SCALA Q&A

Q1. What is Scala?

Ans: Scala is a Java-based Hybrid programming language which is the fusion of both Functional and Object-Oriented Programming Language features. It can integrate itself with Java Virtual Machine and compile the code written.

Q2. Explain how Scala is both Functional and Object-oriented Programming Language?

Ans: Scala treats every single value as an Object which even includes Functions. Hence, Scala is the fusion of both Object-oriented and Functional programming features.

Q3.Write a few Frameworks of Scala

Ans: Some of the Frameworks supported by Scala are as follows:

-
    - Akka Framework
    - Spark Framework
    - Play Framework
    - Scalding Framework
    - Neo4j Framework
    - Lift Framework
    - Bowler Framework

Q4. Mention the types of Variables in Scala? And What is the difference between them?

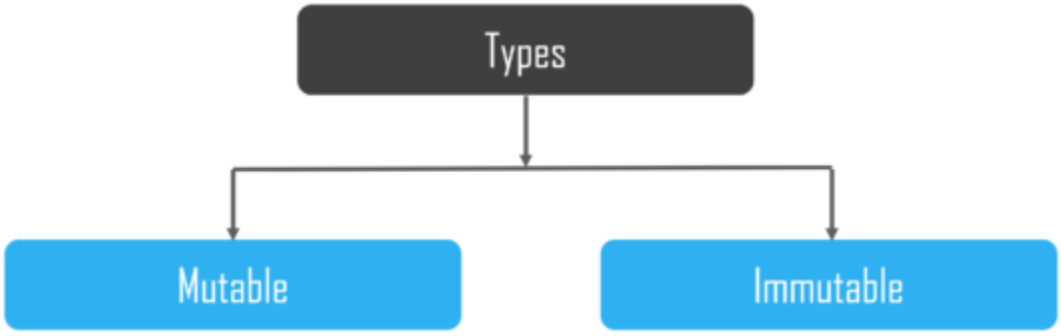
Ans: The Variables in Scala are mainly of two types:

Mutable Variables

-
    -
        - We Declare Mutable Variables by using the var keyword.
        - The values in the Mutable Variables support Changes

Immutable Variables

-
    -
        - We declare Immutable Variables using the val keyword.
        - The values in Immutable Variables do not support changes.



Q5. Explain Streams in Scala.

Ans: In simple words, we define Stream as a Lazy list which evaluates the elements only when it needs to. This sort of lazy computation enhances the Performance of the program.

Q6. Mention the Advantages of Scala

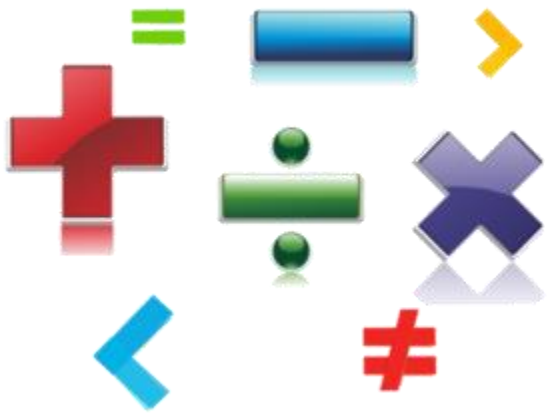
Ans: Some of the major Advantages of Scala are as follows:

-
    - It is highly Scalable
    - It is highly Testable
    - It is highly Maintainable and Productive
    - It facilitates Concurrent programming
    - It is both Object-Oriented and Functional
    - It has no Boilerplate code
    - Singleton objects are a cleaner solution than Static
    - Scala Arrays use regular Generics
    - Scala has Native Tuples and Concise code

Q7. Explain the Operators in Scala

Ans: The following are the Operators in Scala:

-
    - Arithmetic Operators
    - Relational Operators
    - Logical Operators
    - Bitwise Operators
    - Assignment Operators



Q8. What is Recursion tail in Scala?

Ans: 'Recursion' is a function that calls itself. For example, a function 'A' calls function 'B', which calls the function 'C'. It is a technique used frequently in Functional programming. In order for a Tail recursive, the call back to the function must be the last function to be performed.

