

How Proficient Is Generative AI in an Introductory Object-Oriented Programming Course?

Marina Lepp^a and Joosep Kaimre

Institute of Computer Science, University of Tartu, Narva mnt 18, Tartu, Estonia

Keywords: Artificial Intelligence (AI), Object-Oriented Programming, Programming Education, ChatGPT, Copilot, Novice Programmers.

Abstract: In 2022, the release of ChatGPT marked a significant breakthrough in Artificial Intelligence (AI) chatbot usage, particularly impacting computer science education. AI chatbots can now generate code snippets, but their proficiency in solving various tasks remains debated. This study examines how well AI-based chatbots, including ChatGPT and Microsoft Copilot, perform in solving tasks in the "Object-Oriented Programming" course. Both tools were tested on multiple programming tasks and exam questions, with their results compared to those of students. Currently, ChatGPT-3.5 performs below the average student, while Copilot is on par. The chatbots performed better on introductory topics, though their performance varied as task difficulty increased. They also fared better on longer programming test tasks than on shorter exam tasks. A common error was failing to provide all possible solutions and misinterpreting implied requirements. Despite these challenges, both AI tools are capable of passing the course. These findings offer valuable insights for programming instructors by highlighting the strengths and weaknesses of AI chatbots, helping guide potential improvements in computer science education.

1 INTRODUCTION


The year 2022 marked a breakthrough for Artificial Intelligence, especially with the public release of ChatGPT. This event brought widespread attention to AI assistants and chatbots, sparking concerns about AI replacing humans and rendering certain jobs obsolete. One field that is notably affected is computer science and its education (Denny et al., 2024).

The rise of AI has led to a surge of research on its proficiency in university courses (Denny et al., 2024; Finnie-Ansley et al., 2022; Finnie-Ansley et al., 2023; Richards et al., 2024), its use as a tool (Denny et al., 2023; Jukiewicz, 2024; Kiesler et al., 2023), and its impact on student learning (Sun et al., 2024; Yilmaz & Yilmaz, 2023). There is currently no consensus on whether AI outperforms the average student, with some studies suggesting it surpasses students (Finnie-Ansley et al., 2022; Finnie-Ansley et al., 2023; Richards et al., 2024), while others indicate that AI can pass courses but falls short compared to students (Bordt & Luxburg, 2023; Shoufan, 2023).

Specifically, in courses on object-oriented programming, there is limited research on how AI chatbots compare to students and whether there are particular areas where AI struggles. Additionally, there is little research on how well AI assistants handle input in languages other than English (for example, Estonian) and whether this affects their performance.

The main goal of this study is to analyze the proficiency of various AI chatbots in an introductory object-oriented programming course and to compare their performance to that of students enrolled in the course. In this paper, the terms "AI assistants", "chatbots" and "AI chatbots" are used interchangeably to refer to AI-based models that students can interact with. To achieve this goal, the paper will address the following research questions:

- How do different chatbots perform in the "Object-Oriented Programming" course compared to students?
- What are the common mistakes made by AI chatbots while solving the tasks from the "Object-Oriented Programming" course?

^a <https://orcid.org/0000-0003-3303-5245>

The paper opens with a summary of existing research on the proficiency of AI chatbots in computer science courses. Section 3 provides an overview of the "Object-Oriented Programming" course and outlines the evaluation methods used for the AI chatbots in this course. Section 4 presents the results, while Section 5 discusses those results in detail.

2 BACKGROUND

Several studies have evaluated the effectiveness of various AI approaches for solving tasks across different fields, including economics (Geerling et al., 2023), law (Bommarito & Katz, 2022), and medical studies (Lee, 2023). However, this study primarily focuses on computer science and programming courses. The majority of previous studies about computer science have concentrated on ChatGPT (e.g. Richards et al., 2024; Sun et al., 2024; Yilmaz & Yilmaz, 2023), although some have explored other tools like GitHub Copilot (e.g. Finnie-Ansley et al., 2022).

Several studies have shown that AI tools often perform within the top quartile of the respective class in introductory programming courses (e.g. Finnie-Ansley et al., 2022; Richards et al., 2024). AI excels in tasks covering basic topics but shows greater variability in more complex areas like data structures and algorithms (Finnie-Ansley et al., 2023). Bordt and Luxburg (2023) confirmed this variability, finding that while ChatGPT-3.5 could pass a data structures course, it performed below the average student, whereas ChatGPT-4 performed comparably to students. Shoufan (2023) reported similar outcomes, with ChatGPT achieving a passable grade but still trailing behind students. Similarly, it was found that ChatGPT could pass undergraduate courses but struggled at the postgraduate level (Richards et al., 2024). It outperformed students on introductory topics but was outperformed on more advanced tasks. In contrast, in another study, no significant difference was discovered between AI's performance in introductory and intermediate tasks, with AI sometimes excelling in intermediate-level assessments (Savelka et al., 2023). Overall, AI assistants tend to outperform students at the introductory level but struggle more with expert-level tasks. The line between introductory and intermediate tasks is not always clear, and while AI may surpass students in beginner courses, it lags behind on more challenging tasks, though still capable of passing.

