Explain what is Scala?

Scala is an object functional programming and scripting language for general software applications designed to express solutions in a concise manner.

What is a 'Scala set'? What are methods through which operation sets are expressed?

Scala set is a collection of pairwise elements of the same type. Scala set does not contain any duplicate elements. There are two kinds of sets, mutable and immutable.

What is a 'Scala map'?

Scala map is a collection of key or value pairs. Based on its key any value can be retrieved. Values are not unique but keys are unique in the Map.

What is the advantage of Scala?

a) Less error prone functional style

b) High maintainability and productivity

c) High scalability

d) High testability

e) Provides features of concurrent programming

In what ways Scala is better than other programming language?

a) The arrays uses regular generics, while in other language, generics are bolted on as an afterthought and are completely separate but have overlapping behaviours with arrays.

b) Scala has immutable "val" as a first class language feature. The "val" of scala is similar to Java final variables. Contents may mutate but top reference is immutable.

c) Scala lets 'if blocks', 'for-yield loops', and 'code' in braces to return a value. It is more preferable, and eliminates the need for a separate ternary operator.

d) Singleton has singleton objects rather than C++/Java/ C# classic static. It is a cleaner solution

e) Persistent immutable collections are the default and built into the standard library.

f) It has native tuples and a concise code

g) It has no boiler plate code

What are the Scala variables?

Values and variables are two shapes that come in Scala. A value variable is constant and cannot be changed once assigned. It is immutable, while a regular variable, on the other hand, is mutable, and you can change the value.

The two types of variables are

```
var  myVar : Int=0;

val   myVal: Int=1;
```

Mention the difference between an object and a class ?

A class is a definition for a description. It defines a type in terms of methods and composition of other types. A class is a blueprint of the object. While, an object is a singleton, an instance of a class which is unique. An anonymous class is created for every object in the code, it inherits from whatever classes you declared object to implement.

What is recursion tail in scala?

'Recursion' is a function that calls itself. 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.

What is 'scala trait' in scala?

'Traits' are used to define object types specified by the signature of the supported methods. Scala allows to be partially implemented but traits may not have constructor parameters. A trait consists of method and field definition, by mixing them into classes it can be reused.

When can you use traits?

There is no specific rule when you can use traits, but there is a guideline which you can consider.

a) If the behaviour will not be reused, then make it a concrete class. Anyhow it is not a reusable behaviour.

b) In order to inherit from it in Java code, an abstract class can be used.

c) If efficiency is a priority then lean towards using a class

d) Make it a trait if it might be reused in multiple and unrelated classes. In different parts of the class hierarchy only traits can be mixed into different parts.

e) You can use abstract class, if you want to distribute it in compiled form and expects outside groups to write classes inheriting from it.

What is Case Classes?

Case classes provides a recursive decomposition mechanism via pattern matching, it is a regular classes which export their constructor parameter. The constructor parameters of case classes can be accessed directly and are treated as public values.

What is the use of tuples in scala?

Scala tuples combine a fixed number of items together so that they can be passed around as whole. A tuple is immutable and can hold objects with different types, unlike an array or list.

What is function currying in Scala?

Currying is the technique of transforming a function that takes multiple arguments into a function that takes a single argument Many of the same techniques as language like Haskell and LISP are supported by Scala. Function currying is one of the least used and misunderstood one.

What are implicit parameters in Scala?

Implicit parameter is the way that allows parameters of a method to be "found". It is similar to default parameters, but it has a different mechanism for finding the "default" value. The implicit parameter is a parameter to method or constructor that is marked as implicit. This means if a parameter value is not mentioned then the compiler will search for an "implicit" value defined within a scope.

What is a closure in Scala?

A closure is a function whose return value depends on the value of the variables declared outside the function.

What is Monad in Scala?

A monad is an object that wraps another object. You pass the Monad mini-programs, i.e functions, to perform the data manipulation of the underlying object, instead of manipulating the object directly. Monad chooses how to apply the program to the underlying object.

What is Scala anonymous function?

In a source code, anonymous functions are called ‘function literals’ and at run time, function literals are instantiated into objects called function values. Scala provides a relatively easy syntax for defining anonymous functions.

Explain ‘Scala higher order’ functions?

Scala allows the definition of higher order functions. These are functions that take other functions as parameters, or whose result is a function. In the following example, apply () function takes another function ‘f’ and a value ‘v’ and applies function to v.

Example

```
object Test {
def main(args: Array[String]) {
println( apply( layout, 10) )
}
def apply(f: Int => String, v: Int) = f(v)
def layout[A](x: A) = "[" + x.toString() + "]"
```

```
object Test {
def main(args: Array[String]) {
println( apply( layout, 10) )
}
def apply(f: Int => String, v: Int) = f(v)
def layout[A](x: A) = "[" + x.toString() + "]"
```

When the above code is compiled and executed, it produces following result.

```
C:/>scalac Test.scala
C:/>scala Test
[10]
```

```
C:/>scalac Test.scala
C:/>scala Test
[10]
```

What is the difference between var and value?

In scala, you can define a variable using either a, val or var keywords. The difference between val and var is, var is much like java declaration, but val is little different. We cannot change the reference to point to another reference, once the variable is declared using val. The variable defined using var keywords are mutable and can be changed any number of times.

What are option, some and none in scala?

‘Option’ is a Scala generic type that can either be ‘some’ generic value or none. ‘Queue’ often uses it to represent primitives that may be null.

How do I append to the list?

In scala to append into a list, use ":+" single value

```
var myList = List.empty[String]

myList :+= "a"

myList :+= "b"

myList :+= "c"

use++ for appending a list

var myList = List.empty[String]

myList ++= List("a", "b", "c")

var myList = List.empty[String]

myList :+= "a"

myList :+= "b"

myList :+= "c"

use++ for appending a list

var myList = List.empty[String]

myList ++= List("a", "b", "c")
```

How can you format a string?

To format a string, use the .format () method, in scala you can use

Val formatted= "%s %i".format (mystring.myInt)

Why scala prefers immutability?

Scala prefers immutability in design and in many cases uses it as default. Immutability can help when dealing with equality issues or concurrent programs.

What are the four types of scala identifiers ?

The four types of identifiers are

a) Alpha numeric identifiers

b) Operator identifiers

c) Mixed identifiers

d) Literal identifiers

What are the different types of Scala literals?

The different types of literals in scala are

a) Integer literals ; b) Floating point literals ; c) Boolean literals ;

d) Symbol literals ; e) Character literals ;

f) String literals ; g) Multi-Line strings