## List Creation

## In summary, you can create a new Scala List with these approaches:

## Lisp style

## Java style

## range method

## tabulate method

val list = 1 :: 2 :: 3 :: Nil Lisp-style

The :: method takes two arguments, a "head", which is a single element, and a "tail", which is a List.

val list = List(1,2,3) Java-style

Scala can usually infer the type of a List very well.

If you're mixing types in a List constructor, you may need to manually specify the List type.

val x = List[Number](1, 2.0, 33d, 0x1)

val x = List.range(1,10) val x = List.range(1,10,2)

val x = List.fill(3)("foo")

val x = List.tabulate(5)(n => n \* n)

**Object** - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as

behaviors - wagging, barking, eating. An object is an instance of a class.

 **Class** - A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.

 **Methods** - A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

 **Fields** - Each object has its unique set of instant variables, which are called fields. An object's state is created by the values assigned to these fields

**println**("Hello, Scala!");

Comments in Scala // or /\*…\*/

a semicolon (;) is required if you write multiple statements on a single line:

A package is a named module of code. For example, the Lift utility package is net.liftweb.util. The package declaration is the first non-comment line in the source file as follows: Scala packages can be imported so that they can be referenced in the current compilation scope. The following statement imports the contents of the scala.xml package:

import scala.xml.\_

You can import a single class and object, for example, HashMap from the scala.collection.mutable package:

import scala.collection.mutable.HashMap

You can import more than one class or object from a single package, for example, TreeMap and TreeSet from the scala.collection.immutable package:

import scala.collection.immutable.{TreeMap, TreeSet}

A field is a variable that is accessible inside the whole object. This is contrast to local variables, which are only accessible inside the method in which they are declared. Here is a simple field declaration:

class MyClass {

var myField : Int = 0;

}

Scala's type inference can figure out the type of a variable, based on the value assigned to it. Therefore, you could actually omit the type in the field declaration above, like this:

class MyClass {

var myField = 0;

}

<http://tutorials.jenkov.com/scala/index.html>

<https://www.tutorialspoint.com/scala/>

<http://www.scala-lang.org/documentation/>

<https://www.adictosaltrabajo.com/tutoriales/scala/>

http://scalatutorials.com/

https://www.youtube.com/watch?v=DzFt0YkZo8M

<https://twitter.github.io/scala_school/>

<http://joelabrahamsson.com/learning-scala/>

<https://liftweb.net/> web framework

• The Lift Framework • The Play framework • The Bowler framework

<http://www.slideshare.net/jboner/pragmatic-real-world-scala-45-min-presentation?type=powerpoint> ovo kasnije kad vec prodjem ostalo…

Scala is a statically typed language that was conceived in 2001 by Martin Odersky who has also written the Java reference compiler and co-authored Java generics. It is both a functional language, meaning that functions are values, and also an object oriented language where every value is an object. Scala compiles to byte code for the Java Virtual Machine (JVM) making it platform independent. That also means that we from a Scala program can use existing Java libraries and vice versa. While it seems the primary focus is on the JVM it can also be compiled for the .NET framework’s Common Language Runtime (CLR)

The Scala programming language is a newer, very interesting language, with a lot of new features compared to Java. The reason Scala is interesting to Java programmers is, that Scala is compiled to run on the Java Virtual Machine. In other words, Scala is compiled into Java bytecodes. This also means that you can use all Java classes in your Scala code. Even the Java classes you have developed yourself. This makes a transition to Scala cheaper, since a lot of Java code can be reused.

Scala is compiled into Java Byte Code which is executed by the Java Virtual Machine (JVM). This means that Scala and Java have a common runtime platform. It's a different language, but the same runtime.

Some of the new, interesting features in Scala are closures, functions as objects, traits, concurrency mechanisms inspired by Erlang, and its support for DSL's (Domain Specific Languages).

The Scala compiler compiles your Scala code into Java Byte Code which can then be executed by the scala command. The scala command is similar to the java command, in that it executes your compiled Scala code.

Since the Scala compiler can be a bit slow to startup, Scala has a compiler daemon you can run. This daemon keeps running, even when not compiling Scala code. You can then instruct the daemon to compile Scala code for you at will. This saves you the Scala compiler startup overhead when compiling.

The Scala interpreter runs your Scala code directly as it is, without you needing to compile it. The Scala interpreter may come in handy as a Scala script interpreter, a bit like a shell script interpreter on a Unix platform.

