EXP. No.: 5

Aim:

To install scala programming environment and implement conditional Structure and looping constructs.

Scala:

Scala is a powerful, high-level programming. Language and that combines object oriented and functional programming paradims .It was created by Martin Odersky. Scala runs on JVM which allows it to interoperate that can take advantage of both object-oriented and functional programming concepts.

Installation Scala on Linux:

1. Install Java

Scala runs on the Java Virtual Machine. So you need do install Java.

open terminal and install OpenJDK.

Sudo apt update

Sudo apt install open jdk-17-jdk

2. Install Scala

Update package list

Sudo apt update

Install scala- Sudo apt install scala

Verify Scala installation to ensure its installed correctly.

Scala-version.

Installation e Scala on Windows:

Using Scala installer

1. Install Java

Download and install the OpenJDK from jdk. java.net.

During the installation,make sure to check the option "set JAVA-HOME variable" or

manually Set the JAVA-HOME envvionment variable after installation.

2. Install Scala

Go to Scala's official website and download windows installer for Scala .Run the the daonloaded .msi file and follow the On-Screen Scala instruction to install scala.

3. Verify Scala installation.

After installation, open the command prompt and type:

Scala -version

Using Windows Subsystem for Linux:

If you prefer Linux install like environment on windows,you can install WSI.

1. Enable Wsl: Open Powershell as Administrator and run.

    wsl -- install

2. Open the Ubuntu terminal from the start menu.

3. Follow the Linux installation above for installing Java and Scala.

Conditional Statements in Scala

Scala provides several ways to make decisions in code using conditional statements. The most common ones are if-else and match.

1. if-else Statement

The if-else statement in Scala is similar to other languages like Java or Python. It allows you to execute a block of code based on whether a condition is true or false.

Syntax:

```
if (condition) {
  // code to execute if condition is true
} else {
  // code to execute if condition is false
}
```

Program:

```
val number = 10

if (number > 0) {
  println("The number is positive.")
}
```

If-Else Statement:

```
val number = -5

if (number > 0) {
  println("The number is positive.")
} else {
  println("The number is not positive.")
}
```

if-else if Ladder

If there are multiple conditions to check, you can use the else if ladder.

Syntax:

```
if (condition1) {
  // code for condition1
} else if (condition2) {
  // code for condition2
} else {
  // code if none of the conditions are true
}
```

Program:

If -Else if -Else Statement:

```
val number = 0

if (number > 0) {
  println("The number is positive.")
} else if (number < 0) {
  println("The number is negative.")
} else {
  println("The number is zero.")
}
```

Looping Constructs in Scala

Scala provides several ways to execute repetitive code blocks, including while, do-while, and for loops.

1. while Loop

The while loop repeatedly executes a block of code as long as the condition is true.

Syntax:

scala

Copy code

```
while (condition) {
  // code to be executed
```

```
}
```
Program:
```
var count = 0
while (count < 5) {
  println(s"Count is: $count")
  count += 1
}
```
do-while Loop

The do-while loop is similar to while, but it guarantees that the block of code will be executed at least once, even if the condition is false.

Syntax:
```
do {
  // code to be executed
} while (condition)
```
Program:
```
var count = 0
do {
  println(s"Count is: $count")
  count += 1
} while (count < 5)
```

for Loop

Scala's for loop is quite versatile. You can loop over ranges, collections, and more. It also supports **guards** and **yield** for generating new collections.

Syntax for Ranges:
```
for (i <- start to end) {
  // code to execute
}
```
Program:
```
for (i <- 0 until 5) {
  println(s"Index is: $i")
}
```
for Loop with Filters (Guards)

You can add conditions (guards) inside a for loop to filter the values.

Example:
```
val numbers = List(1, 2, 3, 4, 5, 6)

for (num <- numbers if num % 2 == 0) {
  println(num)
}
```

for Loop with yield

Scala's for loop can return a new collection using the yield keyword. This is useful when you want to transform the elements in a collection.

Example:

```
val squaresOfEvens = for {
  num <- numbers if num % 2 == 0
} yield num * num

println(squaresOfEvens)
```

Result:

Hence,Scala was successfully installed and conditional structures and looping constructs are implemented successfully.