In terms of topic-specific performance, AI performed best in algorithms and data structures, followed by operating systems, machine learning, and database management systems (Joshi et al., 2024). Beyond university courses, AI chatbots have also been tested in competitive programming challenges like Leetcode (Kadir et al., 2024). ChatGPT surpassed the average human acceptance rate across nearly all categories and difficulty levels (easy, medium, hard), with the exception of tasks involving bit manipulation. However, these challenges are not directly comparable to university courses, as participants may be less motivated to perform at their best since the results do not impact their GPA.

ChatGPT's ability to generate solutions from non-English (Czech) input has been evaluated in information security courses (Malinka et al., 2023). ChatGPT successfully passed all of the four courses where it was assessed. AI outperformed the students' average in one course, while students outperformed AI in the other three. ChatGPT often outperformed students in full-text exams requiring written answers or solutions. However, students generally performed better in tasks involving projects, essay writing, and coding small snippets.

Research on AI performance in object-oriented programming courses taught in Java, particularly when comparing AI assistants to student results, is limited. It was found that while ChatGPT can handle simpler tasks, it struggles with more complex ones (Bucaloni et al., 2024; Ouh et al., 2023; Wang et al., 2024). However, it often provides partial solutions that give students a useful starting point. Another issue noted is that when tasks involve data presented as UML diagrams or API documentation, chatbots have difficulty parsing the full input. This challenge is not unique to Java or object-oriented programming but highlights the broader limitations of AI in handling non-text inputs. Camara et al. (2023) support this, reporting that ChatGPT struggles to reliably generate UML diagrams and often encounters syntax issues. Cipriano and Alves (2024) further confirmed AI's difficulties with object-oriented tasks, finding that AI-generated code frequently contained compilation errors, required multiple prompts to complete all necessary classes and functions, and failed some unit tests used for task evaluation. In addition, code generated by ChatGPT is prone to several quality issues, such as runtime errors, incorrect outputs, and challenges with maintainability (Liu et al., 2024). The limited research comparing AI assistants to students makes it difficult to determine whether AI faces similar

challenges or if it outperforms students in some areas while underperforming in others.

Overall, AI has advanced to a level where it can successfully pass university courses, but it has yet to consistently outperform the average student. This may be due to the significant variation in AI's proficiency across different tasks. While AI can easily handle small code snippets and exam questions, it struggles with larger, more complex projects, programs, and essays. For essays, AI often encounters issues with proper citations and may generate false references (Malinka et al., 2023). Whether AI exceeds the students' average in a course often depends on the grading scheme and the weight of different tasks in the final grade. Another factor affecting AI performance is the presentation of task descriptions. AI can also pass non-English courses, but more research is needed to determine if this capability extends beyond Czech to other languages, such as Estonian. Research on AI in Java-based object-oriented programming courses is limited, but the general trend aligns with other fields: AI excels at simpler, smaller tasks but struggles as complexity increases. The lack of studies comparing AI to students in Java courses makes it difficult to pinpoint where AI's performance differs from that of students or whether they make similar mistakes.

3 RESEARCH METHOD

3.1 Course Details

Object-oriented programming (OOP) is a widely used paradigm in contemporary programming. It revolves around the concept of objects, which are instances of classes that encapsulate both data and the behaviors associated with that data (Gabbrielli & Martini, 2023). At the University of Tartu, this paradigm is taught to first-year students in the course "Object-Oriented Programming", using the Java programming language. The data, tasks, and results from this course were utilized to compare the proficiency of AI chatbots in object-oriented programming tasks against that of novice programmers.

The "Object-Oriented Programming" course is primarily taught to first-year Computer Science students as a mandatory part of their curriculum. However, it is also selected as an elective by many other students, making it one of the largest courses at the University of Tartu, with annual enrollment ranging from 270 to 330 students. To enroll in the "Object-Oriented Programming" course, students must first complete an introductory course in Python

(Leping et al., 2009). Since students are not required to have prior experience with object-oriented programming or Java, most enter the course as complete beginners in both the language and the subject matter.

The course lasts for 16 weeks and consists of weekly lectures, homework assignments, and practice seminars, employing a flipped-classroom approach (Lepp & Tõnisson, 2015). Each week, students are required to watch lecture videos, complete a short quiz, tackle homework related to the weekly topic, and attend practice seminars to enhance their understanding. Throughout the course, they must take two major tests and complete two group assignments, culminating in a final exam. The final grade primarily depends on the exam (33 points out of 102) and the tests (worth 16 points each). Points earned in practice seminars are based on attendance alone. Most homework assignments feature automated tests that provide immediate feedback, allowing students to resubmit their work until they achieve the highest score. Group projects are open-ended tasks with specific requirements, granting students the freedom to choose their topics and implementations, which can lead to less uniform assessments since graders also take the students' skill levels into account. As a result, our analysis will focus on comparing AI and student performance on the exam and tests, as these assessments have clear and consistent grading criteria, enabling more accurate comparisons. Since these two components significantly influence the final grade (almost 64% in total), this analysis will offer insights into AI's proficiency relative to students, including the potential for academic dishonesty and the extent to which AI could assist students.

The two tests are scheduled for weeks 7 and 13, each covering the topics taught in the preceding weeks. Students are provided with a program outline listing the required classes, the methods each class should include, any interfaces the classes must implement or properties to inherit from a superclass, and the expected main workflow of the program. Students have 105 minutes to complete the program, during which they are allowed to use any materials except for communicating with others or seeking help from an AI chatbot. The beginning of the sample test task (translated into English) can be seen in Figure 1.

