Software Engineering WiSe 25/26

Assignment 0: OOP and Java

Hand-In via Moodle: Oct. 22, 10:00

1. Get started with Java

Java will be the programming language for the examples and programming exercises in this course. If you are not yet familiar with Java, there is plenty of online material available, and we do not have a particular book recommendation.

There are websites like (1) and (2) that help you get started. There is, of course, the official documentation (3). There are also online books, e.g, “Java ist auch eine Insel” (4) that also has exercises. You can find the book here in German (5). Perhaps the material of the “Intro to Programming" course from Princeton (6) might be interesting as well.

As a development environment, we recommend IntelliJ IDEA. You can either use the Community Edition or request a free Student version.

(1) https://dev.java/learn/getting-started/

(2) https://www.oracle.com/java/technologies/jdtt-jsp.html

(3) https://docs.oracle.com/en/java/javase/25/

(4) https://tutego.de/javabuch/index.html

(5) https://tutego.de/javabuch/Java-ist-auch-eine-Insel

(6) https://introcs.cs.princeton.edu/java/home/

2. Conversion Table

You have to implement a **Conversion Table for Celsius and Fahrenheit values**! A source code for the class 'CelsiusFahrenheit' is given below and must be *completed* and *adapted* at various points. Add the method that converts a given number in Celsius to Fahrenheit. Print the result to the console as shown below. The difficulty lies in writing a method for converting from Celsius to Fahrenheit and determining the correct scaling for displaying the values.

```java
import static java.lang.Math.round;
public class CelsiusFahrenheit {

    // Add missing method here

    public static void main(String[] args) {
        System.out.println("Celsius Fahrenheit Converter");
        System.out.println("============================");
        System.out.println("Celsius \t Fahrenheit");

        for (int c = 5; c < 20; c++) {

            System.out.println(c + "\t" + celsiusFahrenheit(c));
        }

    }
}
```

The following output is expected:

```
Celsius-Fahrenheit-Konverter
============================
Celsius     Fahrenheit
5           41
6           43
7           45
8           46
9           48
10          50
```

3. Car Racing Game

You are going to create a simple **Car Racing Game**. First, we create a class ‘RacingCar‘ with the following attributes:

- ‘racer‘ stores the name of the driver. This attribute must be a non-empty ‘string‘ and should be initialized when the object is instantiated.
- ‘speed‘ stores the speed of the car. This attribute can only contain non-negative integer values and must be **less than or equal to** a maximum speed.
- ‘pos‘ is an integer that specifies the position of the car and can only have non-negative values.

Each car also has the following additional attributes:

- ‘maxSpeed‘, which specifies the maximum speed the car can have. This attribute should be initialized when the object is instantiated.
- ‘finish‘, which stores the target distance that the car must travel. It should be set to -1 during initialization.

The class has the following methods:

- ‘start(initSpeed, finishDistance)‘: sets the speed of the car to an initial value, sets the target distance to the passed value and also sets the position of the car to 0.
- ‘race(acceleration)‘: takes an integer value for the acceleration; first adjusts the speed of the car and then updates the position of the car.
- ‘isFinished()‘: calculates a Boolean value (‘true‘ or ‘false‘) and indicates whether the car has reached the finish line.

Now your tasks:

1. Create a **Class Diagram** to match the textual description. Discuss with your team members.
2. **Implement** the class! Use the program code provided as a template. The ‘main‘ method must be executable without errors.

4. Savings Account

The task is to implement the transactions *withdraw money* and *deposit money* as well as *display account balance* in a program. If the account balance is below 0 Euro, a warning is to be issued.

1. **Design** and **implement** a class ‘Sparbuch‘ that provides the appropriate methods for these transactions. Make sure that your class ‘Sparbuch‘ can be integrated into the already provided class ‘Bankautomat‘. The ‘Bankautomat‘ class serves as the main program and provides an info screen for interaction with the user.
2. The class ‘Bankautomat‘ currently only asks for one option and then terminates the program. Can you extend the program so that it stays in the menu until the user selects the “Exit” option?
3. Currently, things can still go wrong if the user does not adhere to the expected input formats. Test the program with different inputs and try to implement the ‘Bankautomat‘ class more robustly.
4. When the program is terminated, the current account balance is lost. How could this be solved?

5. Hangman Game

Hangman is a guessing game for two players. The first player thinks of a word and the second player tries to guess it by consecutively suggesting letters with only a fixed amount of wrong guesses allowed.

Implementation: Implement hangman as a console application. Note the following key points:

- The first player types the secret word in the console.
- Scan the word and store it letter by letter in an array.
- The second player repeatedly guesses a letter by typing it in the console.
- Scan each letter and compare it to the letters in the array.
- Reveal the correct letters at their positions within the word.
- The game is over if all letters are correctly guessed or after 6 failed guesses.

Important: Check the template code provided in Moodle. The file 'src/test/java/HangmanGameTest.java' contains automated tests that run when you build the project with Gradle. Make sure of the following:

- If the player guesses the secret word, print the keyword "Congratulations!".
- If the player fails to guess the word, print the keyword "Game over!".

This allows the automatic tests to detect whether the player has won or lost.

Example run:

```
java

Current progress:
_ _ _ _
You have 6 wrong guesses left.
Guess a letter: j

Current progress:
j _ _ _
You have 6 wrong guesses left.
Guess a letter: w
Wrong guess!

Current progress:
j _ _ _
You have 5 wrong guesses left.
Guess a letter: a

Current progress:
j a _ a
You have 5 wrong guesses left.
Guess a letter: v
Congratulations! You have guessed the word: java
```

Deliverables

This assignment is meant for your preparation and does **not** provide any bonus points! However, we will provide feedback for submitted solutions.

Upload the following documents in Moodle by the deadline!

For Task 2, Task 3, Task 4 and Task 5 (Task 1 does not need any submission):

- One team member submits it on behalf of the team (Exception: Assignment 0 also can be submitted without a team!)
- Submit one PDF with the solution for all tasks (font size 12). For code, submit the zipped project in addition to the PDF.
- The PDF must start with a title page that mentions all group members their names, student ids, and email addresses. Invalid submissions are desk-rejected, i.e., zero points without further checking or resubmissions.
- Please use appropriate tools to create the solution (e.g., LibreOffice/Word for texts or draw.io for graphics). Scanned handwritten solutions will only be accepted if they are very legible.
- If you have any screenshots/pictures, make sure they are in high resolution.
- Provide well-formulated, but concise and specific answers to the tasks (no bulletpoints).

Any kind of plagiarism is prohibited, as well as the use of generative AI (e.g., ChatGPT) to create the final content of this hand-in (except for non-essential content, such as illustrative images for talks).