CSE3040: Java Programming (Fall 2020)
Homework 2
Due: October 27, 11:59PM (KST)

※ Write a Java code for each problem. Make sure your program satisfies all specified requirements. Carefully read the submission instructions at the end of the document.

Problem 6 (20 points)

• We would like to write a Java program that prints the Fibonacci sequence. The definition of Fibonacci sequence is as follows.

F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)

So starting from F(0), the sequence will be as follows:

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 …

• In order to fulfill the goal, we have implemented a part of the code. Complete the program so that the program runs correctly and produces the expected output.

```
public class Problem06 {
    public static void main(String[] args) {
        IntSequence seq = new FibonacciSequence();
        for(int i=0; i<20; i++) {
            if(seq.hasNext() == false) break;
            System.out.print(seq.next() + " ");
        }
        System.out.println(" ");
    }
}
```

• You MUST NOT modify class Problem06.

• In the added code, you must NOT use while loops or for loops.

• IntSequence must be defined as an interface.

• Expected output

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

Problem 7 (20 points)

• We would like to write a program that takes an integer from user and prints the integer in a binary number format.

• In order to fulfill the goal, we have implemented a part of the code. Complete the program so that the program runs without error and produces correct results.

```
public class Problem07 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        String str = in.nextLine();
        int num = Integer.parseInt(str);
        in.close();
        System.out.println("Integer: " + num);
        IntSequenceStr seq = new BinarySequenceStr(num);
        System.out.print("Binary number: ");
        while(seq.hasNext()) System.out.print(seq.next());
        System.out.println(" ");
    }
}
```

• You MUST NOT modify class Problem07.

• IntSequenceStr should be defined as an interface.

• Example run

```
Enter a positive number: 64
Integer: 64
Binary number: 1000000
```

• Tip: You may use Math.pow().

Problem 8 (20 points)

• We would like to write a Java program that calculates the sum of area of multiple shapes.

• Supported shapes are circles, squares, and rectangles.

• We have implemented a part of the code. Complete the missing part so that the program produces correct results.

```
public class Problem08 {
  public static void main(String[] args) {
    Shape[] arr = { new Circle(5.0), new Square(4.0),
                    new Rectangle(3.0, 4.0), new Square(5.0)};
    System.out.println("Total area of the shapes is: " + sumArea(arr));
  }
}
```

• You must not modify the main method in class Problem08. Code can be added outside the main method.

• If you define new classes, all instance variables must be defined as private variables.

• Expected output

```
Total area of the shapes is: 131.53981633974485
```

• You can use the constant math.PI for calculating area of circles.

Problem 9 (20 points)

• We would like to write a Java program that calculates distance between two points in a N-dimensional space.

• The program supports two distance metrics: EuclideanDistance and ManhattanDistance.

• Euclidean distance between two N-dimensional points (p_1, p_2, ..., p_n), (q_1, q_2, ..., q_n) is:

$$d = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

• Manhattan distance between two N-dimensional points (p_1, p_2, ..., p_n), (q_1, q_2, ..., q_n) is:

$$d = |p_1 - q_1| + |p_2 - q_2| + \cdots + |p_n - q_n|$$

• When the compared two points have different dimensions, then the printed distance is -1.

• We have implemented a part of the code. Complete the missing part so that we get the expected output when we run the program.

```
public class Problem09 {
 public static void main(String[] args) {
  Point p1 = new Point(new double[] {1.0, 2.0, 3.0});
  Point p2 = new Point(new double[] {4.0, 5.0, 6.0});
  System.out.println("Euclidean Distance: " + EuclideanDistance.getDist(p1, p2));
  System.out.println("Manhattan Distance: " + ManhattanDistance.getDist(p1, p2));
  Point p3 = new Point(new double[] {1.0, 2.0, 3.0});
  Point p4 = new Point(new double[] {4.0, 5.0});
  System.out.println("Euclidean Distance: " + EuclideanDistance.getDist(p3, p4));
  System.out.println("Manhattan Distance: " + ManhattanDistance.getDist(p3, p4));
 }
}
```

• You MUST NOT modify class Problem09.

• If you define new classes, all instance variables must be defined as private variables.

• Expected output

```
Euclidean Distance: 5.196152422706632
Manhattan Distance: 9.0
Euclidean Distance: -1.0
Manhattan Distance: -1.0
```

Problem 10 (20 points)

• We would like to write a Java program that compares two instances of class Points.

• Class Points has an array of floating-point numbers.

• Two Points objects are regarded as equal if the sums of the floating-point numbers are equal.

• Also, if the user prints a Points object, it should be printed as shown in the expected output.

• Class Problem10 is provided. Complete the missing part so that we get the expected output when we run the program.

```
public class Problem10 {
  public static void main(String[] args) {
    Points p1 = new Points(new double[] {1.0, 2.0, 3.0});
    Points p2 = new Points(new double[] {4.0, 5.0, 6.0});
    System.out.println(p1);
    System.out.println(p2);
    System.out.println(p1.equals(p2));
    Points p3 = new Points(new double[] {1.0, 4.0, 7.0});
    Points p4 = new Points(new double[] {3.0, 9.0});
    System.out.println(p3);
    System.out.println(p4);
    System.out.println(p3.equals(p4));
    Points p5 = new Points(new double[] {1.0, 2.0});
    Points p6 = null;
    System.out.println(p5);
    System.out.println(p6);
    System.out.println(p5.equals(p6));
  }
}
```

• You MUST NOT modify class Problem10.

• You do not need to implement method hashCode.

• Example output

```
[sum of points: 6.0]
[sum of points: 15.0]
false
[sum of points: 12.0]
[sum of points: 12.0]
true
[sum of points: 3.0]
null
false
```

※ Java Naming Convention (same as hw1)

When you write Java code it is good to follow the Java Naming Convention, which is a rule for naming variables, constants, methods, and classes.

(1) Variable Names

Start with a lowercase letter and use uppercase letters as separates. Do not use under bars (‘_’).

```
int myVar;
```

(2) Constant Names

Use all capital letters and use under bars as separators.

```
final int MY_CONST = 1;
```

(3) Method Names

Start with a lowercase letter and use uppercase letters as separators. Do not use under bars.

```
int myMethod()
```

(4) Class Names

Start with an uppercase letter and use uppercase letters as separators. Do not use under bars.

```
Public class MyClassName
```

※ Submission

※ Carefully read this part and follow the instructions. Not properly following the instructions here may result in point deduction.

(1) For this homework, you are going to submit only the .java files. You are going to submit one .java file for each problem.

(2) For problem 6, your file name should be Problem06.java. This means that your public class name should also be Problem06. For other problems, you should name your .java files this way.

(3) Once you are ready to submit, you should make the .java files into a single zip file named cse3040_hw2_20180001.zip. The numbers in the file name should be your student ID.

When you make the zip file, make sure that you are not using subfolders inside the zip file. When the zip file is extracted, the .java files should appear without having any subdirectory structures.

(4) Submit your zip file on the cyber campus.

※ Evaluation Criteria

Your solution to each problem will be tested with various test cases. You will get full points if your program passes all tests. If not, points will be given based on percentage of test cases passed.

For this homework, each problem is worth 20 points. The perfect score is 100.

※ Academic Integrity

- You should write your own code. You can discuss ideas with other students but must not copy their work. You can also get help from the Internet, but you must not copy the source code from the Internet either. We have a duplicate check program which tests whether your source code is similar to other students' code as well as codes that are on the Internet.

- Duplicate work will receive zero grade.

※ Late Policy

- 10% of the score is deducted for each day, up to three days. Submissions are accepted up to three days after the deadline.