Experiment No 8

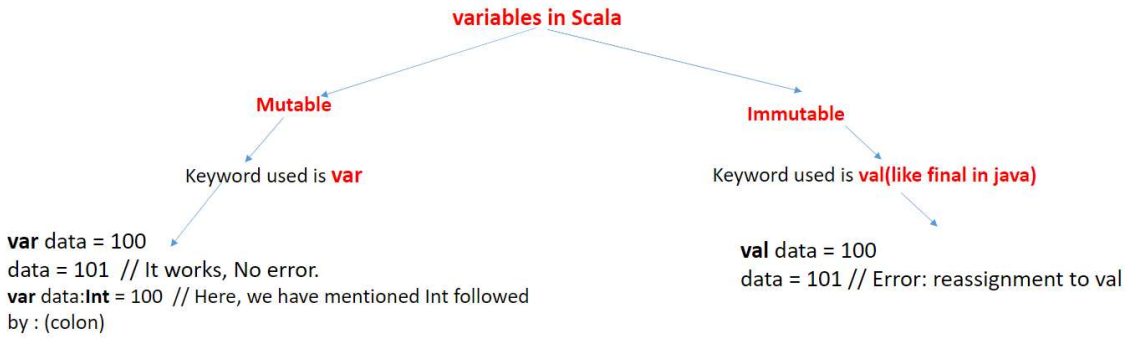
NAME:DIVYESH KHUNT SAPID:60009210116 BATCH:D12

Aim:-Implement a program to demonstrate Scala programming basic Variable, Data types, String interpolation Operators, Precedence Rules, Mathematical Functions, Conditional Statements and Loops

Theory

Basics of Variables, Data Types

Variable is a name which is used to refer memory location. You can create mutable and immutable variable in Scala.



Data Types in Scala

Data types in Scala are much similar to java in terms of their storage, length, except that in Scala there is no concept of primitive data types every type is an object and starts with capital letter.

Data Type	Default Value	Size
Boolean	False	True or false
Byte	0	8 bit signed value (-2^7 to 2^7-1)
Short	0	16 bit signed value(-2^{15} to $2^{15}-1$)
Char	'\u0000'	16 bit unsigned Unicode character(0 to $2^{16}-1$)
Int	0	32 bit signed value(-2^{31} to $2^{31}-1$)
Long	0L	64 bit signed value(-2^{63} to $2^{63}-1$)
Float	0.0F	32 bit IEEE 754 single-precision float
Double	0.0D	64 bit IEEE 754 double-precision float
String	Null	A sequence of characters

String interpolation

Scala offers a new mechanism to create strings from your data. It is called string interpolation. String interpolation allows users to embed variable references directly in processed string literals. Scala provides three string interpolation methods: s, f and raw.

S String Interpolation

The s method of string interpolation allows us to pass variable in string object. You don't need to use + operator to format your output string

F String Interpolation

The f method is used to format your string output. It is like printf function of c language which is used to produce formatted output. You can pass your variables of any type in the print function.

Raw String Interpolation:

The raw method of string interpolation is used to produce raw string. It does not interpret special char present in the string

Example: -

```
object HelloWorld {
  var pi = 3.14
  var s1 = "Scala string example"
  var version = 2.12
  var s2 = "Scala \tstring \nexample"
  var s3 = raw"Scala \tstring \nexample"
    def main(args: Array[String]): Unit = {
    println("value of pi = "+pi)  //String Interpolaqtion
    println(s"value of pi = $pi")  //s Sting Interpolation
    println(s"This is $s1") //s String interPolation with String
    println(f"This is $s1%s, scala version is $version%2.2f")  //f string Interpolation
    println(s2)
    println(s3)  //raw string interpolation
    }
}
```

Output:

```
value of pi = 3.14
value of pi = 3.14
This is Scala string example
This is Scala string example, scala version is 2.12
Scala   string
example
Scala \tstring \nexample
```

Conditional Statements and Loops in Scala

Scala provides if statement to test the conditional expressions. It tests Boolean conditional expression which can be either true or false. Scala use various types of if else statements.

