassing the median and once the upper quartile. The chatbots performed better at introductory topics with more variance when more complex subjects were introduced. These findings are similar to previous research findings which have found that depending on the AI used the results differ and ChatGPT is often passing the courses, but not surpassing the average students with the performance dependent on the complexity of the topic. Interestingly, AI assistants are more capable of solving larger programming tasks based on a long textual description than shorter exam tasks with shorter descriptions.

5.2 Common AI Mistakes

The second research question focused on the common mistakes that AI chatbots make whilst solving the tasks of the course. When grading the solutions provided by ChatGPT and Copilot the mistakes were written down to see whether there are repeating mistakes.

When looking at test 1, the most prevalent mistake was not specifying the encoding of the files. This mistake was present in all ChatGPT solutions and once in Copilot's solutions. This requirement was mentioned in the text as a fact that files are in a specific encoding with the assumption being that reading data from files uses that encoding. Additionally, Copilot displayed null values in a *toString* method once and ChatGPT had problems reading data from a file and used wrong method names once. Another repeating mistake for ChatGPT was not making a superclass method abstract and defining subclass method logic in it. It worked correctly in the context of the program but went against the provided specification.

Test 2 had more mistakes, with both ChatGPT and Copilot not making some methods private. However, this requirement was written in Estonian as the methods should not be callable outside of the class, which could be a reason for the AI assistants not understanding it correctly as they did not make such mistakes in test 1. An additional repeating mistake was sorting in the wrong direction. However, this was specified in Estonian as "non-decreasing" which could create confusion. Copilot had problems with generating *get* and *set* methods, whilst ChatGPT had many problems when the test task required it to use Queue data structures and confused the way the task was supposed to work.

Interestingly enough, whilst there were similar mistakes for ChatGPT and Copilot for the same tests, there were no repeating mistakes when comparing tests 1 and 2. This could stem from the tests focusing on different topics and reducing the possibility of similar mistakes happening. When looking at previous research on object-oriented programming [32], a common problem was the prevalence of compilation errors and needing multiple prompts to generate all the necessary functions. Whilst ChatGPT and Copilot did not have compilation problems, there was the problem of not generating some *get* and *set* methods.

The exam covered all of the topics of the course, meaning that the AI assistants had to solve the tasks on a variety of topics. A common mistake that was not related to topics, but to the presentation of the question was the fact that AI did not choose its answer from the multiple choices given. Often the AI assistants chose or gave one of the correct options, but did not list all of them. Both ChatGPT and Copilot had problems with String comparison, making mistakes based on case and other String methods. ChatGPT specifically had problems with comprehending the values saved in variables. Additionally, ChatGPT had problems with lists as it also made mistakes regarding them being reference-based, with their indexes and sort methods.

Interfaces, abstract classes and class hierarchy were topics that were all intertwined in the exam as they have similarities and differences. Abstract methods and their implementation in subclasses was a topic in which ChatGPT made mistakes in the tests and both AI chatbots had problems regarding them in the exam as well, indicating overall that they fully do not comprehend it. An additional problem was failing to understand how method implementations are searched starting from subclasses and moving upwards with additional problems failing to realise how superclass constructors are called in subclasses. ChatGPT and Copilot also had problems when to use the keywords *abstract*, *extends* and *implements* and which access keywords can be used in interfaces and abstract classes. Collectively, these mistakes illustrate that when having to deal with less common problems and more edge-cases then AI assistants make more mistakes.

The graphics questions had the problem of the AI assistants being unable to parse image data, similar to the problems described in previous research regarding UML diagrams [30-31]. On the topic of events, the AI assistants usually understood event logic correctly but made previously described mistakes regarding String comparison and logical comparison, with them even confusing variable values as mentioned previously. Regarding data streams, both ChatGPT and Copilot had problems with how methods *readInt* and *writeUTF* interact with each other with additional problems of comprehending input file size. Both AI assistants made similar mistakes in exception handling where they just did not take into consideration some *print* statements, which is not a problem with exception handling. However, both of the AIs made the mistake of assuming that an exception thrown in a catch block would be caught. Data structures were another topic where the AI assistants made mistakes. The stack data structure was the only topic in which the AI assistants always made mistakes. ChatGPT specifically had problems with Queues, which was similar to problems seen in test 2, which seems to indicate a problem distinct to it.

The exam ended with a longer question which necessitated knowledge regarding all of the topics covered in the course and required giving explanations for the answers. The mistakes made here were similar to those made in previous questions, with the AI having problems with how interfaces, abstract classes and class hierarchy interact with each other and when to add access modifiers. Additionally, here the AI ran into the problem of only proposing one possible solution, not all of them as was required in the task. However, when their answers were correct, then their given explanations were also sufficient. This seems to align with previous research [23], which found that AI was more capable than students in giving explanations for questions.

All in all, both ChatGPT and Copilot made mistakes whilst solving the tasks. One common problem was not comprehending what was asked of them, with not choosing the answer from the given answer list. An additional problem was regarding answer generation, when asked for all possible solutions they often just gave one correct answer, not all possible ones. Many times it seemed that AI was not as capable of parsing hidden intent in tasks, as it failed to specify encoding or use keyword private for methods that need not be accessed outside of their class. Some of those mistakes could stem from the tasks being in Estonian, as some of the sorting direction mistakes could come from misunderstanding the text. Similarly to previous research, there was a problem of not just generating all of the methods that were mentioned in the text. When looking at specific topics, there were many mistakes made regarding abstract classes, interfaces and class hierarchy. An additional common problem was the stack data structure and ChatGPT specifically had problems with queues. Still, both of the AI assistants were in general quite capable of solving the different tasks, giving explanations and proposing at least one correct solution.

5.3 Students Usage and Perception Regarding AI Assistants

The third research question focused on how much and in what ways students use various artificial intelligence-assisted methods during the course. Data relating to this question was gathered with a questionnaire that the students taking this course could answer for an additional point.

79.8% of the surveyed students had used them, which is similar to a survey conducted in Japan [46], which found that 78.8% of the students surveyed had used ChatGPT for programming exercises. This seems to indicate that about 4/5 students try using AI assistants for programming.

The most popular answer among non-users when inquired about reasons for not using AI assistants was "I want to learn the material myself" with 68% of them choosing it. This overall seems to align with a study conducted in Turkey [38] which found that students perceived the biggest negative to using AI assistants to be programmer laziness. Hence it makes sense that the biggest reason not to use them would be the desire to learn the materials themselves. The least popular answer being “I am afraid of plagiarism” with only 15% of the non-users picking it indicates that students are not afraid of plagiarism or they might not perceive using AI assistants as plagiarism. In the open feedback section, a common reason given for not using AI assistants was that googling information was actually quicker, which indicates that previous bad experiences have limited students’ willingness to use AI. This aligns with previous findings [36-37] which found that positive experiences and good performance help promote the usage of AI, meaning inversely that negative experiences inhibit the adoption of AI chatbot usage into the workflow.

The students who had used AI assistants were asked about the frequency of use. The students were roughly divided in half with 47% of them having used AI assistants only a couple of times and 53% of them using them every other week or more often. However the number of students using AI assistants every week is small (3.9%) when taking into account all users and non-users. This is good as it reduces the likelihood of students becoming solely dependent on them for programming and still personally learning the material. Overall, the users found the AI assistants to generally be helpful as about 72% of the students chose a positive rating when having to describe how much the AI assistant helped them.

When looking at when and how students used AI assistants, then the most popular answers were related to coding. Students used AI assistants the most to solve homework tasks and to understand code examples and the most popular use case was for finding mistakes in code. These findings are similar to Sun et al. [12] who found that students mostly used AI assistants for debugging and coding and less for theory and task decomposition. Additionally, this aligns with previous research [33-34], that proposed the usage of AI chatbots for getting explanations about existing code snippets. A surprise was the fact that not many students used AI assistants for generating example solutions as it was the least popular choice, which might indicate that students are not comfortable with generating solutions with AI chatbots as they might feel it veers into plagiarism or that they might not learn the material. Some interesting uses for AI were mentioned in the open-ended question, where AI chatbots were used to generate test data for tasks and AI chatbots were used to translate code from Python to Java indicating future research possibilities.

When asked about what students liked about AI, the most common answers were related to their speed and availability, understanding of Estonian and the possibility of repeating questions. An interesting contrast was that some who had used it felt that using AI chatbots was quicker than googling, which directly contradicts the feelings of some of the non-users. These answers are mostly in line with previous research [33-34, 38], which also mentioned speed, availability and AI's commenting skill as the main positives perceived by students. The most often mentioned negatives were AI making mistakes or misunderstanding the input, with AI sometimes answering off-topic or changing the code snippet which was given to them. Additionally, occupational anxiety was mentioned, with feelings of AI limiting the amount of different solutions tried by students. These reasons align with previous research [38], which also found that mistakes and occupational anxiety were some of the biggest perceived negatives. What was interesting, was that nobody mentioned the plagiarism and academic integrity aspect, which seems to continue to indicate that students are not thinking about that aspect.