During the test, the students can use automatic tests to verify that all required classes and methods are present. However, these tests do not assess the internal logic of the program. It is the student's responsibility to test and debug their code to ensure it aligns with the provided task outline.

The test consists of writing a program about planes and airport operations. The program must meet the requirements below (even if they seem strange). The program must contain the classes `Airplane`, `CargoAirplane`, `PassengerAirplane`, `Airport` and a main class. The main class reads in airplanes' data and simulates that some airplanes leave the airport. The main class also tests the work of various instance methods. All instance variables of all classes must be private.

1. (3 p) In the abstract class `Airplane`, there must be a private instance variable for the airplane registration number (`String`).
 1. The class must have a one-parameter constructor for setting the instance variable. The class must ensure that the registration number cannot be changed later.
 2. The class must have an abstract double-type method `airportTax`.
 3. The class must also have a `toString` method for displaying airplane information as text, including the registration number and airport tax.
 4. The `Airplane` class must implement the `Comparable<Airplane>` interface, with the `compareTo` method implemented so that the comparison is based on airport tax.

Figure 1: The beginning of the sample task for test 1.

What will be displayed on the screen?

```
LinkedList<Integer> ll = new LinkedList<>();
ll.add(1);
ll.add(2);
ll.poll();
ll.add(3);
System.out.println(ll);
```

Answer:

There is a program

```
public _____ A {
    protected abstract void a();
}
```

Which of the following options would fit the gap to make the program correct?

Select one or more:

☐ class

☐ interface

☐ abstract class

Figure 2: Examples of the short exam tasks.

The exam is taken at the end of the course through a computer-based test on Moodle. Students are allowed to use course materials, review code examples and documentation, and search on Google, but they are prohibited from communicating with others, using AI assistants, opening IDEs, compiling code, or running it. The exam lasts 60 minutes and covers all topics taught in the course. The first question is a declaration of honesty, worth 1 point, and the final question is a longer, open-ended task worth 6 points. Between these are 13 other questions, worth 2 points each, presented in a random order. These questions include multiple-choice, single-choice, fill-in-the-blank, and matching types, with the specific set of options varying depending on the question. Students cannot revisit previous questions once they have moved on to the next. Examples of the short question (translated into English) can be seen in Figure 2.

The longer, open-ended question requires students to explain and justify their answers. There

are two types of this question: one involves filling in the gaps of a code snippet and providing all possible answers along with reasoning, while the other presents a code snippet with errors, asking students to evaluate statements about what is wrong with the code and justify their decisions. Examples of both types of longer tasks (translated into English) are shown in Figure 3.

The program was given 1 and 2 as command line arguments and expected to get 0.5 on the screen. Unfortunately, this **did not work**. The following list contains various possible reasons. Explain for **each** option how relevant it is in this case. It can be assumed that the necessary things have been imported.

```
public class Arithmetic {
    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. A public method cannot call a private method.
2. With the given arguments, the function `f1` does not return the number 0.5.
3. There is no `throws ArithmeticException` in the header of the main method.
4. The output is written to a file.
5. An `ArrayIndexOutOfBoundsException` is thrown

Three files are given. Fill in the gaps so that the rest of the code does not need to be changed. A gap can contain a **single word** or **be left blank**. If there are dependencies between any gaps, note them as well (in the style of "if you put ... in the first gap, then you can put ... in the third gap). If there are multiple options, write down **all the options**. Explain your choices!

