SWE 434
Software Testing and Validation

Lecture slides are available on following website through your university student login
https://lms.ksu.edu.sa/

Raja Majid Mehmood
rmehmood@ksu.edu.sa
Department of Software Engineering,
King Saud University, Riyadh, Saudi Arabia.

Some of the material in these slides is derived from slides produced by Dr. M. Shamim Hossain, Prof. Some, Alan of U Ottawa, Thanks to them

Lecture's Agenda

- Assertions (JUnit Assert Class)
- Exceptions Handling in *Junit Test Class*
- Practical Example

Assertions

- An *assertion* is a statement in the Java™ programming language that enables you to test your assumptions about your program.
- Assertions are defined in the JUnit class **`Assert`**
  - **If an assertion is true, the method continues the further execution.**
  - **If any assertion is false, then**
    - the method stops the execution at that point, and
    - the result for the test case will be fail.
  - **If any other *exception* is thrown during the test method, then**
    - the result for the test case will be error.
  - **If no assertions were violated for the entire method, then**
    - the test case will pass.
- All assertion methods are static methods

Assertions

MyMath.java | MyMathTest.java

```
1  package lab1;
2
3  import static org.junit.Assert.*;
4  import org.junit.Test;
5
6  public class MyMathTest {
7
8      MyMath mathObject=new MyMath();
9
10     @Test
11     public void testDiv() {
12         int actual=mathObject.div(6, 2);
13         int expected=3;
14         assertEquals(expected , actual);
15     }
16 }
```

Assert Class

- static import

assertEquals()

- method from Assert

Common Methods of Assert Class

Methods	Description
fail([String Message])	This method is used to fail test case with given message, e.g; **fail**("*Test case is not implemented yet*").
assertFalse([String Message], boolean)	Boolean value must be *false* to **pass** the test case otherwise test should be **fail** with optional message.
assertTrue([String Message], boolean)	Boolean value must be *true* to **pass** the test case otherwise test should be **fail** with optional message.
assertsEquals([String Message], expected, actual)	Both expected and actual object values must be same to **pass** the test.
assertNull([String Message], object)	Object must be *null* to **pass** the test.
assertNotNull([String], object)	Object should not be *null* to **pass** the test.
assertArrayEquals([String Message], expected, actual)	Both arrays must be same to **pass** the test. Message is optional here.

assertArrayEquals()

	assertArrayEquals(byte[] expecteds, byte[] actuals) Asserts that two byte arrays are equal.
	assertArrayEquals(char[] expecteds, char[] actuals) Asserts that two char arrays are equal.
	assertArrayEquals(int[] expecteds, int[] actuals) Asserts that two int arrays are equal.
	assertArrayEquals(long[] expecteds, long[] actuals) Asserts that two long arrays are equal.
	assertArrayEquals(java.lang.Object[] expecteds, java.lang.Object[] actuals) Asserts that two object arrays are equal.
	assertArrayEquals(short[] expecteds, short[] actuals) Asserts that two short arrays are equal.

assertArrayEquals()

assertArrayEquals(java.lang.String message, byte[] expecteds, byte[] actuals) Asserts that two byte arrays are equal.
assertArrayEquals(java.lang.String message, char[] expecteds, char[] actuals) Asserts that two char arrays are equal.
assertArrayEquals(java.lang.String message, int[] expecteds, int[] actuals) Asserts that two int arrays are equal.
assertArrayEquals(java.lang.String message, long[] expecteds, long[] actuals) Asserts that two long arrays are equal.
assertArrayEquals(java.lang.String message, java.lang.Object[] expecteds, java.lang.Object[] actuals) Asserts that two object arrays are equal.
assertArrayEquals(java.lang.String message, short[] expecteds, short[] actuals) Asserts that two short arrays are equal.

assertEquals()

- assertEquals(double expected, double actual, double delta)
  Asserts that two doubles or floats are equal to within a positive delta.
- assertEquals(long expected, long actual)
  Asserts that two longs are equal.
- assertEquals(java.lang.Object expected, java.lang.Object actual)
  Asserts that two objects are equal.
- assertEquals(java.lang.String message, double expected, double actual, double delta)
  Asserts that two doubles or floats are equal to within a positive delta.
- assertEquals(java.lang.String message, long expected, long actual)
  Asserts that two longs are equal.
- assertEquals(java.lang.String message, java.lang.Object expected, java.lang.Object actual)
  Asserts that two objects are equal.

