BIRLA INSTITIUTE OF TECHNOLOGY AND SCIENCE, PILANI
CS F213
LAB-4 [Unit Testing Using JUnit Framework]

AGENDA **DATE: 17/09/2019** **TIME: 02 Hours**

- Integrating Junit framework in Eclipse
- Assertions and their types
- Creating a sample class and testing its methods: Descriptive examples
- Exercise questions

What is a software test?

A software test is a piece of software, which executes another piece of software. It validates if that code results in the expected state or executes the expected sequence of events. The code which is tested is typically called the *code under test*.

Unit test:A unit test is a piece of code that executes a specific functionality in the code to be tested and asserts a certain behaviour or state.

The percentage of code which is tested by unit tests is typically called *test coverage*.A unit test targets a small unit of code, e.g., a method or a class.

JUnit: It is a Java library to perform unit testing. It uses **annotations** to identify methods that specify a test. A JUnit *test* is a method contained in a class which is only used for testing. This is called a *Test class*. To define that a certain method is a test method, annotate it with the **@Test** annotation.

The test method executes the code under test. An **assert** method, provided by JUnit framework, is used to check an expected result versus the actual result. These method calls are typically called asserts or assert statements.

General methodology: A unit test generally consists of various testing methods that each interact with the class under test in some specific way to make sure it works as expected.

Assertion	Description
void assertEquals([String message], expected value,actual value)	Asserts that two values are equal. Values might be type of int, short, long, byte, char or java.lang.Object. The first argument is an optional String message.
void assertTrue([String message], boolean condition)	Asserts that a condition is true.
void assertFalse([String message],boolean condition)	Asserts that a condition is false.
void assertNotNull([String message], java.lang.Object object)	Asserts that an object is not null.
void assertNull([String message], java.lang.Objectobject)	Asserts that an object is null.
void assertSame([String message], java.lang.Objectexpected, java.lang.Object	Asserts that the two objects refer to the same object.

actual)	
void assertNotSame([String message], java.lang.Object unexpected, java.lang.Object actual)	Asserts that the two objects do not refer to the same object.
void assertArrayEquals([String message], expectedArray, resultArray)	Asserts that the array expected and the resulted array are equal. The type of Array might be int, long, short, char, byte or java.lang.Object.

Example program 1:

A class named MyClass1.java with two integer variables and a multiply method:

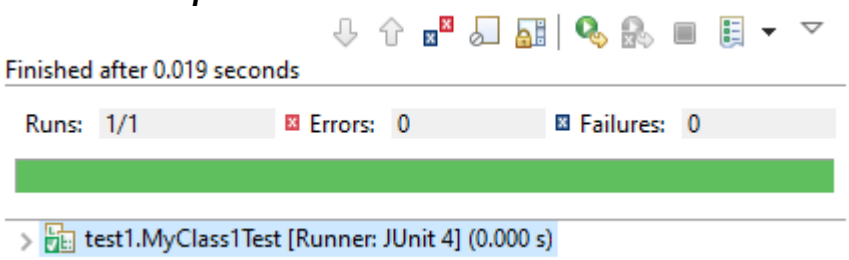
```
public class MyClass1 {
	int x, y;
	public int multiply(int x, int y) {
		return x*y;
	}
}
```

Test class of MyClass1.java:

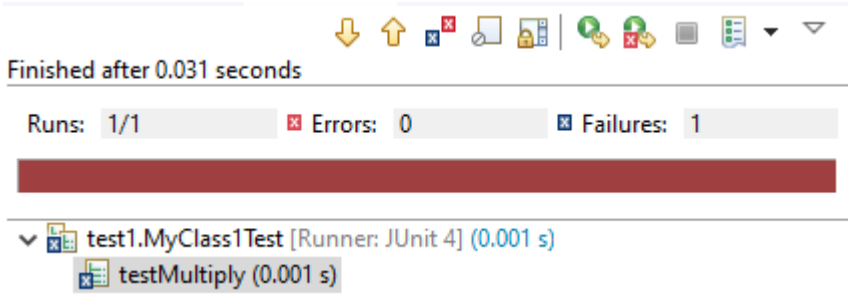
```
import static org.junit.Assert.*;
import org.junit.Test;

public class MyClass1Test {
	@Test
	public void testMultiply() {
		MyClass1 m1 = new MyClass1();
		assertEquals(6, m1.multiply(2, 3));/* Assertion */
	}
}
```

How to understand test success or failure in Eclipse?

