

Classes and Objects

- Classes
 - Declaring Classes
 - Declaring Member Variables
 - Defining Methods
 - Providing Constructors for Your Classes
 - Passing Information to a Method or a Constructor

- Objects
 - Creating Objects
 - Using Objects

- More on Classes
 - Returning a Value from a Method
 - Using the this Keyword
 - Controlling Access to Members of a Class
 - Understanding Class Members
 - Initializing Fields
 - Summary of Creating and Using Classes and Objects
- Questions and Exercises
- Questions and Exercises
- Nested Classes
 - Inner Class Example
- Local Classes
- Anonymous Classes
- Lambda Expressions
 - Method References
- When to Use Nested Classes, Local Classes, Anonymous Classes, and Lambda Expressions
- Questions and Exercises
- Enum Types
- Questions and Exercises

« Previous • Trail • Next »

Home Page > Learning the Java Language > Classes and Objects

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 and might use technology no longer available.
See [Java Language Changes](#) for a summary of updated language features in Java SE 9 and subsequent releases.
See [JDK Release Notes](#) for information about new features, enhancements, and removed or deprecated options for all JDK releases.

Classes

The introduction to object-oriented concepts in the lesson titled [Object-oriented Programming Concepts](#) used a bicycle class as an example, with racing bikes, mountain bikes, and tandem bikes as subclasses. Here is sample code for a possible implementation of a `Bicycle` class, to give you an overview of a class declaration. Subsequent sections of this lesson will back up and explain class declarations step by step. For the moment, don't concern yourself with the details.

```
public class Bicycle {  
  
    // the Bicycle class has  
    // three fields  
    public int cadence;  
    public int gear;  
    public int speed;  
  
    // the Bicycle class has  
    // one constructor  
    public Bicycle(int startCadence, int startSpeed, int startGear) {  
        gear = startGear;  
        cadence = startCadence;  
        speed = startSpeed;  
    }  
  
    // the Bicycle class has  
    // four methods  
    public void setCadence(int newValue) {  
        cadence = newValue;  
    }  
  
    public void setGear(int newValue) {  
        gear = newValue;  
    }  
  
    public void applyBrake(int decrement) {  
        speed -= decrement;  
    }  
  
    public void speedUp(int increment) {  
        speed += increment;  
    }  
  
}
```

A class declaration for a `MountainBike` class that is a subclass of `Bicycle` might look like this:

```
public class MountainBike extends Bicycle {  
  
    // the MountainBike subclass has  
    // one field  
    public int seatHeight;  
  
    // the MountainBike subclass has  
    // one constructor  
    public MountainBike(int startHeight, int startCadence,  
                        int startSpeed, int startGear) {  
        super(startCadence, startSpeed, startGear);  
        seatHeight = startHeight;  
    }  
  
    // the MountainBike subclass has  
    // one method  
    public void setHeight(int newValue) {  
        seatHeight = newValue;  
    }  
  
}
```

`MountainBike` inherits all the fields and methods of `Bicycle` and adds the field `seatHeight` and a method to set it (mountain bikes have seats that can be moved up and down as the terrain demands).

« Previous • Trail • Next »