CSC 455: Assignment 02 | Due date: Sept. 20, 11:59pm

Instructor: Dr. Razib Iqbal | www.razib.info

Note: This is a group assignment. CSC455 Project team members must work together and submit as a group.

Q1: *5 points*

There is a faulty program/code snippet in the following textbox.

```
public int countPositive (int[] x)
{
//Effects: If x==null throw NullPointerException
//   else return the number of
//        positive elements in x.
   int count = 0;
   for (int i=0; i < x.length; i++)
   {
      if (x[i] >= 0)
      {
         count++;
      }
   }
   return count;
}
   // test:  x=[-4, 2, 0, 2]
   //        Expected = 2
```

Answer the following questions: (*1 point each)*

a) Identify the fault.
b) Identify a test case that does NOT execute the fault.
c) Identify a test case that executes the fault, but does not result in an error state.
d) Identify a test case that results in an error, but not a failure. (Hint: Error results in failure if not handled by the program)
e) For the given test case in the textbox, identify the first error state during runtime. Describe the complete state including the values of each of the variables, expected and actual results.

Q2: *10 points*

Download the A02_Codes.zip. Do not modify the attached java files. Create a driver to test the attached java files in Eclipse. Name and save the driver as: TestDriver.java.

a) (*7 points*) Add main method and additional methods (as appropriate) to test the attached java files. Driver will read test inputs from a txt file (user.txt). The user.txt will contain test inputs in the following format:
3224400
3224410

3224420
Expected console output from the driver:
3224400 is a local prepaid user
3224410 is a local prepaid user
3224420 is a local prepaid user

b) Answer the following questions based on Q2a: *(3 Points)*
   i. Identify any debugging strategy you used.
   ii. Explain why did you use/choose that debugging strategy.
   iii. Which other debugging strategy could be used?

Q3: *5 points*

a) After solving Q2a, run the driver for one sample input. What is the initial code coverage for PolicyUser, LocalUser, and RoamingUser classes using EclEmma code coverage tool? Submit a screen capture. *(1 point)*

b) Add additional codes in the driver to increase code coverage for the attached java classes to ~100%. Save the driver as TestDriverM.java, and submit a screen capture showing the increased code coverage in EclEmma. *(4 Points)*

Q4: *10 points*

See the JAVA code on the next few pages. Use this code in a new JAVA project (name the Project/File as FindBugsGroupNumber.java), and perform a static code analysis using the FindBugs tool.

a) Identify the reported bugs from the tool, and give a brief overview of these bugs.
b) Find a solution to fix each of these bugs, and submit the modified (working version) code/project.
c) Use a table to answer a) and b)

Submit your assignment **in Blackboard** only:

1. Submit a zip file containing the Eclipse project and the written answers (and screen captures) in a MS Word/PDF file
2. Include full name of the group members on the first page of the MS Word/PDF file
3. Name your zip file: *455_Group_Number_A02*