Common Methods of Assert Class

assertFalse(boolean condition) Asserts that a condition is false.
assertFalse(java.lang.String message, boolean condition) Asserts that a condition is false.
assertNotNull(java.lang.Object object) Asserts that an object isn't null.
assertNotNull(java.lang.String message, java.lang.Object object) Asserts that an object isn't null.
assertNull(java.lang.Object object) Asserts that an object is null.
assertNull(java.lang.String message, java.lang.Object object) Asserts that an object is null.
assertTrue(boolean condition) Asserts that a condition is true.
assertTrue(java.lang.String message, boolean condition) Asserts that a condition is true.
fail() Fails a test with no message.
fail(java.lang.String message) Fails a test with the given message.

Practical Lab (Example-2)

```
1  package calculator;
2
3  public class MyMath
4  {
5      /**
6       * perform division of two integer numbers
7       *
8       * @param a, must be integer
9       * @param b, must be integer
10      * @return, division result of a and b
11     */
12     public int div(int a, int b)
13     {
14         if (b==0)
15             throw new IllegalArgumentException( "'b' must be greater than 0" );
16         else {
17             int result = a / b;
18             return result;
19         }
20     }
21 }
```

Application Class (SUT)

Practical Lab (Example-2)

- Problem Description:

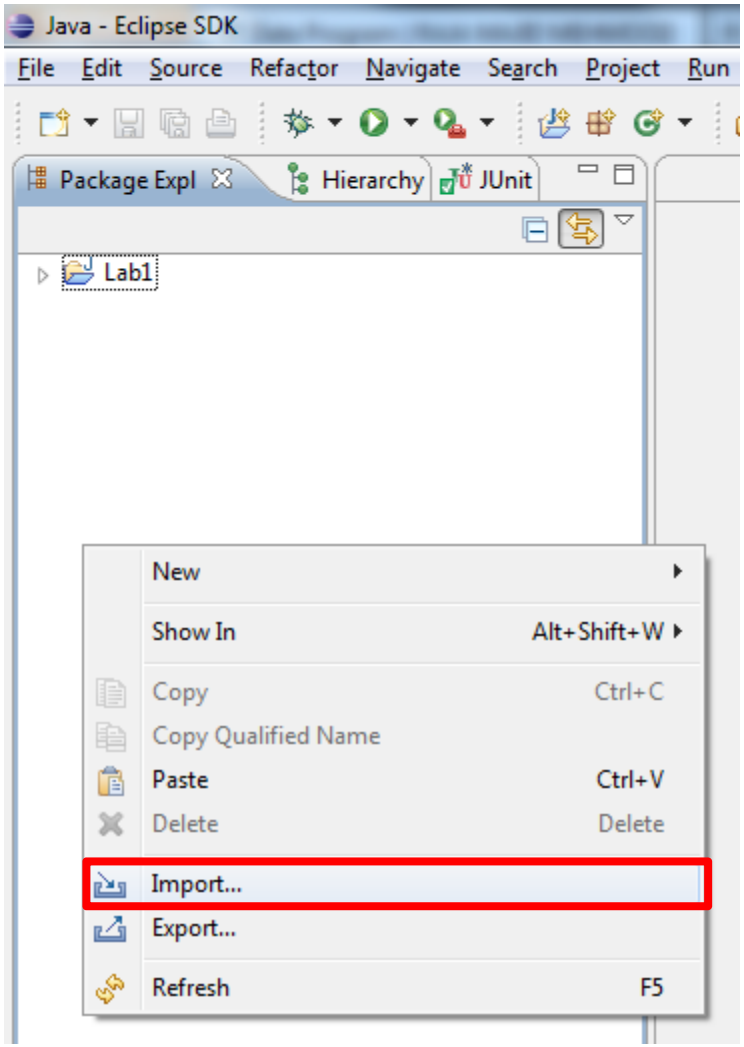
1. How to test all possible cases of MyMath.div(*int*, *int*)?
   1. TC-1: 0 >= a > 0 **AND** 0 > b > 0
   2. TC-2: 0 >= a > 0 **AND** b == 0

