Middle East Technical University **Department of Computer Engineering<**

CENG 305

Object Oriented Programming with Java

Spring 2023-2024

Programming Assignment 1

Due Date: 25th March 2024, 08:00, via ODTUCLASS

1. Objectives

This assignment aims to make you familiar with basic concepts of Object Oriented Programming. You are expected to write java code for the following four problems.

2. Questions

1. **Textbook: P3.8**
   Implement a class `Student`. For the purpose of this exercise, a student has a name and a total quiz score. Supply an appropriate constructor and methods `addQuiz()`, `getTotalScore()`, and `getAverageScore()`. To compute the average,you also need to store the number of quizzes that the student took. Your implementation will be tested with different scenarios. Below is a sample test code:

```
Student student = new Student("Ali");
student.addQuiz();
student.addQuiz();
student.addQuiz();
student.getTotalScore();
student.getAverage();
```

2. **Textbook: P3.11**
   Implement a class `RoachPopulation` that simulates the growth of a roach population. The constructor takes the size of the initial roach population. The `breed` method simulates a period in which the roaches breed, which doubles their population. The `spray(double percent)` method simulates spraying with insecticide, which reduces the population by the given percentage. The `getRoaches` method returns the current number of roaches. Below is a sample test code:

```
RoachPopulation roach = new RoachPopulation(50);
roach.breed();
roach.breed();
roach.spray(25);
roach.breed();

roach.getRoach();
```

3. **Textbook: P3.13**

In this exercise, you are expected to implement a Java class named `BankAccount` that simulates a bank account with 7 methods as given in the class diagram. Include two constructor methods that construct a bank account with either a zero balance or a given balance. The bank will allow a fixed number of free transactions (deposits or withdrawals) every month and charge for transactions exceeding the free allotment. The charge is not levied immediately but at the end of the month. Both of the constructor methods must take the allowed number of free transactions as a parameter.

To modify the balance in the bank account, provide deposit and withdraw methods that modify the balance information in the `BankAccount` class. Additionally, implement `getBalance` and `setTransactionFee` methods to provide the current balance and set the fee for each transaction, respectively.

Lastly, supply a method, `deductMonthlyCharge`, in the `BankAccount` class that deducts the monthly charge and resets the transaction count. (Hint: Use `Math.max(actual transaction count, free transaction count)` in your computation.)

4. **Complete the Soccer Pitch - *SoccerPitch* Class**

The workers who came to paint the field of the football club you support have only painted the left side. Your task for this problem is to complete the painting symmetrically on the right side. Make the necessary changes in the provided code to complete the drawing. You can inspect Figure 1 to see the initial and resulting drawings.

