CS2030 Programming Methodology II
Semester 1 2024/2025

Week of 9 September – 13 September 2024
Problem Set #3
Interface and Polymorphism

1. Given the following interfaces.

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

interface Scalable {
    public Scalable scale(double factor);
}
```

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

```
Circle c = new Circle(10);
Shape s = c;
Scalable k = c;
```

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

   i. `s.scale(0.5);`
   ii. `k.scale(0.5);`
   iii. `s.getArea();`
   iv. `k.getArea();`

   (b) Do the following statements compile?

   i. `c.scale(0.5).getArea()`
   ii. `k.scale(0.5).getArea()`

   (c) How about defining another combined interface `ScalableShape` as

```
interface ScalableShape {
    public Scalable scale(double factor);
    public double getArea();
}
```

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

2. During the lecture, we have seen how we can create `Circle` and `Rectangle` as concrete implementations of the `Shape` interface, and pass it to the `findVolume` method:

```
double findVolume(Shape shape, double height) {
    return shape.getArea() * height;
}
```

Now your friend decided to create `Shape` as a class to represent both a circle and rectangle:

```
class Shape {
    private final String type;
    private final double a;
    private final double b;

    Shape(double radius) {
        this.type = "Circle";
        this.a = radius;
        this.b = 0;
    }

    Shape(double length, double width) {
        this.type = "Rectangle";
        this.a = length;
        this.b = width;
    }

    double getArea() {
        if (this.type.equals("Circle")) {
            return Math.PI * this.a * this.a;
        } else {
            return this.a * this.b;
        }
    }

    public String toString() {
        if (this.type.equals("Circle")) {
            return "Circle with radius " + this.a;
        } else {
            return "Rectangle " + this.a + " x " + this.b;
        }
    }
}
```

which when passed to `findVolume` would still return the same outcome. Justify why this is considered bad program design? *Hint*: what if we need to include a `Square` into our implementation?

3. Complete the method `and` that takes in two `IntPredicate p1` and `p2` and returns a `IntPredicate` that evaluates to `true` if and only if both `p1` and `p2` evaluate to `true`.

```
IntPredicate and(IntPredicate p1, IntPredicate p2) { ... }
```

Express your solution in three different ways:

(a) as an implementation of a concrete class;

(b) as an implementation of an anonymous inner class;

(c) as a lambda expression.