```
public _____ Measurable { //1-2 gap
    public int height();
}
public abstract class Building {
    public _____ String buildingOwner(); //3 gap
    public _____ String toString() { //4 gap
        return "Building - owner: " + buildingOwner();
    }
}
public class ApartmentBuilding _____ Building _____ Measurable
{ //5, 6 gap
}
```

Gap 1-2, explanation:

Gap 3, explanation:

Gap 4, explanation:

Figure 3: Examples of the two long tasks.

3.2 Procedure and Data Analysis

ChatGPT-3.5 and Microsoft Copilot were selected for testing. ChatGPT-3.5 was chosen due to its free availability, unlike ChatGPT-4, which requires a paid subscription. Since students are more likely to use the free version, it was assumed they would opt for ChatGPT-3.5. Microsoft Copilot was included because students at the University of Tartu have free access to its paid version through their university email, making it a likely choice for students.

To evaluate how well ChatGPT and Copilot could handle tasks in the introductory object-oriented programming course, they were provided with the full text of the tests and tasks, and their outputs were graded accordingly. While ChatGPT processed the full text with no issues, Copilot had a 4000-character limit, requiring some tasks to be split into multiple queries. No additional adjustments were made, which also meant that the AI chatbots received the tasks in Estonian.

To assess how effectively the AI tools completed the tasks, the standard grading scheme of the course was used. Common errors were documented to identify recurring issues and potential problem areas. Each AI chatbot was given three different versions of the tests to gather more data points. For the exam, which consists of a range of questions from a Moodle question bank, the AI assistants were tested on ten questions from each set to establish a reliable average performance for each topic. The same set of questions was presented to both AI assistants.

4 RESULTS

4.1 Programming Test 1

Test 1 covers key topics such as Java classes, objects, Strings, files, lists, polymorphism, interfaces, and abstract, super- and subclasses. The test is worth 16 points, with a passing score set at 12 out of 16. ChatGPT and Copilot outputs were evaluated. On average, ChatGPT achieved a score of 14.65, while Copilot scored 15.85 out of 16, both exceeding the required minimum score. A total of 285 students took this test, with the average score being 14.61 points. In comparison, both ChatGPT and Copilot performed better, with their averages being higher. However, the difference between ChatGPT and the students was relatively small, while Copilot showed a more significant advantage. When looking at individual tests, ChatGPT outperformed the students' average on two of them. However, the students' average was

lowered by non-compiling solutions, which were automatically scored as 0. Copilot, on the other hand, outperformed the students on all three tests.

Figure 4 shows the boxplot comparing the results of the students and the AI chatbots for test 1. To improve clarity, outliers were excluded from the visualization since non-compiling solutions automatically received a score of zero. The AI chatbot results are marked with logo pictures, and the students' average is represented as a cross on the boxplot. When examining the quartile distribution, ChatGPT's scores were below the lower quartile for T1.2 and T1.3 and only slightly above for T1.1, suggesting that ChatGPT underperforms compared to the average student on more lengthy and complex tasks. In contrast, Copilot performed better, scoring 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.

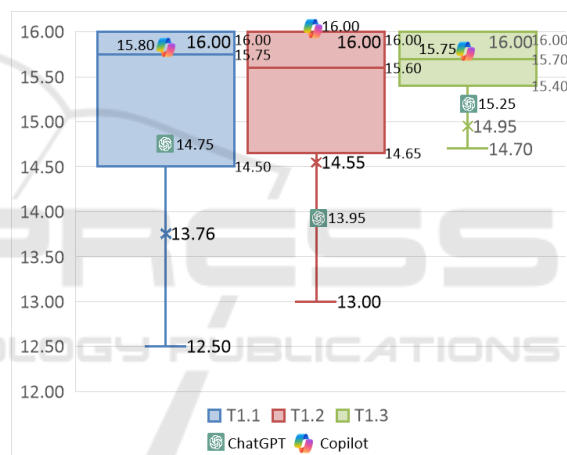


Figure 4: Students' and AI chatbots' results in test 1.

ChatGPT consistently made two mistakes (see Table 1): it failed to specify the required encoding for files and defined logic in abstract methods within the superclass without overriding it in the subclasses. Copilot, however, had no recurring errors. It only missed specifying file encoding once and added null values to a toString method in just one of the three tests.

4.2 Programming Test 2

Test 2 covers streams, exception handling, and data structures, in addition to the topics covered in test 1. The test was worth 16 points and unlike test 1, there was no minimum score required to pass. ChatGPT and Copilot were given the problem descriptions of three test variants, and their results were graded.

Table 1: ChatGPT and Copilot mistakes in test 1.

Test version	ChatGPT mistakes	Copilot mistakes
T1.1	1. Did not use the required encoding for files. 2. A method that needed to be abstract was defined.	Displayed null values in toString method .
T1.2	1. Did not use the required encoding for files. 2. A method that needed to be abstract was defined. 3. Used wrong method names. 4. A mistake in application logic.	Made no mistakes.
T1.3	1. Did not use the required encoding for files. 2. Did not read data from a file.	Did not use the required encoding for files.

On average, ChatGPT achieved a score of 12.13, while Copilot scored 14.87 points out of 16. The students' average for these tests was 13.39 points. ChatGPT scored lower than the students' average, whereas Copilot scored higher. However, when analyzing the individual tests, ChatGPT outperformed students in T2.3 but scored lower in T2.1 and T2.2. Copilot, on the other hand, outperformed students in all test variants.

Figure 5 illustrates the performance of the students and the AI chatbots in test 2. Outliers, such as non-compiling solutions that received a zero score, were excluded for clarity. In terms of quartile distribution, ChatGPT's scores were below the lower quartile for T2.1 and T2.2 and fell between the lower quartile and the median for T2.3. In contrast, Copilot's scores were above the median for T2.1 and T2.3 and just below the median for T2.2. These findings confirm that ChatGPT underperformed compared to the average student, while Copilot consistently outperformed the students' average across the tests.