Whilst some mentioned trying fewer solutions as a negative about using AI chatbots, generally the students disagreed with the sentiment that AI chatbots made them try less different solutions to different tasks. The majority disagreed with statements "I have tried fewer solutions due to AI assistants" and "Due to AI assistants I did not try as much whilst solving homeworks and other graded tasks" as only 14% and 17% agreed or somewhat agreed with the respective statements. Overall this seems to indicate that the average student was not negatively affected by the possibility of using AI assistants and still tried different solutions to learn and study the course material. This is reinforced by the fact that the majority of students (90%) disagree with the statement "I did not work through the course materials, but used an AI assistant" and only 1.6% somewhat agree with this statement.

When looking at how the existence of AI impacts students' interactions with lecturers and teaching assistants, there was no clear majority when indicating agreement or disagreement with the statement "I asked for help less from teaching assistants due to AI assistants". However, as there were 81 students (44%) that agreed with the statement at least somewhat, it indicates a possibility of easing teaching assistant and lecturer workload by propagating the usage of AI chatbots for answering questions initially and following up with the teaching assistants and lecturers when it is necessary. Overall, this shows the need for teaching assistants and lecturers still as there are students who feel that AI chatbots are unable to replace humans.

Some previous research [15] has found that introducing a supplementary chatbot to a course increased students' motivation to study and engage with the material. However, when the students taking this specific course had to indicate their agreement with the statement "The existence of AI assistants motivated me to solve more homework tasks", the most popular answer was the neutral one. This seems to indicate that the existence of AI assistants does not really have that strong of an impact on student motivation. However, the difference in results could stem from the fact that this course had no specialised AI assistant whilst the previous research was focused on a chatbot trained to answer questions regarding the specific course. Still, those who agreed might have gotten help for solving their homework tasks from AI chatbots indicating they could help to motivate some students, but clearly not all.

Research regarding how AI chatbot usage impacts students' skills is limited and contradictory. Some [11] have found that students who used AI chatbots improved their computational thinking skills and programming self-efficacy more. However, others [12] have found no differences between users and non-users. In this course, those who had used AI assistants rarely or never had a higher average score in test 1 than those who had used them regularly with the difference being statistically significant. However, it is unclear whether this stems from the fact that using chatbots inhibited learning or that weaker students use supplementary study materials more often, as in a similar course it was found that students with a lower grade used supplementary materials more often [47].

All in all, the majority of students have used AI assistants, primarily for coding-related tasks such as debugging and understanding code examples. However, only a tiny minority use AI assistants every week, indicating that currently, the students have not developed a dependency on them. While some expressed concerns about chatbots potentially limiting their exploration of different solutions, overall it seems that AI has not reduced the amount of various solutions the students try and helps them when studying the materials.

Conclusions

The aim of this master's thesis was to understand how proficient different AI-based chatbots are in solving tasks in the course “Object-Oriented Programming”, what are the common mistakes and how students use these tools. To achieve this, ChatGPT and Copilot were made to solve tests and exams and their results were compared to students’ results. Additionally, data regarding students’ usage of AI chatbots was gathered with a questionnaire with a focus on frequency of use, ways of use and impact on learning.

To gauge the proficiency of ChatGPT and Copilot, they were given task descriptions of the multiple variants of the tests and multiple exam questions. Their answers were evaluated in accordance with the grading guide and their scores were compared to students’ scores. ChatGPT often performed worse than the median score and multiple times it placed below the bottom quartile. Copilot fared better, with it performing about on par with the median students. The AI chatbots performed better in introductory topics with more variance when more difficult themes were introduced. Additionally, they performed better in the longer test tasks than the shorter exam tasks.

One of the common mistakes of the AI chatbots was not giving all of the possible solutions when asked. Additionally, they struggled with understanding requirements that were written more as an implication than a clear statement. When looking at topics, both of the chatbots made mistakes when interfaces, abstraction and class inheritance were intertwined. Additionally, they were incapable of solving tasks that had some of the information given as a picture. The only topic in which the chatbots were always wrong was the data structure Stack. Still, both of the AI assistants were quite capable of solving the different tasks, giving explanations and proposing at least one correct solution.

About 80% of the students had used an AI chatbot for solving tasks related to the course, but only a small minority (3.9%) used them every week. The students mainly used them for tasks related to coding and valued the AI assistants' speed and availability and disliked the mistakes AI made and how AI misunderstood their input. It was shown that students who used AI assistants more frequently scored lower on test 1 compared to the students who used AI assistants less. Still, the students did not feel that using AI tools made them not learn the material or restricted their exploration of various solutions.

This thesis gave an overview of ChatGPT and Copilot proficiency in an introductory object-oriented course and compared their results to students. Additionally, students' perceptions of

AI chatbots were gathered. These findings are valuable to the conductors of this course and other similar courses as they give an insight into how these AI chatbots compare to students, what common mistakes these AI assistants make and how students perceive, use and feel about them to help make changes to computer science courses and education.

A limitation of this thesis was the fact that it focused only on one course and one year due to the timescale of AI chatbots becoming prevalent. This makes future research on more courses and more years an interesting prospect. This thesis mainly focused on AI proficiency and students' usage of them, but the AI tools are also available to lecturers and how to combine teaching and grading automation with AI assistants is another way to approach the topic, which could lead to some interesting discoveries. As more AI assistants are developed and existing ones are continuously updated, additional research on the changes in proficiency in relation to updates and upgrades is another interesting research avenue.

References

[1] Introducing ChatGPT. https://openai.com/blog/chatgpt (04.04.2024)

[2] Hu, K. ChatGPT Sets Record for Fastest-Growing User Base - analyst note. 2023. https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/ (04.04.2024)

[3] Vogels, E. A. A Majority of Americans Have Heard of ChatGPT, but Few Have Tried It Themselves. 2023. https://www.pewresearch.org/short-reads/2023/05/24/a-majority-of-americans-have-heard-of-chatgpt-but-few-have-tried-it-themselves/ (04.04.2024)

[4] Denny, P., Prather, J., Becker, B., A., Finnie-Ansley, J., Hellas, A., Leinonen J., Luxton-Reilly, A., Reeves, B., N., Santos, E., A., Sarsa, S. Computing Education in the Era of Generative AI. *Communications of the ACM*, Vol. 67, No. 2, pp. 56–67, 2024. https://doi.org/10.1145/3624720

[5] Finnie-Ansley, J., Denny, P., Becker, B., Luxton-Reilly, A., Prather, J. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. *In Proceedings of the 24th Australasian Computing Education Conference*, pp. 10–19, 2022. https://doi.org/10.1145/3511861.3511863

[6] Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E., Prather, J., Becker, B. My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. *In Proceedings of the 25th Australasian Computing Education Conference*, pp. 97–104, 2023. https://doi.org/10.1145/3576123.3576134

[7] Richards, M., Waugh, K., Slaymaker, M., Petre, M., Woodthorpe, J., Gooch, D. Bob or Bot: Exploring ChatGPT's Answers to University Computer Science Assessment. *ACM Transactions on Computing Education*, Vol. 24, No. 5, pp. 1–32, 2024. https://doi.org/10.1145/3633287

[8] Denny, P., Khosravi, H., Hellas, A., Leinonen, J., Sarsa, S. Can We Trust AI-Generated Educational Content? Comparative Analysis of Human and AI-Generated Learning Resources, 2023. https://doi.org/10.48550/arXiv.2306.10509

[9] Kiesler, N., Lohr, D., Keuning, H. Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. *In 2023 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5, 2023. https://doi.org/10.48550/arXiv.2309.00029

[10] Jukiewicz, M. The Future of Grading Programming Assignments in Education: The Role of ChatGPT in Automating the Assessment and Feedback Process. *Thinking Skills and Creativity*, Vol. 52, 2024. https://doi.org/10.1016/j.tsc.2024.101522

[11] Yilmaz, R., Yilmaz, F., G., K. The Effect of Generative Artificial Intelligence (AI)-based Tool Use on Students' Computational Thinking Skills, Programming Self-efficacy and Motivation. *Computers and Education: Artificial Intelligence*, Vol. 4, 2023. https://doi.org/10.1016/j.caeai.2023.100147

[12] Sun, D., Boudouaia, A., Zhu, C., Li, Y. Would ChatGPT-facilitated Programming Mode Impact College Students' Programming Behaviors, Performances, and Perceptions? An Empirical Study. *International Journal of Educational Technology in Higher Education*, Vol. 21, No. 14, 2024. https://doi.org/10.1186/s41239-024-00446-5

