Homework 2

Chapter 3: Test Automation

Name: ______________________

What to do?

Complete the problems below and submit a zip file containing this word document and the code file requested below.

1. **Refer to Chapter 3, problem #6. (40 points)**

   Consider `PrimeNumbers.java` class (`PrimeNumbers.java` is attached to this assignment). This class has three methods as follows:

   - `computePrimes()` → takes one integer input and computes that many prime numbers
   - `iterator()` → returns an Iterator that will iterate through the primes
   - `toString()` → returns a string representation of primes.

   `computePrimes()` has a fault that causes it **not** to include prime numbers whose last digit is 9. For example, it omits `19, 29, 59, 79, 89, 109,…`

   Answer the following questions `a` through `f`, if the test could be created describe what the test case would be or explain why it cannot be created:

   a. A test that does not reach the fault
   b. A test that reaches the fault, but does not infect
   c. A test that infects the state, but does not propagate
   d. A test that propagates, but does not reveal
   e. Write a Junit test that reveals the fault for the following input.

   ```
   computePrimes(8);
   Expected = [2, 3, 5, 7, 11, 13, 17, 19]
   Result = [2, 3, 5, 7, 11, 13, 17, 23]
   ```

   f. After repairing the fault, write a data-driven Junit test for the `toString` method with at least 3 different values. In particular, this data-driven test method should test `toString` method using @ParameterizedTest based on the results of `computePrimes method for n = 7, 8 and 9.`

   Note: For e and f, submit the code for a Junit test class, `PrimeNumbersTest.java`.

2. **Refer to Chapter 3, problem #9 (15 points)**

When overriding the `equals()` method, programmers are also required to override the `hashCode()` method; otherwise clients cannot store instances of these objects in common `Collection` structures such as `HashSet`. For example, the `Point` class from Chapter 1 is defective in this regard (`Point.java` is attached to this assignment).

Answer the following questions and make it easy to evaluate your work. Specifically, since you are evaluating different versions of the same code, provide evidence of execution (screenshots) at each phase, and briefly describe all of your work.

a) Demonstrate the problem with `Point` using a `HashSet`

   Hint: For example, you can create two same instances of point class with the same x and y values and store them in a HashSet to demonstrate in fact the Point class is defective because of not overriding the hashcode method, while the equals method works as expected.

b) Write a simple Junit test to show that `Point` objects do not enjoy this property (i.e., before fixing the problem)

c) Repair the `Point` class to fix the fault

3. **You can take a look at `MinTest.class` under in-class example for reference (45 points)**

Develop JUnit tests for the `MyArrayList` class, which is attached to this assignment on Blackboard. Make sure that your tests check every method in this class with test cases that include exceptional and normal behaviors. Submit the code for a Junit test class (e.g., you can name the class containing your test methods as `MyArrayListTest`).

```
The following can be helpful for you to understand how you can utilize the methods in
this class. In your test methods, you do not really need to print anything, though.
```

```
// Array can be any type. I choose String below as an example
MyArrayList<String> myArrayList = new MyArrayList<>();

// Array can be declared with an initial capacity of the list
// MyArrayList<String> myArrayList = new MyArrayList<>(2);

myArrayList.add("cat");
myArrayList.add("dog");
myArrayList.add("rabbit");

// add() method can be used to add an element to a certain index such as shown below
myArrayList.add(3, "bear");

// get() method can be used to access an element at certain index
System.out.println("Element at index 1: " + myArrayList.get(1));

// size() method can be used to get the number of elements in this list.
System.out.println("List size: "+ myArrayList.size());

// remove() method removes the element at index provided
System.out.println("Removing element at index 1: " + myArrayList.remove(2));

// Print the list for fun
System.out.println("List: " + myArrayList);
```

Due Date

This homework is due by **Friday, February 18, 2022, 11:59 pm.** A penalty of 10% per day will be deducted from your grade, starting at 12:00:01 am.

What to submit?

Submit a zip file that contains the following on Blackboard:

- A word document describing your answers to the questions above.
- The code files requested above