TEST CASE	INPUT		EXPECTED OUTPUT
	INTEGER-A	INTEGER-B	INTEGER-A/INTEGER-B = ?
TC-1	6	2	3
TC-2	5	0	IllegalArgumentException

- **Objective:**
  - Importing the java project
  - Implementing the test case method from above **Test Case Table**

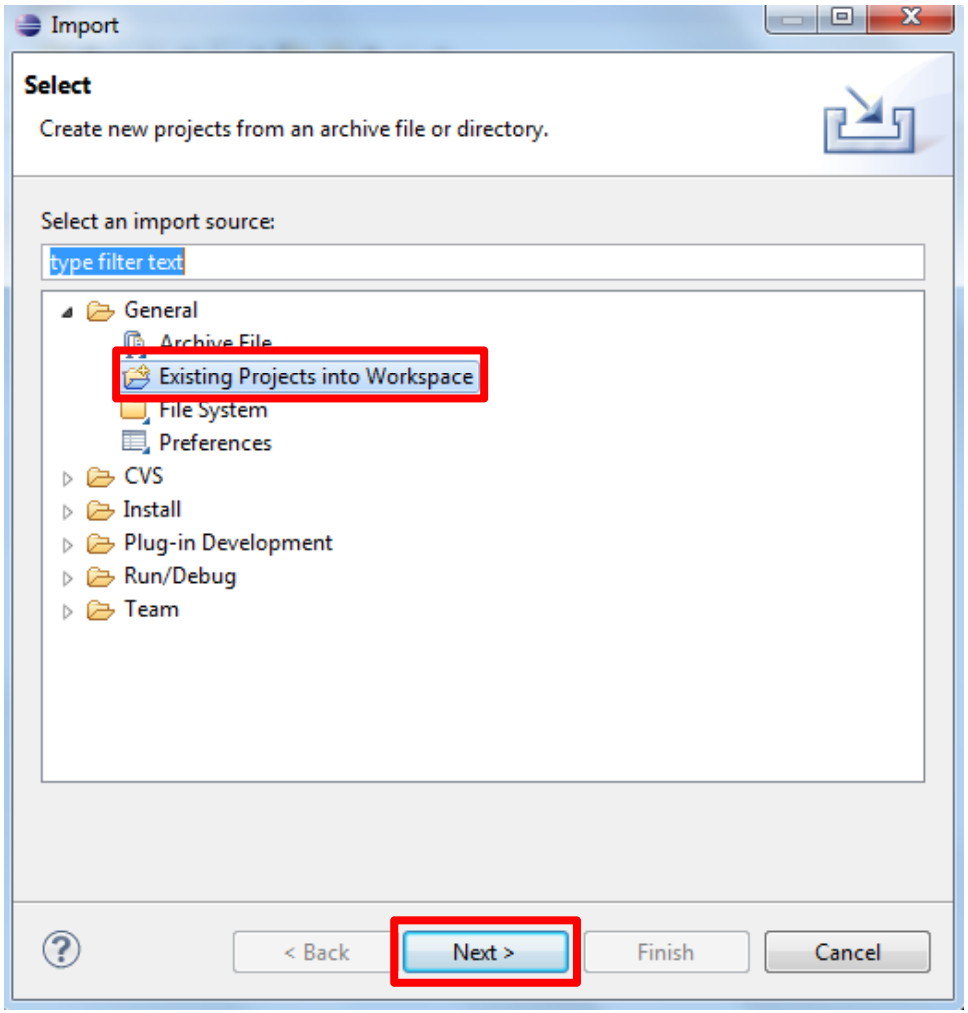
