



Understanding OOP and Java

Object-Oriented Programming Concepts

Written by Paul Pu All Right
Reserved www.torontocollege.com



Object-Oriented Programming Concepts

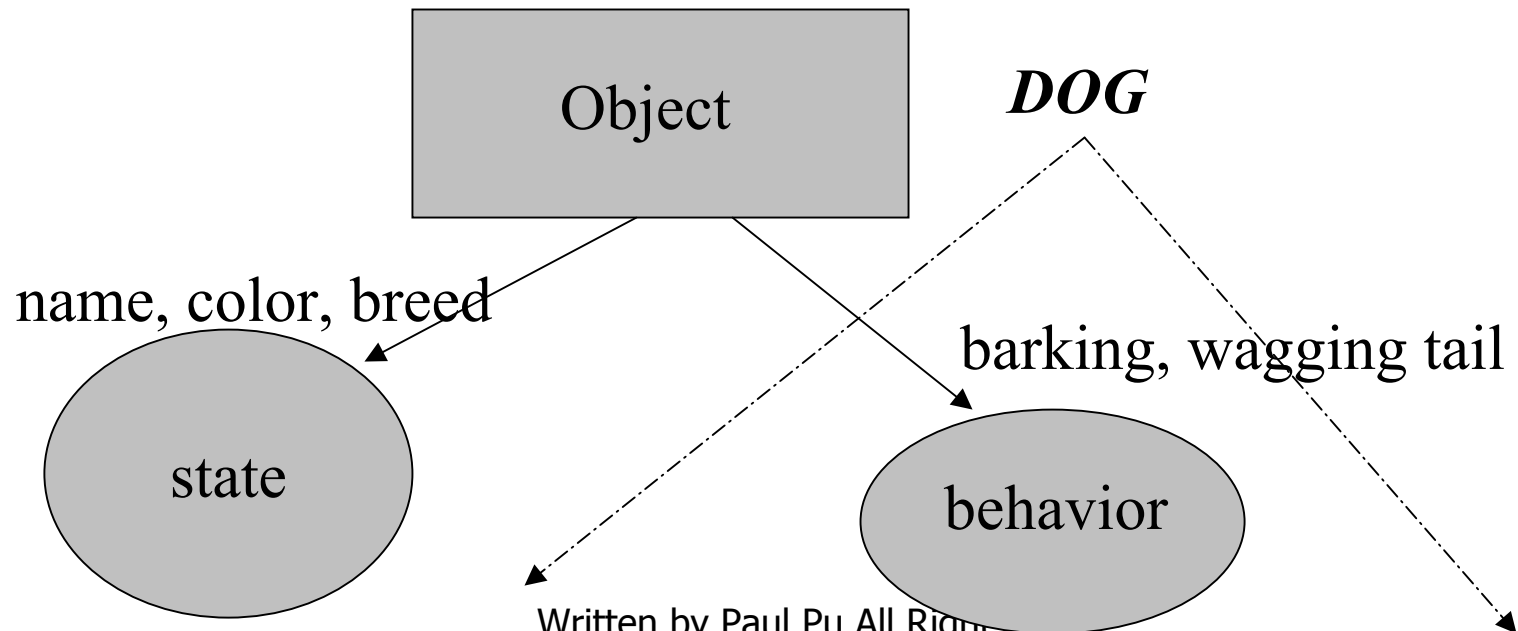
What is Objects?

- **An object** is a software bundle of related variables and methods. Software objects are often used to model real-world objects you find in everyday life.
- **Objects** are key to understanding OOP technology. You can look around you now and see many examples of real-world objects: your dog, your desk, your television set, your bicycle.



What is Objects?

