What is Scala? Is it a Language or Platform? Does it support OOP or FP? Who is the father of Scala?

Scala stands for SCAlable LAnguage. Martin Odersky is the father of Scala.
Scala is a Multi-Paradigm Programming Language, which supports both Object-Oriented and Functional Programming concepts. It is designed and developed by Martin Odersky.
Scala is a Type-Safe Object-Functional Programming JVM Language. Scala runs on JVM(Java Virtual Machine).
Scala is a Hybrid Functional (Object-Oriented and Functional) Programming JVM Language. Scala has a Strong and Statically Type System. In Scala, all types are checked at compile-time.

Is Scala Statically-Typed Language? What is Statically-Typed Language and What is Dynamically-Typed Language? What is the difference between statically typed and dynamically typed languages?

Yes, Scala is a Statically-Typed Language.

Statically-Typed Language means that Type checking is done at compile-time by compiler, not at run-time. The main Advantage of these kinds of Languages is: As a Developer, we should care about writing right code to avoid all compile-time errors. As Compiler checks many of the errors at compile-time, we don't get much issues or bugs at run-time.

Examples:- Java, Scala, C, C++, Haskell etc.

Dynamically-Typed Language means that Type checking is done at run-time, not at compile-time by compiler. As a compiler won't check any type checking at compile-time, We can expect more run-time issues or bugs.

Example:- Groovy, JavaScript, Ruby, Python, Smalltalk etc.

Is Scala a Pure OOP Language? Is Java a Pure OOP Language?

Pure Object-Oriented Programming Language means that everything should be an Object.

Java is not a Pure Object-Oriented Programming (OOP) Language because it supports the following two Non-OOP concepts:

Java supports primitive data types. They are not objects.

Java supports Static members. They are not related to objects.

Yes, Scala is a Pure Object-Oriented Programming Language because in Scala, everything is an Object and everything is a value. Functions are values and values are Objects.

Scala does not have primitive data types and also does not have static members.

Does Scala support all Functional Programming concepts? Does Java 8 support all Functional Programming concepts?

Yes, Scala supports all Functional Programming (FP) concepts. Java 8 has introduced some Functional Programming constructs, but it does NOT support all Functional Programming concepts.

For instance, Java 8 does not support Pattern Matching, Function Currying, Implicits etc.

What are the major advantages of Scala Language? Are there any drawbacks of Scala Language?

If we use Scala Language to develop our applications, we can get the following benefits or advantages and drawbacks:

- Advantages of Scala Language:-

Simple and Concise Code , Very Expressive Code , More Readable Code , 100% Type-Safe Language , Immutability , and No Side-Effects , More Reusable Code , More Modularity , Do More With Less Code , Very Flexible Syntax , Supports all OOP Features , Supports all FP Features. Highly Functional , Less Error Prone Code , Better Parallel and Concurrency Programming , Highly Scalable and Maintainable code , Highly Productivity , Distributed Applications , Full Java Interoperability , Powerful Scala DSLs available , REPL to learn Scala Basics

- Drawbacks of Scala Language:-

Less Readable Code , Bit tough to Understand the Code for beginners , Complex Syntax to learn , Less Backward Compatibility

NOTE:- We can write Scala Code either more readable or less readable way.

What is the Main drawback of Scala Language?

Apart from many benefits of Scala, it has one major Drawback: Backward Compatibility Issue. If we want to upgrade to latest version of Scala, then we need to take care of changing some package names, class names, method or function names etc.

For instance, If you are using old Scala version and your project is using BeanProperty annotation. It was available in "scala.reflect" like "scala.reflect.BeanProperty" in old versions. If we want to upgrade to new Scala versions, then we need to change this package from "scala.reflect" to "scala.beans".

What is the main motto of Scala Language?

Like Java's Motto "Write Once Run Anywhere", Scala has "Do More With Less" or "Do More With Less Code" Motto.

"Do More With Less" means that we can develop more complex program or logic with less code.

What are the popular JVM Languages available now?

Java, Scala, Groovy and Closure are most popular JVM (Java Virtual Machine) languages.

Scala, Groovy and Closure JVM languages supports both Object-Oriented Programming Features and Functional Programming Features.