[13] Bordt, S., Luxburg, U. ChatGPT Participates in a Computer Science Exam. 2023. https://doi.org/10.48550/arXiv.2303.09461

[14] Shoufan, A. Can Students without Prior Knowledge Use ChatGPT to Answer Test Questions? An Empirical Study. *ACM Transactions on Computing Education*, Vol. 23, No. 45, pp. 1–29, 2023. https://doi.org/10.1145/3628162

[15] Ting-Ting Wu, T-T., Li, P.H., Huang, C.-N., Huang., Y.-M. Promoting Self-Regulation Progress and Knowledge Construction in Blended Learning via ChatGPT-Based Learning Aid. *Journal of Educational Computing Research*, Vol. 61, No. 8, pp. 3–31, 2024. https://doi.org/10.1177/07356331231191125

[16] Yenduri, G., Murugan, R., Govardanan, C., S, Supriya, Y, Srivastava, G., Maddikunta, P., K., R., Deepti Raj, G., Jhaveri, R., H., Prabadevi, B., Wang, W., Vasilakos, A., Gadekallu, T., R. Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. *IEEE Access*, Vol. 12, pp. 54608–54649, 2023. https://doi.org/10.48550/arXiv.2305.10435

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I. Attention Is All You Need. *In Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010, 2017. https://doi.org/10.48550/arXiv.1706.03762

[18] Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q., L., Tang, Y. A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. *IEEE/CAA Journal of*

Automatica Sinica, Vol. 10, No. 5, pp. 1122–1136, 2023. https://doi.org/10.1109/JAS.2023.123618

[19] Mehdi, Y. Reinventing search with a new AI-powered Microsoft Bing and Edge, your copilot for the web. 2023. https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-for-the-web/ (24.04.2024)

[20] Microsoft Copilot Pro. https://www.microsoft.com/en-us/store/b/copilotpro (24.04.2024)

[21] Chen, E., Huang, R., Chen, H.-S., Tseng, Y.-H., Li, L.-Y. GPTutor: A ChatGPT-Powered Programming Tool for Code Explanation. *Communications in Computer and Information Science*, Vol. 1831, pp. 321–327, 2023. https://doi.org/10.1007/978-3-031-36336-8_50

[22] Wang, F., H. Efficient Generation of Text Feedback in Object-oriented Programming Education Using Cached Performer Revision. *Machine Learning with Applications*, Vol. 13, No. 34, 2023. https://doi.org/10.1016/j.mlwa.2023.100481

[23] Malinka, K., Peresíni, M., Firc, A., Hujnák, O., Janus, F. On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree? *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education*, pp. 47–53, 2023. https://doi.org/10.1145/3587102.3588827

[24] Geerling, W., Mateer, G., D., Wooten, J., Damodaran, N. ChatGPT Has Mastered the Principles of Economics: Now What?, *SSRN Electronic Journal*, 2023. https://doi.org/10.2139/ssrn.4356034

[25] Bommarito, M., Katz, D., M. GPT Takes the Bar Exam. 2022. https://doi.org/10.48550/ARXIV.2212.14402

[26] Lee, H. The Rise of ChatGPT: Exploring its Potential in Medical Education. *Anatomical sciences education*, 2023. https://doi.org/10.1002/ase.2270

[27] Savelka, J., Agarwal, A., Bogart, C., Song, Y., Sakr, M. Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses?. *In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education*, pp. 117–123, 2023. https://doi.org/10.1145/3587102.3588792

[28] Joshi, I., Budhiraja, R,, Dev, H,, Kadia, J., Ataullah, M., Mitra, S., Akolekar, H., Kumar, D. ChatGPT in the Classroom: An Analysis of Its Strengths and Weaknesses for Solving Undergraduate Computer Science Questions. *In Proceedings of the 55th ACM Technical*

Symposium on Computer Science Education, pp. 625–631. 2024. https://doi.org/10.1145/3626252.3630803

[29] Kadir, M., Rahman, T., Barman, S., Al-Amin, M. Exploring the Competency of ChatGPT in Solving Competitive Programming Challenges. *International Journal of Advanced Trends in Computer Science and Engineering*, Vol 13, No. 1, pp. 13–23, 2024. https://doi.org/10.30534/ijatcse/2024/031312024

[30] Ouh, E., L., Gan, B., K., S., Jin Shim, K., Wlodkowski, S. ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an Undergraduate Java Programming Course. *In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education*, pp. 54–60, 2023. https://doi.org/10.1145/3587102.3588794

[31] Cámara, J., Troya, J., Burgueño, L., Vallecillo, A. On the Assessment of Generative AI in Modeling Tasks: An Experience Report with ChatGPT and UML. *Software and Systems Modeling (SoSyM)*, Vol 22, No. 3, pp. 781–793, 2023. https://doi.org/10.1007/s10270-023-01105-5

[32] Cipriano, B., P., Alves, P. LLMs Still Can't Avoid Instanceof: An Investigation Into GPT-3.5, GPT-4 and Bard's Capacity to Handle Object-Oriented Programming Assignments. 2024. https://doi.org/10.48550/arXiv.2403.06254

[33] Becker, B., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., Santos, E. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. *In Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, pp. 500–506, 2023. https://doi.org/10.1145/3545945.3569759

[34] Sarsa, S., Denny, P., Hellas, A., Leinonen, J. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. *In Proceedings of the 2022 ACM Conference on International Computing Education Research*, Vol. 1, pp. 27–43, 2022. https://doi.org/10.1145/3501385.3543957

[35] Jury, B., Lorusso, A., Leinonen, J., Denny, P., Luxton-Reilly, A. Evaluating LLM-generated Worked Examples in an Introductory Programming Course. *In Proceedings of the 26th Australasian Computing Education Conference*, pp. 77–86, 2024. https://doi.org/10.1145/3636243.3636252

[36] Strzelecki, A. To Use or Not to Use ChatGPT in Higher Education? A Study of Students' Acceptance and Use of Technology. *Interactive Learning Environments*, pp. 1–14, 2023. https://doi.org/10.1080/10494820.2023.2209881

[37] Lai, C., Y., Cheung, K., Y., Chan, C., S. Exploring the Role of Intrinsic Motivation in ChatGPT Adoption to Support Active Learning: An Extension of The Technology Acceptance Model. *Computers and Education: Artificial Intelligence*, Vol. 5, 2023. https://doi.org/10.1016/j.caeai.2023.100178

[38] Yilmaz, R., Yilmaz, F., G., K. Augmented Intelligence in Programming Learning: Examining Student Views on the Use of ChatGPT for Programming Learning. *Computers in Human Behavior: Artificial Humans*, Vol. 1, No. 2, 2023. https://doi.org/10.1016/j.chbah.2023.100005

[39] Gabbrielli, M., Martini, S. Object-Oriented Paradigm. In: Programming Languages: Principles and Paradigms. Undergraduate Topics in Computer Science. Second Edition. Cham: Springer Nature. 2023.

[40] TIOBE index. https://www.tiobe.com/tiobe-index/ (15.03.2024)

[41] Capretz, L. A Brief History of the Object-oriented Approach. *ACM SIGSOFT Software Engineering Notes*, Vol. 28, No. 2, 2003. https://doi.org/10.1145/638750.638778

[42] Arnold, K., Gosling, J. The Java Programming Language. Massachusetts: Addison-Wesley, 1996.

[43] OpenJDK webpage. https://openjdk.org/projects/jdk/21/ (15.03.2024)

[44] University of Tartu Study Infosystem. https://ois2.ut.ee (15.03.2024)

[45] Object-oriented course home webpage.
https://courses.cs.ut.ee/2023/OOP/spring/Main/KursuseKorraldus (15.03.2024)

[46] Rahman M., M., Watanobe Y. ChatGPT for Education and Research: Opportunities, Threats, and Strategies. *Applied Sciences*, Vol. 13, No. 9, 2023. https://doi.org/10.3390/app13095783

[47] Lepp, M., Kaimre, J. Providing Additional Support in an Introductory Programming Course. *In 2022 IEEE Global Engineering Education Conference (EDUCON)*, pp. 210–216, 2022. https://doi.org/10.1109/EDUCON52537.2022.9766661

Appendix

I. Test sample task

This section contains an example of a test task.

Kontrolltöö 1 aines *Objektorienteeritud programmeerimine*

Automaattestimise võimaldamiseks peavad kõik klassid asuma kindlas paketis ja kõik Java failid peavad olema kindla kodeeringuga. Antud juhul lepime kokku, et klassid asuvad vaikepaketis (st failide alguses ei ole `package` direktiivi) ja failide kodeering on UTF-8.

