CORNERSTONE
International Community College of Canada

Module 2
Java Development

Assignment 9- MADP202- July 30, 2019

Due: Tuesday July 30, 6:30 pm

Submission: Please create a pull-request to the corresponding git branch.

Late submission Penalty Policy:

- Lack of submission of 3 or more assignments: 50% of your entire assignments (7.5% of your course mark).
- Lack of submission of 5 or more assignments: 100% of your entire assignments (15% of your course mark).
- Remember you need to get at least 70% to pass the course.

Topics:

Inner/Local/Anonymous/Nested Class

Problem 1: Inner Class

Consider the class Foo1.

```
1.  public class Foo1 {
2.
3.      private int bar1;
4.
5.      public Foo1(int x) {
6.          bar1 = x;
7.      }
8.  }
```

- Add a class called Foo2 as an inner class of the class Foo1. The class Foo2 has the 2 two following instance method and constructor:

```
1.  public Foo2(int x) {
2.              super(x * 10);
3.   }
4.
5.   public void show() {
6.              System.out.println("bar1 = " + bar1);
7.              System.out.println("((Foo1)this).bar1 = " + ((Foo1)this).bar1);
8.              System.out.println("Foo1.this.bar1 = " + Foo1.this.bar1);
9.  }
```

- Add the following class Test64 to your project and run it. What is the output?

```
1.  public class Test64 {
2.
3.      public static void main(String[] args) {
4.          Foo1 f1 = new Foo1(5);
5.          Foo1.Foo2 f2 = f1.new Foo2(6);
6.          f2.show();
7.      }
8.
9.  }
```

- Remove the Test64 class from your Project and add it as an inner class to the Foo1 class. Fix the bugs if you get any. Run it again. What would be the output?

Problem 2: Anonymous class

Consider the following class:

```
1.  abstract class Shape{
2.    abstract float perimeter();
3.    abstract float area();
4.  }
```

Now consider the following class and complete the implementation of the anonymous classes.

```
1.  class ApplicationDriver{
2.
3.  enum ShapeType
4.      {
5.          Rectangle,
6.          Triangle,
7.          Circle,
8.          Square,
9.          Custom
10.     }
11.
12.  public static void main(String args[]){
13.
14.   int[] s1Sides = {10,20};
15.   Shape s1=new Shape(ShapeType.Rectangle, s1Sides){
16.
17.   //complete the implementation of the anonymous inner class here
18.
19.   System.out.println("S1's permiter is: "s1.perimeter());
20.   System.out.println("S1's area is: "s1.area());
21.
22.  }
23.
24.
25.   int[] s2Sides = {10};
26.   Shape s2=new Shape(ShapeType.Square, s2Sides){
27.
```

```
28.   //complete the implementation of the anonymous inner class here
29.
30.   System.out.println("S2's permiter is: "s2.perimeter());
31.   System.out.println("S2s area is: "s2.area());
32.
33.  }
34.
35.
36.
37.   int[] s3Sides = {12};
38.   Shape s3=new Shape(ShapeType.Circle, s3Sides){
39.
40.   //complete the implementation of the anonymous inner class here
41.
42.   System.out.println("S3's permiter is: "s3.perimeter());
43.   System.out.println("S3s area is: "s3.area());
44.
45.  }
46.
47.
48. int[] s4Sides = {8, 12, 12};
49.   Shape s4=new Shape(ShapeType.Triangle, s4Sides){
50.
51.   //complete the implementation of the anonymous inner class here
52.
53.   System.out.println("S4's permiter is: "s4.perimeter());
54.   System.out.println("S4s area is: "s4.area());
55.
56.  }
57.
58. }
```

Problem 3: Anonymous class as a parameter of a method

We are going to implement a problem similar to Problem 2, but this time we are going to use inline anonymous class. Consider the same Shape class as in Problem 2. Change the Application Drive class as following and add a MyGeometry class as following. Complete implementation of the main method.

```
1.  class ApplicationDriver{
2.
3.  class MyGeometry
4.  {
5.          public void calculateAreaAndPerimeter(Shape shape)
6.         {
7.               if(!Custom.class.isInstance(shape))
8.              {
9.                    shape.perimeter();
10.                   shape.area();
11.             }
12.             else
13.            {
14.                   shape.perimeter();
15.            }
16.       }
17.
18. }
19.
20. enum ShapeType
21.     {
22.         Rectangle,
23.         Triangle,
24.         Circle,
25.         Square,
26.         Custom
27.     }
28.
29.  public static void main(String args[]){
30.
31.   MyGeometry mg = new  MyGeometry();
32.
33.   int[] s1Sides = {10,20};   //10 is width and 20 is length
34.   ShapeType s1Type = ShapeType.Rectangle;
35.   //use mg.calculateAreaAndPerimeter to calculate perimeter and area for the Rectangle
    above
36.
37.
```

```
38.   int[] s2Sides = {10};   //10 is the width
39.   ShapeType s2Type = ShapeType.Square;
40.   //use mg.calculateAreaAndPerimeter to calculate perimeter and area for the Square abo
    ve
41.
42.
43.   int[] s3Sides = {12};  //12 is the radius
44.   ShapeType s3Type = ShapeType.Circle;
45.   //use mg.calculateAreaAndPerimeter to calculate perimeter and area for the Circle abo
    ve
46.
47.
48.   int[] s4Sides = {8, 12, 12};
49.   ShapeType s3Type = ShapeType.Custom;
50.   //use mg.calculateAreaAndPerimeter to calculate perimeter and area for the Custom abo
    ve
51.
52.  }
53. }
```

Problem 4: Nested class

A zero-indexed array A consisting of N integers is given. Rotation of the array means that each element is shifted right by one index, and the last element of the array is also moved to the first place.

For example, the rotation of array A = [3, 8, 9, 7, 6] is [6, 3, 8, 9, 7]. The goal is to rotate array A K times; that is, each element of A will be shifted to the right by K indexes.

Write a method called solution, which takes and array and a number k and rotates the array for k times and returns the result: The signature of the method *solution* is given as below:

```
1.  public static int[] solution(int[] A, int K)
2.  {
3.     //body of the method
4.  }
```

Create an ApplicationDriver class with a *main* method. The main method will class the *solution* method.
Based on Single Responsibility principle, the class ApplicationDriver only contains the *main* method and cannot contain the *solution* method.

- Implement the *solution* method as a method in an inner class in the *ApplicationDriver* class.
- Test your implementation properly.

N and K are integers within the range [0..100];
each element of array A is an integer within the range [−1,000..1,000].