Q9. Explain the use of Tuples in Scala?

Ans: Scala tuples combine a Finite number of items together so that the programmer can Pass a tuple around as a Whole. Unlike an Array or List, a tuple is Immutable and can hold objects with different Datatypes.

Q10. How is a Class different from an Object?

Ans: Class combines the data and its methods whereas an Object is one particular Instance in a class.

Q11. Why do we need App in Scala?

Ans: App is a helper class that holds the main method and its Members together. The App trait can be used to quickly turn Objects into Executable programs. We can have our classes extend App to render the executable code.

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

Q12. What are Higher-order functions?

Ans: A Higher-order function is a function that does at least one of the following: takes one or more Functions as Arguments, returns a Function as its result.

Q13. Explain the scope provided for variables in Scala.

Ans: There are three different scopes depending upon their use. Namely:

Fields:

-
    - Fields are variables declared inside an object and they can be accessed anywhere inside the program depending upon the access modifiers. Fields can be declared using var as well as val.

Method Parameters:

-
    - Method parameters are strictly Immutable. Method parameters are mainly used to Pass values to the methods. These are accessed inside a method, but it is possible to access them from outside the method provided by a Reference.

Local Variables:

-
    - Local variables are declared inside a method and they are accessible only inside the method. They can be accessed if you return them from the method.

Q14. What is a Closure?

Ans: Closure is considered as a Function whose return value is Dependent upon the value of one or more variables declared outside the closure function.

Q15. Mention how Scala is different from Java

Ans: A few scenarios where Scala differs from Java are as follows:

-
    - All values are treated as Objects.

- Scala supports Closures
- Scala Supports Concurrency.
- It has Type-Inference.
- Scala can support Nested functions.
- It has DSL support [Domain Specific Language]
- Traits

Q16. Explain the access Modifiers available in Scala

Ans: There are mainly three access Modifiers available in Scala. Namely,

Private:

- The Accessibility of a private member is restricted to the Class or the Object in which it declared.

The following program will explain this in detail.

```
1 class Outer {
  class Inner {
    private def f() { println("f") }
     class InnerMost {
       f() // OK
    }
  }
  (new Inner).f() // Error: f is not accessible
}
```

Protected:

- A protected member is only Accessible from Subclasses of the class in which the member is defined.

The following program will explain this in detail.

1	**package** p

```
2   class Super {
3     protected def f() { println("f") }
4   }
5   class Sub extends Super {
6     f()
7   }
8   class Other {
9     (new Super).f() // Error: f is not accessible
10  }
11 }
```

Public:

-
    - Unlike Private and Protected members, it is not required to specify Public keyword for Public members. There is no explicit modifier for public members. Such members can be accessed from Anywhere.

Following is the example code snippet to explain Public member

```
1  class Outer {
2    class Inner {
3      def f() { println("f") }
4      class InnerMost {
5        f() // OK
6      }
7    }
8    (new Inner).f() // OK because now f() is
9  public
   }
```

Q17. Why Scala prefers Immutability?

Ans: Scala prefers Immutability in design and in many cases uses it as default. Immutability can help when dealing with Equality issues or Concurrent programs.

Q18. Mention the Identifiers in Scala.

Ans: There are four types of Scala Identifiers:

-
    - Alphanumeric identifiers
    - Operator identifiers
    - Mixed identifiers
    - Literal identifiers

```
1  //Scala program to demonstrate Identifiers in
2  Scala.
3  object Main
4  {
5  //Main method
6      def main(args: Array[String])
7      {
8          //Valid Identifiers
9          var 'name = "Hari'"
10         var age = 20;
11         var Branch = "Computer Science"
12         println()
13         println()
14         println()
15     }
   }
```

Q19. How do you define a function in Scala?

Ans: def keyword is used to define the Function in Scala.

```
1  object add {
2    def addInt( a:Int, b:Int ) : Int = {
3      var sum:Int = 0
4      sum = a + b
5      return sum
6    }
7  }
```

Q20. How is the Scala code compiled?