ChatGPT exhibited several recurring mistakes (Table 2), with the most common being that methods were marked as public instead of private. This requirement was outlined in the test, as methods were not supposed to be accessible outside the class, which likely impacted the outcome. However, a similar problem with private methods was also mentioned previously (Wang et al., 2024). The test variants differed in that two required the use of queues, while one required the use of maps. ChatGPT made more mistakes in the tests involving queues (T2.1, T2.2), particularly with reading user input, while it performed better in the test that involved maps (T2.3). Another recurring issue was sorting: the tests required

a non-decreasing order, but ChatGPT sorted in the opposite direction.

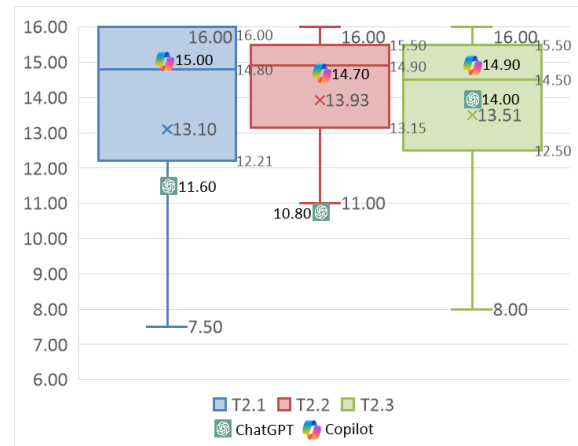


Figure 5: Students' and AI chatbots' results in test 2.

Table 2: ChatGPT and Copilot mistakes in test 2.

Test version	ChatGPT mistakes	Copilot mistakes
T2.1	1. Methods were not private. 2. The logic for asking user input did not work correctly. 3. Did not use user input. 4. Multiple mistakes in application logic.	1. Did not generate get and set methods.
T2.2	1. Methods were not private. 2. The logic for asking user input did not work correctly. 3. Did not use user input. 4. Sorted in the wrong direction. 5. Did not generate some functionalities.	1. Did not generate some get methods. 2. Sorted in the wrong direction.
T2.3	1. Methods were not private. 2. Sorted in the wrong direction. 3. Small mistake in reading input.	1. Did not generate some get methods. 2. A method was not private.

Copilot had a recurring issue in all three test variants where it failed to generate some required getter methods and found workarounds instead. Other mistakes were more unique to individual test variants. Like ChatGPT, Copilot struggled with the sorting direction and setting a method as private, but these issues occurred only in one test variant, unlike the recurring mistakes seen with ChatGPT. Additionally, Copilot did not show any noticeable performance

differences between the queue-based tests (T2.1, T2.2) and the map-based test (T2.3).

4.3 Exam

The AI assistants performed similarly on questions related to objects and classes (see Appendix). However, a noticeable difference emerged in questions, which focused on strings, files, and lists (Q3). Both struggled with various string methods, but ChatGPT also encountered issues with `Collections.sort` and list indexing. The most significant discrepancy, which contributed to the larger gap in points, was that ChatGPT struggled to accurately handle string values stored in variables, often confusing values between variables or introducing new ones. This issue did not occur in the Copilot's responses.

Copilot consistently outperformed ChatGPT on topics related to interfaces, sub-, super- and abstract classes (see Appendix Q4-Q9)—even though they made similar mistakes, Copilot did so less frequently. Both assistants struggled with class hierarchy, specifically confusing the order of method declaration searches and how a superclass's constructor is called during instance creation. Another shared mistake was assuming that an abstract class must implement an interface's methods. A unique error for Copilot was attempting to define an abstract method in an abstract class without using the keyword "abstract". ChatGPT, on the other hand, had more distinct errors, such as confusing when to use "extends" versus "implements," misunderstanding when access modifiers are required, and mixing up what is permissible in classes versus abstract classes.

The graphics questions could not be analyzed since they included a picture of a JavaFX program, which could not be input into the AI assistants. For the events topic (Q11), all issues were due to logical and string comparison errors, rather than a misunderstanding of how events and changes function. Overall, Copilot and ChatGPT made similar mistakes, but the main difference in their scores came from the fact that ChatGPT made these errors more frequently than Copilot.

The only topic where ChatGPT outperformed Copilot was exception handling (Q13), which is notable since Copilot outscored ChatGPT in all other areas. Both AI assistants struggled with print statements, often ignoring them, and incorrectly assumed that if an exception is thrown in a catch block, it would be caught automatically, which is not the case. Another shared error was their misunderstanding that data streams cannot

interchangeably use `readUTF/writeUTF` for `readInt/writeInt` (Q12). Additionally, ChatGPT had more issues understanding file lengths. Both AIs struggled with the stack data structure (Q14), consistently failing on this topic. As there was no time when they answered correctly about stacks, this likely indicates a lack of sufficient training data. ChatGPT also showed difficulties with queues and sets, a recurring issue that also appeared in the results of test 2. This suggests that Copilot has a better grasp of less common data structures compared to ChatGPT.

The long tasks (Q15) required a comprehensive understanding of the topics from the entire course, leading to a variety of mistakes, many of which were similar to errors made in earlier questions, such as issues with superclass constructor calls and string-to-number conversions. A recurring problem was that when the task required providing all possible solutions, the AI assistants almost always gave only one correct answer instead of multiple possibilities. For example, they would provide only "public" as the correct keyword, even though other keywords were also valid, or they would use only an abstract class or the superclass for instance creation. The tasks also required explanations for the proposed solutions, where the AI assistants performed well if their initial answer was correct. As with other tasks, Copilot outperformed ChatGPT in this area.

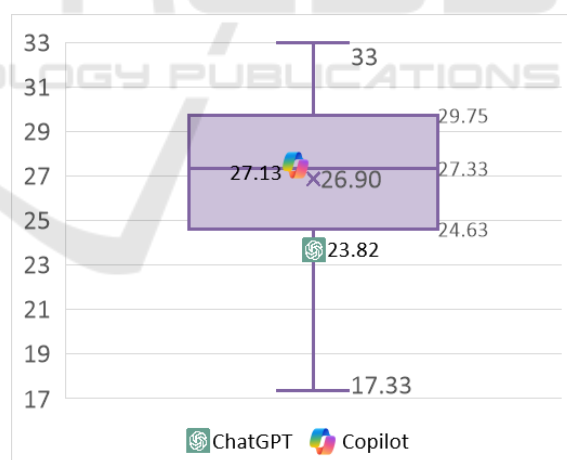


Figure 6: Students' and AI chatbots' results in the exam.

