Villanova University
Final Examination
Semester: Fall  Year: 2020

Student Name _____

Student Id _____

| | |
|---|---|
| Course Abbreviation and Number | CSC1051 |
| Course Title | Algorithms and Data Structures I |
| Instructor | Xue Qin |

| | |
|---|---|
| Time Period | Start time: 2:30pm  End time: 5:00pm |
| Duration of Exam | 150 minutes |
| Number of Exam Pages | 11 pages (including cover page) |
| Exam Type | Programming Questions |
| Addition Materials | Open Book |

| Question | Point | Max | Question | Point | Max |
|---|---|---|---|---|---|
| 1 | | 5 | 5 | | 10 |
| 2 | | 5 | 6 | | 10 |
| 3 | | 7 | 7 | | 20 |
| 4 | | 8 | 8 | | 15 |
| Total | | 80 | Bonus | | 15 |

## Program 1 (Loop & Condition):

1. Please download **`OddNumbers.java`** in Final folder
2. In this program, you are required to print all the odd numbers from 1 to 100 (Include 1 and 100).
3. The output looks like this, please print all the numbers in one line:

```
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75
77 79 81 83 85 87 89 91 93 95 97 99 1 3 5 7 9 11 13 15
17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51
53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87
89 91 93 95 97 99
```

4. Compile and run your program to make sure there is no error.
5. Submit **`OddNumbers.java`** through blackboard under the "Final program question submission" in Final folder.

## Program 2 (Conditional Operator & Interactive Program):

1. Please download **`CompareNumbers.java`** in Final folder
2. In this program, please use **`Scanner`** to collect two integers and assign them to `n1` and `n2`
3. Then, replace the if-else statement with **conditional operator** but print out the same output.
4. The sample run looks like this:

```
input two integers:
4 6
4 is not greater than 6
```

```
input two integers:
10 9
10 is greater than 9
```

5. Compile and run your program to make sure there is no error.
6. Submit **`CompareNumbers.java`** through blackboard under the "Final program question submission" in Final folder.

## Program 3 (Switch Statement):

1. Please download **`LetterValue.java`** in Final folder
2. In this program, you will calculate the value of any given letter sequence from user input.
3. Please use **`Scanner`** to collect the letter sequence from user input and assign it to String `s`.
4. Please use switch statement when calculating the numeric value.
5. The numeric value for each letter is shown below, and the final value will the summation of all the numeric values.

| Letter | Numeric Value |
|---|---|
| **a, b** | 3 |
| **c, d** | 8 |
| **e, f** | 5 |
| **g, h** | 6 |
| **other** | 1 |

6. **Attention**, the input letter sequence is **case insensitive**.
7. The sample run looks like this:

```
input a letter sequence:
1Ag2&
The value is: 12
```

```
input a letter sequence:
EggApple6
The value is: 29
```

8. Compile and run your program to make sure there is no error.
9. Submit **`LetterValue.java`** through blackboard under the "Final program question submission" in Final folder.

## Program 4 (Advanced loop & condition):

1. Please download **`PrimeNumber.java`** in Final folder
2. In this program, please use **`Scanner`** to collect an integer (greater than 1) from user. No input validation is needed.
3. Design an algorithm to decide **if the integer is a prime number or not**.
4. The prime number is only divisible by 1 and itself. For example, the factors of prime number **`n`** are 1 and **`n.`**
5. The sample run looks like this:

```
input an integer greater than 1:
5
It's a prime number.
```

```
input an integer greater than 1:
6
It's not a prime number.
```

6. Compile and run your program to make sure there is no error.
7. Submit **`PrimeNumber.java`** through blackboard under the "Final program question submission" in Final folder.

## Program 5 (Data Conversion, Method definition & overloading):

1. Please download **`Average.java`** in Final folder
2. In this program, the main method has been completed, please **DO NOT** modify it.
3. In main method, we calculate two averages, `average1` and `avarage2`, by invoking the average method.
4. Please design the `average()` method by observing the method invocation.
5. The sample run looks like this:

```
average1: 2.5
average2: 4.333333333333333
```

6. Compile and run your program to make sure there is no error.
7. Submit **`Average.java`** through blackboard under the "Final program question submission" in Final folder.

