

CSCE 740 - Unit Testing Activity

Name(s):

You are testing the following method:

```
public double max(double a, double b);
```

Devise four executable test cases for this method in the JUnit notation. See the attached handout for a refresher on the notation.

jUnit Basics

(You may keep this handout)

JUnit is a Java-based toolkit for writing executable tests.

- Choose a target from the code base.

```
public class Calculator {
    public int evaluate (String expression) {
        int sum = 0;
        for (String summand: expression.split("\\+"))
            sum += Integer.valueOf(summand);
        return sum;
    }
}
```

- Write a “testing class” containing a series of unit tests centered around testing that target.
 - Each test is denoted **@test**

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

```
public class CalculatorTest {
    @Test
    public void evaluatesExpression() {
        Calculator calculator = new Calculator();
        int sum = calculator.evaluate("1+2+3");
        assertEquals(6, sum);
        calculator = null;
    }
}
```

```
@Test
public void test<MethodName><TestingContext>() {
    //Define Inputs
    try{ //Try to get output.
    }catch(Exception error){
        fail("Why did it fail?");
    }
    //Compare expected and actual values through assertions or through if statements/fails
}
```

- **@Before** annotation defines a common test initialization method:

```
@Before
public void setUp() throws Exception
{
    this.registration = new Registration();
    this.registration.setUser("ggay");
}
```

- **@After** annotation defines a common test tear down method:

```
@After
public void tearDown() throws Exception
{
    this.registration.logout();
    this.registration = null;
}
```

- **@BeforeClass** defines initialization to take place before any tests are run.

```
@BeforeClass
public static void setUpClass() {
    myManagedResource = new
        ManagedResource();
}
```

- **@AfterClass** defines tear down after all tests are done.

```
@AfterClass
public static void tearDownClass() throws IOException {
    myManagedResource.close();
    myManagedResource = null;
}
```

- Assertions are a "language" of testing - constraints that you place on the output.
 - assertEquals, assertEqualsArray
 - Compares two items for equality.
 - For user-defined classes, relies on .equals method.
 - Compare field-by-field
 - assertEquals(studentA.getName(), studentB.getName()) rather than assertEquals(studentA, studentB)

```
@Test
public void testAssertEquals() {
    assertEquals("failure - strings are not equal", "text", "text");
}
```

- assertEquals compares arrays of items.

```
@Test
public void testAssertArrayEquals() {
    byte[] expected = "trial".getBytes();
    byte[] actual = "trial".getBytes();
    assertEquals("failure - byte arrays
    not same", expected, actual);
}
```

- assertFalse, assertTrue

- Take in a string and a boolean expression.
- Evaluates the expression and issues pass/fail based on outcome.
- Used to check conformance of solution to expected properties.

```
@Test
public void testAssertFalse() {
    assertFalse("failure - should be false", (getGrade(studentA,
    "CSCE747").equals("A"));
}
@Test
public void testAssertTrue() {
    assertTrue("failure - should be true", (getOwed(studentA) > 0));
}
```

- assertNull, assertNotNull

- Take in an object and checks whether it is null/not null.
- Can be used to help diagnose and void null pointer exceptions.

```
@Test
public void testAssertNotNull() {
    assertNotNull("should not be null", new Object());
}
```

```
@Test
public void testAssertNull() {
    assertNull("should be null", null);
}
```

- assertEquals, assertNotSame

- Checks whether two objects are clones.
- Are these variables aliases for the same object?

- assertEquals uses .equals().
- assertEquals uses ==

```
@Test
public void testAssertNotSame() {
    assertNotSame("should not be same Object", studentA, new Object());
}
```

```
@Test
public void testAssertSame() {
    Student studentB = studentA;
    assertEquals("should be same", studentA, studentB);
}
```