1. By observing the red/green status bar in the left window.
   - ***Green for success***



   - ***Red for failure***

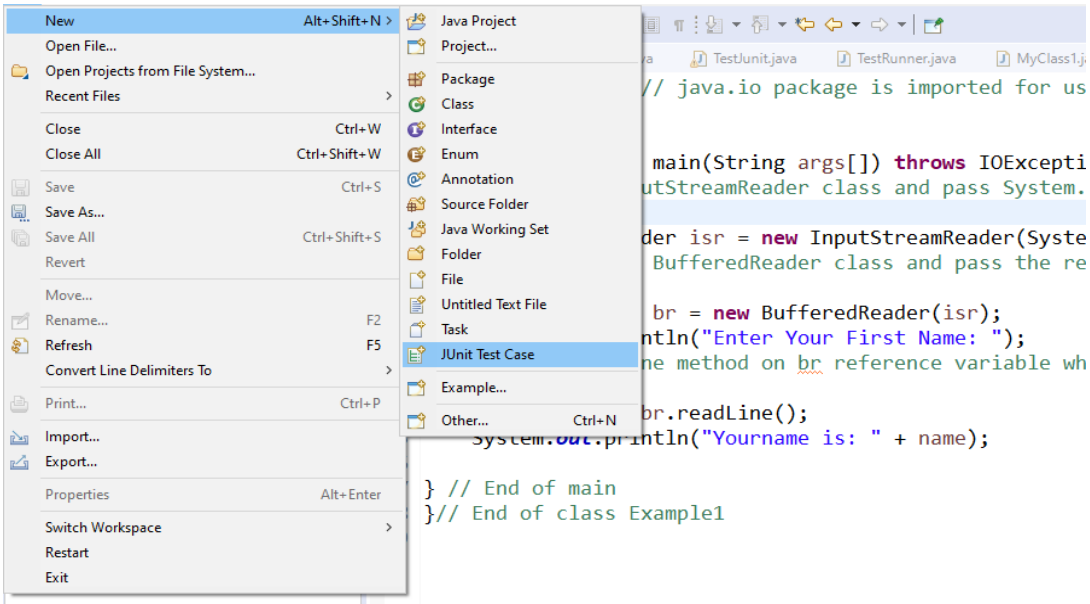


2. Using try and catch mechanisms to print the message

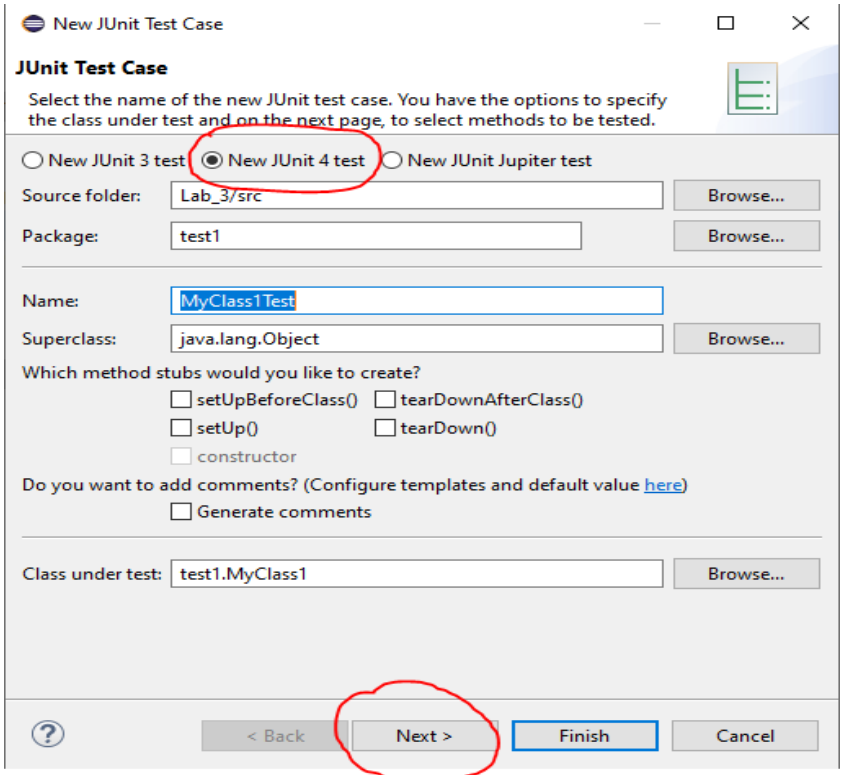
    - To be discussed later.