Raamatupoes müüakse nii raamatuid (mõned nendest on õppevahendid) kui ka ajakirju. Eesti käibemaksuseaduse järgi on erinevate toodete jaoks erinev käibemaksumäär: ajakirjade jaoks 5%, raamatute jaoks 9%, aga õppevahendeid käibemaksuga ei maksustata. Nimekiri toodetest on salvestatud tekstifaili järgmiselt:

```
raamat;3239363520;O. Luts;Kevade;4.45
ajakiri;(01)0123128TEC-IT;Tehnikamaailm;2023-1;7.60
raamat;06-000-00-0034;P. Mancini;Õpime lõbusalt!
Tehnoloogia;12;õppevahend
raamat;9789916164358;N. d'Estienne d'Orves;Eiffel;22.75
ajakiri;123423-IT;Imeline teadus;2023-1;11.90
```

Iga rida algab toote tüübiga. Tüübile järgneb tootekood. Kui tegu on raamatuga, siis järgnevad tootekoodile raamatu esimene autor, raamatu pealkiri ja hind ilma käibemaksuta. Kui tegu on õppevahendiga, siis on lõpus vastav kommentaar. Kui tegu on ajakirjaga, siis järgnevad tootekoodile ajakirja pealkiri, aasta ja number eraldatud kriipsuga ning hind käibemaksuta. Eraldajaks on semikoolon.

Kontrolltöö seisneb toodete ja raamatupoe tööd käsitleva programmi koostamises. Programm peab vastama alltoodud nõuetele (isegi kui need kummalised tunduvad). Programm peab sisaldama klasse `Toode`, `Ajakiri`, `Raamat`, `Klient` ning peaklassi. Peaklassis loetakse sisse toodete andmed ning kliendid ostavad raamatuid ja ajakirju. Peaklassis testitakse ka erinevate isendimeetodite tööd. Kõikide klasside kõik isendiväljad peavad olema privaatsed.

1. (3 p) Abstraktses klassis `Toode` peavad olema privaatsed isendiväljad tootekoodi (`String`), pealkirja (`String`) ja hinna (`double`) jaoks.
    1. Klassis peab olema kolme parameetriga konstruktor koodi, pealkirja ja hinna määramiseks. Klass peab hoolitsema, et hiljem koodi muuta ei saaks.
    2. Vajadusel võib teha isendiväljade jaoks `get`- ja `set`-meetodid.
    3. Klassis peab olema abstraktne `double`-tüüpi parameetriteta meetod `hindMaksuga`, mis tagastab hinna käibemaksuga.
    4. Klassis peab olema ka meetod `toString` toote info mõistlikult tekstina esitamiseks, tuues välja tootekoodi, pealkirja ja hinna käibemaksuga.

5. Klass `Toode` peab realiseerima liidese `Comparable<Toode>`, kusjuures `compareTo` meetod realiseeritakse nii, et võrdlemine toimub käibemaksuga hinna alusel.

2. (2 p) Klass `Ajakiri` on klassi `Toode` alamklass. Ülemklassis olemasolevaid isendivälju siin uuesti mitte kirjeldada. Lisaks peavad olema privaatsed isendiväljad ajakirja aasta (`int`) ja numbri (`int`) jaoks.
   1. Klassis peab olema viie parameetriga konstruktor, mille abil saab määrata ajakirja koodi, pealkirja, hinna, aasta ja numbri (nimetatud järjekorras).
   2. Klassis peab olema meetod `hindMaksuga`, mis käibemaksuga hinna arvutamisel arvestab ajakirjade käibemaksumääraga (5%). Vihje: hind käibemaksuta * 1.05.
   3. Klassis peab olema ka meetod `toString` ajakirja info mõistlikult tekstina esitamiseks, mille ülekatmisel on rakendatud ülemklassi meetodit `toString`, lisades ajakirja aasta ja numbri.

3. (2 p) Klass `Raamat` on klassi `Toode` alamklass. Ülemklassis olemasolevaid isendivälju siin uuesti mitte kirjeldada. Lisaks peavad olema privaatsed isendiväljad raamatu autori (`String`) ja õppevahendi staatuse (`boolean`; `true`, kui raamat on õppevahend) jaoks.
   1. Klassis peab olema viie parameetriga konstruktor, mille abil saab määrata ajakirja koodi, pealkirja, hinna, autori ja õppevahendi staatuse (nimetatud järjekorras).
   2. Klassis peab olema meetod `hindMaksuga`, mis käibemaksuga hinna arvutamisel arvestab sellega, kas raamat on õppevahend (siis käibemaksuga ei maksustata) ning raamatute käibemaksumääraga (9%).
   3. Klassis peab olema ka meetod `toString` raamatu info mõistlikult tekstina esitamiseks, mille ülekatmisel on rakendatud ülemklassi meetodit `toString`, lisades raamatu autori ja teate, kas raamat on õppevahend või mitte.

4. (4 p) Klassis `Klient` peavad olema privaatsed isendiväljad kliendi nime (`String`) ja ostetud toodete nimekirja (`List<Toode>`) jaoks.
   1. Klassis peab olema ühe parameetriga konstruktor kliendi nime määramiseks.
   2. Äsjaloodud `Klient`-tüüpi isendil ei olegi ühtegi ostetud toodet. Toodete lisamiseks peab olema `void`-tüüpi meetod `lisaToode`, mis jätab argumendiks antud `Toode`-tüüpi isendi meelde.
   3. Klassis peab olema `double`-tüüpi parameetriteta meetod `toodeteSumma`, mis tagastab ostetud toodete käibemaksuga hindade summa.
   4. Klassis peab olema `void`-tüüpi parameetriteta meetod `prindiTooted`, kus ostetud tooted sorteeritakse vastavalt meetodis `compareTo` kirjeldatud järjekorrale ja väljastatakse ekraanile nii, et iga toode on eraldi real.
   5. Klassis peab olema ka meetod `toString` kliendi info mõistlikult tekstina esitamiseks, näidates kliendi nime, ostetud toodete arvu ja toodete summat.

5. (5 p) Peaklass peab olema nimega `Peaklass`. Klassis peab olema staatiline avalik meetod `loeTooted` tagastustüübiga `List<Toode>`, mis võtab argumendiks faili nime (sõnena) ja tagastab selles failis olevad toodete andmed. Meetod võib visata erindi (st meetodi signatuuris võib olla `throws Exception`). Toodete faili formaat on ülalpool toodud. Toodete arv failis ei ole teada (programm peaks töötama suvalise arvu toodetega). Kui failist lugemist ei õnnestu programmeerida, siis kirjutage selles meetodis vastav list programmi sisse (vähendab tulemust 2 punkti võrra).

Peameetodis tehakse järgmised tegevused.

1. Rakendatakse vastavat staatilist meetodit, et lugeda failist *tooted.txt* toodete andmed.
2. Luuakse 5 klienti (nimed mõtelge ise välja).
3. Tehakse kõikidest klientidest `Klient[]`-tüüpi massiiv. (Massiivi võib ka enne klientide tegemist luua ja järjest täita.)
4. Iga kliendi jaoks genereeritakse üks arv *n* vahemikust [0; toodete arv] ja see klient ostab nii palju juhuslikult valitud tooteid. Iga kliendi jaoks peab toodete listi ka segama. Selleks tuleb kasutada `Collections.shuffle` meetodit. See meetod võtab argumendiks listi ning järjestab selle suvalises järjekorras. Toodete list järjestada iga kliendi jaoks uuesti ümber ning lisada kliendile esimesed *n* toodet.
5. Väljastatakse ekraanile iga kliendi info.
6. Väljastatakse ekraanile iga kliendi tooted.