Java SE 8 supports all Object-Oriented Programming Features. However, it supports very few Functional Programming Features like Lambda Expressions, Functions, Type Inference, Higher-Order Functions.

Like Java's java.lang.Object class, what is the super class of all classes in Scala?

As we know in Java, the super class of all classes (Java API Classes or User Defined Classes) is java.lang.Object. In the same way in Scala, the super class of all classes or traits is "Any" class.

Any class is defined in scala package like "scala.Any".

What is default access modifier in Scala? Does Scala have "public" keyword?

In Scala, if we don't mention any access modifier to a method, function, trait, object or class, the default access modifier is "public". Even for Fields also, "public" is the default access modifier.

Because of this default feature, Scala does not have "public" keyword.

What is "Type Inference" in Scala?

Types can be inferred by the Scala Compiler at compile-time. It is known as "Type Inference". Types means Data type or Result type. We use Types at many places in Scala programs like Variable types, Object types, Method/Function Parameter types, Method/Function return types etc.

In simple words, determining the type of a variable or expression or object etc at compile-time by compiler is known as "Type Inference".

What are the similarities and differences between Scala's Int and Java's java.lang.Integer? What is the relationship between Int and RichInt in Scala?

- Similarities between Scala's Int and Java's java.lang.Integer:

Both are classes ; Both are used to represent integer numbers ; Both are 32-bit signed integers ;

- Differences between Scala's Int and Java's java.lang.Integer:

Scala's Int class does not implement Comparable interface ; Java's java.lang.Integer class implements Comparable interface.
Java's Integer is something similar to Scala's Int and RichInt. RichInt is a final class defined in scala.runtime package like "scala.runtime.RichInt".
In Scala, the Relationship between Int and RichInt is that when we use Int in a Scala program, it will automatically convert into RichInt to utilize all methods available in that Class. We can say that RichInt is an Implicit class of Int. (We will discuss "What is Implicit and the advantages of Implicits in my next post).

What is Nothing in Scala? What is Nil in Scala? What is the relationship between Nothing and Nil in Scala?

In Scala, Nothing is a Type (final class). It is defined at the bottom of the Scala Type System that means it is a subtype of anything in Scala. There are no instances of Nothing.

Use Cases of Nothing In Scala:-

If Nothing does not have any instances, then when do we use this one in Scala Applications?

Nil is defined using Nothing (See below for example).

None is defined using Nothing.

object None extends Option[Nothing]

We can use Nothing as a return type of methods which never return.

We can use Nothing as a return type of methods which terminates abnormally.

Nil is an object, which is used to represent an empty list. It is defined in "scala.collection.immutable" package as shown below:

object Nil extends List[Nothing]

Example:-

scala> Nil

res5: scala.collection.immutable.Nil.type = List()

scala> Nil.length

res6: Int = 0

What is Null in Scala? What is null in Scala? What is difference between Null and null in Scala?

Null is a Type (final class) in Scala. Null type is available in "scala" package as "scala.Null". It has one and only one instance that is null.

In Scala, "null" is an instance of type scala.Null type.

Example:-

scala> val myNullRef : Null = null

myNullRef: Null = null

We cannot assign other values to Null type references. It accepts only 'null' value.

Null is a subtype of all Reference types. Null is at the bottom of the Scala Type System. As it is NOT a subtype of Value types, we can assign "null" to any variable of Value type.

Example:-

scala> val myInt : Int = null

:10: error: an expression of type Null is ineligible for implicit conversion

val myInt : Int = null

Here type mismatch error. found : Null(null) but required: Int. The implicit conversions between Null and Int are not applicable because they are ambiguous.

What is Unit in Scala? What is the difference between Java's void and Scala's Unit?

In Scala, Unit is used to represent "No value" or "No Useful value". Unit is a final class defined in "scala" package that is "scala.Unit".

Unit is something similar to Java's void. But they have few differences.

Java's void does not any value. It is nothing.

Scala's Unit has one value ()

() is the one and only value of type Unit in Scala. However, there are no values of type void in Java.

Java's void is a keyword. Scala's Unit is a final class.

Both are used to represent a method or function is not returning anything.

What is the difference between val and var in Scala?

In Scala, both val and var are used to define variables. However, they have some significant differences.

var stands for variable.

val stands for value.

