



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 Inheritance chapter that we use the **extends** keyword to inherit from a class.

Now we can create **Pig** and **Dog** 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 Main {  
    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();  
    }  
}
```

Try it Yourself »

Why And When To Use "Inheritance" and "Polymorphism"?

- It is useful for code reusability: reuse attributes and methods of an existing class when you create a new class.

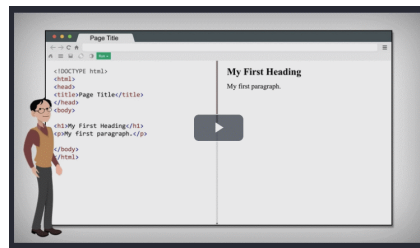
◀ Previous

Next ▶

ADVERTISEMENT

NEW

We just launched
W3Schools videos



Explore now

COLOR PICKER



Get certified
by completing
a Java
course today!



Get started

CODE GAME



Play Game

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT



Report Error

Spaces

Pro

Buy Certificate

Top Tutorials

[HTML Tutorial](#)

[CSS Tutorial](#)

[JavaScript Tutorial](#)

- How To Tutorial
- SQL Tutorial
- Python Tutorial
- W3.CSS Tutorial
- Bootstrap Tutorial
- PHP Tutorial
- Java Tutorial
- C++ Tutorial
- jQuery Tutorial

Top References

- HTML Reference
- CSS Reference
- JavaScript Reference
- SQL Reference
- Python Reference
- W3.CSS Reference
- Bootstrap Reference
- PHP Reference
- HTML Colors
- Java Reference
- Angular Reference
- jQuery Reference

Top Examples

- HTML Examples
- CSS Examples
- JavaScript Examples
- How To Examples
- SQL Examples
- Python Examples
- W3.CSS Examples
- Bootstrap Examples
- PHP Examples
- Java Examples
- XML Examples
- jQuery Examples

Get Certified

- HTML Certificate
- CSS Certificate
- JavaScript Certificate
- Front End Certificate
- SQL Certificate
- Python Certificate
- PHP Certificate
- jQuery Certificate
- Java Certificate
- C++ Certificate
- C# Certificate
- XML Certificate

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using W3Schools, you agree to have read and accepted our terms of use, cookie and privacy policy.

Copyright 1999-2022 by Refsnes Data. All Rights Reserved.
W3Schools is Powered by W3.CSS.

