

## 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 an Interface?

As you've already learned, objects define their interaction with the outside world through the methods that they expose. Methods form the object's *interface* with the outside world; the buttons on the front of your television set, for example, are the interface between you and the electrical wiring on the other side of its plastic casing. You press the "power" button to turn the television on and off.

In its most common form, an interface is a group of related methods with empty bodies. A bicycle's behavior, if specified as an interface, might appear as follows:

```
interface Bicycle {

    // wheel revolutions per minute
    void changeCadence(int newValue);

    void changeGear(int newValue);

    void speedUp(int increment);

    void applyBrakes(int decrement);
}
```

To implement this interface, the name of your class would change (to a particular brand of bicycle, for example, such as `ACMEBicycle`), and you'd use the `implements` keyword in the class declaration:

```
class ACMEBicycle implements Bicycle {

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

    // The compiler will now require that methods
    // changeCadence, changeGear, speedUp, and applyBrakes
    // all be implemented. Compilation will fail if those
    // methods are missing from this class.

    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);
    }
}
```

Implementing an interface allows a class to become more formal about the behavior it promises to provide. Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler. If your class claims to implement an interface, all methods defined by that interface must appear in its source code before the class will successfully compile.

---

**Note:** To actually compile the `ACMEBicycle` class, you'll need to add the `public` keyword to the beginning of the implemented interface methods. You'll learn the reasons for this later in the lessons on [Classes and Objects](#) and [Interfaces and Inheritance](#).

---

---

[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 Inheritance?

**Next page:** What Is a Package?