Montgomery College, CMSC 203
Worksheet 1
Module 9

Objectives

- JUnit

Concept Questions

1) Which of the following is correct about manual testing?
   A) Since test cases are executed by human resources, it can be very slow and tedious.
   B) Since test cases need to be executed manually, more labor (test programmers) are required
   C) Both of the above.
   D) None of the above.

Ans: C

2) What is a Unit Test Case?

Ans: A Unit Test Case is a part of code, which ensures that another part of code (method) works as expected.

3) What is JUnit?

Ans: JUnit is a unit-testing framework for the Java programming language.

4) What is `@Test` and where is it used?

Ans: `@Test` annotation is used to mark a method as test method, result of which is then compared with expected output to check whether the test is successful or not.

5) What is `@Before` and `@After` and its usage?

Ans:

`@Before` – This method should execute before each test. Such methods are generally used for initialization before performing a actual test in test environment.

`@After` – This method should execute after each test and used for cleaning up the test and temporary data to avoid memory issues.

6) Which methods cannot be tested by JUnit test class?
   A) public methods
   B) private methods
   C) protected methods
   D) methods with void return type

Ans: B

Programming Questions

1. Given the following Conversion class, implement the following:

```
 public class Conversion {
      private double temp;  // Temperature

      public Conversion ( double temp)
      {
            this.temp = temp;
      }

      public double tempConversion(String unit)
      {
            if ( unit.equals("F"))
                  return (temp -32 ) * (5.0/9);   //Convert to Celcius
            else
                  return (temp * (9.0/5) ) + 32;  // Convert to Fahrenheit
      }
      public String toString ()
      {
            return "Temperature conversion program! " + temp;
      }

}

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class ConversionTest {
      Conversion c1 ,c2;
      @BeforeEach
      void setUp() throws Exception {
            //Create an object c1 of type Conversion with the temp value set to 100
            c1 = new Conversion (100);
            //Create an object c2 of type Conversion with the temp value set to 32
            c2 = new Conversion (32);
      }

      @AfterEach
      void tearDown() throws Exception {
            //destroy the reference to c1 and c2
            c1=c2=null;

      }

      @Test
      void testtempConversion() {
            // Complete the following test cases
            assertEquals (37.7, c1.tempConversion("F") , .1);
            assertEquals( 89.6, c2.tempConversion("C"),.1);
```

```
	}
	@Test
	void testtoString() {
		// Complete the following test cases
		assertTrue(c1.toString().equals("Temperature conversion program!
100.0"));
	}
}
```

In the code above, under the comments, create these methods:

- Write a test method, which will test the equality of `String str1` and `String str2`.

- Write a test method, which will test the equality of two floating point variables `f1` and `f2`.

- Write a test method, which will assert the truthfulness of this expression: if integer `i1` is greater than integer `i2` the test should pass, otherwise the test should fail.

- In the setup method, change the values of `f1` and `f2` so that the `testFloatingEquality` test passes.

2. Following represent a `Car` Class:

```
public class Car {
	private String name;
	private double price;
	private String ownerName;

	public Car(String name, double price, String ownerName) {
		this.name = name;
		this.price = price;
		this.ownerName = ownerName;
	}

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
		this.price = price;
	}
	public String getOwnerName() {
		return ownerName;
	}
	public void setOwnerName(String ownerName) {
		this.ownerName = ownerName;
	}
}
```

Implement the following method in the Car Class that returns the discounted price of a `Car` object based on a given discount rate:

```
public double discountPrice(double rate)
```

Create the following Junit test class:

- In the setup method create two `Car` instances with the following information:
  `name` : BMW
  `price`: 10000
  `owner`: Bill
- Create a test method which will test the equality of all the fields of the `Car` objects.
- Create a test method which will test discountPrice method. For example a call to
- `discountPrice(10)`, assuming the above information for the Car object should return 9000 .
- In the tear down method delete the `Car` objects which you created.

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

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class CarTest {
	Car c1;
	Car c2;

	@Before
	public void setUp() {
		c1 = new Car("BMW", 10000, "Bill");
		c2 = new Car("BMW", 10000, "Bill");
	}

	@Test
	public void testEquality() {
		assertEquals(c1.getName(), c2.getName());
		assertEquals(c1.getPrice(), c2.getPrice(), 0.0001);
		assertEquals(c1.getOwnerName(), c2.getOwnerName());
	}

	@Test
	public void testSum() {
		assertEquals(c1.getPrice() + c2.getPrice(), 20000,
0.00001);
	}
	@Test
	public void testDiscountPrice() {
```

```
            assertEquals(9000,c1.discountPrice(10),  0.00001);
      }

      @After
      public void tearDown() {
            c1 = null;
            c2 = null;
      }
}
```