CMSC 132 Quiz 1 Worksheet

The first quiz for the course will be on Friday, June 8. The following list provides more information about the quiz:

- The quiz will be a written quiz (no computer).
- Closed book, closed notes quiz.
- Answers must be neat and legible. We recommend that you use pencil and eraser.

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with the TA or instructor during office hours. We strongly recommend you **DO NOT use Eclipse** to write the code associated with these exercises. Try to answer the exercises in a piece of paper and then use Eclipse to verify your solutions. This approach will better prepare you for the quiz.

Exercises

General Questions

1. What is the difference between overriding and overloading?

2. Can every class in Java have a main method?

3. What is the difference between the following two declarations?

```
final static int SIZE = 10;

final int SIZE = 10;
```

4. What is the difference between a class and an instance of a class?

5. What are the advantages/disadvantages of the StringBuffer class vs. String?

6. What are the advantages/disadvantages of ArrayList class vs. Java arrays?

8. What is the value of z after executing the following statements?

```
int y = 4;
int z = ( y < 10 ? (y > 2 ? 1 : 2 ) : 3 );
```

9. What is output by the following code fragment?

```
try {
        String x = null;
        System.out.println( x.length() );
}
catch ( NullPointerException e ) {
        System.out.println( "Null Exception" );
}
catch ( Exception e ) {
        System.out.println( "Exception" );
}
System.out.println( "Why are we here?" );
```

Code Analysis

You be the compiler. All these following classes appear in the same package. For each statement of the main program, indicate whether it generates a compilation error, and if not, what is printed. (When showing the execution, assume that all statements generating compilation errors have been removed.)

```
// in file Foo.java
public class Foo {
    protected void pro( int x ) { System.out.println("Foo:pro:" + x); }
    public void pub( int x ) { System.out.println("Foo:pub:" + x); }

}

// in file Bar.java
public class Bar extends Foo {
    protected void pro( int x ) { System.out.println("Bar:pro:" + x); }
    public void pub( int x ) { System.out.println("Bar:pub:int:" + x); }
    public void pub( double d ) { System.out.println("Bar:pub:dbl:" + d); }
}

// in file Driver.java
public static void main(String[] args) {
    Foo a = new Foo( );
    Foo b = new Bar( );
    Bar c = new Bar( );
    a.pro( 1 );
    a.pub( 2 );
    a.pub( 3.0 );
    b.pro( 4 );
    b.pub( 5 );
    b.pub( 6.0 );
    c.pro( 7 );
    c.pub( 8 );
    c.pub( 9.0 );
}
```

Implementation

1. Write a static method that produces an identical copy of a two-dimensional array of doubles. The signature of the method is:

   public static double[][] duplicate(double[][] array);

2. Write a static method that is given a 2-dimensional array of doubles, and finds the first row that consists of an increasing sequence of values, that is, it finds the smallest i such that:

   array[i][0] < array[i][1] < array[i][2] < ...

   If such a row exists, a copy of the contents of that row is returned as a one dimensional array. Otherwise null is returned. The signature of the method is:

   public static double[] firstIncreasing(double[][] array);

3. Write a method that is given a two-dimensional (ragged) array of String objects and returns a two-dimensional (ragged) array of String objects where all the null entries have been removed. For example, if the original array has the data (NULL represents a null reference):

John (null) Mary George (null)
(null) Pete Rick
(null) (null) (null)

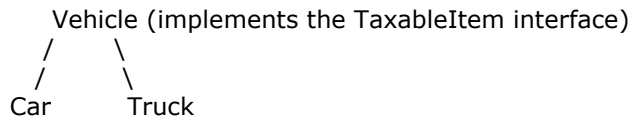
the result generated by your method will be a two-dimensional array with three rows.

John Mary George
Pete Rick
(this row is empty)

4. Implement the classes and interface associated with the inheritance hierarchy below. We are providing the basic idea but it is up to you to develop the full set of classes/interfaces. This problem is intentionally open-ended, and there is no one correct solution. Try as much as possible to have the following constructs in your code:

   CONSTRUCTS: super, this, protected, overriding, overloading

   Inheritance Hierarchy:



   Vehicle Class:
   data members: weight, maximumSpeed, numberOfDoors,
   numberOfTires, color
   methods: get (accessor) methods for data members
   abstract methods: start(), move(), shiftGear()

   Car Class:
   data members: add any members you understand are necessary
   methods: get/set methods for data members, isSedan(), isSportCar(),

   Truck Class:
   data members: maxLoadCapacity, maxSpeedLoaded
   methods: get/set methods for data members,
   unloadCargo(), loadCargo()

   TaxableItem Interface:
   methods: getTaxAmount(), getItemName()

   Feel free to add any data members/methods you understand are necessary.