## Program 6 (Array & Random Numbers):

1. Please download **`RandomArray.java`** in Final folder
2. In this program, you are going to create a **random length array with random value**. And calculate the average in the end.
3. The random length is between **4 to 5**.
4. The random value of each array element is **0 to 9**.
5. You need to print the array after created.
6. And please print the float point average value in the end.
7. You could use **`Random`** class or **`Math`** class to generate the random numbers.
8. The sample run looks like this:

```
Random array is:
2 4 3 8 0
The average is: 3.4
```

```
Random array is:
7 8 5 2
The average is: 5.5
```

```
Random array is:
3 0 6 1 3
The average is: 2.6
```

9. Compile and run your program to make sure there is no error.
10. Submit **`RandomArray.java`** through blackboard under the "Final program question submission" in Final folder.

## Program 7 (Class):

1. Please download **`Dice.java`** and **`GameRoom.java`** in Final folder
2. Class `Dice` represents a general dice concept. It could have multiple instances.

3. We assume all dices' face value start from 1, **so the number of faces for a dice will be the value of max face value.**
4. Please follow the instruction below to **finish the class design**
5. The class features are shown in the table below:

| Class Name | Attributes | Operations |
|---|---|---|
| `Dice` | `faceValue`<br>`MAX_VALUE` | `roll()`<br>`getValue()` |

6. Dice only have two attributes: `faceValue` and `MAX_VALUE`:
    a. **`faceValue`** is an integer, it represents the current value of dice.
    b. **`MAX_VALUE`** is a constant integer, it represents the max face value of a dice, in other words, the number of faces for this dice.
7. **About the operations:**
    a. Constructor
        i. Initialize the **`MAX_VALUE`** to the value of an integer parameter.
        ii. Initialize the **`faceValue`** to 0.
    b. Method `roll()`
        i. **Return integer, public method**
        ii. It will assign the **faceValue** to a new random value that between 1 and **MAX_VALUE.**
        iii. And it will return the faceValue in the end.
    c. Method `getValue()`.
        i. **Return integer, public method**
        ii. It will return the faceValue.

8. In **GameRoom** class, please do the following things:
    a. Create a 4-faces dice, a 6-faces dice, and an 8-faces dice.
    b. Rolling these three dices and print the face values.
    c. Calculate the summation of three values.
9. The sample run will look like this:

```
Rolling three dices:
2, 6, 2
sum: 10
```

```
Rolling three dices:
4, 5, 4
sum: 13
```

10. Compile and run your program to make sure there is no error.
11. Submit **`Dice.java`** and **`GameRoom.java`** through blackboard under the "Final program question submission" in Final folder.

## Program 8 (File Input and Output) :

1. Please download **`PrintStars.java`** and "`input.txt`" in Final folder.
2. In this program, we are going to read the numbers from "`input.txt`" and output the corresponding numbers of stars in "`output.txt`".
3. Please use **`Scanner`** to read "`input.txt`".
4. Please use **`PrintWriter`** to write "`output.txt`".
5. The example looks like this:

`input.txt`

```
4
3
2
1
5
2
```

`output.txt`

```
****
***
**
*
*****
**
```

6. Compile and run your program to make sure there is no error.
7. Submit **`PrintStars.java`** through blackboard under the "Final program question submission" in Final folder.

## Bonus Program (Exception & 2d array) :

1. Please download the program **`RaggedArray.java`** in Final folder
2. In this program, we are going to calculate the summation of a random column from a ragged 2d array.
3. Please try to run the main method yourself first.
4. You may encounter an exception during the execution, **please properly insert the try-catch block to handle the exception,** so that you can successfully calculate the summation.
5. The sample run looks like this:

```
The sum of column 0 is 22
```

```
The sum of column 2 is 7
```

6. Please download another copy of **`RaggedArray.java`** and rename it to **`RaggedArrayNew.java.`**
7. Try to modify the code without exception handling, and make it successfully calculate the summation.
8. Compile and run your programs to make sure there is no error.
9. Submit **`RaggedArray.java`** and **`RaggedArrayNew.java`** through blackboard under the "Final program question submission" in Final folder.