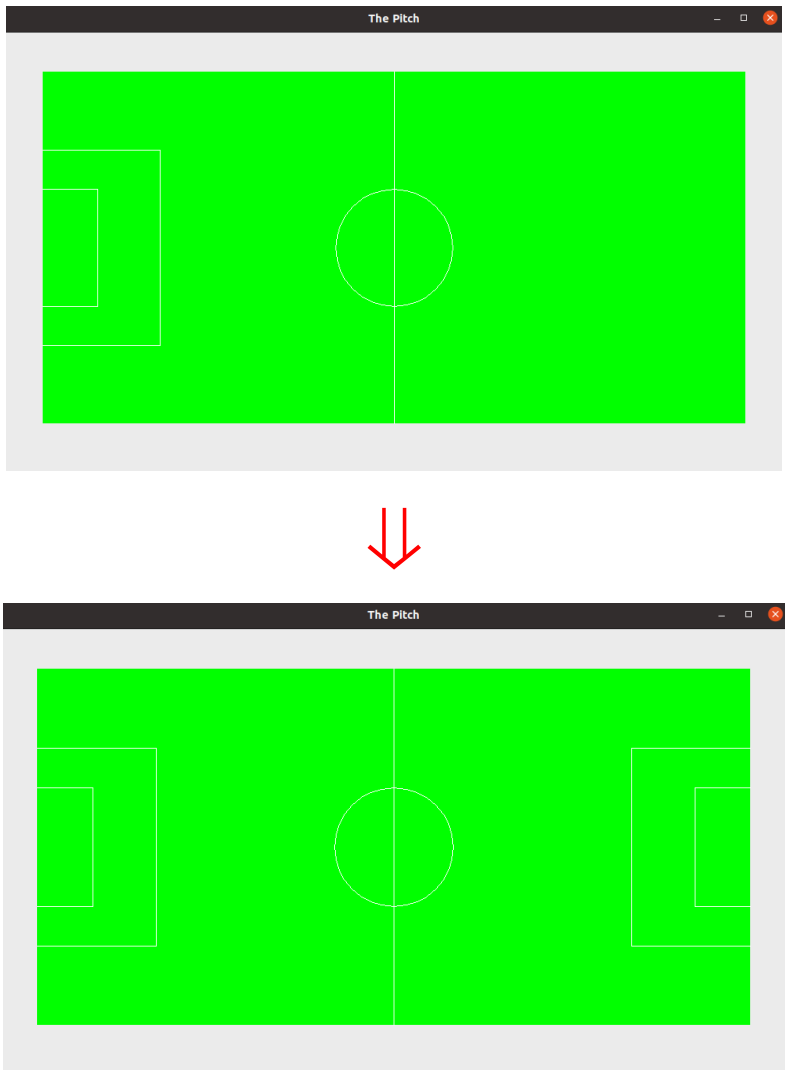

Figure 1: Initial and final drawings of the fourth problem.

3. Project Structure

You can see the final BlueJ project structure in Figure 2.

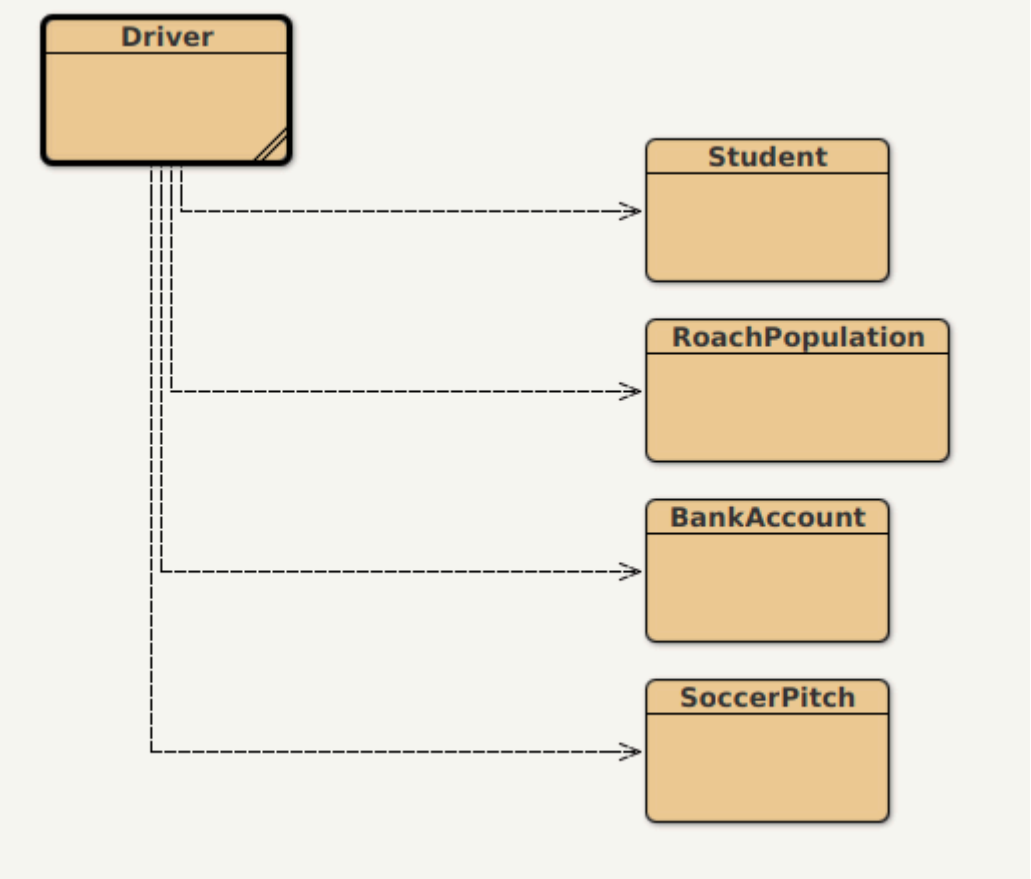

Figure 2: Project Structure

- You can run the project via running the main method of the `Driver` class.
- The context of `Driver` and initial `SoccerPitch` classes will be provided to you. You can edit the `Driver` class as you like for testing purposes. It will not be graded.
- You need to implement `Student`, `RoachPopulation` and `BankAccount` classes from scratch and modify the provided `SoccerPitch` class for your solution. Note that other files in your project will be **overwritten/deleted**. Therefore, make sure to contain your solution in these classes only.

4. Specifications

- **Programming Language:** Java. You should use **BlueJ IDE** to write and implement your solutions. You can download it from www.bluej.org
- **Using javadoc Utility:** You should generate an HTML documentation which explains your classes and their members (variables and methods) using *javadoc*. For reference, you can see **Chapter 3.2.4** of the textbook.
- **Usage of *this* keyword:** The reference variable *this* should be used in your code whenever it is applicable.
- **Code and UML Diagram Consistency:** The variables and the methods defined in the classes and their access modifiers should be consistent with the UML diagram provided in Figure 3.