As we know, variable means changeable and value means constant.

var is used to define Mutable variables that means we can reassign values once its created.

val is used to define Immutable variables that means we cannot reassign values once its created.

In simple Java terminology, var means 'variable' and val means 'final variable'.

What is REPL in Scala? What is the use of Scala's REPL? How to access Scala REPL from CMD Prompt?

REPL stands for Read-Evaluate-Print Loop. We can pronounce it as 'ripple'. In Scala, REPL is acts as an Interpreter to execute Scala code from command prompt. That's why REPL is also known as Scala CLI(Command Line Interface) or Scala command-line shell.

The main purpose of REPL is that to develop and test small snippets of Scala code for practice purpose. It is very useful for Scala Beginners to practice basic programs.

We can access REPL by using "scala" command. When we type "scala" command at CMD Prompt, we will get REPL shell where we can type and execute scala code.

D:\> scala

scala>

What are the Scala Features?

Scala Language supports the following features:

Supports both OOP-style(Imperative-Style) and Functional-Style Programming

Pure Object-Oriented Programming Language

Supports all Functional Features

REPL(Read-Evaluate-Print Loop) Interpreter

Strong Type System

Statically-Typed Language

Type Inference

Supports Pattern Matching

Supports Closures

Supports Persistent Data Structures

Uses Actor Model to develop Concurrency Applications

Interoperable with Java

Available all development tools – IDEs, Build Tools, Web Frameworks, TDD and BDD Frameworks

How do we implement loops functionally? What is the difference between OOP and FP style loops?

We know how to implement loops in Object-Oriented style: Using Mutable Temporary variables, update the variable value and use Loop constructs. It is very tedious and unsafe approach. It is not Thread-Safe.

Object-Oriented style uses the following constructs to implement Loops:

Loop Constructs

Mutability

Side Effects

We can implement same Loops differently in Functional way. It is Thread-Safe. We can use the following two techniques to implement the loops in functional style:

Recursion

Tail-Recursion

Immutability

No Side-Effects

What is "Application" in Scala or What is Scala Application? What is "App" in Scala? What is the use of Scala's App?

Scala Application:

In Scala, App is a trait defined in scala package like "scala.App". It defines main method. If an Object or a Class extends this trait, then they will become as Scala Executable programs automatically because they will inherit main method from Application.

The main advantage of using App is that we don't need to write main method. The main drawback of using App is that we should use same name "args" to refer command line argument because scala.App's main() method uses this name.

Example:-

Without Scala App:

```
object MyApp {
   def main( args: Array[String]){
      println("Hello World!")
   }
}
```

With Scala App:

```
object MyApp extends App{
  println("Hello World!")
}
```

If we observe above two examples, in second example we have not defined main method because we have inherited from Scala App(Application).

Before Scala 2.9, we have scala.Application trait. But it is deprecated by scala.App since Scala 2.9 version.

Does Scala support Operator Overloading? Does Java support Operator Overloading?

Java does not support Operator Overloading. Scala supports Operator Overloading.

The reason is that Java does not want to support some misleading method names like "+*/". Scala has given this flexibility to Developer to decide which methods/functions name should use.

When we call 2 + 3 that means '+' is not an operator, it is a method available in Int class (or it's implicit type). Internally, this call is converted into "2.+(3)".

What is an Expression? What is a Statement? Difference between Expression and Statement?

Expression:

Expression is a value that means it will evaluate to a Value. As an Expression returns a value, We can assign it to a variable.

Example:- Scala's If condition, Java's Ternary operator.

Statement:

Statement defines one or more actions or operations. That means Statement performs actions. As it does not return a value, we cannot assign it to a Variable.

Example:- Java's If condition.

What is the difference between Java's "If..Else" and Scala's "If..Else"?

Java's "If..Else":

In Java, "If..Else" is a statement, not an expression. It does not return a value and cannot assign it to a variable.

