PROGRAMMING ASSIGNMENT 4
HUBBM-Dino: A Game with GUI in *Java*

Subject: **Simple Game Development with Java GUI framework (JavaFX)**
TAs: Ali Burak Erdogan

Release Date: 15/12/2022
Due Date: 05/01/2023 (until 23:59)

1 Introduction

In this assignment, you are expected to gain practice on developing a Graphical User Interface (GUI) application using Java programming Language. As opposed to the *command-line programs* where the interaction between the user and the computer often relies on string of text, a GUI program offers a much richer type of interface where the user uses a mouse and keyboard to interact with GUI components such as windows, menus, buttons, check boxes, text input boxes, scroll bars, and so on. The fact that most people today interact with their computers exclusively through GUI, a developing a GUI-based application has become a must for the new developers.

There are lots of frameworks to develop a GUI application (such as Swing, SWT, AWT, and the like). In this assignment, you are to employ `JavaFX` framework to complete this assignment. JavaFX is a software platform for creating and delivering desktop applications, as well as Rich Internet Applications (RIAs) that can run across a wide variety of devices. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS.

In this assignment, a similar version of *Dinosaur Game*, a hidden game on Google Chrome web browser[1], is expected from you to develop through JavaFX framework. All details that you need while designing the game is explained in the following section.

2 HUBBM-Dino

As expected, the goal in HUBBM-Dino game is to score points as much as possible, which depends on the number of obstacles (cactuses and dinosaur birds) that you have avoided —details regarding point calculation has been explained later. The initial scene of the game is given in the Fig. 1. As you see, the scene comprises of green ground and the dinosaur endlessly runs while avoiding cactuses and flying dinosaur birds by jumping over them. The scoreboard as well as level information is placed at the top-left corner of the scene. The title of the window is set with the name of the game that is 'HUBBM-Dino'. A short screen capture can be found in the announcement on *Piazza* and you are strictly advised to watch it at least once so that you better grasp how to design the game. The executable jar file has and will not be shared with you as there are lots of decompiler on Internet that converts jar to source code.

[1] `https://en.wikipedia.org/wiki/Dinosaur_Game`

You are to design a scene and feel free to design it with your imagination. We have provided in the template project the example images for the dinosaur, the birds (two images for making wing flapping animation), the ground, the big and middle sized cactuses. You can use them, or replace them with other images you like. However, the base scene of your implementation should be the one given in this paper. User should use up, right and left arrow keys to move the dinosaur. Up arrow key makes it jump; while left and right arrow keys enable horizontal move. You should animate all the objects of interest (dinosaur birds, big and middle size cactuses, and the dinosaur). When user releases keys, however, horizontal move should be stopped. When up arrow key is pressed, it should make our dinosaur jump with an initial vertical acceleration, and after that the dinosaur should gradually fall down with a negative vertical acceleration (simulating gravity).

Figure 1: A screenshot from the gameplay

The dinosaur is not allowed to leave the screen, so you should adjust your parameters so that the dinosaur cannot jump above the scene nor move horizontally out of the scene. Every time a bird or cactus leaves the scene, the score should be incremented by one. Also, every time the score reaches the multiples of 10, the level gets incremented by one. For example, score of 21 makes the game reach to level 3. Current level simply determines the animation speed —the higher the level is the faster the objects enters. However, the birds should appear only when level 2 is reached, so initially only cactuses should enter the game scene.

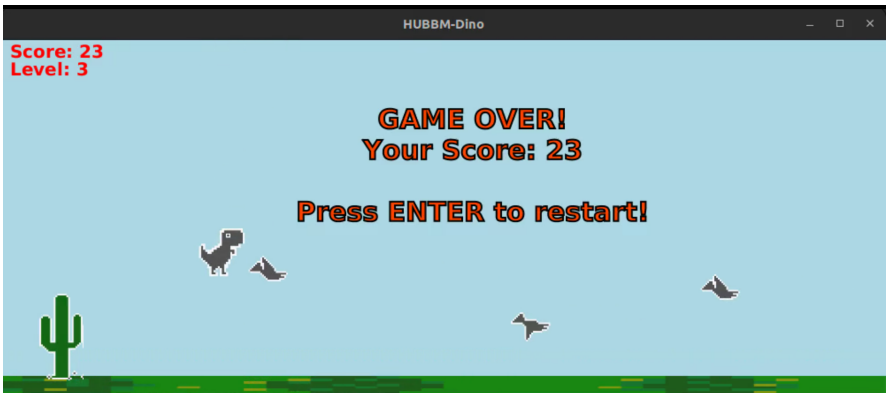

Figure 2: An example of a game-over scenario

You should always check if the dinosaur 'touches' on other objects (cactuses or birds) throughout the game. Such case is illustrated into the Fig. 2. It is very important to denote that there should be initially no crash among the dinosaur and between any object —see the figure at first page. So, keep that in your mind while generating the initial scene of the game. You should play a sound effect (choose anything you like) when a collision happens indicating the failure. Also, jumping and moving movements should have suitable sound effects.

Once the dinosaur has collided with an object, it is the end of the game, and a message text into the middle screen will be displayed. The message content should be same as the one in the capture. In addition, user will no more be allowed to control the game, namely all the key attempt should be ignored except `ENTER` button. Do mind to ignore `ENTER` button throughout the game. When user pressed it, user should be welcomed with the initial scene as given in the first page. Every time, user switches into the new game, level and score should be set as 1 and 0, respectively.

3 Grading Policy

Task	**Point**
`Object-oriented design and clean code`	15
`Graphical design`	15
`Animating objects`	15
`Keyboard control`	10
`Sound effects`	10
`Detecting collisions correctly`	10
`Score and level calculation`	5
`Animation speed with respect to level`	5
`Loading game with no crash`	5
`Enabling new game`	5
`Animating the birds' wing flapping correctly`	5
`Extra features in gameplay`	10 (bonus)

Note: The requirements for having full grade from *Object-oriented design and clean code* criteria:

- Using separate classes for different objects in the game like birds, dinosaur, cactuses. Also, using inheritance and polymorphism concepts properly.
- Avoiding putting the big part of code into a single function. Instead, breaking your code into separate functions and calling them when necessary.
- Avoiding putting all functionality into main class.
- Avoiding duplicate codes and following the DRY (Don't Repeat Yourself) principle.
- Following Java's naming conventions for variable names and methods.

4 Notes

- Accept the GitHub classroom assignment invitation from this link: https://classroom.github.com/a/1AyAmijm
- Design your implementation in Java 8 in which Lambda expressions are introduced, which considerably simplify the development. So, you are highly encouraged to make use of lambda expressions in this assignment.
- **Use Java 1.8 (a.k.a. Java 8).** Lower or higher versions of Java may cause errors in grading and losing points. Our grading environment works with Java 8.
- You should use IntelliJ IDEA for development environment. Please ensure that your code is well working.
- **Don't change file hierarchy of the template project.**
- **Don't change the names of classes which are provided to you**.
- The file hierarchy should match the following:

```
src/
   --GameObject.java
   --MyGame.java
   --*.java (You can add additional java classes)
   res/
      --bird1.png
      --bird2.png
      --cactus.png
      --cactus_big.png
      --platform.png
      --player.png
      --Any additional assets like images, sounds etc.
```

- The assignment must be original, individual work. All the duplicate or Internet works (even if a citation is provided) are both going to be considered as cheating.
- You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza under this link: https://piazza.com/hacettepe.edu.tr/fall2022/bbm104

5 Late Submission Policy

You may use up to 3 extension days for the assignment. But each extension day will bring about additional 10% degradation for evaluation of the assignment.