The - and + symbols before the names denote private and public access to the members, respectively.

- **Provided Code Files:** For simplicity, the "*Driver*" class is provided in such a way that the main function will properly invoke the methods of the classes above in a loop in accordance with the menu shown in the sample output (Figure 4). Additionally initial codes for Question 4 are provided.
- **Code Clarity:** You should also consider the clarity of your code. Even if your code runs perfectly, obscure code will cause you to lose points.
- **Grading:** Your submission will be graded as follows:
    - Student class: 20 points.
    - RoachPopulation class: 20 points.
    - BankAccount class: 25 points.
    - SoccerPitch class: 15 points.
    - Documentation: 15 points.
    - Coding Style: 5 points.
- In the Driver class's main method, utilize the "javax.swing.JOptionPane" library to obtain inputs for all classes except the first question, as demonstrated in the figures below. Similarly, display all outputs of the classes using "javax.swing.JOptionPane".
- An example on the use of javax.swing.JOptionPane was provided in ODTUCLASS and here are the sample outputs shown below Figures 4 - 7.

5. Regulations

1. **Submission type:** You will submit a zip file named as **e1234567_ceng305_pa1.zip** which includes all of your BlueJ project files and generated javadoc files. e1234567 should be your student identification number. Submission will be done via **ODTUClass**. Only your last submission will be graded. If your submission fails to follow the specifications or does not compile, there will be a significant penalty in points.
2. **Late submission:** In case of late submission your score will be calculated as follows:

   SCORE-(5*day*day)
3. **Cheating: We have zero tolerance policy for cheating.** People involved in cheating will be punished according to the university regulations and will get 0 from the assignment. You can discuss algorithmic choices, but sharing code between students is strictly forbidden. Your code will be compared with those of your friends both semantically and visually. Please be aware that there are "very advanced tools" that detect if twho codes are similar.
4. **No grouping:** The assignment has to be done individually.
5. **Communication:** You can use the 'discussion forum' on ODTUClass for your questions and share your ideas. Check the 'news forum' for announcements regularly. Also, you can contact with 'atakan@ceng.metu.edu.tr' for your problems or questions.

