Lab3: Interface (Worksheet)

Section\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Important Rules

* Any cheating (copying code, answer, picture, etc.) cause the lab’s score to be “**0**”.
* Any screenshots must be from YOUR own program.
* If the captured screenshot (e.g., JUnit Test Case) and the result of your program are different, the lab score will be “**0**”.

# Part A: Problem Statement

1. Download the lab package and unzip it
2. Read and try to understand the lab sheet and run the given executable file.
   1. Capture the game screen of the GUI when you start the game.

All configuration parameters are stored in the variables in ConfigurableOpton.java. From the code below, answer Questions 2.b – 2.e.

// ConfigurableOption.java

**public** **class** ConfigurableOption {

**public** **static** **int** *screenWidth* = 640;

**public** **static** **int** *screenHeight* = 480;

**public** **static** **int** *timelimit* = 180; //Second

**public** **static** **int** *objectCreationMinDelay* = 100; //ms

**public** **static** **int** *objectCreationMaxDelay* = 200; //ms

**public** **static** **int** *objectMinDuration* = 200; //ms

**public** **static** **int** *objectMaxDuration* = 700; //ms

}

* 1. Which parameters are related to the screen size? (You must click the “Apply” button to see the change.)
  2. Which parameters are related to the duration of one object in the game?
  3. Which parameters are related to how often the new object will be created?
  4. Which parameters are related to the length (in seconds) for one game?

1. Click the “High Score” button and capture the game screen
2. Click the “New Game” button, press enter to pause the game, and, then, capture the screen.
3. Start playing the game according to the guide in the lab sheet.
   1. Explain about all of the shootable objects (color). Do you think that their life points are equal?
   2. Explain about all of the collectible objects? Is it easier to destroy the shootable objects using the special gun you collected than the simple gun? How many bullets do you have in the special gun?
4. Finish the game and capture the game screen. Now, you should understand the game logic

# Part B: A Preparation before Your Implementation

1. Add all of the given library file to the project, then, capture the screenshot of “Java Build Path”
2. In the given folder “resource (place contents in bin)”, copy the folder “res” and paste in the bin folder under the Eclipse project. What do you have in the “res” folder?
3. From the UML diagram, how many interface, abstract, and simple classes in the diagram? Explain how to represent them in the UML diagram.
4. Study the lab sheet, list all of the class in “lab3\_lib\_o.jar”
5. Start your implementation
   1. All classes you’ll need to implement are inside package “logic”.
   2. Code the class PlayerStatus and, then, run JUnit Test Case TestPlayerStatus. Next, capture the screenshot of the JUnit Test Case
   3. Code the class Gun and, then, run JUnit Test Case TestGun, Next, capture the screenshot of the JUnit Test Case
6. In the class TargetObject, do we need to implement all of the IRenderableObject’s methods? Please give an answer along with your reason.

# Part C: Result

1. Can you run your runnable jar? (Yes/ No) Capture the game screen (after clicking “New Game”).
2. Open the runnable jar file with the zip program, such as WinZip, WinRAR. Copy and paste the folder “res” into the runnable jar file. Now run your program and, then, capture the game screen again.
3. Start and pause the game, and, then, capture the game screen
4. Finish game, and, then, capture the game screen

# Part D: JUnit Test Case

1. Run TestTargetObject and, then, capture the result of Junit Test Case in Eclipse.
2. Run TestShootableObject and, then, capture the result of Junit Test Case in Eclipse.
3. Run TestTestSimpleTarget and, then, capture the result of Junit Test Case in Eclipse.
4. Run TestTestSmallTarget and, then, capture the result of Junit Test Case in Eclipse.
5. Run TestSplitterTarget and, then, capture the result of Junit Test Case in Eclipse.
6. Run TestColectibleObject and, then, capture the result of Junit Test Case in Eclipse.
7. Run TestItemSpecialGun and, then, capture the result of Junit Test Case in Eclipse.
8. Run TestItemBullet and, then, capture the result of Junit Test Case in Eclipse.
9. Run TestSpecialGun and, then, capture the result of Junit Test Case in Eclipse.