If statement

If-else statement
Nested if-else statement
If-else-if ladder statement

If Statement Example-

```
var age: Int = 20;
if (age > 18) {
   println ("Age is greater than 18")
}
```

Scala if-else example

```
var number:Int = 21
if(number%2==0){
   println("Even number")
}else{
   println("Odd number")
}
```

Scala If-Else-If Ladder Example

```
var number:Int = 85
if(number>=0 && number<50){
   println ("fail")
}
else if(number>=50 && number<60){
   println("D Grade")
}
else if(number>=60 && number<70){
   println("C Grade")
}
else if(number>=70 && number<80){
   println("B Grade")
}
else if(number>=80 && number<90){
   println("A Grade")
}
else if(number>=90 && number<=100){
   println("A+ Grade")
}
else println ("Invalid")
```

Scala If Statement as better alternative of Ternary Operators

In Scala, you can assign if statement result to a function. Scala does not have ternary operator concept like C/C++ but provides more powerful *if* which can return value.

```
 object MainObject {
   def main (args: Array[String]) {
      val result = checkIt (-10)
```

```
    println (result)
  }
  def checkIt (a: Int) = if (a >= 0) 1 else -1    // Passing an if expression value to function
}
```

Scala Pattern Matching

Pattern matching is a feature of Scala. It works same as switch case in other programming languages. It matches best case available in the pattern.

```
object MainObject {
  def main (args: Array[String]) {
      var a = 1
      a match {
         case 1 => println("One")
         case 2 => println("Two")
         case _ => println("No")
      }
      }
}
```

Scala while loop

In Scala, while loop is used to iterate code till the specified condition. It tests boolean expression and iterates again and again. You are recommended to use while loop if you don't know number of iterations prior.

```
object MainObject {
  def main(args: Array[String]) {
    var a = 10;                  // Initialization
    while( a<=20 ){              // Condition
      println(a);
      a = a+2                    // Incrementation
    }
     }
  }
```

Scala do-while loop example

```
object MainObject {
  def main (args: Array[String]) {
      var a = 10;       // Initialization
      do {
         println(a );
         a = a + 2;     // Increment
      }
      while(a <= 20 )    // Condition
  }
}
```

Scala for loop

Syntax

```
for(i <- range){
   // statements to be executed
}
```

Scala for-loop example by using *to* keyword

```
object MainObject {
  def main(args: Array[String]) {
      for( a <- 1 to 10 ){
       println(a);
     }
  }
}
```

Scala for-loop Example by using *until* keyword

```
object MainObject {
  def main(args: Array[String]) {
      for( a <- 1 until 10 ){
       println(a);
     }
  }
}
```

Scala for-loop filtering Example

You can use *for* to filter your data. In the below example, we are filtering our data by passing a conditional expression. This program prints only even values in the given range.

```
object MainObject {
  def main(args: Array[String]) {
      for( a <- 1 to 10 if a%2==0 ){
       println(a);
     }
  }
}
```

Scala for-loop in Collection

In scala, you can iterate collections like list, sequence etc, either by using for each loop or for-comprehensions.

```
object MainObject {
  def main(args: Array[String]) {
      var list = List(1,2,3,4,5,6,7,8,9)          // Creating a list
```

```
      for( i <- list){                    // Iterating the list
         println(i)
      }

   }
}
```

