ITCS-2830 Applications Implementation & Testing

Project #6 – Creating JUnit Unit Tests & TDD

Objectives

Now that you have a solid grasp on **Eclipse, Java and JUnit**, this project will introduce you to writing ***your own Java class AND JUnit unit tests***. Remember, practice makes perfect!

Overview

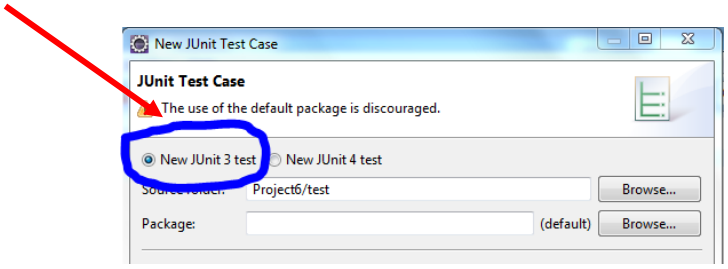
In this project you are going to create JUnit unit tests for a class you will write from scratch. *Don't worry; you don't need a Ph.D. in Java for this assignment.* I will give you several hints along the way. You will write the class and it is up to you to design solid JUnit **passing** tests.

Project 6

You are currently developing a system that requires a class to convert an integer to the name of the day of the week. For example, 0 would be "Sunday", 1 would be "Monday", and so on. Create a class called "**Week.java**" that will represent the class for days of the week. It is a simple class – it contains one method called **`getDayOfWeek()`** and takes an integer. For example, **`getDayOfWeek(0)`** would return "**`Sunday`**". How you design this class is up to you.

When you develop the unit tests, try your hand at TDD by creating the unit tests first, running the failing test, developing the class code, and then creating the passing test.

Your task is to write the source code (Week.java), and design, implement, and execute passing JUnit unit tests for the Week.java. Create a new project in Eclipse called "**Project 6**". Create a class called "Week.java" in your "**src"** directory. Create a new directory called "**test**" to hold your JUnit **VERSION 3** (not **VERSION 4**) unit tests. When you create the JUnit test, don't forget to check the box for Version 3. Let's call this test class "DayOfWeekTest.java".



A. Create the two .java files listed above.

B. Use the *appropriate assert* methods to test your class. Make sure you test for each day of the week. [Note: you may want to reference previous videos you watched for hints and tips on how to write those tests.]

C. Try misspelling a day of the week or entering an integer that is not in the switch statement; does the test fail? No need to take screenshots here – just practice making a test fail to see what happens.

D. In the Week.java file create a single method called **`getDayOfWeek()`** that returns a String (a week day) and takes a integer (use Sunday as day zero, Monday as day one, etc).

E. Use a switch statement to implement the day of the week values (i.e. "Sunday", "Monday", etc.). Switch statements are quite easy to understand and, you may already have had some experience with other languages that use case/switch statements. For more information on the Java switch statement please turn your attention to the following site by Oracle: http://goo.gl/FZUft

You are now complete with Project #6.

Deliverables

Locate the workspace for your project.

(1) Take a screenshot of your **PASSING** JUnit tests for each week day. (.png please)

(2) Take a screenshot of your **Week.java** file. (.png please)

(3) Take a screenshot of your **DayOfWeekTest.java** file. (.png please)

(4) Zip your entire **Project6**. A total of FOUR separate files (3 - .png screenshots, and 1 – zipped Eclipse project) (no .rar files please, only .zip)

Check List

1 - .png screenshot of Deliverable (1)…….☐

2 - .png screenshot of Deliverable (2)…….☐

3 - .png screenshot of Deliverable (3)…….☐

4 - .zip file Project6 Deliverable (4)…….☐

Upload your **FOUR** project *files* to the dropbox for **Project #6**l.

Please do not zip the screenshots with the zipped Eclipse project – you will have FOUR files to upload into the dropbox.