Test 2 Review: CompSci 6

Name (print): ____________________________

Honor Acknowledgment (signature): ____________________________

DO NOT SPEND MORE THAN 15 MINUTES ON ANY OF THE QUESTIONS! If you do not see the solution to a problem right away, move on to another problem and come back to it later. The final page is a list of common methods of classes we have studied in class so that you do not need to memorize such details.

Before starting, make sure your test contains 14 pages.

If you think there is a syntax error, then ask.

<table>
<thead>
<tr>
<th>Problem</th>
<th>value</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>5 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>15 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>15 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>55 pts.</td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM 1 :  (Parse, Parse, Baby: (5 pts))

Write the method `average` which, given a string of positive whole numbers (each separated by a comma), returns the average of those numbers. The given string is guaranteed to contain at least one value and does not contain any spaces or other punctuation.

For example, the code given below should assign `i`, `j`, and `k` the values 1.0, 2.33, and 277.75, respectively.

```java
String s = "1,1,1,1,1";
String t = "4,2,1";
String u = "10,1,100,1000";

double i = average(s);
double j = average(t);
double k = average(u);
```

Complete the method `average` below.

```java
public double average (String values)
{
```
PROBLEM 2 :  (No Credit for This (10 points))

Credit card numbers can be validated using what’s called a checksum algorithm. The algorithm basically works as follows, where the left-most digit has index 0 (as it does for strings) and the algorithm works from left to right examining each digit of the credit card.

Accumulate a sum based on adding a value obtained from each digit of the credit card as follows, where each \( k^{th} \) digit is examined starting with \( k = 0 \).

- If \( k \) is even, add the \( k^{th} \) digit to the sum.
- If \( k \) is odd, multiply the \( k^{th} \) digit by two. If the result is greater than or equal to 10, subtract 9. Add this computed value to the sum.

If the resulting sum is divisible by 10, the credit card number passes the checksum test, otherwise the card number is not valid. Credit card numbers are often 16-digits long, but they can be of any length.

Here’s an example for the card number 7543210987654347 showing that it is a valid card number.

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit Card Number</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Value Added</td>
<td>7</td>
<td>10-9</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>18-9</td>
<td>8</td>
<td>14-9</td>
<td>6</td>
<td>10-9</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>14-9</td>
</tr>
<tr>
<td>Total</td>
<td>7</td>
<td>8</td>
<td>12</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>22</td>
<td>31</td>
<td>39</td>
<td>44</td>
<td>50</td>
<td>51</td>
<td>55</td>
<td>61</td>
<td>65</td>
<td>70</td>
</tr>
</tbody>
</table>

Since this is valid, the call isValid("2543210987654321") evaluates to true. However, the call isValid("1234") evaluates to false since the computed sum is 1 + 4 + 3 + 8 = 16 which isn’t divisible by 10.

Write the boolean-valued method isValid whose header is given on the next page.

In writing isValid you might want to use the method total below as a model — in particular this method shows how to use the method Integer.parseInt to convert a string to a number.

```java
/*
 * Add up values of digits in s, return total.
 * This only works if s consists of digits 0..9,
 * e.g., total("12345") returns 15
 * @param s is the string whose digits are totalled
 * @return total of digits in s
 */
public int total (String s) {
    int total = 0;
    for(int k = 0; k < s.length(); k++) {
        total += Integer.parseInt(s.substring(k, k + 1));
    }
    return total;
}
```

(method on next page ⇒)
/**
 * Returns true if and only if credit card has a valid checksum.
 * @param ccard consists only of digits '0', ... '9'
 */

public boolean isValid (String ccard)
{

}
PROBLEM 3 :  (It’s all in the differences: (10 pts))

Write the method maxDifference whose header is given below. maxDifference returns the largest difference between two values in its ArrayList parameter, numbers.

For example, if numbers represents the values \{1, 5, 2, 7, 4\} then maxDifference(numbers) should evaluate to 6 since the largest difference between values in numbers is 7 - 1 = 6. Note that the difference between 1 and 7 is also considered to be 6. You may find the static method Math.abs useful, it returns the absolute value of a number.

In writing maxDifference you can call the method maxValue shown below. You may also write other methods to use in the code you write for maxDifference

```java
/**
 * Returns maximal value in ArrayList.
 * @param numbers is the ArrayList whose max value is returned
 * @return maximal value in parameter numbers
 */
public int maxValue(ArrayList<Integer> numbers) {
    int max = Integer.MIN_VALUE;
    for (Integer value : numbers) {
        max = Math.max(max, value);
    }
    return max;
}

/**
 * Returns maximal difference between values in a list
 */
public int maxDifference(ArrayList<Integer> numbers) {
    return maxValue(numbers) - maxValue(numbers);
}
```
PROBLEM 4 :  (Glory Days: 15 pts))

Implement a class that represents a music jukebox. To support its use in as many diners as possible, your class should read its possible songs from a data file, one answer per line. Each time the jukebox is requested to play a song, a random one from those read from the file should be returned.