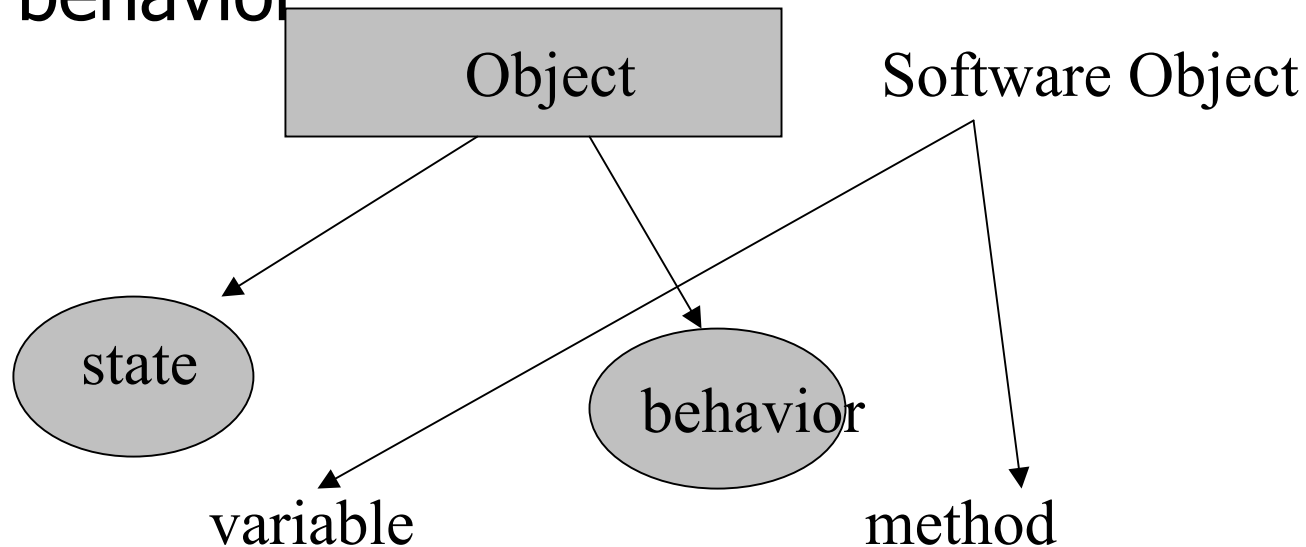
- These real-world objects share two characteristics: They all have *state* and *behavior*





What is Objects?

- **Software objects** are modeled after real-world objects in that they too have state and behavior





Dog class

- public class Dog {
- public int age;
- public String color;
- public String name;
- public void barking(){
- System.out.println("Wow,wow");
- }



Dog class

```
public void setName(String myName){  
    name=myName;  
}  
public void setAge(int myAge) {  
    age=myAge;  
}
```



Dog class

```
public void setColor(String myColor) {  
    color=myColor;  
}  
public String getName(){  
    return name;  
}
```



Dog class

```
public int getAge(){  
    return age;  
}  
public String getColor() {  
    return color;  
}  
}
```



CreateDog

```
public class CreateDog {  
    public static void main(String[] args) {  
        Dog myDog=new Dog();  
        Dog yourDog=new Dog();  
        myDog.name="tt";  
        myDog.setColor("Yellow");  
        myDog.setAge(10);  
        System.out.println("My dog's name  
is"+myDog.getName());  
    }  
}
```



CreateDog

```
System.out.println("My dog's age is"+myDog.getAge());  
System.out.println("My dog's color  
is"+myDog.getColor());  
yourDog.setName("ff");  
yourDog.setColor("Yellow");  
yourDog.setAge(5);  
System.out.println("your dog's name  
is"+yourDog.getName());
```



CreateDog

```
System.out.println("your dog's age  
is"+yourDog.getAge());  
System.out.println("your dog's color  
is"+yourDog.getColor());  
}  
}
```



Create Cat class

Assignment 1

Create Cat Class

Create CreateCat Class



Create Base_Calculator

Method:

add method

subtract method



Create a Common_Calculator

Methods

add

subtract

multiply

divide



Base_Calculator

```
package test;
```

```
/**
```

```
* Title:      Test how to use JBuilder
```

```
* Description: This is my first Java class.I will test my Java code  
               inside JBuilder
```

```
* Copyright:  Copyright (c) 2001
```

```
* Company:    99it
```

```
* @author Paul Pooh
```

```
* @version 1.0
```

```
*/
```



