Polymorphism

By Daniel Schnipke, Daniel Lehman, Ian Ludden, and Tyler Whitehouse

Sources

Big Java, Fifth Edition (Horstmann); Section 9.4

Oracle.com – The Java Tutorials: Polymorphism

Summary

Definition:

Broadly, polymorphism means "the ability to take many forms." Polymorphism is the capability of a method of a parent class to be overridden by each subclass and operate uniquely for each subclass. In other words, the same method can be called for an object of each subclass, and the method will operate using the definition specified by each subclass.

Uses:

The purpose of polymorphism is to allow similar objects of classes that all extend the same class to call the same method. This is especially useful when iterating through a list or array in which each object is of the type of the parent class. For instance:

```
Vehicle[] inventory = {new Corvette(), new F150(), new Camry()};
for(Vehicle auto : inventory) {
        auto.build();
}
```

In the above example, each Vehicle in the inventory calls the build() method, but since the build() method is defined differently within each subclass, each call uses the build() method of that Vehicle's subclass.

Hint

You can use the "super" keyword to represent the parent class and access its constructors and methods.

Example

In the example on the reverse of this page, Animal is the parent class, with subclasses Dog and Bird. Each class overrides the printDescription() method of Animal to add its own unique information.

```java
public class Animal {
	private String name, sound;
	public Animal(String name, String sound) {
		this.name = name;
		this.sound = sound;
	}
	public void printDescription() {
		System.out.println(this.name + " the " + this.getClass().getSimpleName()
				+ " goes " + this.sound + "");
	}
}

class Dog extends Animal {
	private String furColor;
	public Dog(String name, String sound, String furColor) {
		// The keyword super calls the parent class; in this case, it uses the
		// constructor of Animal.
		super(name, sound);
		this.furColor = furColor;
	}
	public void printDescription() {
		// The keyword super calls the parent class, using the definition of
		// printDescription() in Animal.
		super.printDescription();

		// New print statement specific to the Dog class.
		System.out.println(this.name + "'s fur color is " + this.furColor
		+ ".\n");
	}
}

class Bird extends Animal {
	private String featherColor;
	public Bird(String name, String sound, String featherColor) {
		super(name, sound);
		this.featherColor = featherColor;
	}
	public void printDescription() {
		super.printDescription();
		System.out.println(this.name + "'s feather color is "
		+ this.featherColor + ".\n");
	}
}

//This class creates three Animal objects and prints their descriptions to
demonstrate polymorphism.
public class PolymorphismTest {
	public static void main(String[] args) {
		Animal dog1 = new Dog("Fido", "\"growl.\"", "black");
		Animal bird1 = new Bird("Polly", "\"Polly wanna cracker?\"", "white");
		Animal dog2 = new Dog("Lassie", "\"bark!\"", "brown and white");
		dog1.printDescription();
		bird1.printDescription();
		dog2.printDescription();
	}
}
```