For example, below is the first five lines of a file that holds some of my favorite songs in no particular order:

Kool and The Gang - Ladies Night
Tom Jones - Its Not Unusual
Bee Gees - Night Fever
Captain and Tennille - Love Will Keep Us Together
Donna Summer - Dim All The Lights

For example, the code below represents how your class might be used in a program.

```java
JukeBox box = new JukeBox(new Scanner(new File("old_songs.txt"))); while (box.numSongsPlayed() < 5) {
    System.out.println(box.playSong());
}
```

Below is one possible sample output from the program above given the data file above.

Captain and Tennille - Love Will Keep Us Together
Tom Jones - Its Not Unusual
Donna Summer - Dim All The Lights
Kool and The Gang - Ladies Night
Tom Jones - Its Not Unusual

(problem continued on next page ⇒)
**Part A:** (3 points)

Complete the declaration of the class’ instance variables below. You should add any private state (instance variables) you think your `JukeBox` class needs in order to implement the constructor and other two methods of the class so that they function as shown in the example above.

```java
public class JukeBox {
    // Part A: declare instance variables here

    public JukeBox (Scanner input) {
        // implemented in Part B
    }

    public int numSongsPlayed () {
        // implemented in Part B
    }

    public String playSong () {
        // implemented in Part B
    }
}
```
Part B: (12 points)

Write the implementation for the class member functions previously declared below. Note, you may lose points on Part A for instance variables that are declared there, but not used in this Part.

```java
/**
 * All songs from the list passed in are added to another
 * list kept by this class. All other instance variables
 * are initialized.
 */
public JukeBox (Scanner input)
{
}

/**
 * @returns number of songs that have been played (i.e.,
 * the number of times playSong has been called)
 */
public int numSongsPlayed ()
{
}

/**
 * @returns one random string from possible songs
 */
public String playSong ()
{
}
```
Different people want different prizes when they enter a sweepstakes. To accommodate this, you are to write several classes that inherit from a generic `Winner` class, but differ only in the prize that is offered to the contestant.

Consider the hierarchy of classes given below to solve this problem (their method implementations are not important yet, so they are not included below):

```java
public abstract class Winner {
    private string myFirstName;
    private string myLastName;
    private int myTicketNumber;

    public Winner (String firstName, String lastName)
    public void printLetter ()
    protected void printGreeting ()
    protected void printTicket ()
    protected void printPrize ()

    private void generateTicket (int numDigits)
}

public class CarWinner extends Winner {
    private String myCar;

    public CarWinner (String firstName, String lastName, String car)
    protected void printGreeting ()
    protected void printPrize ()
}

class CruiseWinner extends Winner {
    private String myDestination;

    public CruiseWinner (String firstName, String lastName, String destination)
    protected void printGreeting ()
    protected void printPrize ()
}
```
Part A: (2 points)

If an instance of `CarWinner` was created for the contestant “Amanda Harris” and an instance of `CruiseWinner` was created for “Michael Abernathy”, the following output should be printed.

Amanda Harris, you’ve won a NEW CAR!
Amanda, that’s what you will hear if your
ticket 387452289 matches the winning ticket.
Imagine, Amanda, you could be cruising
in a Mercedes down Main Street!

Michael Abernathy, you’ve won a 7-DAY CRUISE!
Michael, that’s what you will hear if your
ticket 126755804 matches the winning ticket.
Imagine, Michael, you could be cruising
to beautiful Hawaii with your sweetheart!

Assuming that all the classes declared previously have been correctly implemented, complete the code started in `main` below such that the output above is printed.

```java
public static void main (String[] args)
{
    Winner c1 = new CarWinner( , , );
    Winner c2 = new CruiseWinner( , , );

    c1.printLetter();
    c2.printLetter();
}
```
Below the code for the base class `Winner` is given (except for the function `generateTicket` which you will complete in the final part of this problem). On the next page, you will complete the code for the two sub-classes of `Winner` so that they generate the output in the previous part.

*In writing your subclasses, you should write as little new code as possible. In other words, make use of calls to already implemented base class functions where possible.*

```java
public Winner (String firstName, String lastName)
{
    myFirstName = firstName;
    myLastName = lastName;
    myTicketNumber = generateTicket(9);
}

public void printLetter ()
{
    printGreeting();
    printTicket();
    printPrize();
    System.out.println();
}

protected void printGreeting ()
{
    System.out.print(myFirstName + " " + myLastName + ", ");
}

protected void printTicket ()
{
    System.out.println(myFirstName + ", that's what you will hear if your");
    System.out.println("ticket " + myTicketNumber + " matches the winning ticket.");
}

protected void printPrize ()
{
    System.out.println("Imagine, " + myFirstName + ", you could be cruising");
}
```

(problem continued on next page ⇒)
Part B: (5 points)

```java
public CarWinner (String firstName, String lastName, String car)
{
}

protected void printGreeting ()
{
}

protected void printPrize ()
{
}
```
Part C: (5 points)

public CruiseWinner (String firstName, String lastName, String destination);
{

}

protected void printGreeting ()
{

}

protected void printPrize ()
{

}
Part D: (3 points)
Write the function `generateTicket` that takes a number of digits to generate and creates a random number guaranteed to have that many digits. Note, this means the number can have zeros within it, like the five-digit number 10020, but it cannot start with a zero. For example, the number 01234 is not a valid five-digit number.

```java
private int generateTicket (int numDigits)
{
    int result = 0;
    Random rand = new Random();
    return result;
}
```