Base_Calculator

```
public class Base_calculator {  
    public double add(double a,double b) {  
        return a+b;  
    }  
    public double subtract(double a,double b) {  
        return a-b;  
    }  
}
```



Use_calculator

```
package test;
```

```
/**
```

```
 * Title:      Test how to use JBuilder
```

```
 * Description: This is my first Java class.I will test my Java code  
                inside JBuilder
```

```
 * Copyright:  Copyright (c) 2001
```

```
 * Company:    99it
```

```
 * @author Paul Pooh
```

```
 * @version 1.0
```

```
 */
```



Use_calculator

```
public class Use_calculator {  
    public static void main(String[] args) {  
        double a = Double.parseDouble(args[0]);  
        double b= Double.parseDouble(args[1]);  
        Base_calculator calculator=new Base_calculator();  
        double x=calculator.add(a,b);  
        double y=calculator.substract(a,b);  
        System.out.println("a+b="+x);  
        System.out.println("a-b="+y);  
    }  
}
```



Use_Calculator



$$X=18773.77+9394738$$

$$Y= 18773.77+9394738$$

X? Y?



Regular_Calculator

```
package test;
public class Regular_calculator extends Base_calculator
{

    public double divide(double a,double b) {
        return a/b;
    }
    public double times(double a,double b) {
        return a*b;
    }
}
```



Use Regular Calculator

```
public class Use_Regular_Calculator {  
    public static void main(String[] args){  
        double a = Double.parseDouble(args[0]);  
        double b= Double.parseDouble(args[1]);  
        Regular_calculator calculator=new Regular_calculator();  
        double x=calculator.add(a,b);  
        double y=calculator.substract(a,b);  
        double z=calculator.times(a,b);  
        double w=calculator.divide(a,b);  
    }  
}
```



Use Regular Calculator

```
System.out.println(a+" "+b+"="+x);
    System.out.println(a+" "+b+"="+y);
    System.out.println(a+" "+b+"="+z);
    System.out.println(a+" "+b+"="+w);
}
}
```



Use_Vector.java

```
import java.util.Vector ;  
public class Use_Vector {  
public static void main(String[] args) {  
Vector box=new Vector();  
    Dog myDog=new Dog();  
    Dog yourDog=new Dog;  
        myDog.setAge(10);  
        yourDog.setAge(5);  
        box.addElement(myDog);  
        box.addElement(yourDog);
```



Use_Vector.java

```
Dog dog1=(Dog)box.get(0);  
Dog dog2=(Dog)box.get(1);  
int dog1_age=dog1.getAge();  
int dog2_age=dog2.getAge();  
System.out.println("dog1's age="+dog1_age);  
System.out.println("dog2's age="+dog2_age);  
}  
}
```



What is Objects?

method

A function defined in a class. There are *instance method*, *class method*. Unless specified otherwise, a method is not static.

instance method

Any method that is invoked with respect to an instance of a class.



What is Objects?

class method

A method that is invoked without reference to a particular object. Class methods affect the class as a whole, not a particular instance of the class.

Also called a *static method*.



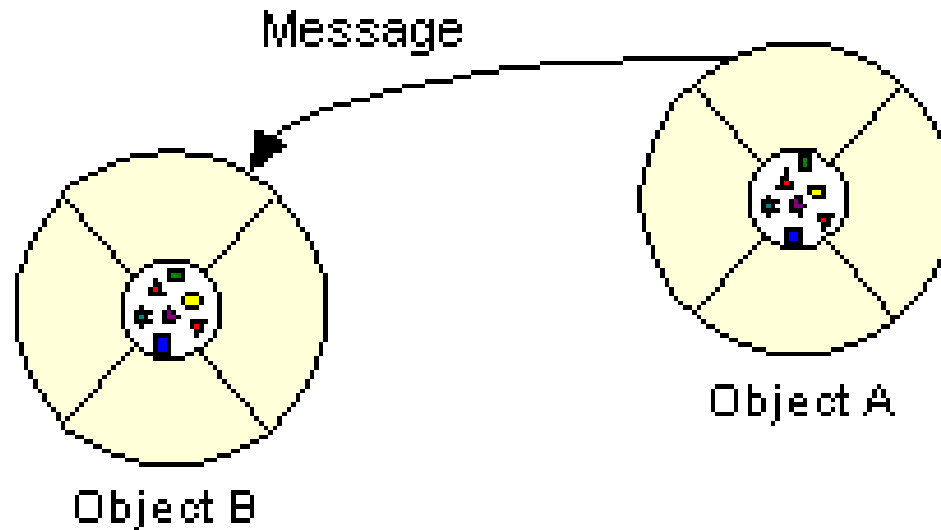
What Is a Message?