Importing the Java Project

- Right click on package explorer
  - Click on **import**



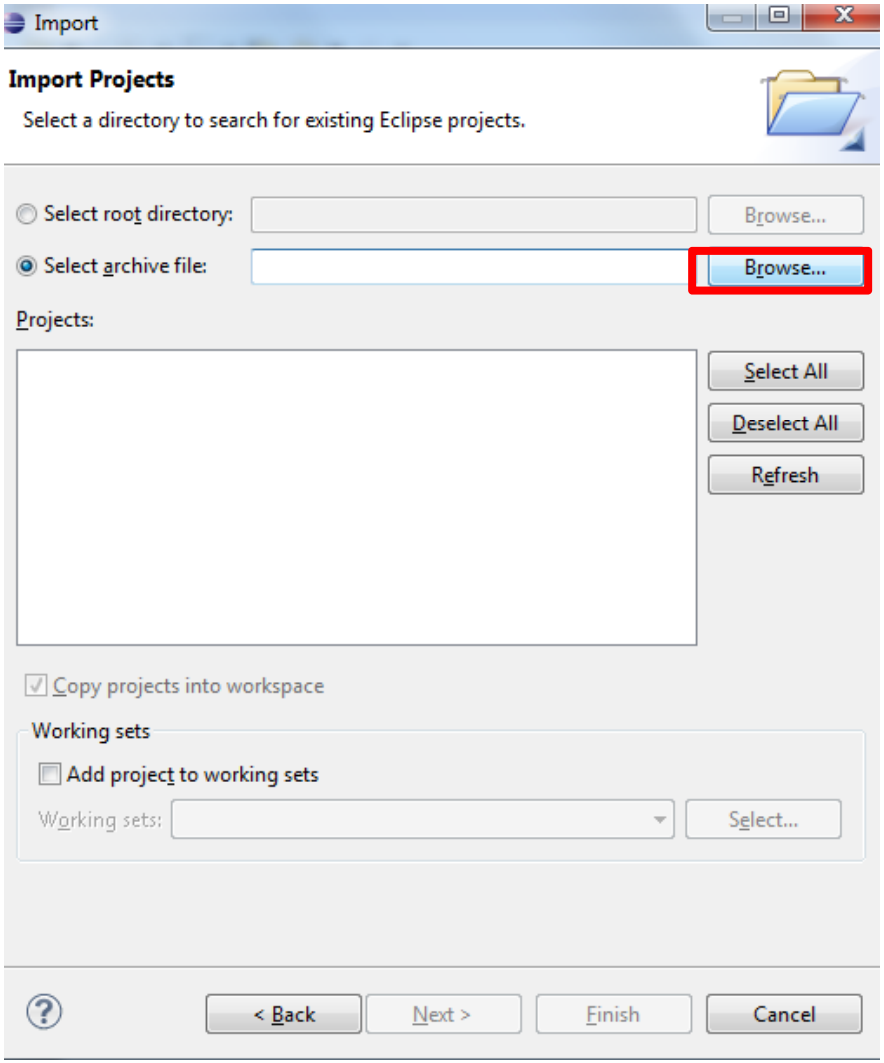
Importing the Java Project

- Choose **Existing Projects…**
  - Click on **Next**



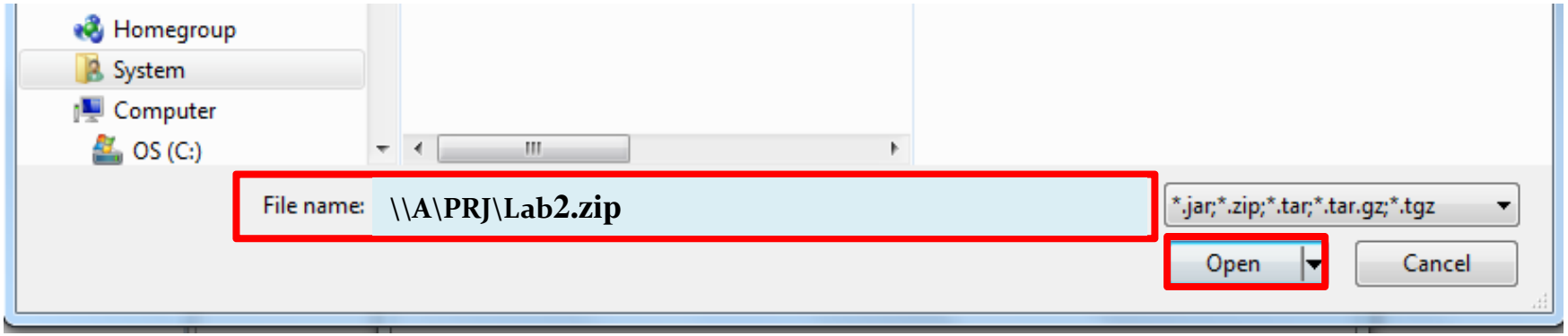
Importing the Java Project

- Choose **Browse**