Steps to use JUnit in Eclipse:

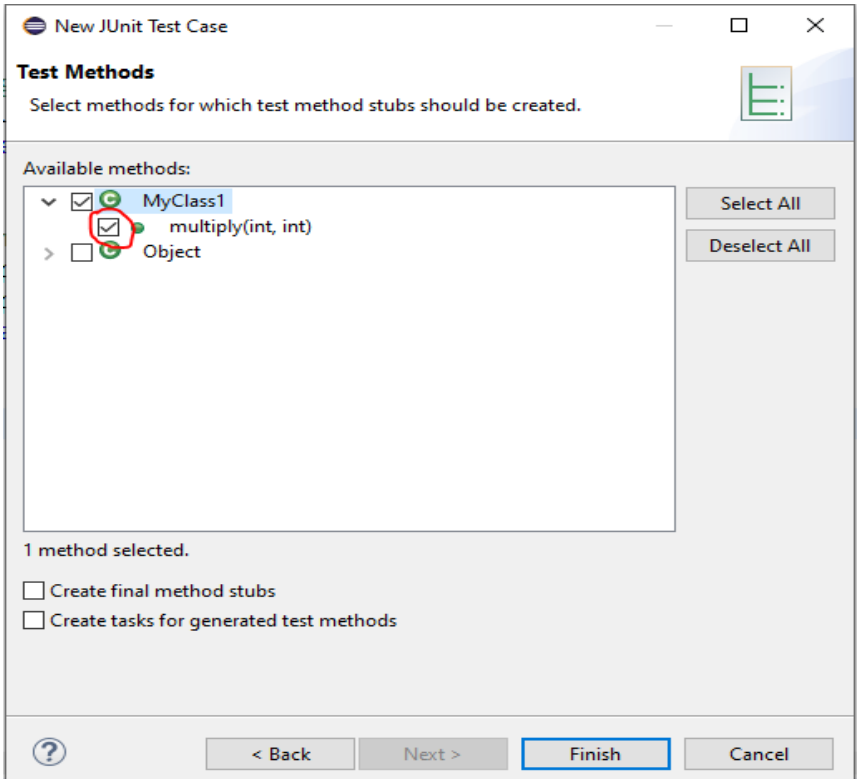
Step 1. Create MyClass1.java (mentioned above) class in your project.
Step 2. Now, create a JUnit Test Case of MyClass1.java by right clicking on your java class and selecting Junit Test Case.



a. A dialog box will pop up to help you create your test case. Make sure that the option at the top is set to JUnit 4. Click Next.



b. You will see a set of checkboxes to indicate which methods you want to test (right now only one method is displayed as per our program). Choose the method to test and click Finish.



Step 3. You will find the Junit test class for MyClass1.java as follows –

```
import static org.junit.Assert.*;
import org.junit.Test;
public class MyClass1Test {

	@Test
	public void testMultiply() {
		fail("Not yet implemented");
	}

}
```

Replace **fail("Not yet implemented")** with

```
MyClass1 m1 = new MyClass1();
assertEquals(6, m1.multiply(2, 3));
```

Step 4. Now, run your program and observe the success/failure status (Green/Red colour bar) on the left hand window.

Note: A more detailed description of using Junit with Eclipse can be found at

https://courses.cs.washington.edu/courses/cse143/11wi/eclipse-tutorial/junit.shtml

Example program 2: Create a class "Student". It should contain three variables *name* (String), *id_no* (int) and *marks* (float). Make suitable constructors, followed by getter and settermethods for class variables.Also create a user defined function

```
public class Student {

	/* create variables */
	/* create constructors */
	/* create getter and setter methods */

	/* User defined method to compare two students*/
	public booleanComparison(Student x, Student y) {
		if(x.get_marks()>y.get_marks())
			return true;
		else
			return false;
	}
}
```