In terms of average performance, Copilot achieved a higher average score (27.13 points) than the students (26.90, $SD=3.62$), while ChatGPT had a lower average (23.82). The exam results for the students and the AI chatbots are shown as a boxplot in Figure 6. Although Copilot had a higher average score than the students, it fell below the median, suggesting that more than half of the students

performed better. ChatGPT fared even worse, with an average score below the bottom quartile, indicating that 75% of the students outperformed it. Overall, this suggests that half of the students have a better understanding of the topics than the AI chatbots.

5 DISCUSSION

5.1 Comparison of AI vs. Students

One of the goals of the study was to compare chatbot performance with that of students in the "Object-Oriented Programming" course. In test 1, both AI chatbots outperformed the students' average, but this was mainly because non-compiling student solutions received automatic zeros, lowering the average. Quartile analysis showed ChatGPT being consistently below the bottom quartile, meaning 75% of students outperformed it. Copilot, however, performed better, scoring above the median in all three tests and reaching the upper quartile in one. These results align with Bordt and Luxburg's (2023) findings, where ChatGPT-3.5 passed a course but underperformed compared to students, while ChatGPT-4 performed similarly to them. As Copilot is based on GPT-4, the results are consistent with this observation. However, other studies (Finnie-Ansley et al., 2022; Finnie-Ansley et al., 2023; Richards et al., 2024) have found that AI tools like ChatGPT and Codex ranked above the upper quartile in introductory CS courses, possibly due to the tasks being in English, whereas this study used Estonian tasks. Similar language-related performance variance was observed in Czech information security courses (Malinka et al., 2023), where students often outperformed AI.

In test 2, the AI results differed from the students' average: ChatGPT performed worse, while Copilot did better. ChatGPT was in the bottom 25% for two test variants and in the bottom half for the third. Copilot scored above the median for two variants but below it for one. This continues the trend from test 1 where ChatGPT-3.5 passed but underperformed compared to students, while Copilot performed closer to the students' median. However, both chatbots performed worse overall in test 2 compared to test 1. These results align with studies (Finnie-Ansley et al., 2023; Richards et al., 2024), showing that AI tools do better on simpler tasks but struggle with more complex ones, even though other research (Savelka et al., 2023) has found no such difference. The exam results confirmed this pattern: ChatGPT scored lower than the students' average while Copilot

outperformed it, but both AIs still fell below the median student. A key issue was the AIs' inability to parse one question presented as an image, costing them 2 points. This reflects previous challenges with interpreting UML diagrams (Cámara et al., 2023; Ouh et al., 2023).

An interesting observation is that the AI performed better on the tests than on the exam. The tests involved creating programs with multiple classes and functions from longer textual descriptions, while the exam focused on shorter questions requiring decisions about smaller code snippets. One might expect more errors in longer texts, yet the AI struggled more with the shorter questions. This aligns with previous research (Malinka et al., 2023), which found that students excelled at solving small code snippets compared to AI, but the AI's ability to handle more complex tasks from detailed descriptions stands out as noteworthy.

5.2 Common AI Mistakes

The second research question examined common mistakes made by AI chatbots in solving course tasks. The mistakes by ChatGPT and Copilot were documented to identify patterns. In test 1, the most frequent error, present in all ChatGPT solutions and once in Copilot's, was failing to specify file encoding. This was a requirement stated in the task, as the files are in a particular encoding, with the assumption that data would be read using that encoding. Copilot also displayed null values in a toString method, and ChatGPT had issues reading files and used incorrect method names. Another recurring error for ChatGPT was failing to make a superclass method abstract. Instead, it defined logic in it, which worked but did not follow the task's specification. Test 2 had more errors. Both chatbots failed to make certain methods private, likely due to the requirement being written in Estonian as the methods are not meant to be callable outside the class, and this may have caused confusion for the AI assistants, as they did not make similar mistakes in test 1. Both also sorted data incorrectly, possibly due to confusion over the term "non-decreasing," which was in Estonian. Copilot struggled with generating get and set methods, while ChatGPT had difficulties with Queue data structures, misunderstanding how the task should be handled. Interestingly, while ChatGPT and Copilot made similar mistakes on the same tests, no repeated errors were observed between tests 1 and 2. This may be due to the different topics each test covered, which reduced the likelihood of similar mistakes. Although ChatGPT and Copilot failed to generate some

required get and set methods, they did not encounter compilation issues, which have been highlighted by previous research (Cipriano & Alves, 2024).