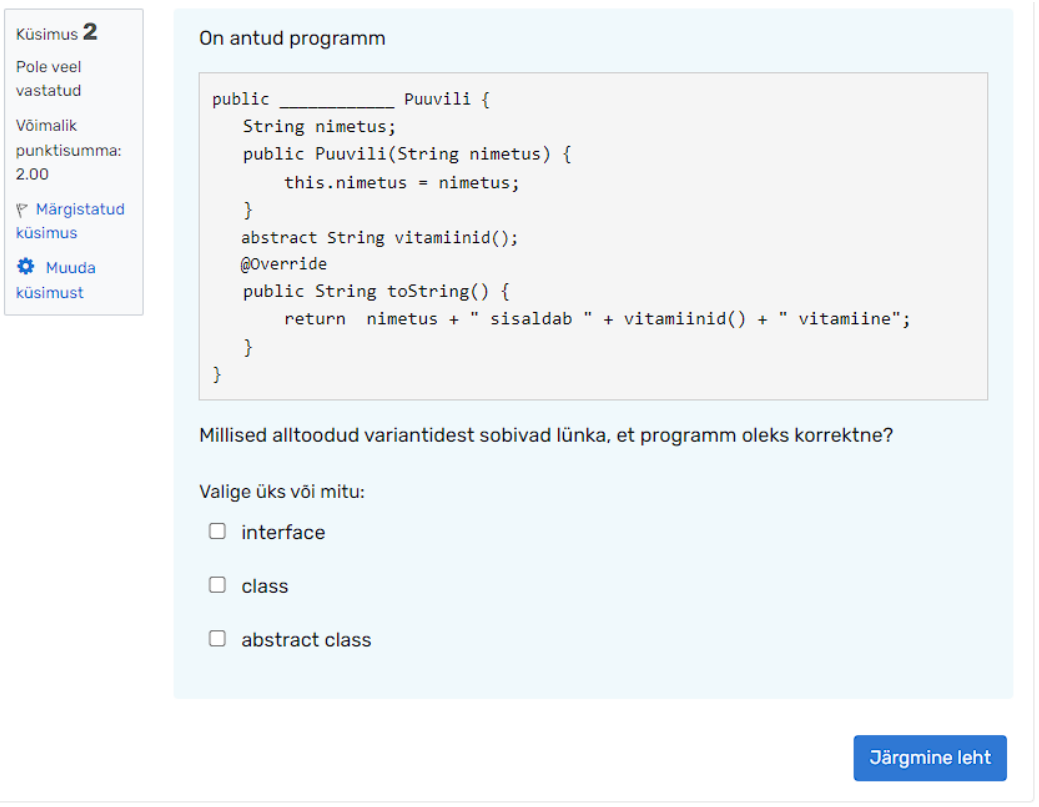
Programmi väljund peab olema arusaadav ja loetav. Andmete fail on aadressil https://kodu.ut.ee/~marinai/tooted.txt. Salvestage see oma arvutisse. Fail on kodeeringus UTF-8.

Mittekompileeruva programmi eest punkte ei saa. Kontrolltöö ajal on Moodle'is kättesaadav automaatne test, mis kontrollib, kas lahendus sisaldab nõutud komponente. Meetodite sisu see test ei kontrolli.

Palun esitada viimane töötav versioon! Palun esitada Moodle'isse (Kontrolltöö nr 1 järeltöö).

II. Example of Short Exam Tasks

This section contains examples of the short exam tasks.



Küsimus 2

Pole veel vastatud

Võimalik punktisumma: 2.00

Märgistatud küsimus

Muuda küsimust

On antud programm

```
public __________ Puuvili {
   String nimetus;
   public Puuvili(String nimetus) {
       this.nimetus = nimetus;
   }
   abstract String vitamiinid();
   @Override
   public String toString() {
       return  nimetus + " sisaldab " + vitamiinid() + " vitamiine";
   }
}
```

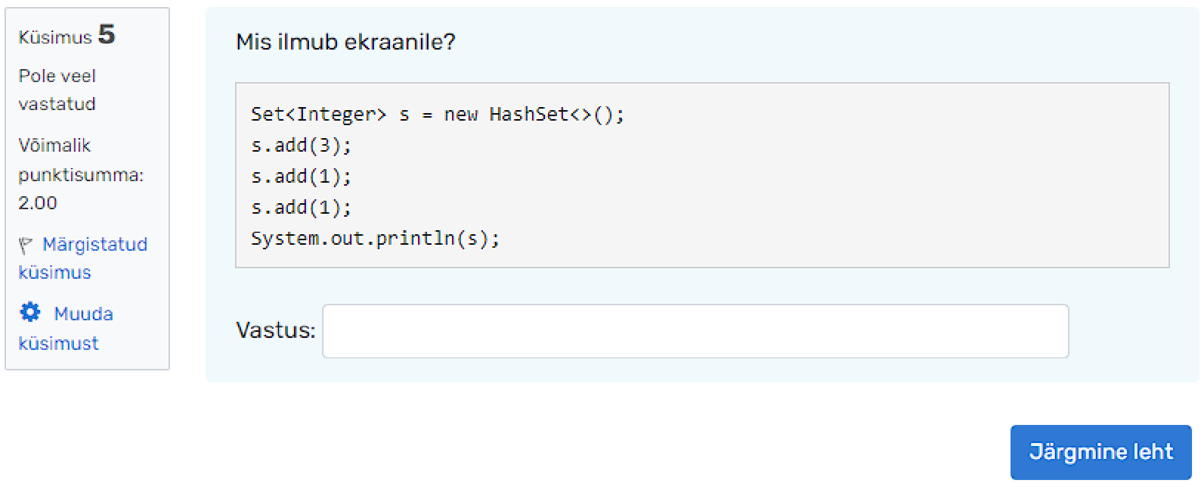
Millised alltoodud variantidest sobivad lünka, et programm oleks korrektne?

Valige üks või mitu:

☐ interface

☐ class

☐ abstract class



Küsimus 5

Pole veel vastatud

Võimalik punktisumma: 2.00

Märgistatud küsimus

Muuda küsimust

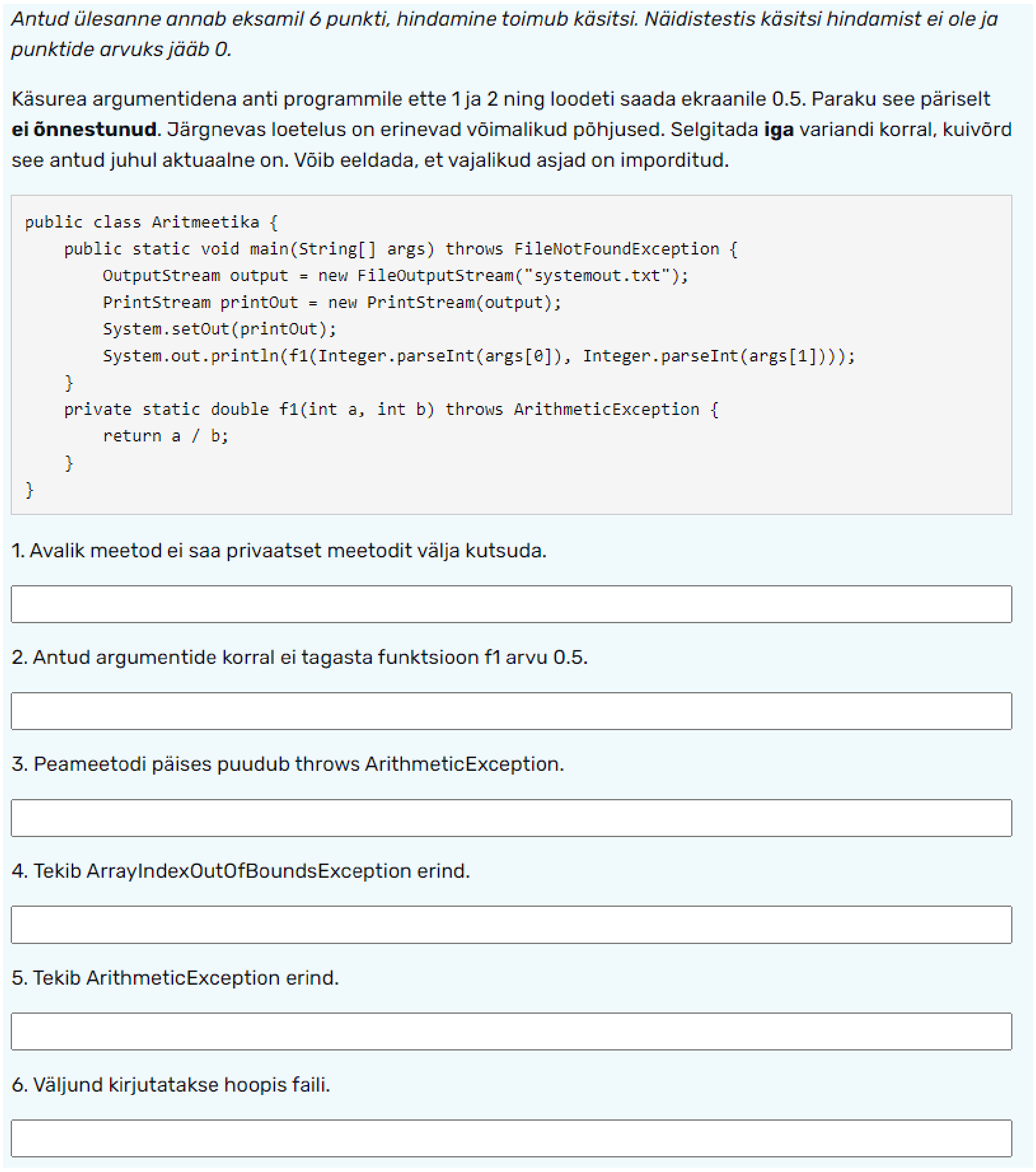
Mis ilmub ekraanile?

```
Set<Integer> s = new HashSet<>();
s.add(3);
s.add(1);
s.add(1);
System.out.println(s);
```

Vastus:

III. Example of Long Exam Tasks

This section contains examples of the two long tasks.