A single object alone is generally not very useful. Instead, an object usually appears as a component of a larger program or application that contains many other objects. Through the interaction of these objects, programmers achieve higher-order functionality and more complex behavior.



What Is a Message?

- **Software objects** interact and communicate with each other by sending *messages* to each other. When object A wants object B to perform one of B's methods, object A sends a message to object B





What Is a Message?

Messages provide two important benefits.

An object's behavior is expressed through its methods, so (aside from direct variable access) message passing supports all possible interactions between objects.

Objects don't need to be in the same process or even on the same machine to send and receive messages back and forth to each other.



What Is a Class?

- **Definition:** A class is a blueprint, or prototype, that defines the variables and the methods common to all objects of a certain kind.
- *In the real world, you often have many objects of the same kind. For example, your bicycle is just one of many bicycles in the world. Using object-oriented terminology, we say that your bicycle object is an instance of the class of objects known as bicycles. Bicycles have some state (current gear, current cadence, two wheels) and behavior (change gears, brake) in common. However, each bicycle's state is independent of and can be different from that of other bicycles.*



Object-Oriented Programming Concepts

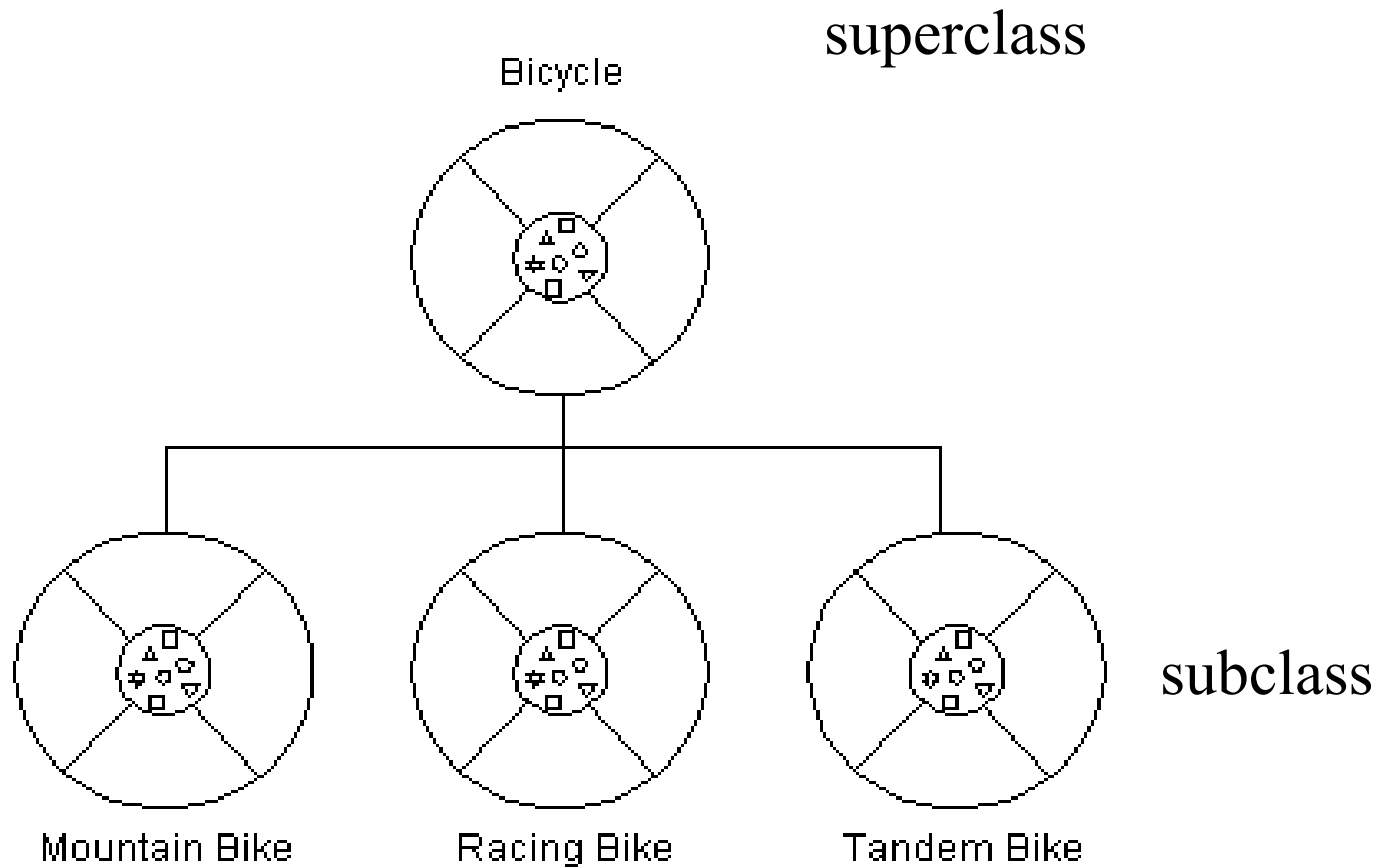
What Is Inheritance?

