Department of
Computer Science and Engineering

Title: Software automation testing (JUnit)

Software Testing & Quality Assurance Lab CSE 434

Green University of Bangladesh

1 Objective(s)

- To be familiar with JUnit.
- To gain practical knowledge about JUnit.

2 Introduction

Unit Testing is used to verify a small chunk of code by creating a path, function or a method. The term “unit” exist earlier than the object-oriented era. It is basically a natural abstraction of an object oriented system i.e. a Java class or object (its instantiated form).

Unit Testing and its importance can be understood by below-mentioned points:

- Unit Testing is used to identify defects early in software development cycle.
- Unit Testing will compel to read our own code. i.e. a developer starts spending more time in reading than writing.
- Defects in the design of code affect the development system. A successful code breeds the confidence of developer.

3 JUnit

JUnit for Automation Testing Testing is the process of checking the functionality of an application to ensure it runs as per requirements. Unit testing comes into picture at the developers' level; it is the testing of single entity (class or method). Unit testing plays a critical role in helping a software company deliver quality products to its customers. Unit testing can be done in two ways − Manual Testing and Automated Testing. Software testing tools are required for the betterment of the application or software.

JUnit is a unit testing framework for Java programming language. It plays a crucial role test-driven develop ment, and is a family of unit testing frameworks collectively known as xUnit. JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. This approach is like "test a little, code a little, test a little, code a little." It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.

3.1 Features of JUnit

- JUnit is an open source framework, which is used for writing and running tests.
- Provides annotations to identify test methods.
- Provides assertions for testing expected results.
- Provides test runners for running tests.
- JUnit tests allow you to write codes faster, which increases quality.
- JUnit is elegantly simple. It is less complex and takes less time.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
- JUnit tests can be organized into test suites containing test cases and even other test suites.
- JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

3.2 What is a Unit Test Case ?

A Unit Test Case is a part of code, which ensures that another part of code (method) works as expected. To achieve the desired results quickly, a test framework is required. JUnit is a perfect unit test framework for Java programming language.

A formal written unit test case is characterized by a known input and an expected output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post-condition.

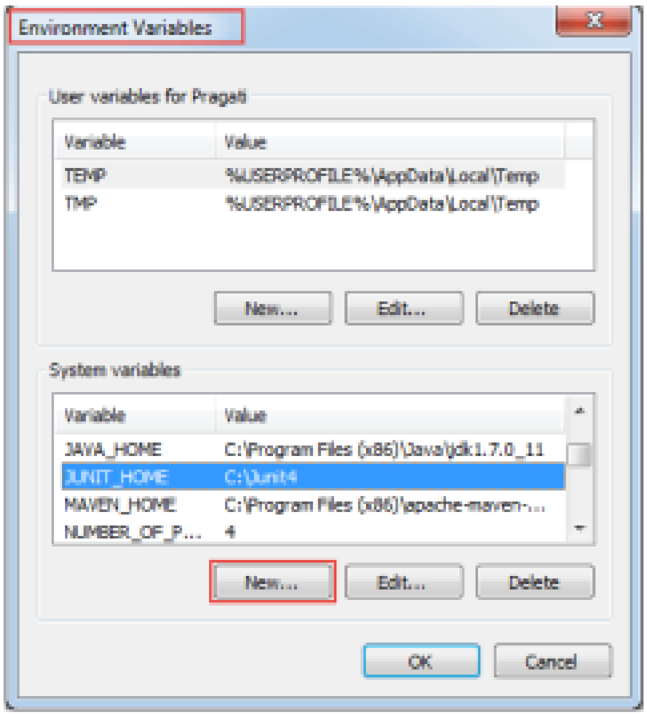
There must be at least two unit test cases for each requirement − one positive test and one negative test. If a requirement has sub-requirements, each sub-requirement must have at least two test cases as positive and negative.

3.3 JUnit - Environment Setup

Step 1) You need to set JUNIT_HOME environment variable to point out the base location where you have placed JUnit Jars.

For example, if you have created a JUnit folder in c: drive and placed jars there, then for environment settings you need to open control panel ->advanced ->environment variable.

1. Under environment window clicks on “new” button.



Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Variable	Value
JAVA_HOME	C:\Program Files (x86)\Java\jdk1.7.0_11
JUNIT_HOME	C:\Junit4
MAVEN_HOME	C:\Program Files (x86)\apache-maven-...
NUMBER_OF_P...	4

When you click on new button in environment variables, it will open another window

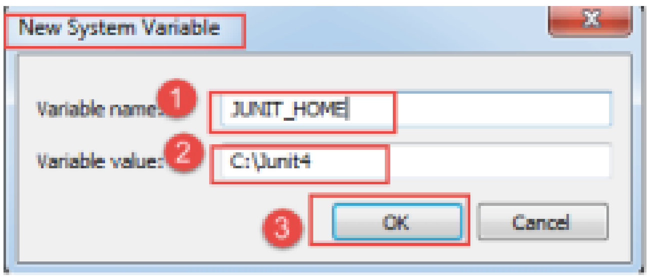
Step 2) A “New System Variable” window will open:

- Provide variable name as “JUNIT_HOME”.
- Provide JUnit value as JUnit path where you have copied JUnit jar files.
- Click on OK like figure ??

When you click on OK, it will create a new system variable with the given name and value. Which you can verify in environment variable window as shown in step 1 image.

