STUDENT ID#__________________________

CS 207, Spring 2016, Programming II PRACTICE FINAL EXAM
Saturday, April 30, 2016
11:00 a.m. to 1:00 p.m.
Room LWH-1001

Instructions:

1. **Do not turn this page until told to do so.**

2. This exam is ***closed book*** and ***closed notes***.

3. ***Write your STUDENT ID Number on every page***. If you do not write your ID Number on a certain page, you ***will not receive credit for the question on that page!***

4. ***Do not write your name on any page (except this one)***. If you write your name on a certain page, ***you will not receive credit for the question on that page!***

5. There are **6** questions on the exam, one per page. Once you start, verify you have all 6 questions.

6. You must give your answer to a question on the ***same page that the question appears*** (or the page where you are asked to write your answer). If you put your answer on a different page, ***you will not receive any credit for it!***

7. For problems that ask you to write a ***method***, you must write the method header ***exactly*** as shown, but you do not need to write `main()`.

8. For problems that ask you to write a ***program***, you should write the `main()` method ***only***—you can assume the following import statement and keyboard declarations.

```
        import java.util.*;

        Scanner keyboard = new Scanner(System.in);
   or   Scanner kb = new Scanner(System.in);
   or   Scanner kbd = new Scanner(System.in);
```

9. For tracing problems, assume that any appropriate import statements are provided.

10. You may use "SOP" as an abbreviation for "`System.out.print`" and "SOPln" or "SOPL" for "`System.out.println`".

11. You do not need to do any error checking of input values, ***unless the problem specifically asks you to do so!***

12. If you are caught looking at other papers or communicating with other students in any way, you will receive an **F** for the course.

STUDENT ID#____________________________

Question 1 (20 pts)

Write a method named `arrayOfDigits`. The method receives one parameter, an integer named `n`, and returns a 2-dimensional (2D) integer array. Each row in the 2D array should be created in the following way:

1. The first row should contain the digits from `0`, up to and including, the left-most digit of `n`.
2. The second row should contain the digits from `0`, up to and including, the second-to-left digit of `n`.
3. Continue to create each row of the 2D array in this way so that the last row contains the digits from `0`, up to and including, the right-most digit of `n`.
4. You can assume that `n` will never be negative.
5. See sample usage below.

```
arrayOfDigits(247) returns:
{ { 0, 1, 2 },
  { 0, 1, 2, 3, 4 },
  { 0, 1, 2, 3, 4, 5, 6, 7 } }
```

```
arrayOfDigits(9031) returns:
{ { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 },
  { 0 },
  { 0, 1, 2, 3 },
  { 0, 1 } }
```

```
public static int[][] arrayOfDigits(int n)
{












}
```

STUDENT ID#____________________________

Question 2 (20 pts)

Create a properly encapsulated class named `LetterArray` that has the following:

1. A `String` instance variable named `line`.
2. A 2-dimensional (2D) `char` array instance variable named `letters`.
3. A constructor that accepts one `String` parameter and sets the instance variable `line` to the value of the parameter.
4. A getter (accessor) method for the `line` instance variable.
5. A method named `createLetterArray` that does not return anything. The method should create a 2D `char` array where each row is created in the following way: The elements of the first row should contain the characters of the first word in `line`, the elements of the second row should contain the characters of the second word in `line`, and so on. The method should also set the `letters` instance variable to this 2D `char` array. You can assume that the `line` instance variable will only contain letters (a-z and A-Z) and spaces.
6. A method named `printLetterArray` that does not return anything and prints each row of the `letters` instance variable on its own line, with the elements separated by spaces.
7. Sample usage is provided below.
8. Complete and place your code in the box on the next page. **Any code on this page will not be considered as part of your solution.**

`Sample usage:`

```
LetterArray sa1 = new LetterArray("aspire to inspire");
System.out.println(sa1.getLine());
sa1.createLetterArray();
sa1.printLetterArray();
System.out.println();

LetterArray sa2 = new LetterArray("veni vidi vici");
System.out.println(sa2.getLine());
sa2.createLetterArray();
sa2.printLetterArray();
System.out.println();

LetterArray sa3 = new LetterArray("verylongword");
System.out.println(sa3.getLine());
sa3.createLetterArray();
sa3.printLetterArray();
```

`Output:`

```
aspire to inspire
a s p i r e
t o
i n s p i r e

veni vidi vici
v e n i
v i d i
v i c i

verylongword
v e r y l o n g w o r d
```

STUDENT ID#____________________________

Question 2 (continued)

STUDENT ID#______________________________

Question 3 (15 points)

What is the **exact** output of the `main` method in the `FlowerTest` class (found on the following page)?

```java
public class Flower
{
  public static int num = 1;
  public static String pot = "";
  private String species;
  private int petals;

  public Flower(String s)
  {
    System.out.println(this.species);
    this.species = s;
    num += 2;
    System.out.println(num);
  }

  public Flower(String s, int p)
  {
    this.species = s;
    this.petals = p;
    System.out.println(this.species + " " +
                       this.petals);
    pot += s;
    System.out.println(pot);
  }

  public String flowerPower()
  {
    String s = this.species + " " +
               this.petals;
    pot += this.petals;
    return s;
  }
}
```

```java
public class Lily extends Flower
{
  private String genus;

  public Lily(String s, String g, int p)
  {
    super(s, p);
    System.out.println(this.genus + " " +
                       num);
  }

  public String flowerPower()
  {
    String s = super.flowerPower();
    s += " " + this.genus;
    num++;
    return s;
  }

  public boolean equals(Object o)
  {
    Flower f = (Flower) o;
    String p1 = this.flowerPower();
    String p2 = f.flowerPower();
    boolean isEqual = false;
    if (p1.equals(p2))
      isEqual = true;

    return isEqual;
  }
}
```

