Interfaces

Overview

Interfaces are like contracts between the interfaces and the classes that implement them. They define characteristics (functions, variables) that must be implemented by the class. You can think of it as interfaces define what a class must do, but not how it's done. That is up to the classes themselves.

```
interface Shape {
    // Returns the area of the shape
    double area();
    // Returns the circumference/perimeter of the shape
    double circumference();
}
public Square implements Shape {
    private double length;
    public double area() {
        return length * length;
    }
    public double circumference() {
        return 4 * length;
    }
}
public Circle implements Shape {
    private double radius;
    public double area() {
        return Math.pow(radius, 2) * Math.PI;
    }
    public double circumference() {
        return 2 * radius * Math.PI;
    }
}
```

We can see in this example that the Square and Circle classes both implement the Shape interface. Since these two classes implement the Shape interface, they must provide implementations for the functions specified in the Shape interface, which you can see above. We use interfaces to achieve abstraction (something we coders luuuuv). This is why all of the methods in an interface don't have implementations, since it's up to the class to provide implementations. They are also useful for helping us quickly identify certain functionalities of classes since if a class implements a specific interface, then it must have a specific set of functionalities (methods).
***Note: any variable in an interface must be public, static, and final**

TL;DR

1. Interfaces provide specific methods/functionalities a class must have
2. Methods in the interface **must not** have bodies/implementations
3. Classes that implement an interface **must** provide implementation for all methods in that interface
4. All variables in an interface must be **public, static, and final**