- **Object-oriented systems** take a step further and allow classes to be defined in terms of other classes.
- For example, mountain bikes, racing bikes, and tandems are all kinds of bicycles. In object-oriented terminology, mountain bikes, racing bikes, and tandems are all subclasses of the bicycle class. Similarly, the bicycle class is the superclass of mountain bikes, racing bikes, and tandems. This relationship is shown in the following figure.



Object-Oriented Programming Concepts

What Is Inheritance?





Object-Oriented Programming Concepts

What Is Inheritance?

subclass

- A class that is derived from a particular class, perhaps with one or more classes in between.

superclass

- A class from which a particular class is derived, perhaps with one or more classes in between.

*Each subclass **inherits** state (in the form of variable declarations) from the superclass. Mountain bikes, racing bikes, and tandems share some states:*



Object-Oriented Programming Concepts

What Is Inheritance?

inheritance

- The concept of classes automatically containing the variables and methods defined in their *supertypes*.

supertype

- The supertypes of a type are all the interfaces and classes that are extended or implemented by that type.



Object-Oriented Programming Concepts

What Is Inheritance?

hierarchy

- A classification of relationships in which each item except the top one (known as the root) is a specialized form of the item above it. Each item can have one or more items below it in the hierarchy. In the Java(TM) class hierarchy, the root is the Object class. **You are not limited to just one layer of inheritance. The inheritance tree.**



Object-Oriented Programming Concepts

What Is Inheritance?

Inheritance offers the following benefits:

- Subclasses provide specialized behaviors from the basis of common elements provided by the superclass. Through the use of inheritance, programmers can reuse the code in the superclass many times.
- Programmers can implement superclasses called *abstract classes* that define "generic" behaviors. The abstract superclass defines and may partially implement the behavior, but much of the class is undefined and unimplemented. Other programmers fill in the details with specialized subclasses.



Object-Oriented Programming Concepts

What Is Inheritance?

- **The Object class** is at the top of class hierarchy, and each class is its descendant (directly or indirectly). A variable of type Object can hold a reference to any object, such as an instance of a class or an array. Object provides behaviors that are required of all objects running in the Java Virtual Machine. For example, all classes inherit **Object's toString method**, which returns a string representation of the object.