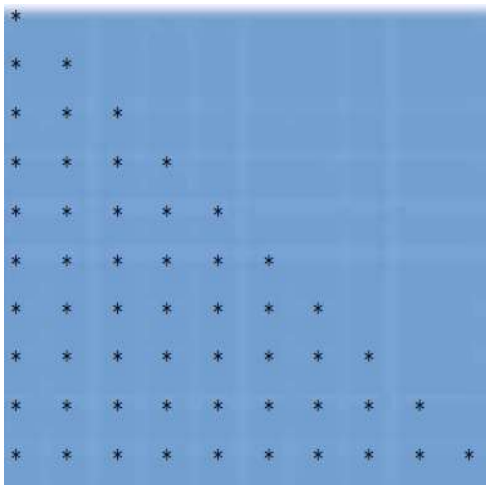
Type Casting in Scala

Type conversion is done automatically by compiler. A type conversion is alteration from one data type to another data type. Scala supports automatic type conversion as follows: Byte->Short->Int->Long->Float->Double.

Note that Scala does not supports reverse conversion e.g. Double->Float->Long->Int->Short->Byte.

Lab Experiments to be Performed in this Session: -

1. Write a program to find max of 3 Nos.
2. Write a program to print given no in words using pattern matching and while loop .eg 123 output one two three.
3. Write a program to find whether the no is prime or not using do while loop.
4. Write a program in Scala to demonstrate string interpolation.
5. Write a program that prints the following patterns.



CODE:

```scala
object q1 {
    def main(args: Array[String]): Unit = {
    println("Enter number1:")
    val n1 = scala.io.StdIn.readInt()
    println("Enter number2:")
    val n2 = scala.io.StdIn.readInt()
    println("Enter number3:")
    val n3 = scala.io.StdIn.readInt()
    println(s"You entered the integer: $n1 $n2 $n3")
    if(n1>n2 && n1>n3 ) {
      println(s"The greatest number is $n1 ")}
    if(n2>n1 && n2>n3 ) {
      println(s"The greatest number is $n2")}
    if(n3>n2 && n3>n1 ) {
      println(s"The greatest number is $n3 ")
    }
    }
}
```

✔ **EXECUTE MODE, VERSION, INPUTS & ARGUMENTS**

Stdin Inputs

```
10
2
12
```

RESULT

```
Enter number1:
Enter number2:
Enter number3:
You entered the integer: 10 2 12
The greatest number is 12
```

CPU Time: 8.13 sec(s), Memory: 151204 kilobyte(s)

```scala
object q1 {
    def main(args: Array[String]): Unit = {
        val a = Array("zero","one","two","three","four","five", "six","seven","eight","nine")
        println("Enter number:")
        var n = scala.io.StdIn.readInt()

        var result = ""

        while(n>0){
         var digit=n%10
        n=n/10
        result=a(digit)+" "+result
        }
        print(s"$result")
    }
    }
```

Stdin Inputs

```
123
```

RESULT

```
Enter number:
one two three
```

```scala
object PrimeCheckRecursive {
  def isPrime(n: Int, i: Int): Boolean = {
    if (n == 0 || n == 1)
      false
    else if (n == i)
      true
    else if (n % i == 0)
      false
    else
      isPrime(n, i + 1)
  }

  def main(args: Array[String]): Unit = {
    val n = scala.io.StdIn.readInt()
    if (isPrime(n, 2))
      println(s"$n number is prime")
    else
      println(s"$n number is not prime")
  }
}

```

Stdin Inputs

```
12
```

Interactive

CommandLine Arguments

RESULT

```
12 number is not prime
```

```scala
object PrimeCheckRecursive {

  def main(args: Array[String]): Unit = {
    val name = scala.io.StdIn.readLine()
    val age = scala.io.StdIn.readInt()

    val message = s"My name is $name and I am $age years old."
    println(message)
  }
}

```

Stdin Inputs

```
Divyesh
19
```

Interactive

CommandLine Arguments

RESULT

```
My name is Divyesh and I am 19 years old.
```

```
object PrimeCheckRecursive {

  def main(args: Array[String]): Unit = {
        for (i <- 1 to 10) {
            println("")
                for (j <- 1 to i) {
                    print("*")
  }
}
  }
}
```

RESULT

```

*
**
***
****
*****
******
*******
********
*********
**********
```