



**Getting Started** 

Learn ▼

Tutorials •

i

# TOUR OF SCALA CLASSES

Classes in Scala are blueprints for creating objects. They can contain methods, values, variables, types, objects, traits, and classes which are collectively called *members*. Types, objects, and traits will be covered later in the tour.

## Defining a class

A minimal class definition is simply the keyword class and an identifier. Class names should be capitalized.

```
class User
val user1 = new User
```

The keyword new is used to create an instance of the class. User has a default constructor which takes no arguments because no constructor was defined. However, you'll often want a constructor and class body. Here is an example class definition for a point:

```
class Point(var x: Int, var y: Int) {
    def move(dx: Int, dy: Int): Unit = {
        x = x + dx
        y = y + dy
    }

    override def toString: String =
        s"($x, $y)"
}

val point1 = new Point(2, 3)
println(point1.x) // 2
println(point1) // prints (2, 3)
```

This Point class has four members: the variables x and y and the methods move and toString. Unlike many other languages, the primary constructor is in the class signature (var x: Int, var y: Int). The move method takes two integer arguments and returns the Unit value (), which carries no information. This corresponds roughly with void in Java-like languages. toString, on the other hand, does not take any arguments but returns a String value. Since toString overrides toString from AnyRef, it is tagged with the override keyword.

#### Constructors

Constructors can have optional parameters by providing a default value like so:

```
class Point(var x: Int = 0, var y: Int = 0)

val origin = new Point // x and y are both set to 0

val point1 = new Point(1)
println(point1.x) // prints 1
```

In this version of the Point class, x and y have the default value 0 so no arguments are required. However, because the constructor reads arguments left to right, if you just wanted to pass in a y value, you would need to name the parameter.

```
class Point(var x: Int = 0, var y: Int = 0)
val point2 = new Point(y = 2)
println(point2.y) // prints 2
```

This is also a good practice to enhance clarity.

### Private Members and Getter/Setter Syntax

Members are public by default. Use the private access modifier to hide them from outside of the class.

```
class Point {
  private var _x = 0
  private var _y = 0
  private val bound = 100
  def x = _x
  def x = (newValue: Int): Unit = {
    if (newValue < bound) _x = newValue else printWarning</pre>
  }
 def y = _y
 def y_= (newValue: Int): Unit = {
    if (newValue < bound) _y = newValue else printWarning</pre>
  private def printWarning = println("WARNING: Out of bounds")
}
val point1 = new Point
point1.x = 99
point1.y = 101 // prints the warning
```

In this version of the Point class, the data is stored in private variables  $\_x$  and  $\_y$ . There are methods def x and def y for accessing the private data.  $def x_=$  and  $def y_=$  are for validating and setting the value of  $\_x$  and  $\_y$ . Notice the special syntax for the setters: the method has  $\_=$  appended to the identifier of the getter and the parameters come after.

Primary constructor parameters with val and var are public. However, because val s are immutable, you can't write the following.

```
class Point(val x: Int, val y: Int)
val point = new Point(1, 2)
point.x = 3 // <-- does not compile</pre>
```

Parameters without val or var are private values, visible only within the class.

```
class Point(x: Int, y: Int)
val point = new Point(1, 2)
point.x // <-- does not compile</pre>
```

#### More resources

- Learn more about Classes in the Scala Book
- How to use Auxiliary Class Constructors

 $\leftarrow$  previous next  $\rightarrow$ 

Contributors to this page:



heathermiller



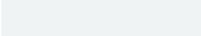












**DOCUMENTATION** 

**Getting Started** 

API

Overviews/Guides

Language Specification

**DOWNLOAD** 

**Current Version** 

All versions

**COMMUNITY** 

Community

Mailing Lists

Chat Rooms & More

Libraries and Tools

The Scala Center

**CONTRIBUTE** 

How to help

Report an Issue

**SCALA** 

Blog

Code of Conduct

License

**SOCIAL** 

GitHub

Twitter

