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 "`Main`" with two attributes: `x` and `y`:

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

  public static void main(String[] args) {


  }
}
```

Try it Yourself »

Modify Attributes

You can also modify attribute values:

Example

Set the value of `x` to 40:

```
public class Main {
  int x;

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

```
    Main myObj = new Main();
    myObj.x = 40;
    System.out.println(myObj.x);
  }
}
```

Try it Yourself »

Or override existing values:

Example

Change the value of `x` to 25:

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

  public static void main(String[] args) {
    Main myObj = new Main();

    System.out.println(myObj.x);
  }
}
```

Try it Yourself »

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

Example

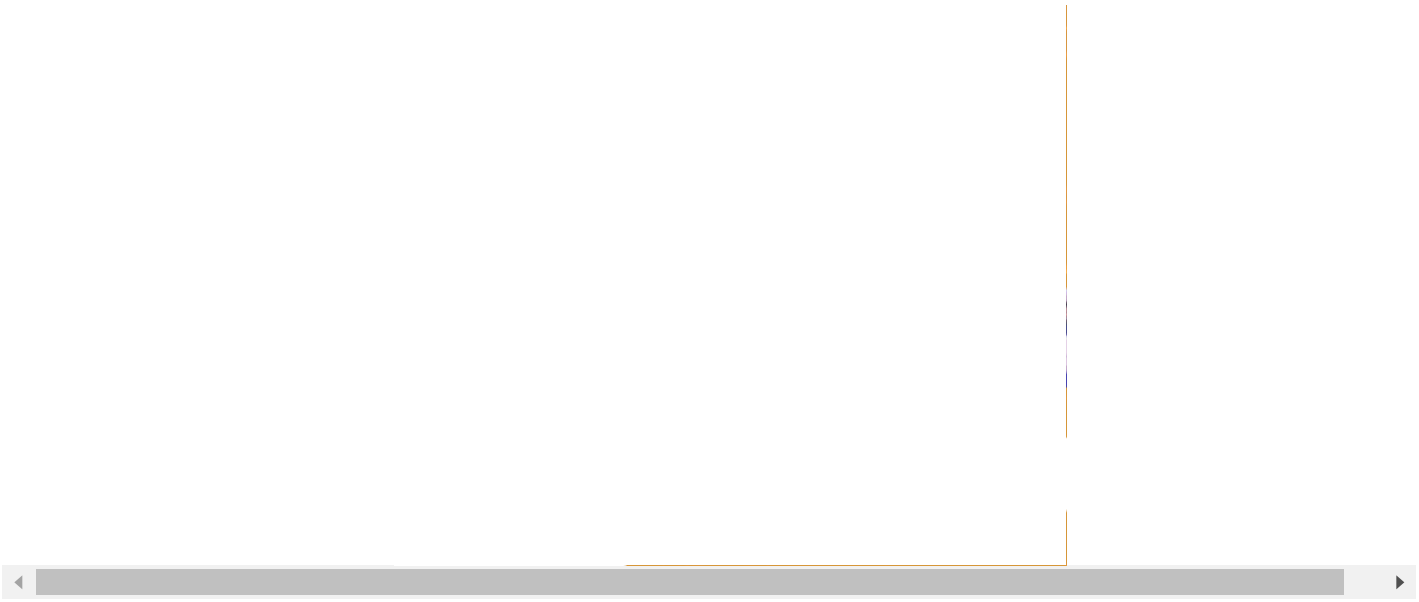
```
public class Main {


  public static void main(String[] args) {
    Main myObj = new Main();
```

```
    myObj.x = 25; // will generate an error: cannot assign a value to a final va
    System.out.println(myObj.x);
  }
}
```

Try it Yourself »

The `final` keyword is useful when you want a variable to always store the same value, like PI (3.14159...).

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



Multiple Objects

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

Example

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

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

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

Try it Yourself »

Multiple Attributes

You can specify as many attributes as you want:

Example

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

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

Try it Yourself »

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

❮ Previous

Next ❯

Top Tutorials

PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate