CS-140 — Spring 2017

Project 2

Objectives

The objective of this project is to learn how create a class on your own and how to test code with the unit testing framework `JUnit`.

Background

The project contains one class – `Project2Test`.

The `project2Test` class represents a test suite for a class `RoachPopulation.java` that you will be writing.

The `Project2Test` class is a test suite that is used to test some aspects of the `RoachPopulation` class. It does this by creating a RoachPopulation object and printing the results of calling some of their methods.

In class, we wrote a driver class for testing our `BankAccount` class. It relies on a human to read the output and to compare it to the expected output to determine if the code is working correctly. The problem with this testing method is that the human has to carefully check the output, and needs to do so every time the code is changed, compiled and run. What we would like to have is a more automated way of testing our code.

Unit testing provides a way of automatically testing individual classes and methods in an automated and repeatable way. By writing tests that test the behavior of classes and methods and comparing them automatically to expected values, we can let the computer run through our suite of tests every time we make a change to the code. This way we can be sure that any changes we made did not break some code that was already working. In many large software projects the test suite is run automatically every time a developer pushes new code to repository.

We will be using `JUnit`, a unit testing framework written specifically for Java. JUnit can be run on its own, or as a plug-in to integrated development environments like Eclipse. JUnit is already built in to BlueJ.

Objectives

This project provides you with the opportunity to put together nearly all of the material you have learned so far to create a single program. Most of what you are asked to do is very similar to what you did in the labs and in-class work, so look back at those labs and class examples for ideas.

You will be writing code for one class - `RoachPopulation`.

Procedure

In this programming project, you will write a simple program to do the following.

1. Implement a class `RoachPopulation` that simulates the growth of a roach population. The class has the following methods:

    1. The constructor takes the size of the initial roach population (an `int`).
    2. The `breed` method simulates a period in which the roaches breed, which doubles their population. This method has no arguments and no return value.
    3. The `spray` method simulates spraying with insecticide, which reduces the population by the given percentage. This method takes a percentage as a double value and has no return value
    4. The `getRoaches` method returns the current number of roaches.
    5. The `toString` method returns a String. For example, if the current number of roaches in the population is 50, this method will return "Number of roaches in the population: 50"
    6. **You need to write javadoc comments for all the methods in `RoachPopulation` class**. Please refer to your text book and BankAccount.java for more information

2. A program `Project2Test` is already provided for you. To use this code in your Bluej package, do the following:

    1. Create a folder Project2Code within Project2 folder
    2. Create empty RoachPopulation class.
    3. Create empty RoachPopulationTest class by doing the following steps. To turn on the testing tools in BlueJ:

        .

        1. Open `Preferences…`
        2. Click on the `Interfaces` tab
        3. Check the `Show Unit Testing Tools` check-box
        4. Right-click on the box for the `RoachPopulation` class you just created, and choose `Create Test Class`. A new box will appear (colored green) called `RoachPopulationTest`, and with the annotation `<<unit test>>`.
        5. Double-click the `RoachPopulationTest` class to view the source code.
        6. The `RoachPopulationTest` class imports some code from `org.junit`, and sets up three methods – a constructor, a `setUp` method and a `tearDown` method.

7. Replace the entire `RoachPopulationTest.java` with the contents of `Project2Test.java` file.

4. Compile your code and run the tests. This should pass because you have an empty class and an empty test class.
5. You should get in the habit of writing and testing a single method before moving on to the next.
6. Write your constructor for the `RoachPopulation` class, and then test it by uncommenting the `setup` method in `RoachPopulation`, compiling both classes, and running the tests. If your test passes, move on to the next method. If your test fails, make corrections to your constructor until the test passes.
7. Continue this process for each of the remaining methods and their associated tests:

    1. `getRoaches` and `test_getRoaches`
    2. `toString` and `test_toString()`
    3. `breed` and `test_breed`
    4. `spray` and `test_spray`
    5. `test_breedAndSpray`

Specifications

You need to submit your Project2 folder with the subfoler Project2Code with `RoachPopulation.java` and `RoachPopulationTest.java` files. A log file must address **what you learned in this project and the difficulties you faced**, time you spent on the project, web sources you referenced etc.

This is an Individual Assignment - No Partners

As this is a Project (and not a Lab) you will be working on your own, not with a partner. You should not be sharing your code with anyone else, other than the instructor.

You will need to fork your own private Project2 repository on GitLab for this project. The only person who should have any access to your repository is your instructor.

You can ask questions on Piazza about setting up your repository on GitLab, about using Git to send code to the instructor, and general questions about how to write your code. However you should not be posting sections of code and asking others to find your errors.

Deliverables

Be sure that you have your name and an explanation of what your program does in the Javadoc comments. Be sure that you have indented consistently.

You have to submit a discussion log along with the project. Please refer to DiscussionLogGuidelines repository or Lab2 for more information.

The instructor will pull your Project2 from your GitLab repository to grade it. Make sure:

1. You have pushed all changes to your shared repository. (I can't access local changes on your computer.)
2. You have added your instructor as Master to your shared GitLab repository.

Due Date/Time

Your project must be pushed to GitLab by Thursday 2/16 at 11:59pm.

Copyright and License