The exam covered various course topics, requiring the AI assistants to tackle a wide range of tasks. A frequent issue, which was not related to the content but to the question format, was that the AI often failed to choose answers from the multiple-choice options provided, usually selecting just one correct answer instead of listing all. Both ChatGPT and Copilot struggled with String comparison, making case-sensitive errors and using incorrect methods. ChatGPT also had trouble understanding variable values and made mistakes related to lists, including issues with indexes, reference-based handling, and sorting. Complex topics like interfaces, abstract classes, and class hierarchies posed challenges. ChatGPT consistently made errors with abstract methods, and both AI tools struggled with method implementations and understanding how superclass constructors are called. They also had difficulty using the keywords "abstract," "extends," "implements," and understanding access modifiers in interfaces and abstract classes. These issues suggest that when faced with less common or edge-case problems, AI assistants are more prone to mistakes.

The graphics questions revealed that AI assistants could not parse image data, similar to issues noted in prior research on UML diagrams (Cámara et al., 2023; Ouh et al., 2023). For events, the AIs generally understood event logic but made recurring mistakes in String and logical comparisons, often confusing variable values. In data streams, both ChatGPT and Copilot struggled with how methods like readInt and writeUTF interact, as well as with input file size. Both AIs also failed to account for some print statements in exception handling and incorrectly assumed that exceptions thrown in a catch block would be caught. Data structures were another weak area, with both always making errors on stack-related tasks, and ChatGPT having recurring problems with Queues, similar to the issues in test 2. The final exam question, requiring knowledge across all course topics, showed familiar errors—trouble with interfaces, abstract classes, and class hierarchies, along with incorrect use of access modifiers. Additionally, the AIs tended to propose only one solution when multiple were required. However, when their answers were correct, their explanations were solid, consistent with previous research where AI outperformed students in explaining answers (Malinka et al., 2023).

6 CONCLUSIONS

This paper provided an overview of ChatGPT and Copilot's proficiency in an introductory object-oriented programming course, comparing their performance to that of students. The findings offer valuable insights for instructors, highlighting how these AI tools stack up against students and identifying common mistakes made by the chatbots. This information can be informative for improvements to computer science courses and education.

A limitation of this study is its focus on just one course in a single year, due to the rise of AI chatbots being only a recent development. Future research across more courses and years would be beneficial. While this research primarily explored AI proficiency, another promising area of study is how AI tools can assist lecturers with teaching and grading automation. Additionally, as AI assistants evolve and improve through updates, research into how their proficiency changes over time would be an intriguing direction for further exploration.

ACKNOWLEDGEMENTS

This work was supported by the Estonian Research Council grant "Developing human-centric digital solutions" (TEM-TA120).

REFERENCES

- Bommarito, M., & Katz, D. M. (2022). *GPT takes the bar exam*. <https://doi.org/10.48550/ARXIV.2212.14402>
- Bordt, S., & Luxburg, U. (2023). *ChatGPT participates in a computer science exam*. <https://doi.org/10.48550/arXiv.2303.09461>
- Bucaioni, A., Ekedahl, H., Helander, V., & Nguyen, P. T. (2024). Programming with ChatGPT: How far can we go? *Machine Learning with Applications*, 15, 100526. <https://doi.org/10.1016/j.mlwa.2024.100526>
- Cámara, J., Troya, J., Burgueño, L., & Vallecillo, A. (2023). On the assessment of generative AI in modeling tasks: An experience report with ChatGPT and UML. *Software and Systems Modeling (SoSyM)*, 22(3), 781–793. <https://doi.org/10.1007/s10270-023-01105-5>
- Cipriano, B. P., & Alves, P. (2024). LLMs still can't avoid instanceof: An investigation into GPT-3.5, GPT-4 and Bard's capacity to handle object-oriented programming assignments. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training* (pp. 162–169). ACM. <https://doi.org/10.1145/3639474.3640052>

- Denny, P., Khosravi, H., Hellas, A., Leinonen, J., & Sarsa, S. (2023). Can we trust AI-generated educational content? Comparative analysis of human and AI-generated learning resources. <https://doi.org/10.48550/arXiv.2306.10509>
- 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. (2024). Computing education in the era of generative AI. *Communications of the ACM*, 67(2), 56–67. <https://doi.org/10.1145/3624720>
- Finnie-Ansley, J., Denny, P., Becker, B., Luxton-Reilly, A., & Prather, J. (2022). 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). <https://doi.org/10.1145/3511861.3511863>
- Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E., Prather, J., & Becker, B. (2023). 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). <https://doi.org/10.1145/3576123.3576134>
- Gabbrielli, M., & Martini, S. (2023). Object-oriented paradigm. In *Programming languages: Principles and paradigms* (2nd ed.). Springer Nature.
- Geerling, W., Mateer, G. D., Wooten, J., & Damodaran, N. (2023). ChatGPT has mastered the principles of economics: Now what? *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.4356034>
- Jukiewicz, M. (2024). The future of grading programming assignments in education: The role of ChatGPT in automating the assessment and feedback process. *Thinking Skills and Creativity*, 52. <https://doi.org/10.1016/j.tsc.2024.101522>
- Joshi, I., Budhiraja, R., Dev, H., Kadia, J., Ataullah, M., Mitra, S., Akolekar, H., & Kumar, D. (2024). ChatGPT in the classroom: An analysis of its strengths and weaknesses for solving undergraduate computer science questions. In *Proceedings of the Technical Symposium on Computer Science Education* (pp. 625–631). <https://doi.org/10.1145/3626252.3630803>
- Kadir, M., Rahman, T., Barman, S., & Al-Amin, M. (2024). Exploring the competency of ChatGPT in solving competitive programming challenges. *International Journal of Advanced Trends in Computer Science and Engineering*, 13(1), 13–23. <https://doi.org/10.30534/ijatcse/2024/031312024>
