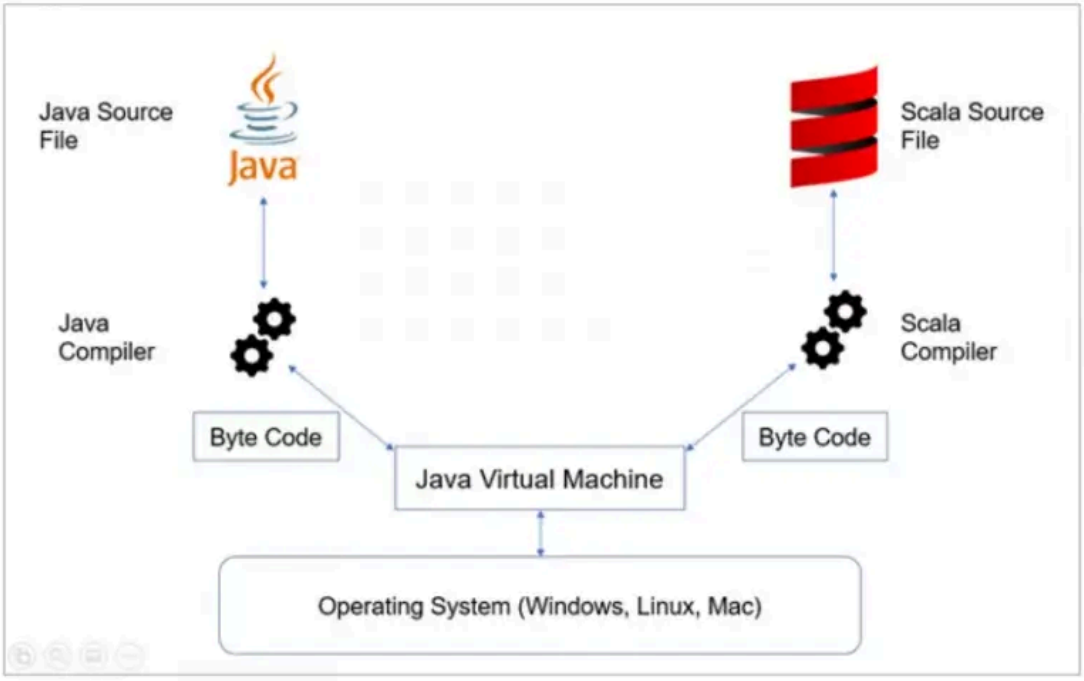
Scala Fundamentals



1. What is Scala?

Scala is a modern, concise, and flexible programming language that combines both object-oriented and functional programming paradigms. It runs on the Java Virtual Machine (JVM) and is fully interoperable with Java libraries.

2. Basic Syntax and Data Types

Declaring Variables

- `val` for immutable values (like `final` in Java)
- `var` for mutable variables

```scala
val immutableValue: Int = 10
var mutableValue: String = "Hello"
mutableValue = "World"
```

Common Data Types

- `Int` (integer numbers)
- `Double` (floating point)
- `Boolean` (`true` or `false`)
- `String`
- `Unit` (represents no meaningful value, similar to `void` in Java)

Example:

```scala
val isScalaFun: Boolean = true
val pi: Double = 3.14159
val name: String = "Scala"
```

3. Control Structures

If-else Statement:

```scala
val x = 5
if (x > 0) println("Positive")
else println("Non-positive")
```

Match Expression (like switch-case):

```scala
val day = "Monday"
day match {
  case "Monday"    => println("Start of week")
```

```scala
  case "Friday"    => println("End of work week")
  case _           => println("Midweek")
}
```

For Loop

```scala
for (i <- 1 to 5) println(i)
```

4. Functions

Defining functions with or without return types:

```scala
def add(a: Int, b: Int): Int = a + b

def greet(name: String): Unit = {
  println(s"Hello, $name!")
}

println(add(3, 4))   // Output: 7
greet("Scala")       // Output: Hello, Scala!
```

Anonymous (lambda) function example:

```scala
val square = (x: Int) => x * x
println(square(5))  // Output: 25
```

5. Classes and Objects

Defining a simple class with fields and methods:

```scala
class Person(var name: String, var age: Int) {
  def greet(): Unit = {
```

```
    println(s"Hi, my name is $name and I'm $age years
old.")
  }
}

val p = new Person("Shiva", 25)
p.greet()  // Output: Hi, my name is Shiva and I'm 25
years old.
```

Singleton object (like a static class):

```scala
object Main {
  def main(args: Array[String]): Unit = {
    println("Hello from the singleton object!")
  }
}
```

6. Collections

Basic list example:

```scala
val fruits = List("Apple", "Banana", "Cherry")
fruits.foreach(println)
```

Using map on collections:

```
val numbers = List(1, 2, 3)
val doubled = numbers.map(_ * 2)  // List(2, 4, 6)
```

7. Pattern Matching Example

```
def describe(x: Any): String = x match {
```

```
  case 5 => "Five"
  case "hello" => "Greeting"
  case _: Int => "Some integer"
  case _ => "Unknown"
}

println(describe(5))          // Five
println(describe("hello"))    // Greeting
println(describe(42))         // Some integer
println(describe(true))       // Unknown
```

8. Program Entry Point

Scala programs require an entry point defined by a main method inside an object:

```
scala

object HelloWorld {
  def main(args: Array[String]): Unit = {
    println("Hello, Scala developer!")
  }
}
```

Recap

- You declare values and variables with `val` and `var`.
- Control flow with `if`, `match`, and loops.
- Define functions easily, including anonymous ones.
- Work with classes for OOP, and objects for singletons.
- Use collections and powerful pattern matching.

If you want, I can provide a hands-on example project, detailed explanations on specific topics like functional programming, or more advanced Scala concepts!