كلية الامير الحسين بن عبد الله الثاني لتكنولوجيا المعلومـــات
Faculty of Prince Al-Hussein Bin Abdullah II for Information Technology

الجـــــــــــامعة الهـــــــــــاشمية
The Hashemite University

Software Testing Course
Black-box Testing using JUnit
2021/2022 Summer Semester

Introduction

This project applies the use of JUnit 4 in a Java application using Eclipse IDE.

Prerequisites

- Same as the first assignment, if not already done, refer to the first assignment.
- In addition, you need to download the project jar file from Moodle.

Problem Description

An insurance company offers car insurance based on the following criteria:

(1) Young people, aged between 18 and 25 years, pay 5% of the insured car's price.
(2) People aged up to 75 years pay 2% of the insured car's price.
(3) Cars cheaper than 5k are not to be insured.
(4) Expensive cars are not insurable too, which cost more than 50k.
(5) Young people without a driving license can't insure a car.
(6) Senior people are not insured as well.

You are asked to test a class called **`Evaluator`** that applies the company policy. The class has a public method for calculating the car insurance amount, which has the following signature:

```
double evaluate(int age, int price) throws InvalidEvaluationException;
```

The method throws an exception of type **`InvalidEvaluationException`** when invalid inputs are passed to the method.

The jar file called **`Evaluator.jar`**, which contains classes **`Evaluator`** and **`InvalidEvaluationException`**, is posted on Moodle. To use the jar, add it to the **`classpath`** of your code and import package **`jo.edu.hu.bash`**.

Work to be done

You are asked to perform the following:

1. **(29 marks)** Derive test cases using **Equivalence partitioning/Boundary value analysis**.
2. **(25 marks)** Implement the test cases in **`JUnit 4`**. The test cases should be organized in two files:
   a. **`EvaluatorValidTest`**. This class contains the identified test cases with valid output.
   b. **`EvaluatorInvalidTest`**. This class contains the identified test cases with invalid output.
3. **(3 marks)** Implement a test suite class called **`EvaluatorTestSuite,`** which executes both the above two test classes when executed.
4. **(8 marks)** Implement a class that is executable from command line called **`Tester`** that takes a parameter to decide which test class to run:
   a. If the parameter value is **`1`**, **`Tester`** runs the **`EvaluatorValidTest`** class.
   b. If the parameter value is **`2`**, **`Tester`** runs the **`EvaluatorInvalidTest`** class.

   c. If the parameter value is `3`, `Tester` runs both classes.
   d. The `Tester` class should keep running until it receives 0, then it terminates.
   e. If no or incorrect value is given, class `Tester` continues showing a suitable error message asking the user to provide another correct value.
   f. Prepare a command-line statement that is used to execute the `Tester` class.
5. **(23 marks)** Write a test report that shows the results of running the test cases. It must contain the following:
   a. A table that contains; test case id, inputs, actual output, expected output, and comment.
   b. In the comment column, if the test **passes**, just write **"pass"**, if the test **fails**, write "**Fail**, …" and explain why. For example: Fail, the method should throw `InvalidEvaluationException` because the `age` was not valid.

What to submit

1. The generated complete test classes java files from step 2.
2. The generated complete test class java file from step 3.
3. The generated complete test class java file from step 4.
4. **(2 marks)** A project report file containing:
   a. The names and student ids of the group.
   b. The details of applying the test case design technique along with the designed test cases from **step 1**.
   c. The command-line statement from **step 4**.
   d. The test report described in **step 5**.

Notes

1. Submission using Moodle
   a. **One zipped file** to be submitted containing the above-described files.
   b. You might be asked to present your work to the instructor.
2. This project can be performed as a group of **3 students.**
   a. **For a group, only one submission** is required per group.
3. Some faults are seeded in the code! Your test cases need to find some or all these faults.
4. The test cases with invalid inputs pass if the `InvalidEvaluationException` exception is thrown. For these test cases, there is no need to write an assertion in the test method; just call the tested method `evaluate` inside the test method.
5. Write only one assertion in each test method (or one call to `evaluate` method in test cases with invalid inputs).