Antud ülesanne annab eksamil 6 punkti, hindamine toimub käsitsi. Näidistestis käsitsi hindamist ei ole ja punktide arvuks jääb 0.

Käsurea argumentidena anti programmile ette 1 ja 2 ning loodeti saada ekraanile 0.5. Paraku see päriselt **ei õnnestunud**. Järgnevas loetelus on erinevad võimalikud põhjused. Selgitada **iga** variandi korral, kuivõrd see antud juhul aktuaalne on. Võib eeldada, et vajalikud asjad on imporditud.

```
public class Aritmeetika {
    public static void main(String[] args) throws FileNotFoundException {
        OutputStream output = new FileOutputStream("systemout.txt");
        PrintStream printOut = new PrintStream(output);
        System.setOut(printOut);
        System.out.println(f1(Integer.parseInt(args[0]), Integer.parseInt(args[1])));
    }
    private static double f1(int a, int b) throws ArithmeticException {
        return a / b;
    }
}
```

1. Avalik meetod ei saa privaatset meetodit välja kutsuda.

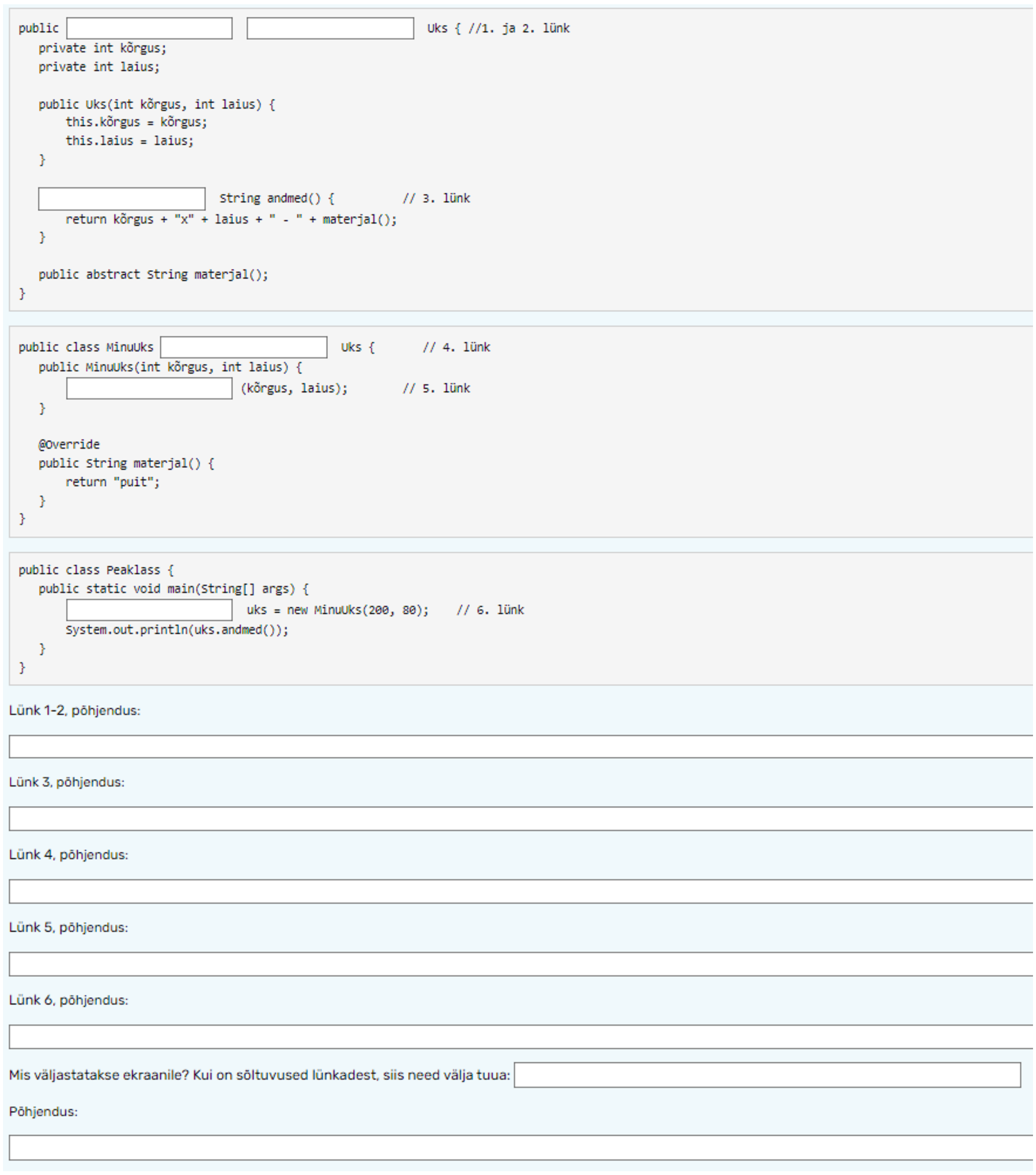
2. Antud argumentide korral ei tagasta funktsioon f1 arvu 0.5.

3. Peameetodi päises puudub throws ArithmeticException.

4. Tekib ArrayIndexOutOfBoundsException erind.

5. Tekib ArithmeticException erind.

6. Väljund kirjutatakse hoopis faili.



```
public                          Uks { //1. ja 2. lünk
   private int kõrgus;
   private int laius;

   public Uks(int kõrgus, int laius) {
       this.kõrgus = kõrgus;
       this.laius = laius;
   }

                String andmed() {          // 3. lünk
       return kõrgus + "x" + laius + " - " + materjal();
   }

   public abstract String materjal();
}
```

```
public class MinuUks              Uks {       // 4. lünk
   public MinuUks(int kõrgus, int laius) {
                  (kõrgus, laius);        // 5. lünk
   }

   @Override
   public String materjal() {
       return "puit";
   }
}
```

```
public class Peaklass {
   public static void main(String[] args) {
                  uks = new MinuUks(200, 80);    // 6. lünk
       System.out.println(uks.andmed());
   }
}
```

Lünk 1-2, põhjendus:

Lünk 3, põhjendus:

Lünk 4, põhjendus:

Lünk 5, põhjendus:

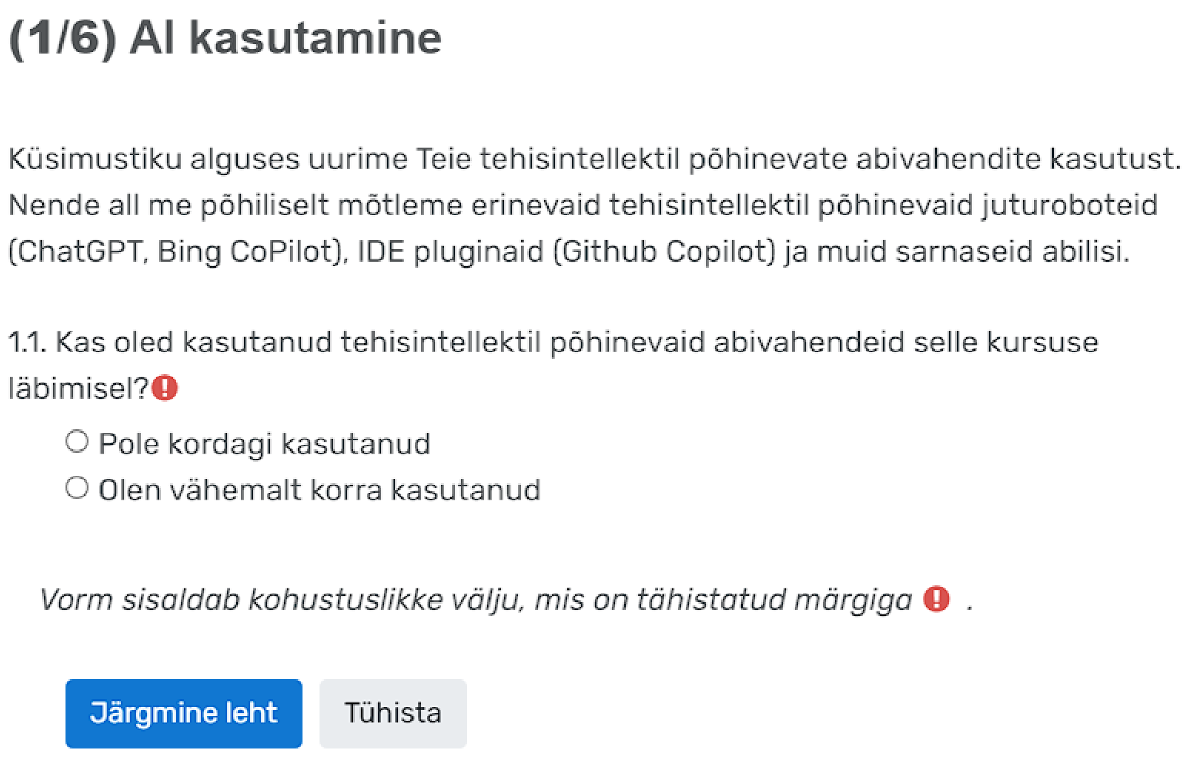
Lünk 6, põhjendus:

Mis väljastatakse ekraanile? Kui on sõltuvused lünkadest, siis need välja tuua:

Põhjendus:

IV. Student Questionnaire

The questionnaire is presented as pictures.