Scala has a set of features which differ from Java. Some of these are:

1. All types are objects.
2. Type inference.
3. Functions are objects.
4. Domain specific language (DSL) support.
5. Traits.
6. Closures.
7. Concurrency support inspired by Erlang.

## Scala for the Web

One of the popular Scala web frameworks is called Lift. You can find it here:

[**http://liftweb.net**](http://liftweb.net/)

A Scala class is a template for Scala objects. That means, that a class defines what information objects of that class holds, and what behaviour (methods) it exposes. A class can contain information about:

* Fields
* Constructors
* Methods
* Superclasses (inheritance)
* Interfaces implemented by the class
* etc.

Object - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as

behaviors - wagging, barking, eating. An object is an instance of a class.

 Class - A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.

 Methods - A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

 Fields - Each object has its unique set of instant variables, which are called fields. An object's state is created by the values assigned to these fields

**object HelloWorld {**

/\* This is my first java program.

\* This will print 'Hello World' as the output

\*/

def main(args: Array[String]) {

println("Hello, world!") // prints Hello World

}

}

Let's look at how to save the file, compile and run the program. Please follow the steps given below:

1. Open notepad and add the code as above.

2. Save the file as: HelloWorld.scala.

3. Open a command prompt window and go to the directory where you saved the program file.

4. Type 'scalac HelloWorld.scala' and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line.

5. Above command will generate a few class files in the current directory. One of them will be called

HelloWorld.class. This is a bytecode, which will run on Java Virtual Machine (JVM).

6. Now, type 'scala HelloWorld' to run your program.

7. You will be able to see 'Hello, World!' printed on the window.

**Basic Syntax:**

About Scala programs, it is very important to keep in mind the following points.

 Case Sensitivity - Scala is case-sensitive, which means identifier Hello and hello would have different

meaning in Scala.

 Class Names - For all class names, the first letter should be in Upper Case.

If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

Example class MyFirstScalaClass

 Method Names - All method names should start with a Lower Case letter.

If several words are used to form the name of the method, then each inner word's first letter should be in

Upper Case.

Example def myMethodName()

 Program File Name - Name of the program file should exactly match the object name.

When saving the file, you should save it using the object name (Remember scala is case-sensitive) and

append '.scala' to the end of the name (if the file name and the object name do not match your program will

not compile).

Example: Assume 'HelloWorld' is the object name. Then, the file should be saved as 'HelloWorld.scala'

 def main(args: Array[String]) - Scala program processing starts from the main() method, which is a

mandatory part of every Scala Program.

**Scala Keywords:**

The following list shows the reserved words in Scala. These reserved words may not be used as constant or

variable or any other identifier names.

Abstract Case catch class

Def Do else extends

False Final finally for

forSome If implicit import

Lazy Match new null

Object Override package private

Protected Return sealed super

This Throw trait try

True Type val var

While With yield

- : = =>

<- <: <% >:

# @

<https://docs.oracle.com/javase/tutorial/java/concepts/>

<https://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep>

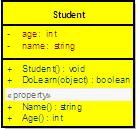
In order to clearly understand the object orientation model, let’s take your “hand” as an example. The “hand” is a class. Your body has two objects of the type "hand", named "left hand" and "right hand". Their main functions are controlled or managed by a set of electrical signals sent through your shoulders (through an interface). So the shoulder is an interface that your body uses to interact with your hands. The hand is a well-architected class. The hand is being reused to create the left hand and the right hand by slightly changing the properties of it.

### **4.4.****What is an Object?**

An object can be considered a "thing" that can perform a set of related activities. The set of activities that the object performs defines the object's behavior. For example, the Hand (object) can grip something, or a Student(object) can give their name or address.

In pure OOP terms an object is an instance of a class.

### **4.5.****What is a Class?**



A class is simply a representation of a type of object. It is the blueprint, or plan, or template, that describes the details of an object. A class is the blueprint from which the individual objects are created. Class is composed of three things: a name, attributes, and operations.

<http://tutorials.jenkov.com/scala/classes.html>

A Scala class is a template for Scala objects. That means, that a class defines what information objects of that class holds, and what behaviour (methods) it exposes. A class can contain information about:

* Fields
* Constructors
* Methods
* Superclasses (inheritance)
* Interfaces implemented by the class
* etc.