Example:-

```
 int year;
 if( count == 0)
 year = 2014;
```

```
else
  year = 2015;
```

Scala's "If..Else":

In Scala, "If..Else" is an expression. It evaluates a value i.e. returns a value. We can assign it to a variable.

```
val year = if( count == 0) 2014 else 2015
```

NOTE:-Scala's "If..Else" works like Java's Ternary Operator. We can use Scala's "If..Else" like Java's "If..Else" statement as shown below:

```
val year = 0
if( count == 0)
  year = 2014
else
  year = 2015
```

Is Scala an Expression-Based Language or Statement-Based Language? Is Java an Expression-Based Language or Statement-Based Language?

In Scala, everything is a value. All Expressions or Statements evaluates to a Value. We can assign Expression, Function, Closure, Object etc. to a Variable. So Scala is an Expression-Oriented Language.

In Java, Statements are not Expressions or Values. We cannot assign them to a Variable. So Java is not an Expression-Oriented Language. It is a Statement-Based Language.

Tell me some features which are supported by Java, but not by Scala and Vice versa?

Java does not support Operator Overloading, but Scala supports it.

Java supports ++ and — operators , but Scala does not support them.

Java has Checked and Unchecked Exceptions, but Scala does not have Checked Exceptions.

Scala does not support break and continue statements, but Java uses them.

Scala does not have explicit Type casting, but Java supports this feature.

Scala supports Pattern Matching, but Java does not.

Java uses Primitive Data types, but Scala does not have.

Java supports static members, but Scala does not have static members concept.

Scala supports Implicits and Traits, Java does not support them.

NOTE:-This list goes beyond one page. However, these are some important points to remember about differences in Scala and Java features to face Scala Interviews.

What is the difference between Function and Method in Scala?

Scala supports both functions and methods. We use same syntax to define functions and methods, there is no syntax difference.

However, they have one minor difference:

We can define a method in a Scala class or trait. Method is associated with an object (An instance of a Class). We can call a method by using an instance of a Class. We cannot use a Scala Method directly without using object.

Function is not associated with a class or trait. It is defined in a Scala Package. We can access functions without using objects, like Java's Static Methods.

NOTE:- We will discuss about Class, Trait,Package, Object etc in my coming posts.

How many public class files are possible to define in Scala source file?

In Java, we can define at-most one public class/interface in a Source file. Unlike Java, Scala supports multiple public classes in the same source file.

We can define any number of public classes/interfaces/traits in a Scala Source file.

Like Java, what are the default imports in Scala Language?

We know, java.lang is the default package imported into all Java Programs by JVM automatically. We don't need to import this package explicitly.

In the same way, the following are the default imports available in all Scala Programs:

java.lang package ; scala package ; scala.PreDef

How many operators are there in Scala and Why?

Unlike Java and like C++, Scala supports Operator Overloading. Scala has one and only operator that is "=" (equalto) operator. Other than this all are methods only.

For instance 2 + 3, here "+" is not an Operator in Scala. "+" is method available in Int class. Scala Compiler observes 2 and 3 are Integers and tries to find that "+" method in Int class. So Scala Compiler converts "2 + 3" expression into "2.+(3)" and make a call to "+" method on integer object "2" and pass integer object "3" as parameter to "+" method.

Both "2 + 3" and "2.+(3)" are equal. It's just Scala's syntactic sugar to write programs in Functional style.

Mention Some keywords which are used by Java and not required in Scala? Why Scala does not require them?

Java uses the following keywords extensively:

'public' keyword – to define classes, interfaces, variables etc.

'static' keyword – to define static members.

Scala does not required these two keywords. Scala does not have 'public' and 'static' keywords.

In Scala, default access modifier is 'public' for classes,traits, methods/functions, fields etc. That's why, 'public' keyword is not required.

To support OOP principles, Scala team has avoided 'static' keyword. That's why Scala is a Pure-OOP Langauge. It is very tough to deal static members in Concurrency applications.

What is PreDef in Scala? What is the main purpose of PreDef in Scala?

In Scala, PreDef is an object defined in scala package as "scala.PreDef". It is an utility object.

It defines many utility methods as shown below:

Console IO (print,println etc) ; Collection utility methods ; String utility methods ;

Implicit conversion methods ; Assertion utility methods etc.

For instance, print, println, readLine, readInt, require etc methods are defined in PreDef object.

In Scala, PreDef is available to use its methods without importing in all Scala Programs because Scala Compiler imports this object into all compilation units like Class, Object, Trait etc. automatically.