CS2030S Programming Methodology II
Semester 2 2021/2022

2 & 3 February 2022
Problem Set #2

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:

```
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
```

super(side, side)

   (b) Now implement two separate methods to set the width and height of the rectangle:

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

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

   What undesirable design issues would this present?

Square no longer square

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

```
@Override
public void setHeight(double height) {
    super.setHeight(height);
    super.setWidth(height);
}
```

```
@Override
public void setWidth(double width) {
    super.setHeight(width);
    super.setWidth(width);
}
```

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

Yes

2. Given the following interfaces.

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

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

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

```
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();

N, Y, Y, Y

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

No, can only inherit from one class

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

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

No, circle has to go through shape or printable

and let class `Circle` implement `PrintableShape` instead?

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

Class: parent class needs to be instantiated, so there will be conflicting methods
Interface: the methods are just copied over, so it is just checking method signature