The Java™ Tutorials

Trail: Learning the Java Language
Lesson: Object-Oriented Programming Concepts

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.

What Is a Class?

In the real world, you'll often find many individual objects all of the same kind. There may be thousands of other bicycles in existence, all of the same make and model. Each bicycle was built from the same set of blueprints and therefore contains the same components. In object-oriented terms, we say that your bicycle is an *instance* of the *class of objects* known as bicycles. A *class* is the blueprint from which individual objects are created.

The following `Bicycle` class is one possible implementation of a bicycle:

```
class Bicycle {

    int cadence = 0;
    int speed = 0;
    int gear = 1;

    void changeCadence(int newValue) {
         cadence = newValue;
    }

    void changeGear(int newValue) {
         gear = newValue;
    }

    void speedUp(int increment) {
         speed = speed + increment;
    }

    void applyBrakes(int decrement) {
         speed = speed - decrement;
    }

    void printStates() {
         System.out.println("cadence:" +
             cadence + " speed:" +
             speed + " gear:" + gear);
    }
}
```

The syntax of the Java programming language will look new to you, but the design of this class is based on the previous discussion of bicycle objects. The fields `cadence`, `speed`, and `gear` represent the object's state, and the methods (`changeCadence`, `changeGear`, `speedUp` etc.) define its interaction with the outside world.

You may have noticed that the `Bicycle` class does not contain a `main` method. That's because it's not a complete application; it's just the blueprint for bicycles that might be *used* in an application. The responsibility of creating and using new `Bicycle` objects belongs to some other class in your application.

Here's a `BicycleDemo` class that creates two separate `Bicycle` objects and invokes their methods:

```
class BicycleDemo {
    public static void main(String[] args) {

        // Create two different
        // Bicycle objects
        Bicycle bike1 = new Bicycle();
        Bicycle bike2 = new Bicycle();
```

```
        // Invoke methods on
        // those objects
        bike1.changeCadence(50);
        bike1.speedUp(10);
        bike1.changeGear(2);
        bike1.printStates();

        bike2.changeCadence(50);
        bike2.speedUp(10);
        bike2.changeGear(2);
        bike2.changeCadence(40);
        bike2.speedUp(10);
        bike2.changeGear(3);
        bike2.printStates();
    }
}
```

The output of this test prints the ending pedal cadence, speed, and gear for the two bicycles:

```
cadence:50 speed:10 gear:2
cadence:40 speed:20 gear:3
```

About Oracle | Contact Us | Legal Notices | Terms of Use | Your Privacy Rights

Copyright © 1995, 2017 Oracle and/or its affiliates. All rights reserved.

Previous page: What Is an Object?
Next page: What Is Inheritance?