//Codes for Q4

```
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Collection;
import java.util.UnknownFormatConversionException;

public class Findbugs1 {
	public static void main(final String[] args) {

		System.out.println("Findbugs Sample 001 for BC_IMPOSSIBLE_CAST");
		// WRONG
		try {
			Findbugs1.bcImpossibleCastWRONG();
		} catch (final ClassCastException e) {
			System.out.println("   - ERROR:" + e.getMessage());
		}
		// CORRECT
		Findbugs1.bcImpossibleCastCORRECT();

		System.out.println("Findbugs Sample 002 for BC_IMPOSSIBLE_DOWNCAST");
		// WRONG
		try {
			Findbugs1.bcImpossibleDowncastWRONG();
		} catch (final ClassCastException e) {
			System.out.println("   - ERROR:" + e.getMessage());
		}
		// CORRECT
		Findbugs1.bcImpossibleDowncastCORRECT();

		System.out.println("Findbugs Sample 003 for BC_IMPOSSIBLE_INSTANCEOF");
		// WRONG
		Findbugs1.bcImpossibleInstanceOfWRONG();
		// CORRECT
		Findbugs1.bcImpossibleInstanceOfCORRECT();

		System.out.println("Findbugs Sample 004 for BC_IMPOSSIBLE_DOWNCAST_OF_TOARRAY");
		// WRONG
		try {
			Findbugs1.bcImpossibleDowncastOfArrayWRONG();
		} catch (final ClassCastException e) {
			System.out.println("   - ERROR:" + e.getMessage());
		}
		// CORRECT
		Findbugs1.bcImpossibleDowncastOfArrayCORRECT();

		System.out.println("Findbugs Sample 005 for DMI_BIGDECIMAL_CONSTRUCTED_FROM_DOUBLE");
		// WRONG
		Findbugs1.dmiBigDecimalConstructedFromDoubleWRONG();
		// CORRECT
		Findbugs1.dmiBigDecimalConstructedFromDoubleCORRECT();

		System.out.println("Findbugs Sample 006 for ES_COMPARING_STRINGS_WITH_EQ");
		// WRONG
		Findbugs1.esComparingStringsWithEqWRONG();
		// CORRECT
		Findbugs1.esComparingStringsWithEqCORRECT();

		System.out.println("Findbugs Sample 007 for VA_FORMAT_STRING_ILLEGAL");
		// WRONG
		try {
			Findbugs1.vaFormatStringIllegalWRONG();
		} catch (final UnknownFormatConversionException e) {
			System.out.println("   - ERROR:" + e.getMessage());
		}
		// CORRECT
```

```
		Findbugs1.vaFormatStringIllegalCORRECT();

		System.out.println("Findbugs Sample 008 for RV_RETURN_VALUE_IGNORED");
		// WRONG
		Findbugs1.rvReturnValueIgnoredWRONG();
		// CORRECT
		Findbugs1.rvReturnValueIgnoredCORRECT();

		System.out.println("Findbugs Sample 009 for NP_ALWAYS_NULL");
		// WRONG
		try {
			Findbugs1.npAlwaysNullWRONG();
		} catch (final NullPointerException e) {
			System.out.println("   - ERROR:" + e.getMessage());
		}
		// CORRECT
		Findbugs1.npAlwaysNullCORRECT();

		System.out.println("Findbugs Sample 010 for QBA_QUESTIONABLE_BOOLEAN_ASSIGNMENT");
		// WRONG
		Findbugs1.qabQuestionableBooleanAssignmentWRONG();
		// CORRECT
		Findbugs1.qabQuestionableBooleanAssignmentCORRECT();

	}

	private static void bcImpossibleCastWRONG() {
		final Object doubleValue = Double.valueOf(1.0);
		final Long value = (Long) doubleValue;
		System.out.println("   - " + value);
	}

	private static void bcImpossibleCastCORRECT() {
		final Object doubleValue = Double.valueOf(1.0);
		final Double value = (Double) doubleValue;
		System.out.println("   - " + value);
	}

	private static void bcImpossibleDowncastWRONG() {
		final Object exception = new RuntimeException("abc");
		final SecurityException value = (SecurityException) exception;
		System.out.println("   - " + value.getMessage());
	}

	private static void bcImpossibleDowncastCORRECT() {
		final Object exception = new RuntimeException("abc");
		final RuntimeException value = (RuntimeException) exception;
		System.out.println("   - " + value.getMessage());
	}

	private static void bcImpossibleInstanceOfWRONG() {
		final Object value = Double.valueOf(1.0);
		System.out.println("   - " + (value instanceof Long));
	}

	private static void bcImpossibleInstanceOfCORRECT() {
		final Object value = Double.valueOf(1.0);
		System.out.println("   - " + (value instanceof Double));
	}

	private static void bcImpossibleDowncastOfArrayWRONG() {
		final Collection<String> stringVector = new ArrayList<String>();
		stringVector.add("abc");
		stringVector.add("xyz");
		final String[] stringArray = (String[]) stringVector.toArray();
		System.out.println("   - " + stringArray.length);
```

```
}

private static void bcImpossibleDowncastOfArrayCORRECT() {
	final Collection<String> stringVector = new ArrayList<String>();
	stringVector.add("abc");
	stringVector.add("xyz");
	final String[] stringArray = stringVector.toArray(new String[stringVector.size()]);
	System.out.println("   - " + stringArray.length);
}

private static void dmiBigDecimalConstructedFromDoubleWRONG() {
	final BigDecimal bigDecimal = new BigDecimal(3.1);
	System.out.println("   - " + bigDecimal.toString());
}

private static void dmiBigDecimalConstructedFromDoubleCORRECT() {
	final BigDecimal bigDecimal = new BigDecimal("3.1");
	System.out.println("   - " + bigDecimal.toString());
}

private static void esComparingStringsWithEqWRONG() {
	final StringBuilder sb1 = new StringBuilder("1234");
	final StringBuilder sb2 = new StringBuilder("1234");
	final String string1 = sb1.toString();
	final String string2 = sb2.toString();
	System.out.println("   - " + (string1 == string2));
}

private static void esComparingStringsWithEqCORRECT() {
	final StringBuilder sb1 = new StringBuilder("1234");
	final StringBuilder sb2 = new StringBuilder("1234");
	final String string1 = sb1.toString();
	final String string2 = sb2.toString();
	System.out.println("   - " + string1.equals(string2));
}

private static void vaFormatStringIllegalWRONG() {
	System.out.println(String.format("   - %>s  %s", "10", "9"));
}

private static void vaFormatStringIllegalCORRECT() {
	System.out.println(String.format("   - %s > %s", "10", "9"));
}

private static void rvReturnValueIgnoredWRONG() {
	final BigDecimal bigDecimal = BigDecimal.ONE;
	bigDecimal.add(BigDecimal.ONE);
	System.out.println(String.format("   - " + bigDecimal));
}

private static void rvReturnValueIgnoredCORRECT() {
	final BigDecimal bigDecimal = BigDecimal.ONE;
	final BigDecimal newValue = bigDecimal.add(BigDecimal.ONE);
	System.out.println(String.format("   - " + newValue));
}

private static void npAlwaysNullWRONG() {
	final String value = null;
	if (null != value & value.length() > 2) {
		System.out.println(String.format("   - " + value));
	} else {
		System.out.println(String.format("   - value is invalid"));
	}
}

private static void npAlwaysNullCORRECT() {
```

```
		final String value = null;
		if (null != value && value.length() > 2) {
			System.out.println(String.format("   - " + value));
		} else {
			System.out.println(String.format("   - value is invalid"));
		}
	}

	private static void qabQuestionableBooleanAssignmentWRONG() {
		boolean value = false;
		if (value = true) {
			System.out.println(String.format("   - value is true"));
		} else {
			System.out.println(String.format("   - value is false"));
		}
	}

	private static void qabQuestionableBooleanAssignmentCORRECT() {
		final boolean value = false;
		if (value == true) {
			System.out.println(String.format("   - value is true"));
		} else {
			System.out.println(String.format("   - value is false"));
		}
	}

}
```