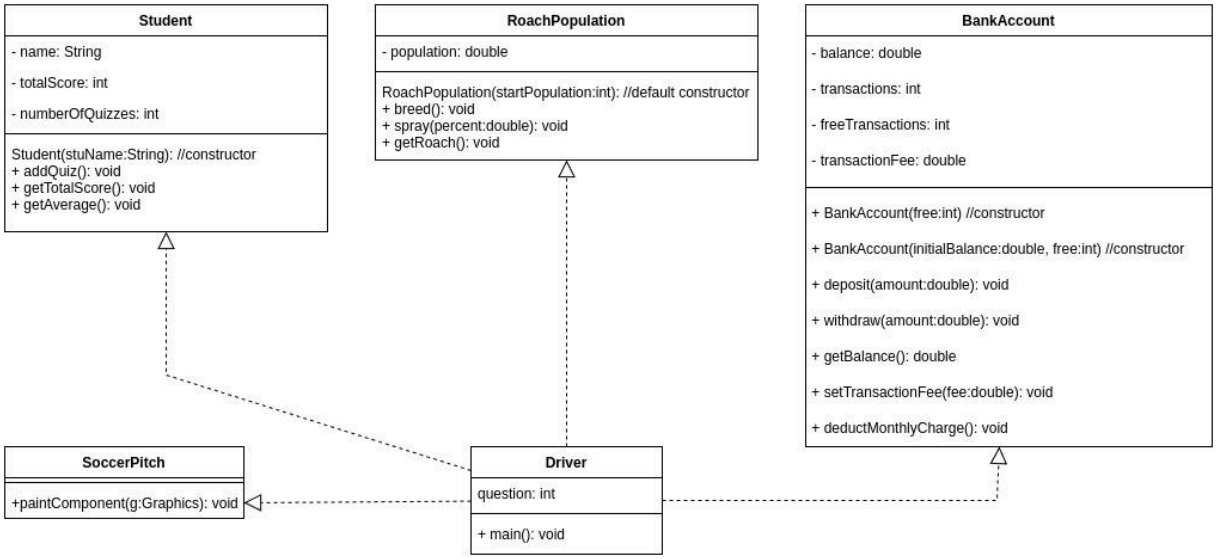

Figure 3: Class Diagram

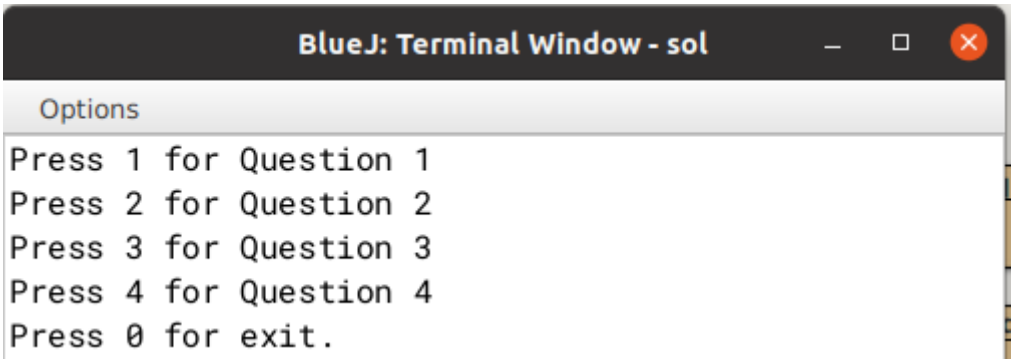

Figure 4: Sample Menu

```
Press 1 for Question 1
Press 2 for Question 2
Press 3 for Question 3
Press 4 for Question 4
Press 0 for exit.
1
Balance: 1071.75
Expected: 1071.75
Balance: 1091.75
Expected: 1091.75
Balance: 1120.75
Expected: 1120.75
```

Figure 5: Sample run of question 1

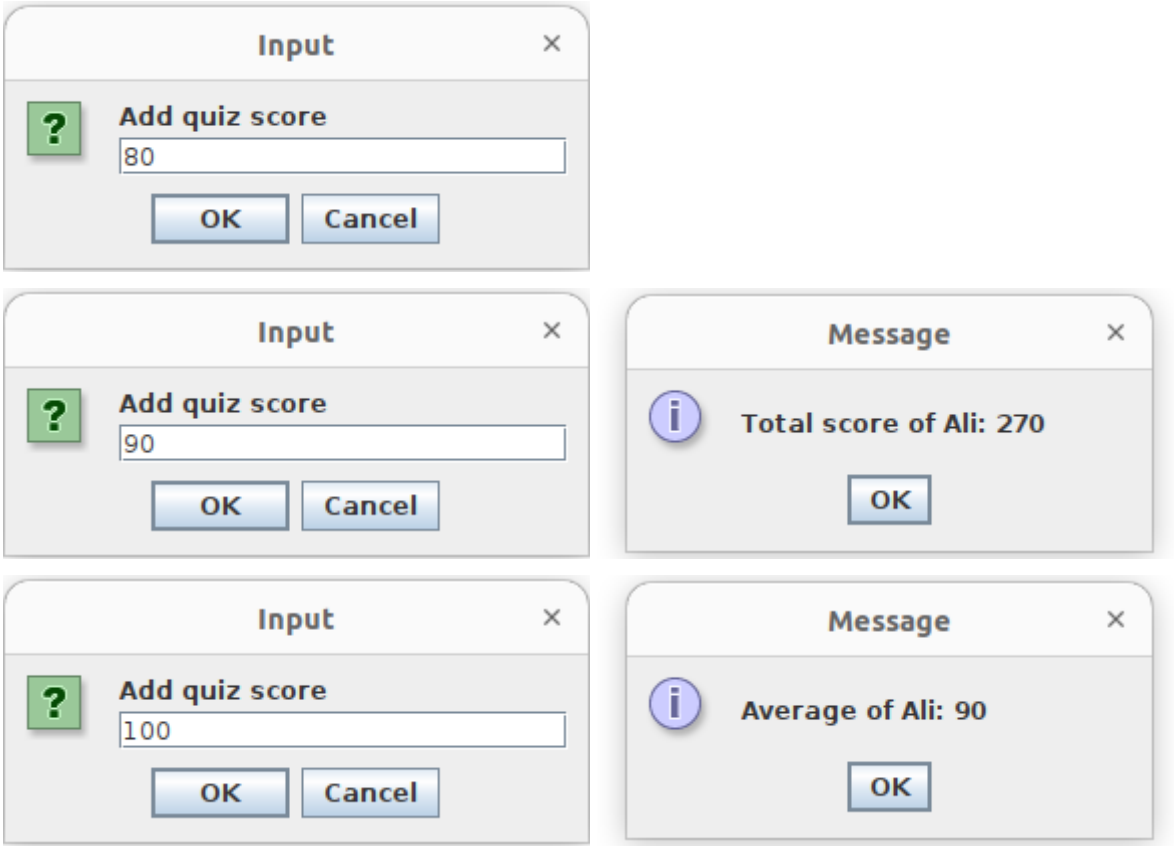

Figure 6: Sample run of question 2

Figure 7: Sample run of question 3