Scala

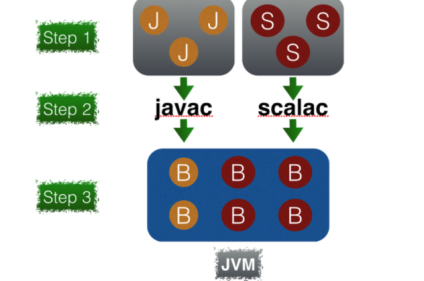
Scala is both Functional And Object Oriented Language that runs on the JVM .

It was developed By **Martin Odersky.**

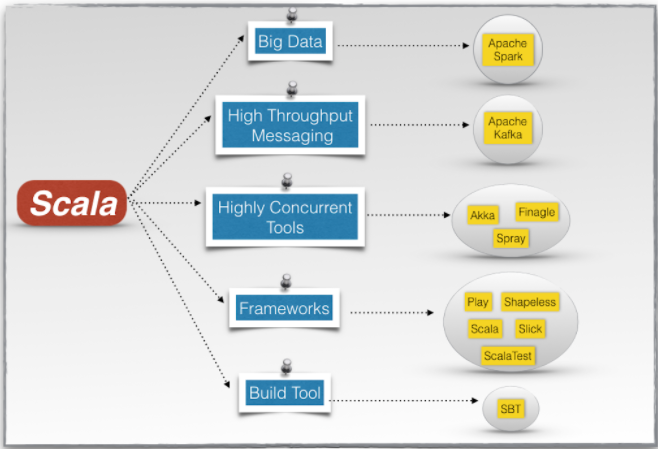
JDK – Contains set of libraries and utilities for developing the Java Application.It also comes with the JRE Java RunTime Environment. you will also hear the term Java Virtual Machine or JVM used to refer to the JRE.

 Scala, there is **NO** need to terminate the statement with a **semi-colon** .Scala can extend the Trait Called APP to run the code. Where the Trait “APP” have the Main Method. Consider Trait as an equivalent to the Interface in Java.

AS per the Below Diagram both Java And Scala Code will be converted into the ByteCode and will be running on top of JVM.



Scala EcoSystem



Setting Up the Development Environment

Installing the Scala Plugin In INTELLIJ

<http://allaboutscala.com/tutorials/chapter-1-getting-familiar-intellij-ide/scala-environment-setup-install-scala-plugin-intellij/>

2.1 Variables in Scala

2.2String interpolation

2.3

3.Functions

* Create a Simple Function named **favouritedonut** which has the return type as String
* Create a Function called leastFavouriteDonut without Paranthesis
* Create a Function “PrintDonutSalesReport” With No Return Type
* Create A Function “CalculateDonutCost” which has the donutName Quantity as Input and it return the total cost
* Modify the Above Function such that it accepts another parameter called CouponCode and it has some default value as “Nocode”.
* Modify the Function “CalculateDonutCost” with couponCode as Option Parameter
* Assign Default Value to the CouponCode Option Parameter
* Create a Function “CalculateCouponCode” which Returns Options[Strings] , if the return value is null display ur custiom code . call the CalculateCouponCode Function Using the Pattern Matching.
* Create A Function Called “ApplyDiscount” and make the Function as a Generic Parameter of Type T
* Lets modify the Above Function and we are returning a sequnce and make the sequence as general one
* Create A Function Called “PrintReport” and it accepts n number of Strings Array of Strings , Seq of Strings and Display it.
* Create A Function Named “Summation” and it should accepts n numbers and it should give back their sum to us.
* Create a Function called “totalCost” as Currying Function and Find the TotalCost for the Donuts

Typed Function :-

We have a Function 1 which calculates the Discount and it accepts String,

We have Another Function which calculates the Discount and it accepts Double.

Instead oF Two Functions we can use a Simple and Common Generic Functions To Handle the above two Scenarios,

we will create a **typed** function which will specify a **genericparameter** of type T as follows:

def applyDiscount[**T**](discount: **T**) {

* Sure you can rely on Scala's type inference to infer the type of your function parameter.
* But in general it's a good practice to explicitly specify your parameter types.

We can also add the retuen Type to the applyDicount Function

Variable Arguments Functions

def printReport(names: String\*) {

println(s"""Donut Report = ${names.mkString(", ")}""")

}

This Function Will Accepts a n number of Strings and Displays It .

We can also pass a list to this Function

Currying if we want to Calculate the students Grade based on the exam mark assignments quizzes then the variable argument Function is not suited Well

We can use Partial Functions

def add(a:Int,b:Int) = a+b;

def add\_curry(a:Int)(b:Int)=a+b;

val plus5 = add\_curry(5)

plus5(3) //8

we can also use the currly Bracessis

add\_curry{5}{3} // 8

add\_curry{

println(“hi”)

5}

{3} //8

Def examResults(marks:Int\*,assign:Int\*,quizzes:Int\*)={} while calling we don’t know whether how many marks assignments quizzes were there

So Better Option is Currying the same method can be re written as

Def examResults(marks:Int\*)(assign:Int\*)(quizzes:Int\*)={}

examResults(32,23,4,3)(23,23,23,2)(34,5,4)

Higher Order Functions

we will learn how to create **Higher Order Function** which is a function that takes another function as its parameter.

def totalCostWithDiscountFunctionParameter(donutType: String)(quantity: Int)(f: Double => Double): Double = {

println(s"Calculating total cost for $quantity $donutType")

val totalCost = 2.50 \* quantity

f(totalCost)

}

Creating the High Order Function With Another Function

println("\nStep 4: How to define and pass a function to a higher order function")

def applyDiscount(totalCost: Double): Double = {

val discount = 2 // assume you fetch discount from database

totalCost - discount

}

Higher Order Functions with Call-by Name

Notes:-

**Scala's typed functions provide greater flexibility with the use of variances which we will see in upcoming tutoria**