CSC 171 LAB 05

Your assignment is to write programs in the Java language by following the instructions below. For each section, write a separate, appropriately named Java class file (e.g. Section01.java). Each class should include a **`main`** method to test the code written in the section. Unless otherwise indicated, points are distributed evenly across the sections. Remember that you may lose up to 10% of your grade for style, and gain up to 10% if you do something impressively creative.

Note: As mentioned in the lecture, labs will be approximately twice as large for the rest of the semester. This lab is worth 200 points.

Note: Giving or receiving electronic copies of CSC 171 lab solutions are violations of the academic honesty policy. More importantly, as each lab builds on the previous labs, and each project builds on the labs, if you do not implement the solutions yourself, you will not learn by doing, and will struggle with later assignments.

Lab Instructions

1. Write a Java method that accepts two integer parameters, *a* and *b*. In the body of the method, print the value of *a* and *b.* Swap the values and print a message to the command line showing that the values have been swapped. For example, executing the method with the values a=5 and b=7 should produce output similar to the following:

```
Before swap: a=5, b=7
After swap: a=7, b=5
```

2. Write a Java method that accepts an integer array with two elements and swaps the value of the elements *within the same array*. Remember, elements in an array can be assigned a new value like any other variable, e.g. **`myArray[0] = 4`**; For example, if the method is called with an integer array containing the elements { 3, 4 }, the array should contain the elements {4, 3} after the method is called. To receive full credit, **do not** use a second array to copy the elements.

3. Write a Java method that accepts an array of chars of arbitrary length and uses a **for loop** to reverse the order of the elements in the array. For example, passing an array containing the elements { ‘F’, ‘u’, ‘t’, ‘u’, ‘r’, ‘a’, ‘m’, ‘a’ } into the method should modify the elements such that they are ordered { ‘a’, ‘m’, ‘a’, ‘r’, ‘u’, ‘t’, ‘u’, ‘F’ } after the method is called. To receive full credit, **do not** use a second array.

4. Write a Java program that prompts the user to enter an integer. Use the integer as the size for a new double array. Use a **for loop** to prompt the user to enter a floating point number for each element in the array, and store the number in the array. Use a second **for loop** to, calculate the average of all of the numbers and print it. Your output should look something like the following:

```
How many numbers will you enter: 5
Enter a decimal value: 21.2
Enter a decimal value: 3.7
Enter a decimal value: 10.5
Enter a decimal value: 2.6
Enter a decimal value: 101.123
The average is 27.824599999999997
```

5. Pick a number between 1 and 10. Write a Java program that asks the user to guess your number. Use a **while loop** to continue prompting the user until they guess the number correctly. If the user guesses a number below 1 or greater than 10 print a message insulting their ability to count and end the game.

6. Pick a number between 1 and at least 100 (you may choose a larger number if you like). Write a Java program that uses a **do while** loop to prompt the user to guess until they guess correctly. Each time the user guesses print a message indicating that their guess is "too high," "too low," or "correct!" For example:

```
Guess a number between 1 and 100.
Enter your next guess: 22
Too low!
Enter your next guess: 50
Too high!
Enter your next guess: 37
Correct!
```

7. The java.util.Random class can generate an integer that is greater than 0 and less than n using the randomInt(int n) method. Write a public static Java method that accepts two integers *x* and *y* and will use the java.util.Random class to return a random number m such that x <= m <= y. Write a loop that tests your method 100 times with a small range (e.g. x is not more than 10 less than y) and prints the results to verify that the numbers are random and that all possible values are generated (including x and y).

8. Create an integer array of 10 elements. Use the method created in section 7 to fill each element with randomly generated numbers between 1 and 1000. To receive full credit you must call the method from a new class created for this section (**do not** copy and paste). Print all of the elements on a single line. Next, print the value and the

index of the minimum and maximum values in the array. Your output should look similar to the following:

```
633 543 226 525 43 650 901 736 203 757
min value is 43 and is at index 4
max value is 901 and is at index 6
```

9. Use the method created in section 7 as part of a simplified game of 21. To receive full credit you must call the method from a new class created for this section (**do not** copy and paste). Randomly "draw cards" by generating integers between 2 and 11. Each time a card is drawn, print the value and the total score (the cumulative value of all the cards). Continue until one of the following occurs, and then print a message describing the outcome:
    a. The score is exactly 21.
    b. The score exceeds 21 (busted).
    c. Five cards are drawn (Charlie).

    Your output should look something like the following:

```
Drew a 2.  Total score is 2.
Drew a 4.  Total score is 6.
Drew a 4.  Total score is 10.
Drew a 6.  Total score is 16.
Drew a 3.  Total score is 19.
Charlie!
```

10. Use a loop and switch statements to print out all of the words to the song "The Twelve Days of Christmas." Note that this is a cumulative song; each verse includes one new line plus all of the lines from the previous verse. Think carefully about how **`break`** statements affect the behavior of a switch statement. If you are unfamiliar with the song, try using a search engine to find "12 days of christmas lyrics." If implemented correctly, the first three verses should look similar to the following:

```
On the first day of Christmas
my true love sent to me:
A Partridge in a Pear Tree

On the second day of Christmas
my true love sent to me:
Two Turtle Doves
and a Partridge in a Pear Tree

On the third day of Christmas
my true love sent to me:
Three French Hens
Two Turtle Doves
```

and a Partridge in a Pear Tree

HAND IN

Before handing in, create two additional files in your lab directory:

1. Create a README that contains the following:
    a. Your contact information (name, class, lab session), TA name, and assignment number.
    b. A brief (one paragraph at most) description of the assignment.
    c. Instructions explaining how to run your code. While it is possible that some TAs will compile, run, and test your code in Eclipse, it is also possible that they will want to compile and run it from the command line. You should include instructions on how to compile and run each of the executable Java classes.
2. Create a file titled “SampleOutput” showing the results of running your code. (You may copy and paste from the Eclipse console). If appropriate, please include a comment above the output for each section (e.g. “#OUTPUT FOR SECTION 1”) and put a few blank lines between each section.

Hand in by uploading the compressed (i.e. “zipped”) folder containing your assignment files to Blackboard.