STUDENT ID#____________________________

Question 3 (continued)

```java
public class FlowerTest
{
  public static void main(String[] args)
  {
    Flower f1 = new Flower("azalea");
    Lily y1 = new Lily("calla", "palustrus", 1);
    Flower f2 = new Lily("astromeria", "?", 30);

    Flower[] pretty = { f2, new Flower("petunia"), y1, f1 };

    for (int i = 0; i < pretty.length; i++)
    {
      String fp = pretty[i].flowerPower();
      System.out.println(fp);
    }

    Lily y2 = new Lily("calla", "palustrus", 1);
    Flower f3 = new Flower("azalea");

    boolean eq;
    eq = y1.equals(y2);
    System.out.println(eq);

    eq = f2.equals(f3);
    System.out.println(eq);
  }
}
```

Output Window

STUDENT ID#___________________________

Question 4 (15 points)

Given the below interface and abstract class, create the class `MyMatrix` as follows:

1. `MyMatrix` should inherit from the `Matrix` class and implement the `Enumerable` interface.
2. The `MyMatrix` constructor should take one parameter, a 2-dimensional (2D) integer array and set the superclass instance variable.
3. The `printRowLengths` method should print the length of each row. Your output format should match the sample output format **exactly**.
4. The `printRowForUser` method should prompt the user to enter a row index and read in the value entered. The method should print out the row at the index specified by the user. You can assume that the `Scanner` object has been created (see instructions on first page) and that the user will only enter valid row indices. Your output format should match the sample output format **exactly**.
5. The `largestNumber` method should return the largest number in the `numbers` instance variable.
6. You should write the code to produce the output in the same format provided below. **Do not add any additional methods other than the ones described in the instructions.**
7. Complete and place your code in the box provided on the next page. **Any code on this page will not be considered as part of your solution.**

Sample Usage

```
int[][] a = { { 18,  0, 2 },
              {  3, -3 },
              { 29,  4, 8, -1 } };

MyMatrix mm1 = new MyMatrix(a);
mm1.printRowLengths();
System.out.println(mm1.largestNumber());
mm1.printRowForUser();
System.out.println();

int[][] b = { { -7,  -5, -19, -34 } };

MyMatrix mm2 = new MyMatrix(b);
mm2.printRowLengths();
System.out.println(mm2.largestNumber());
mm2.printRowForUser();
```

Matrix.java

```
public abstract class Matrix
{
  private int[][] numbers;

  protected Matrix(int[][] nums)
  {
    this.numbers = nums;
  }

  public int[][] getNumbers()
  {
    return this.numbers;
  }

  public abstract void printRowLengths();
}
```

Sample Output

```
Row 0: 3
Row 1: 2
Row 2: 4
29
Enter row index: 1
3, -3,

Row 0: 4
-5
Enter row index: 0
-7, -5, -19, -34,
```

Enumerable.java

```
public interface Enumerable
{
  public abstract void printRowForUser();

  public abstract int largestNumber();
}
```

STUDENT ID#____________________________

Question 4 (continued)

STUDENT ID#___________________________

Question 5 (10 points)

Complete the `evenOddNeither()` method below so that it does the following:

1. Reads the input from the data.txt file (sample provided below). You do **not** know how many lines are in the file. The sample provided is just an example of the file format.
2. Prints out `"Even"` if the two integers in a line are both even, prints out `"Odd"` if they are both odd, and `"Neither"` otherwise. In addition, keep track of the smallest integer read in and print that value out after reading each line, matching the sample output format **exactly.**
3. You can assume that all integers in the file are less than 10000000.
4. Handles a `FileNotFoundException` by printing out "`Nope. Try again.`"
5. You can assume that the `Scanner` and `File` classes have been imported.
6. Complete and place your code in the box provided on the next page. **Any code on this page will not be considered as part of your solution.**

Sample file:

```
12 3
-2 100
91 1
...
```

Sample output:

```
Neither
Current minimum: 3
Even
Current minimum: -2
Odd
Current minimum: -2
...
```

Question 5 (continued)

```
public static void evenOddNeither()
{




}
```

STUDENT ID#____________________________

Question 6 (20 points)

Write a method named `changeDuplicates()` that takes a `String` parameter `s` and does the following:

1. If a character in `s` is the first occurrence of that specific character, then keep it. Any subsequent occurrences of a character (i.e. duplicates) should be changed to a hyphen (`'-'`). Return the modified version of `s`.
2. If `s` contains any capital letters, the method should throw an `IllegalArgumentException` with the message `"Capital letters not permitted"`. You do not have to handle the exception.
3. You may use at most **one** loop.
4. You may use at most **two** conditional statements, and only **one** of those conditional statements can have an else block (no else-if blocks are allowed).
5. Sample usage is provided below.
6. Complete and place your code in the box on the next page. **Any code on this page will not be considered as part of your solution.**

Calling `changeDuplicates("mississippi")` returns:
```
mis-----p--
```

Calling `changeDuplicates("dazzled")` returns:
```
daz-le-
```

Calling `changeDuplicates("duplicates")` returns:
```
duplicates
```

The following examples should result in an exception. Catching and handling the exception by printing `e.toString()` when calling the method would show the following:

Calling `changeDuplicates("Cleopatra")` results in an `IllegalArgumentException`:
```
java.lang.IllegalArgumentException: Capital letters not permitted
```

Calling `changeDuplicates("tiMe")` results in an `IllegalArgumentException`:
```
java.lang.IllegalArgumentException: Capital letters not permitted
```

STUDENT ID#______________________________

Question 6 (cont.)

```
public static String changeDuplicates(String s)
{






}
```