# Polymorphism Summary

By Zachary Stewart, Zoe Rudich, and John Kulczak

The word “polymorphism” refers to a principle of biology that describes an organism or species that has various life stages or forms. Polymorphism in Java describes classes of a similar sort.

In polymorphism, there is an original class, which has a design to which other classes adhere. The lower classes (known as subclasses) each have their own unique attributes that separate themselves from one another, but not from the original class design. The way one can determine if a class is a subclass is to see if after the class name the word “extends” appears followed by an original class name. This brings to mind the idea of inheritance.

Example of Creating a Subclass:

```
public class Hammerhead extends Shark {

//Shark can be considered the “original class” while Hammerhead is the class that adheres to “Shark”

}
```

```
Shark one = new Hammerhead(); //This is allowed because Hammerhead extends Shark

Hammerhead two = new Shark(); //this is not allowed because Hammerhead is a subclass of Shark

Shark.add(one); //one can add subclasses to lists of its superclass
```

Polymorphism can also refer to methods that are overridden in at least one subclass. For example, if the Shark Class had a method getHabitat(), and getHabitat() was overridden in the subclasses HammerHead and GreatWhite, that method would be considered polymorphic.

Polymorphism also exists for Interfaces. Each class that implements an interface IS_A type of the interface it extends, but an interface cannot make its own object. This is similar to saying that all squares are quadrilaterals, but not all quadrilaterals are squares.