CS2030S Recitation: Problem Set #2 — Wk 4, Sem 1, 2022/23

1. We would like to design a class `Square` that inherits from `Rectangle`. A `Square` has the constraint that the four sides are of the same length. Consider the `Rectangle` class below:

```java
/*
    Public Class must be in its own file
    with the filename the same as the class name
*/
public class Rectangle {
  private double width;
  private double height;

  public Rectangle(double width, double height) {
    this.width = width;
    this.height = height;
  }

  @Override
  public String toString() {
    return "Height: " + this.height + " Width: " + this.width;
  }
}
```

   (a) How should `Square` be implemented to obtain the following output from JShell?

```
jshell> new Square(5);
$.. ==> Height: 5.0 Width: 5.0
```

   (b) Now implement two separate methods to set the width and height of the `Rectangle` class as follows:

```java
public void setHeight(double height) {
  this.height = height;
}
public void setWidth(double width) {
  this.width = width;
}
```

   What undesirable design issues would this present?

```java
class Vector2D {
    private double[] coord2D;
    // code omitted
}
```

   (c) Now implement two overriding methods in the `Square` class as follows:

```java
public void setHeight(double height) {
  super.setHeight(height);
  super.setWidth(height);
}
public void setWidth(double width) {
  super.setHeight(width);
  super.setWidth(width);
}
```

Do you think that it is now sensible to have `Square` inherit from `Rectangle`? Or should it be the other way around? Or maybe they should not inherit from each other?

2. Given the following interfaces.

```java
public interface Shape {
  public double getArea();
}

public interface Printable {
  public void print();
}
```

   (a) Suppose the class `Circle` implements both interfaces above. Given the following program fragment,

```java
Circle c = new Circle(new Point(0, 0), 10);
Shape s = c;
Printable p = c;
```

   Are the following statements allowed? Why do you think Java does not allow some of the following statements?

   i. `s.print();`

   ii. `p.print();`

   iii. `s.getArea();`

   iv. `p.getArea();`

   (b) Someone proposed to re-implement `Shape` and `Printable` as abstract classes instead. Would this work?

   (c) Can we define another interface `PrintableShape` as follows:

```java
public interface PrintableShape extends Printable, Shape {
}
```

   and let the class `Circle` implements `PrintableShape` instead?

3. Using examples of overriding methods, illustrate why a Java class cannot inherit from multiple parent classes, but can implement multiple interfaces.