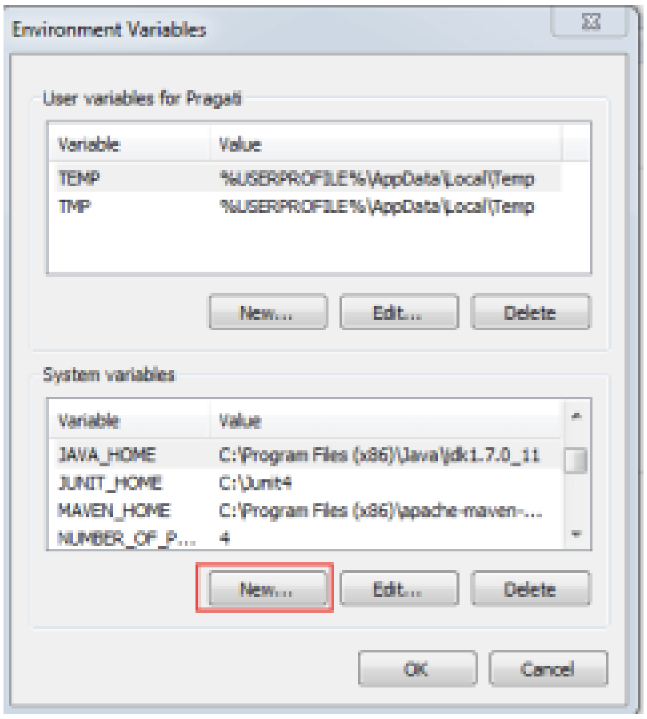
Step 3) After creating JUNIT_HOME, create another variable with the name CLASSPATH. Again go to Environment Variables and follow the below steps.

- Click on “new” button like image 1. When you click on new in environment variables, it will open another window.



Variable name: JUNIT_HOME

Variable value: C:\Junit4



Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Variable	Value
JAVA_HOME	C:\Program Files (x86)\Java\jdk1.7.0_11
JUNIT_HOME	C:\Junit4
MAVEN_HOME	C:\Program Files (x86)\apache-maven-...
NUMBER_OF_P...	4

Figure 1

Step 4) In this step, point out JUNIT_HOME to JUnit.jar which is placed in JUnit folder as given below: 1.

Variable Name: CLASSPATH

2. Variable Value: %CLASSPATH%;%JUNIT_HOME%*JUnit*4.10.*jar*; .;

3. Click on the OK button.

Step 5) Install JUnit jar file in eclipse:

1. Right click on project:

2. Click on "build path" and then

3. Click on "Configure build path".

4. Now click on "Add External JARs" button to add your downloaded JUnit.jar file with eclipse. 5.

After adding a JUnit.jar file, click on 'OK' button to close java build path window.

4 Implementation

Fair knowledge of SDLC, java programming, and basics of software testing process helps in understanding JUnit program.

Let's understand Unit Testing using a live example. We need to create a test class with a test method annotated with @Test as given below:

MyFirstClassTest.java

```
package batch183;

import static org.JUnit.Assert.*;
import org.JUnit.Test;
public class MyFirstClassTest
{

        @Test
        public void myFirstMethod()
        {
                String str= "JUnit is working fine";
                assertEquals("JUnit is working fine",str);
        }
}
```

TestRunner.java

To execute our test method (above) ,we need to create a test runner. In the test runner we have to add test class as a parameter in JUnitCore's runclasses() method . It will return the test result, based on whether the test is passed or failed.

For more details on this see the code below :

```
package batch183;

import org.JUnit.runner.JUnitCore;
import org.JUnit.runner.Result;
import org.JUnit.runner.notification.Failure;

public class TestRunner {
            public static void main(String[] args) {
                                                Result result = JUnitCore.runClasses(MyFirstClassTest.class);

            for (Failure failure : result.getFailures()) {
                        System.out.println(failure.toString());
            }
            System.out.println("Result=="+result.wasSuccessful());
      }
}
```

To run this example, right click on TestLogic class -> Run As -> 1Junit Test. Let's see the output displayed in Eclipse IDE.

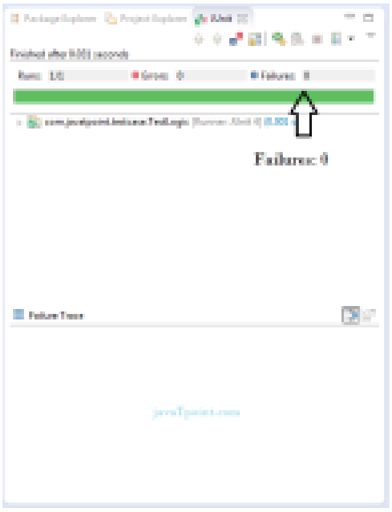
5 Output

Once TestRunner.java executes our test methods we get output as failed or passed. Please find below output explanation

1. In this example, after executing MyFirstClassTest.java , test is passed and result is in green.
2. If it would have failed it should have shown the result as Red and failure can be observed in failure trace. See below JUnit GUI:

6 Discussion & Conclusion

Based on the focused objective(s) to understand about the Selenium Webdriver, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).



7 Lab Task (Please implement yourself and show the output to the instructor)

1. Implement the test cases and evaluate performances for Balance Checking, Balance Transfer and Request Statement of a Banking ATM System for Automated Testing.

8 Lab Exercise (Submit as a report)

1. Generate test cases and evaluate performance for your given project using Automated testing.

9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.

10 Resources

https://www.guru99.com/junit-test-framework.html

https://www.tutorialspoint.com/junit/junit_environment_setup.htm