Now, test the "Student" class using StudentTest as follows (create the test class in Eclipse using the same way as mentioned in the previous example) -

```
import static org.junit.Assert.*;
import org.junit.Test;
public class StudentTest {

	@Test
	public void testStudentStringIntFloat() {
		Student s1 = new Student("George", 1, 90.15f);
		Student s2 = new Student("John", 2, 75.81f);
		Student s3 = new Student("Alice", 3, 95.26f);

		/* Test the getter functions */
		assertEquals("George", s1.get_name());
		assertEquals(1, s1.get_id_no(), 0.001);
		assertEquals(90.15f, s1.get_marks(), 0.001);

		/* Test the setter function */
		float marks=96.38f;
		s1.set_marks(marks);
		assertEquals(marks, s1.get_marks(), 0.001);

		/* Compare two students */
		/* for method based comparison */
		assertEquals(true, s1.Higher(s1, s2));

		/* for direct comparison */
		// assertEquals(s1.get_marks(), s2.get_marks(), 0.001);
	}
}
```

Once finished, remove the last comment and observe the output.

Example program 3 (using Junit with arrays):

Create a java class to be tested FirstDayAtSchool.java as below:

```
import java.util.Arrays;
public class FirstDayAtSchool {

	public String[] prepareMyBag() {
		String[] schoolbag = {"Books", "Notebooks", "Pens"};
		System.out.println("My school bag contains: " +
		Arrays.toString(schoolbag));
		return schoolbag;
	}

	publicString[] addPencils() {
	String[] schoolbag = {"Books", "Notebooks", "Pens", "Pencils"};
	System.out.println("Now my school bag contains: "
				+ Arrays.toString(schoolbag));
	return schoolbag;
	}
}
```

Now, create a JUnit test case for FirstDayAtSchool as –

```
import static org.junit.Assert.*;
import org.junit.Test;

public class FirstDayAtSchoolTest {

	FirstDayAtSchool school = new FirstDayAtSchool();
	String[] bag1 = {"Books", "Notebooks", "Pens" };
	String[] bag2 = {"Books", "Notebooks", "Pens", "Pencils" };

	@Test
	public void testPrepareMyBag() {
		System.out.println("Inside testPrepareMyBag()");
		assertArrayEquals(bag1, school.prepareMyBag());
	}

	@Test
	public void testAddPencils() {
		System.out.println("Inside testAddPencils()");
		assertArrayEquals(bag2, school.addPencils());
	}
}
```

Run the test case by right-clicking on the test class and select Run As -> JUnit Test and observe the status as Green/Red bar.

Exercise questions

1. Below is the code for a simple test of the Card class.

```
import org.junit.*;
import static org.junit.Assert.*;
public class CardTests {

	/* Test the Card constructor */
	@Test
	public void cardConstructor(){
		c1 = new Card("ace", "hearts", 1);
		c2 = new Card("ace", "hearts", 1);
		c3 = new Card("ace", "hearts", 2);
		assertEquals(c1.rank(), c2.rank());
		assertEquals(c1.suit(), c2.suit());
	}
}
```

a. Based on your understanding from the above code (CardTests), create the Card class with its variables, constructors and methods, and test them using CardTests.

b. Write a second test that tests the matches method. Here is an example of the Junit assertion you should use:

```
assertTrue(c1.matches(c2));
```

2. Below is the code a class named Calculator. It performs three functions: (i) finds the maximum number from a given array of integers, (2) finds cube of a given number and (3) reverses a given word. It uses a different method for each operation.

```
import java.util.StringTokenizer;
public class Calculator {

      //method that returns maximum number
      public static intfindMax(intarr[]){
            int max=-100;
            for(inti=0;i<arr.length;i++){
                  if(max<arr[i])
                  max=arr[i];
            }
            return max;
      }

      //method that returns cube of the given number
      public static intcube(int n){
            return n*n*n;
      }

      //method that returns reverse words
      public static String reverseWord(String str){

            StringBuilder result = new StringBuilder();
            StringTokenizer tokenizer =
                        new StringTokenizer(str," ");

            while(tokenizer.hasMoreTokens()){
                  StringBuildersb=new StringBuilder();
                  sb.append(tokenizer.nextToken());
                  sb.reverse();
                  result.append(sb);
            }

            return result.toString();
          }
      }
```

Create a Test class for Calculator.java and test its methods **findMax,cube** and **reverseWord.**