Java Class Attributes

❮ Previous Next ❯

Java Class Attributes

In the previous chapter, we used the term "variable" for x in the example (as shown below). It is actually an **attribute** of the class. Or you could say that class attributes are variables within a class:

Example

Create a class called " MyClass " with two attributes: x and y :

```
public class MyClass {
  int x = 5;
  int y = 3;
}
```

Another term for class attributes is **fields**.

Accessing Attributes

You can access attributes by creating an object of the class, and by using the dot syntax (.):

The following example will create an object of the MyClass class, with the name myObj . We use the x attribute on the object to print its value:

Example

Create an object called " myObj " and print the value of x :

```
public class MyClass {
  int x = 5;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

Run example »

Modify Attributes

You can also modify attribute values:

Example

Set the value of x to 40:

```
public class MyClass {
  int x;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.x = 40;
    System.out.println(myObj.x);
  }
}
```

Run example »

Or override existing values:

Example

Change the value of x to 25:

```
public class MyClass {
  int x = 10;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.x = 25; // x is now 25
    System.out.println(myObj.x);
  }
}
```

Run example »

If you don't want the ability to override existing values, declare the attribute as `final` :

Example

```
public class MyClass {
  final int x = 10;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.x = 25; // will generate an error: cannot assign a value to a
final variable
    System.out.println(myObj.x);
  }
}
```

The `final` keyword is useful when you want a variable to always store the same value.

The `final` keyword is called a "modifier". You will learn more about these in our Java Modifiers Chapter.

Multiple Objects

If you create multiple objects of one class, you can change the attribute values in one class, without affecting the attribute values in the other class:

Example

Change the value of `x` to 25 in `myObj2`, and leave `x` in `myObj1` unchanged:

```
public class MyClass {
  int x = 5;

  public static void main(String[] args) {
    MyClass myObj1 = new MyClass();  // Object 1
    MyClass myObj2 = new MyClass();  // Object 2
    myObj2.x = 25;
    System.out.println(myObj1.x);  // Outputs 5
    System.out.println(myObj2.x);  // Outputs 25
  }
}
```

Run example »

Multiple Attributes

You can specify as many attributes as you want:

Example

```
public class Person {
  String fname = "John";
  String lname = "Doe";
  int age = 24;

  public static void main(String[] args) {
    Person myObj = new Person();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Age: " + myObj.age);
    }
}
```

Run example »

The next chapter will teach you how to create class methods and how to access them with objects.

❮ Previous Next ❯