Importing the Java Project

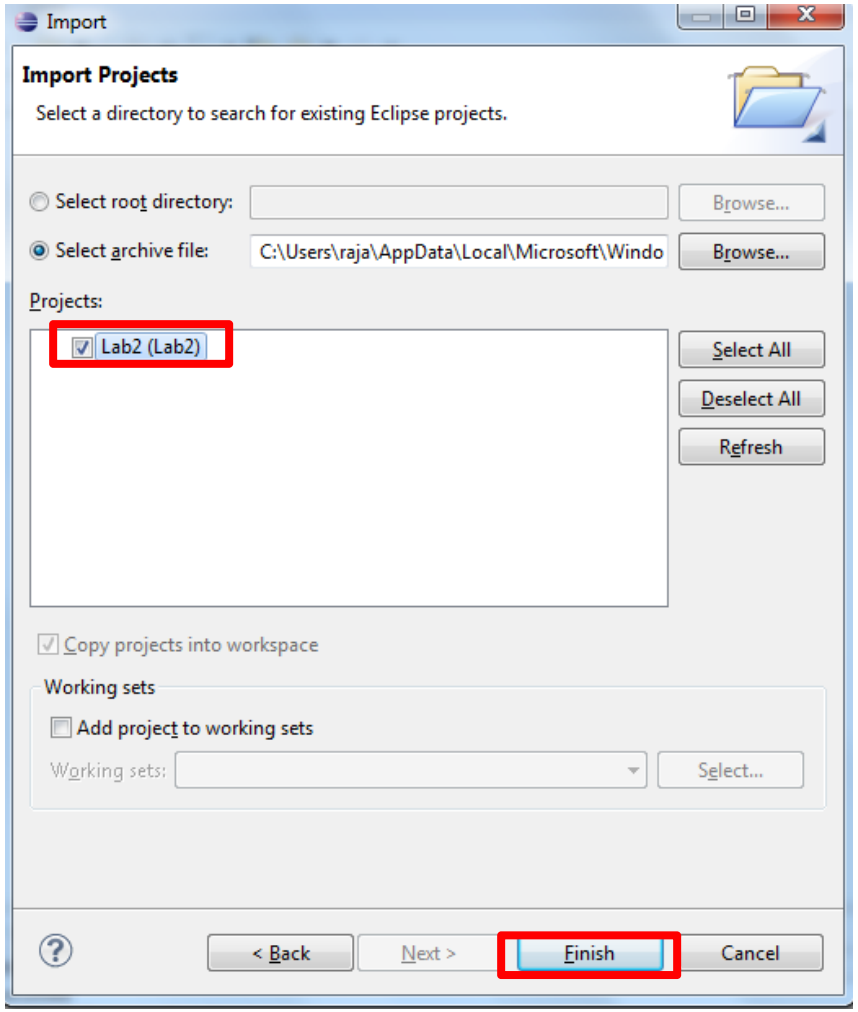
- Enter any *URL* or *Project Source Location*
  - For example;
    - **\\A\PRJ\Lab2.zip**



  - Click on **Open**

Importing the Java Project

- Select **Lab2**
  - Click on **Finish**



Test Class Implementation