Tutorial-01 22001875

1.Scala is a programming language. Scala does support OOP. Father of Scala Martin Odersky.
2.Scala is a statically typed language

Static Type - type checking is performed at compile-time. The type of each variable and expression is known and checked by the compil

Dynamic Type – type checking is performed at runtime the type of each variable is and expression is checkd while the program is running.

Differences

i) **Statically Typed:** Type checking is done at compile-time.
Dynamically Typed: Type checking is done at runtime ii) **Statically Typed:** Often more efficient because the compiler can optimize the code based on known types.
Dynamically Typed: Potentially less efficient due to the overhead of type checking at runtime.
iii) **Statically Typed:** Generally provides better support for advanced IDE features
Dynamically Typed: Tooling support can be more challenging due to the lack of explicit type information

3.Scala is not a pure OOP language it’s a highbred which combines OOP and functional Programming. Java is also not a pure OOP language which doesn’t consider everything as an object
4.Scala does support all the key Functional Programming concepts. Java doesn’t support all the functional programming concepts.

5. Advantage

(1) Scala seamlessly integrates object-oriented and functional programming paradigms
(2) Scala's syntax is concise and expressive, reducing boilerplate code and making programs easier to read.
(3) Scala's strong static type system catches many errors at compile time, while type inference reduces the need for explicit type declarations
(4) Scala encourages immutability, which leads to safer and more predictable code.
(5) Scala's pattern matching is a powerful tool for handling different data structures and simplifying complex control flows

Disadvantage

(1) Scala's rich feature set and flexible syntax can lead to complex codebases
(2) The combination of OOP and FP paradigms, along with advanced language features, can make Scala challenging to learn for beginners
(3) Scala's compile times can be longer compared to some other languages, which can slow down the development process.
(4) Although Scala's tooling has improved over the years, it still lags behind more mature languages
(5) While Scala is interoperable with Java, mixing Scala and Java code can sometimes lead to interoperability issues and subtle bugs

6. The main drawback of the Scala language is its **steep learning curve**. This complexity arises from several factors Because of combination of Paradigms, Rich feature set, Complex syntax
7. The main motto of the Scala language is **"Scalable Language."** This reflects its design philosophy of being scalable in terms of usage and application
8. Java, Kotlin, Scala, Groovy
9. the superclass of all classes is **Any**. The **Any** class is at the top of the Scala type hierarchy and serves a role similar to **java.lang.object** in Java.
10. Public is the default access modifier Scala doesn't have a public keyword.
11. Type inference in Scala refers to the compiler's ability to deduce the types of variables, expressions, and functions based on context and usage, without requiring explicit type annotations from the programmer
12. Int in scala is a primitive datatype, representing 32 bits signed integer.
    Java.lang.Integer is considered as a object and provided additional functionalities.
13.
    Nothing - Represents the absence of a value or a computation that never return a result. Nil
    - Represents an object of empty list

    Nothing is a type representing the absence of a value or a computation that never returns, and it is a subtype of all other types. Nil is an object representing an empty list, and it is an instance of `List[Nothing]`, indicating that it can hold elements of type Nothing.
14.
    Null - is a type representing the null reference in Scala and is a subtype of all reference types.
    null - is the literal representing the null reference itself and can be assigned to variables of reference types.
15.
    Unit – Is a type in Scala that is used to represent the result of an expression that does not return a meaningful value
    Void – also represent absent of a meaning full value returned from a expression.

16.Val is used to declare immutable variables in Scala. Var is used to declare mutable variables in Scala.

17.REPL stands for Read-Eval-Print Loop. It's an interactive programming environment that allows you to enter code, evaluate it, and see the results immediately

i) Experimentation and Testing ii)
Learning and Exploration iii)
Prototyping:
iv) Debugging and Troubleshooting

In the command prompt, type **Scala** and press Enter. This will launch Scala's REPL environment. Now you can start typing Scala code and press Enter to evaluate and see the results immediately

18.Features

i) Static Typing with Type Inference
ii) Object-Oriented Programming (OOP)
iii) Functional Programming (FP
iv) Concurrency and Parallelism - Scala provides built-in support for concurrent and parallel programming using actors, futures, and parallel collections

v) Pattern Matching vi) Higher-Order Functions vii) Immutable Data Structures viii) String Interpolation
ix) Comprehensive Collections Library x)
Java Interoperability

19.In FP loops use recursion and higher order function to achieve iteration ‘ In OOP loops rely on mutable state and imperative constructs for iteration.

In OOP, loops are typically implemented using imperative constructs like `for` loops, `while` loops, or iterators. In FP, loops are implemented using recursion, where a function calls itself with modified arguments to achieve iteration.

20.Application - refers to a program or code snippet that can be executed directly
App - is a trait in Scala's standard library (`scala.App`) that simplifies the creation of Scala Applications.

When an object extends the `App` trait, Scala automatically generates a `main` method that runs the code inside the object. This saves you from having to write a `main` method explicitly, making your code more concise and readable. Handy for interactive sessions and experimenting with Scala code in an easy-to-use and immediate way. Suitable for scripting tasks and automation where you want to run Scala code directly from the command line or as part of a script.

21.Scala supports operator overloading

In Java, operator overloading is not directly supported. Instead, Java provides predefined behaviors for these operators based on the types of operands involved

22.An expression is a combination of variables, operators, and method invocations that produces a value A statement, on the other hand, is a complete unit of execution that does not produce a value by itself

The main difference between an expression and a statement is that an expression evaluates to a value, while a statement performs an action or operation but does not produce a value as a result.

23.Only the syntax is different other than that it has the same features.

24.Scala is primarily an expression-based language. Java is primarily a statement-based language.

25.Java Features Not Supported by Scala:

i) Checked Exceptions - Java supports checked exceptions where methods must declare the exceptions they can throw or handle. Scala don’t have this.
ii) Primitive Data Types
iii) Wildcard Capture in Generics
iv) Strict Object-Oriented Model

Scala Features Not Supported by Java:

i) Pattern Matching
ii) Case Classes and Pattern Matching
iii) Type Inference
iv) Higher-Order Functions and Closures - Scala treats functions as first-class citizens, allowing higher-order functions.

26.In Scala, a function is a standalone piece of code that can be assigned to a variable, passed as an argument, or returned from another function.
A method, on the other hand, is a function that is defined inside a class or object

27.At most one public lass

28.Java.lang._ Sacla._
Predef._

29.There are approximately 40 operators in Scala.
Scala, including arithmetic, comparison, logical, bitwise, assignment, string concatenation, type ascription, range, and symbolic operators.

30.

Public - does not require keywords for method or field visibility within a class
Static - Scala does not have static members like Java. Instead, Scala uses singleton objects and companion objects for similar functionality
Void - uses Unit
Super - refer to a superclass or trait, but it is not required

31. Predef in Scala is an object that provides a set of commonly used definitions and utility functions. It includes commonly used types, functions, and control structures, making code writing more convenient and concise.