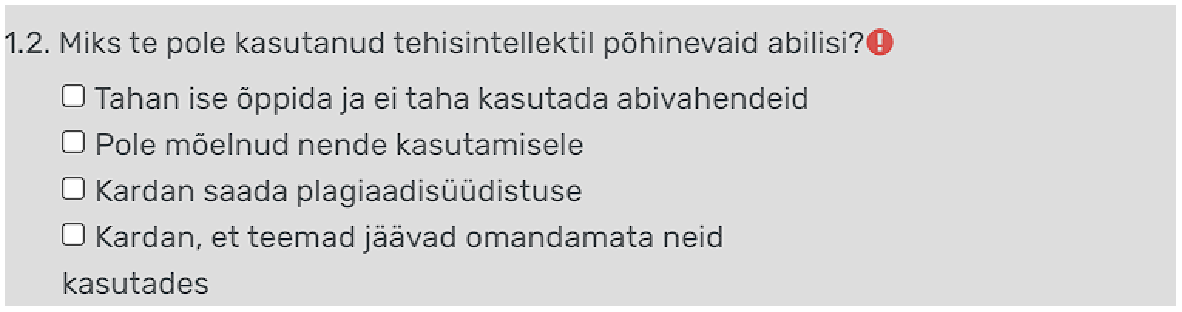
(1/6) AI kasutamine

Küsimustiku alguses uurime Teie tehisintellektil põhinevate abivahendite kasutust. Nende all me põhiliselt mõtleme erinevaid tehisintellektil põhinevaid juturoboteid (ChatGPT, Bing CoPilot), IDE pluginaid (Github Copilot) ja muid sarnaseid abilisi.

1.1. Kas oled kasutanud tehisintellektil põhinevaid abivahendeid selle kursuse läbimisel?❗

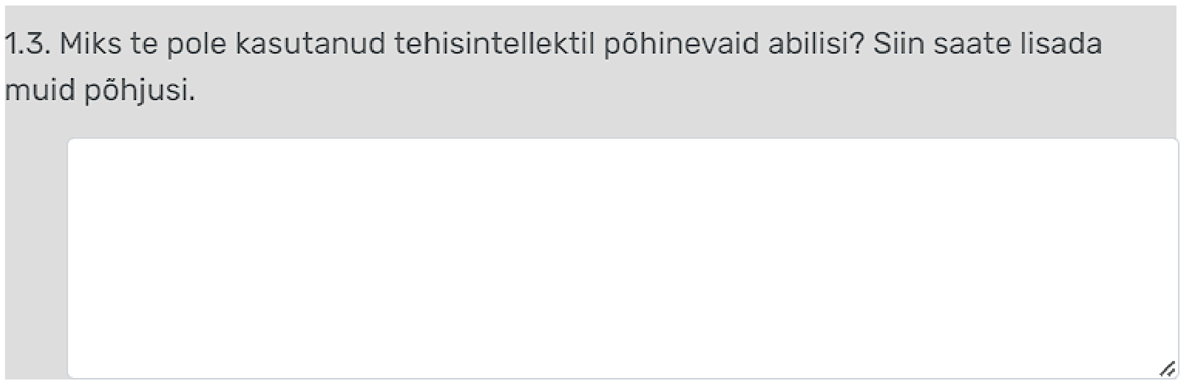
○ Pole kordagi kasutanud
○ Olen vähemalt korra kasutanud

Vorm sisaldab kohustuslikke välju, mis on tähistatud märgiga ❗ .

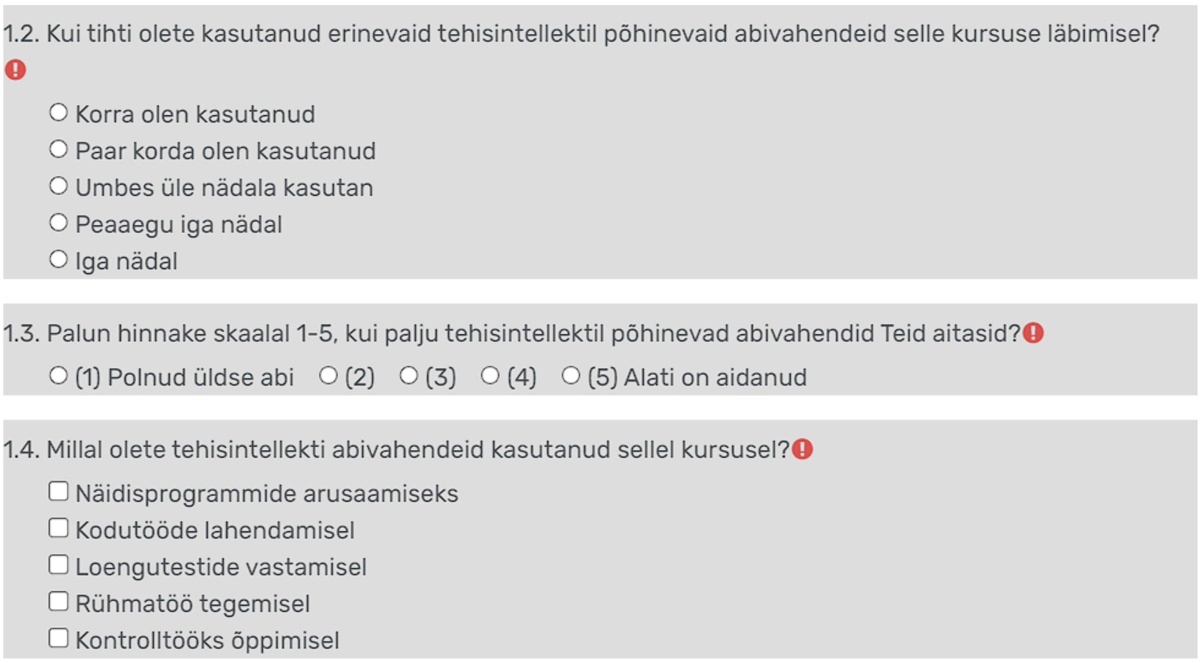


1.2. Miks te pole kasutanud tehisintellektil põhinevaid abilisi?❗

☐ Tahan ise õppida ja ei taha kasutada abivahendeid
☐ Pole mõelnud nende kasutamisele
☐ Kardan saada plagiaadisüüdistuse
☐ Kardan, et teemad jäävad omandamata neid kasutades



1.3. Miks te pole kasutanud tehisintellektil põhinevaid abilisi? Siin saate lisada muid põhjusi.


1.2. Kui tihti olete kasutanud erinevaid tehisintellektil põhinevaid abivahendeid selle kursuse läbimisel?

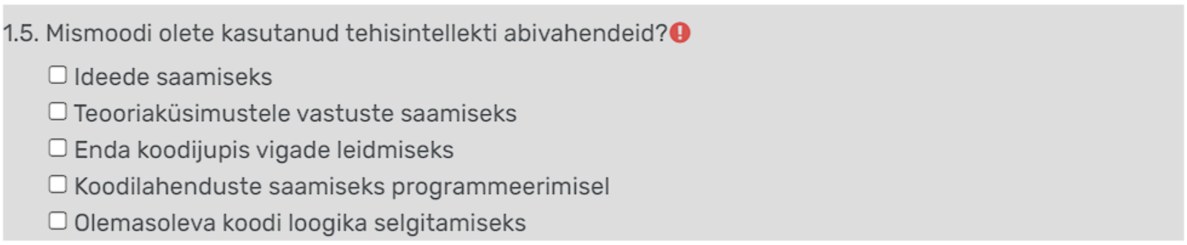
- ○ Korra olen kasutanud
- ○ Paar korda olen kasutanud
- ○ Umbes üle nädala kasutan
- ○ Peaaegu iga nädal
- ○ Iga nädal