- Kiesler, N., Lohr, D., & Keuning, H. (2023). Exploring the potential of large language models to generate formative programming feedback. In *2023 IEEE Frontiers in Education Conference (FIE)* (pp. 1–5). <https://doi.org/10.1109/FIE58773.2023.10343457>
- Lee, H. (2023). The rise of ChatGPT: Exploring its potential in medical education. *Anatomical Sciences Education*. <https://doi.org/10.1002/ase.2270>
- Leping, V., Lepp, M., Niitsoo, M., Tõnisson, E., Vene, V., & Villem, A. (2009). Python prevails. In *Proceedings of the International Conference on Computer Systems and Technologies* (pp. IV.17-1 - IV.17-5).
- Lepp, M., & Tõnisson, E. (2015). Integrating flipped classroom approach and work in pairs into workshops in programming course. In *International Conference on Frontiers in Education: Computer Science and Computer Engineering* (pp. 220-226).
- Liu, Y., Le-Cong, T., Widyasari, R., Tantithamthavorn, C., Li, L., Le, X.-B. D., & Lo, D. (2024). Refining ChatGPT-generated code: Characterizing and mitigating code quality issues. *ACM Transactions on Software Engineering and Methodology*, 33(5), Article 116. <https://doi.org/10.1145/3643674>
- Malinka, K., Peresini, M., Firc, A., Hujnák, O., & Janus, F. (2023). On the educational impact of ChatGPT: Is artificial intelligence ready to obtain a university degree? In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education* (pp. 47–53). <https://doi.org/10.1145/3587102.3588827>
- Ouh, E. L., Gan, B. K. S., Jin Shim, K., & Wlodkowski, S. (2023). 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). <https://doi.org/10.1145/3587102.3588794>
- Richards, M., Waugh, K., Slaymaker, M., Petre, M., Woodthorpe, J., & Gooch, D. (2024). Bob or Bot: Exploring ChatGPT's answers to university computer science assessment. *ACM Transactions on Computing Education*, 24(5), 1–32. <https://doi.org/10.1145/3633287>
- Savelka, J., Agarwal, A., Bogart, C., Song, Y., & Sakr, M. (2023). 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). <https://doi.org/10.1145/3587102.3588792>
- Shoufan, A. (2023). Can students without prior knowledge use ChatGPT to answer test questions? An empirical study. *ACM Transactions on Computing Education*, 23(45), 1–29. <https://doi.org/10.1145/3628162>
- Sun, D., Boudouaia, A., Zhu, C., & Li, Y. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. *International Journal of Educational Technology in Higher Education*, 21(14). <https://doi.org/10.1186/s41239-024-00446-5>
- Wang, S., Ding, L., Shen, L., Luo, Y., Du, B., & Tao, D. (2024). OOP: Object-oriented programming evaluation benchmark for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024* (pp. 13619–13639). Bangkok, Thailand.
- Yilmaz, R., & Yilmaz, F. G. K. (2023). 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*, 4. <https://doi.org/10.1016/j.caeai.2023.100147>

APPENDIX

Table: AI assistants' results in exam questions.

	Topic	AI	Points	Mistakes
Q2	Objects, Classes	Chat-GPT	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	Chat-GPT	Avg=0.995 SD=0.74 min=0 max=2	Comparing substrings indexes Problems with uppercase and lowercase comparison Assumed that Collections.sort returned, not changed the 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 Problems with the String contains method
Q4	Interfaces	Chat-GPT	Avg=1.563 SD=0.37 min=1 max=2	Stated that abstract class needs to implement interface methods An empty method body is still an implementation of a method Used extends for interfaces Stated 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	Stated that abstract class needs to implement interface methods
Q5	Interfaces	Chat-GPT	Avg=1.4 SD=0.78 min=0 max=2	Used keyword class when methods are not implemented Stated that abstract cannot be used in interfaces Stated that the access modifier has to be specified 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	Chat-GPT	Avg=1.64 SD=0.39 min=1 max=2	Failed to realize a superclass's constructor with no arguments is always called when creating an instance of a subclass
		Copilot	Avg=1.904 SD=0.22 min=1.33 max=2	Failed to realize that when a subclass calls a superclass constructor with arguments, the constructor with no arguments is not called
Q7	Class hierarchy	Chat-GPT	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	
Q8	Abstract classes	Chat-GPT	Avg=1.516 SD=0.14 min=0.66 max=2	Stated that abstract classes cannot have realized methods Did not add the keyword abstract to abstract methods Stated 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	Stated that an abstract class needs to implement all superclass methods Used extends with interfaces Stated that you can override method only when superclass is abstract
Q9	Abstract classes	Chat-GPT	Avg=1.663 SD=0.26 min=1.33 max=2	Assumed that interfaces cannot contain variables Assumed that abstract classes cannot contain only non-abstract methods
		Copilot	Avg=1.826 SD=0.29 min=1.33 max=2	Stated that abstract classes cannot have realized methods Implemented methods in interfaces Assumed that interfaces cannot contain variables
Q10	Graphics			Could not be analyzed (the questions contained pictures)

	Topic	AI	Points	Mistakes
Q11	Events	Chat-GPT	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
Q12	Streams	Chat-GPT	Avg=1.799 SD=0.34 min=1.14 max=2	Did not understand 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	Chat-GPT	Avg=1.857 SD=0.12 min=1.75 max=2	Did not notice a print statement An exception thrown in a catch block was not caught
		Copilot	Avg=1.732 SD=0.62 min=0 max=2	
Q14	Data structures	Chat-GPT	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
Q15	Question with explanations	Chat-GPT	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 Assumed that method signatures must contain throws NumberFormatException Non-static methods cannot be called directly in a static context Did not mention creating subclass instances
		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 Stated that protected methods cannot be called from other classes Stated 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