## Java Polymorphism

Previous

Next >

## Java Polymorphism

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Like we specified in the previous chapter; **Inheritance** lets us inherit attributes and methods from another class. **Polymorphism** uses those methods to perform different tasks. This allows us to perform a single action in different ways.

For example, think of a superclass called **Animal** that has a method called **animalSound()**. Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):

## Example

```
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}

class Dog extends Animal {
  public void animalSound() {
    System.out.println("The dog says: bow wow");
}
```

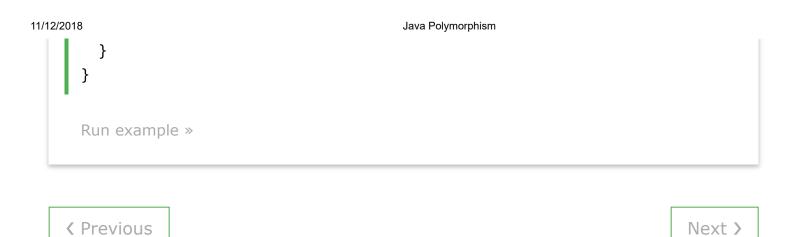
```
}
}
```

Remember from the <u>Inheritance chapter</u> that we use the <u>extends</u> keyword to inherit from a class.

Now we can create Pig and Cat objects, and call the animalSound() method on both of them:

## Example

```
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}
class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}
class Dog extends Animal {
  public void animalSound() {
    System.out.println("The dog says: bow wow");
  }
}
class MyMainClass {
  public static void main(String[] args) {
    Animal myAnimal = new Animal(); // Create a Animal object
    Animal myPig = new Pig(); // Create a Pig object
    Animal myDog = new Dog(); // Create a Dog object
    myAnimal.animalSound();
    myPig.animalSound();
    myDog.animalSound();
```



Copyright 1999-2018 by Refsnes Data. All Rights Reserved.