CSE 1030 Introduction to Computer Science II A Solution of Test 1

$1 \quad (20 \text{ marks})$

Explain why we add a private default constructor to a utility class?

When the Java compiler encounters a class without any constructor at all, it automatically adds a public constructor to it. To prevent this from happening, we add a private constructor.

2 (20 marks)

Consider the Circle class whose API can be found at the end of this test. Three different implementers (1, 2, and 3) declare the attributes as follows.

- private double radius; private double area;
- 2. double radius;
- 3. private double area;

Which choice do you like best, 1, 2, or 3? You have to explain your answer to receive any marks.

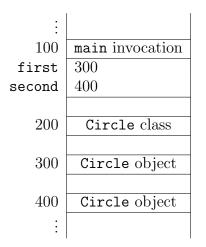
I like 3 best. I do not like 1 because either radius or area is redundant (given the area I can get the radius and given the radius I can get the area). I do not like 2 since non-final attributes should be declared private.

3 (20 marks)

Consider the Circle class whose API can be found at the end of this test. Consider the following main method.

```
Circle first = new Circle(0.0);
Circle second = new Circle(1.0);
output.println(first.compareTo(second));
```

Just before the compareTo method is invoked, memory can be depicted as follows.



Draw the invocation block (and any related blocks) for the compareTo invocation. Only draw those parts that are new or changed.

÷	
400	compareTo invocation
this	300
other	400
÷	

4 (20 marks)

Consider the Circle class whose API can be found at the end of this test. Consider the following compareTo method of the Circle class.

```
public int compareTo(Circle other)
{
    if (this.getClass() == other.getClass())
    {
        return (int) (this.getArea() - other.getArea());
    }
    else
    {
        return -1;
    }
}
```

Mention *three* aspects that can be improved and describe **how** they can be improved.

- Use a single return statement.
- There is no need for this.getClass() == other.getClass() since other is a Circle.
- The app

```
final double DIFFERENCE = 0.3;
Circle one = new Circle(0);
Circle two = new Circle(DIFFERENCE);
Circle three = new Circle(2 * DIFFERENCE);
output.println(one.compareTo(two));
output.println(two.compareTo(three));
output.println(one.compareTo(three));
```

produces the output

0 0 -1

and, hence, compareTo does not satify one of the properties given in the Comparable API.

```
public int compareTo(Circle other)
{
  int difference;
  if (this.getArea() < other.getArea())</pre>
  {
    difference = -1;
  }
  else if (this.getArea() > other.getArea())
  {
    difference = 1;
  }
  else
  {
    difference = 0;
  }
  return difference;
}
```

5 (20 marks)

(a) What is the *singleton* design pattern?

Only a single instance of the class can be created.

- (b) Describe **how** the implementation of the **Circle** class needs to be modified so that the **Circle** class becomes a singleton.
 - Make the constructor private.
 - Introduce a static attribute that keeps track of the created Circle.

```
private static Circle instance = new Circle(0.0);
```

• Introduce a static method getInstance which returns the Circle.

```
public static Circle getInstance()
{
   return Circle.instance;
}
```