Ans: Code is written in Scala IDE or a Scala REPL, Later, the code is converted into a Byte code and transferred to the JVM or Java Virtual Machine for compilation.

Q21.Differentiate between Null, Nil, None and Nothing

Ans: They appear similar but different in their behaviour:

Null	Nil	None	Nothing
Null represents the absence of a value.	Nil denotes the end a List	None is the value of an Option with no value.	Nothing is lowest type in type System.

Q22. Explain If-Else-If terminology

Ans: If-Else-If statement executes one of the three statements.

Example:

```
1  object Demo {
2    def main(args: Array[String]) {
3      var x = 30;
4      if( x == 10 ){
5        println("Value of X is 10")
6      } else if( x == 20 ){
7        println("Value of X is 20");
8      } else if( x == 30 ){
9        println("Value of X is 30");
10     } else{
11       println("This is else statement");
12     }
13   }
14 }
```

Q23. Describe Loops in Scala.

Ans: There are mainly three types of loops in Scala.

-
    - While Loop: Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
    - Do-While: Like a while statement, except that it tests the condition at the end of the loop body.
    - For: Executes a sequence of statements multiple times and abbreviated the code that manages the loop variable.
    - Break: Break is a loop control statement which Terminates the loop statement and transfers execution to the statement immediately following the loop.

Q24. Generate an Infinite loop

Ans: A loop becomes an Infinite loop if a condition never becomes false. If you are using Scala, the while loop is the best way to implement an infinite loop.

The following program implements an infinite loop.

```
1  object Demo {
```

```
2    def main(args: Array[String]) {
3      var a = 10;
4       //An infinite loop.
5      while( true ){
6        println( "Value of a: " + a );
7      }
8    }
9  }
```

Q25. What is the Syntax for function declaration in Scala?

Ans: The Syntax for function declaration is as follows:

```
1  def functionName ([list of parameters]) :
2  [return type] = {
3    function body
4    return [expression]
   }
```

Here, the return type is any valid Scala data type and we separate the list of parameters by comma and list of parameters and return type are optional. Very similar to Java, we use a return statement along with an expression in case function returns a value.

Q26. How do I Concatenate two Strings?

Ans: There are three methods to perform string concatenation in Scala

```
1  string1.concat(string2);
2  "My name is ".concat("Zara");
3  "Hello," + " world" + "!"
```

Q27. Explain any five string methods.

Ans: Following are few String Methods in Scala.

-
    - String trim(): Returns a copy of the string, with leading and trailing whitespace omitted.
    - String toUpperCase: Converts all of the characters in this String to upper case using the rules of the given Locale.
    - Char[] to CharArray(): Converts this string to a new character array.
    - String[] split(String regex): Splits this string around matches of the given regular expression.
    - Int length(): returns the length of this string.

Q28. Explain how to create Arrays

Ans: We use the following methods to create arrays in Scala:

-
    - We use ofDim to declare multidimensional arrays.

```
var myMatrix = ofDim[Int](3,3)
```

-
    - We use Range() method to generate an array containing a sequence of increasing integers in a given range.
    - def range(start: Int, end: Int, step: Int): Array[Int]
    -

```
range (10, 20, 2)
```

Q29. What is Map in Scala?

Ans: Map is a collection of key/value pairs. Scala retrieves a Value based on its Key.

```
1  val colors = Map("red" -> "#FF0000", "azure" -> "#F0FFFF")
```

Q30. Explain Exception Handling in Scala

Ans: Throw Exception: Throwing an exception looks the same as in Java. You create an exception object and then you throw it with the throw keyword as follows.

-
    - Throw new IllegalArgumentException
    - Catching an Exception:

Scala allows you to try/catch any exception in a single block and then perform pattern matching against it using case blocks. Try the following example program to handle the exception.

Example:

```
17  import java.io.FileReader
    import java.io.FileNotFoundException
    import java.io.IOException
    object Demo {
      def main(args: Array[String]) {
        try {
```

	val f = **new** FileReader("input.txt")
	} **catch** {
	case ex: FileNotFoundException ={
	println("Missing file exception")
	}
	case ex: IOException = {
	println("IO Exception")
	}
	}
	}
	}