1.3. Palun hinnake skaalal 1-5, kui palju tehisintellektil põhinevad abivahendid Teid aitasid?

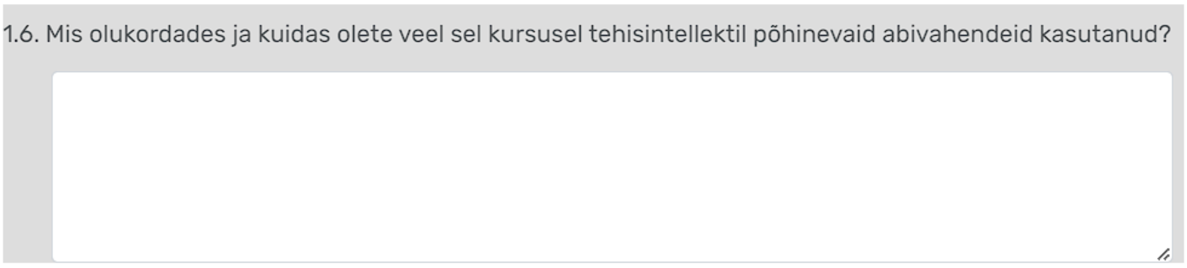
○ (1) Polnud üldse abi ○ (2) ○ (3) ○ (4) ○ (5) Alati on aidanud

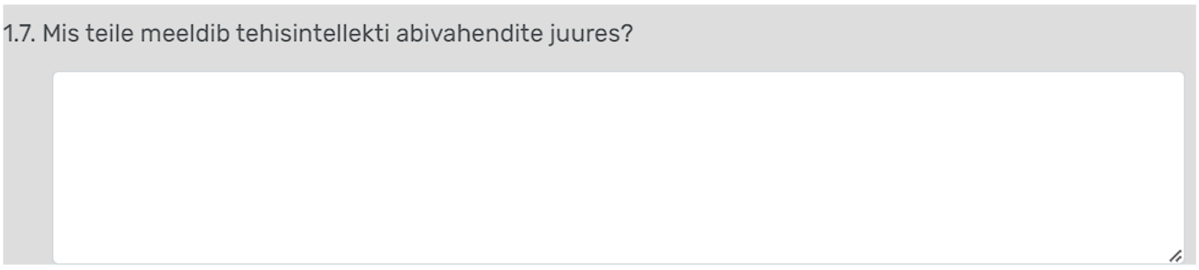
1.4. Millal olete tehisintellekti abivahendeid kasutanud sellel kursusel?

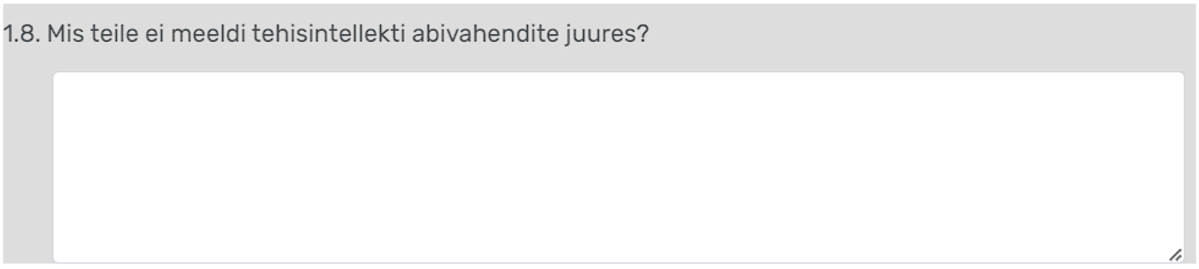
- ☐ Näidisprogrammide arusaamiseks
- ☐ Kodutööde lahendamisel
- ☐ Loengutestide vastamisel
- ☐ Rühmatöö tegemisel
- ☐ Kontrolltööks õppimisel

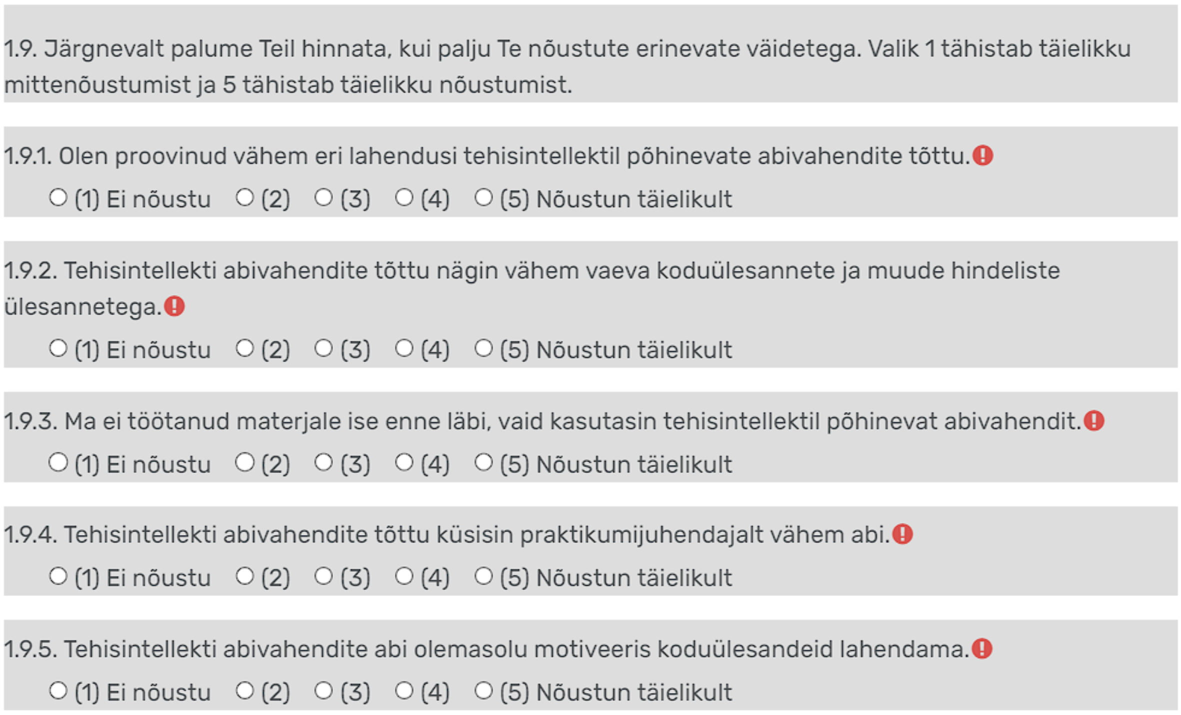

1.5. Mismoodi olete kasutanud tehisintellekti abivahendeid?

- ☐ Ideede saamiseks
- ☐ Teooriaküsimustele vastuste saamiseks
- ☐ Enda koodijupis vigade leidmiseks
- ☐ Koodilahenduste saamiseks programmeerimisel
- ☐ Olemasoleva koodi loogika selgitamiseks


1.6. Mis olukordades ja kuidas olete veel sel kursusel tehisintellektil põhinevaid abivahendeid kasutanud?


1.7. Mis teile meeldib tehisintellekti abivahendite juures?


1.8. Mis teile ei meeldi tehisintellekti abivahendite juures?



1.9. Järgnevalt palume Teil hinnata, kui palju Te nõustute erinevate väidetega. Valik 1 tähistab täielikku mittenõustumist ja 5 tähistab täielikku nõustumist.

1.9.1. Olen proovinud vähem eri lahendusi tehisintellektil põhinevate abivahendite tõttu.❗

○ (1) Ei nõustu ○ (2) ○ (3) ○ (4) ○ (5) Nõustun täielikult

1.9.2. Tehisintellekti abivahendite tõttu nägin vähem vaeva koduülesannete ja muude hindeliste ülesannetega.❗

○ (1) Ei nõustu ○ (2) ○ (3) ○ (4) ○ (5) Nõustun täielikult

1.9.3. Ma ei töötanud materjale ise enne läbi, vaid kasutasin tehisintellektil põhinevat abivahendit.❗

○ (1) Ei nõustu ○ (2) ○ (3) ○ (4) ○ (5) Nõustun täielikult

1.9.4. Tehisintellekti abivahendite tõttu küsisin praktikumijuhendajalt vähem abi.❗

○ (1) Ei nõustu ○ (2) ○ (3) ○ (4) ○ (5) Nõustun täielikult

1.9.5. Tehisintellekti abivahendite abi olemasolu motiveeris koduülesandeid lahendama.❗

○ (1) Ei nõustu ○ (2) ○ (3) ○ (4) ○ (5) Nõustun täielikult

V. Licence

Non-exclusive licence to reproduce the thesis and make the thesis public