AP CS: Lesson 19: Interfaces and compareTo

Name: _____ Period: ___

Interfaces: (BJP Chapter 9.5)

Interface: A list of methods that a class can implement.

- Interfaces give you an is-a relationship *without* code sharing.
  i.e. A Rectangle object can be treated as a Shape but has no common code.

```
public interface name {
    public type name(type name, ..., type name);
    public type name(type name, ..., type name);
    ...
}
```

Example:

```
public interface Shape {
    public double area();
    public double perimeter();
}
```

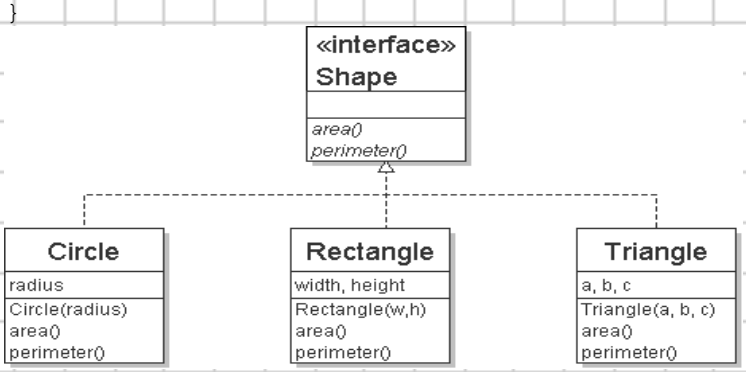
A class can declare that it *implements* an Interface.

This means the class must contain each of the abstract methods in that interface. (Otherwise, it will not compile.)

```
public class name implements interface {
    ...
}
```

Example:

```
public class Circle implements Shape {
    ...
}
```



- Arrow goes up from class to interface(s) it implements.
  - There is a supertype-subtype relationship here;
    e.g., all Circles are Shapes, but not all Shapes are Circles.

AP CS: Lesson 19: Interfaces and compareTo

CompareTo: (BJP Chapter 10.2)

The standard way for a Java class to define a comparison function for its objects is to define a compareTo method.

```
this.compareTo(object parameter_name)
```

Compares this with the parameter

- returns negative number if this is less than parameter
- returns zero if they are equal
- returns positive number if this is greater than parameter

Specifically a call of **A**.compareTo(**B**) will return:

a value < 0 if **A** comes "before" **B** in the ordering,
a value > 0 if **A** comes "after" **B** in the ordering,
or = 0 if **A** and **B** are considered "equal" in the ordering.

Syntax:

```
public interface Comparable {
    public int compareTo(Object obj);
}

public class name implements Comparable<name> {
    ...
    public int compareTo(name other) {
        ...
    }
```

Examples:

In the String class, there is a method:

```
public int compareTo(String other)
```

So `compareTo` can be used as a test in an `if` statement.

```
String a = "alice";
String b = "bob";
if (a.compareTo(b) < 0) {  // true
    ...
}

public class Point implements Comparable<Point> {
    private int x;
    private int y;
    ...
    // sort by x and break ties by y
    public int compareTo(Point other) {
        if (x < other.x) {
            return -1;
        } else if (x > other.x) {
            return 1;
        } else if (y < other.y) {
            return -1;   // same x, smaller y
        } else if (y > other.y) {
            return 1;    // same x, larger y
        } else {
            return 0;    // same x and same y
        }
    }
```