NAME : Priya Kadu

SEAT NO. :994

SUBJECT : PPL PRACTICAL JOURNAL

# Control Structures :

**1) Write a program to calculate average of all numbers between n1 and n2(eg.100 to 300 Read values of n1 and n2 from user)**

**Ans:**

```
object Average_Prime{
var count = 0
var sum:Double = 0
def main(args: Array[String]) {
print("\nEnter the Num1: ")
var num1 = scala.io.StdIn.readInt()
print("Enter the Num2: ")
var num2 = scala.io.StdIn.readInt()
println("Average is: " +avg_cal(num1,num2))
}
def avg_cal(n1:Int, n2: Int): Double ={
var n = 0
if(n1==1)
n = 2
else
n = n2
for (i <- n to n2 if isPrime(i)) {
count = count + 1
sum = sum + i
}
var avg: Double = sum / count
avg
}
def isPrime(n: Int) = {
(2 until n) forall (n % _ != 0)
}
}
```

**Output:**

```
Enter the Num1: 1
Enter the Num2: 30
Average is: 12.9
```

**2) Write a program to calculate factorial of a number.**

**Ans**:

```
object Factorial
{
def factorialIt(n: Int): Int =
{
var factorial = 1
for(i <-1 to n)
factorial *= i
return factorial
}
def main(args: Array[String])
{
print("Enter number : ")
val n = scala.io.StdIn.readInt()
println("The factorial of " + n + " is " + factorialIt(n))
}
}
```

**Output:**

```
Enter number : 8
The factorial of 8 is 40320
```

**3) Write a program to read five random numbers and check that random numbers are perfect number or not**.

**Ans:**

```
object  Perfect
{
def main(args : Array[String])
{
var r=scala.util.Random;
var sum =0;
for(i <- 1 to 5)
{
var n = r.nextInt(50)
for(j <- 1 to n-1)
{
if(n % j == 0)
{
sum += j
}
}
if(sum == n)
println(n + " is a perfect number")
else
println(n + " is not a perfect number")
}
}
}
```

**Output:**

```
34 is not a perfect number
43 is not a perfect number
45 is not a perfect number
28 is not a perfect number
49 is not a perfect number
```

# Arrays :

**1) Write a program to find maximum and minimum of an array**

**Ans:**

```
object MaxMinOfArray
{
def main(args: Array[String]) ={
print("Enter the size of array: ")
var size = scala.io.StdIn.readInt()
var arr = new Array[Int](size)
print("Enter Elements of Array: \n")
for(i <- 0 to size-1)
{
arr(i) = scala.io.StdIn.readInt()
}
println("Max element is: "+ maximum(size, arr))
println("Min element is: "+ minimum(size, arr))
}
def maximum(size: Int,Arr:Array[Int]): Int= {
var max = 0
for(i<- 0 to size -1 ) {
if(Arr(i)>max) {
max = Arr(i)
}
}
return max
}
def minimum(size: Int,Arr:Array[Int]): Int= {
var min = Arr(0)
for(i<- 1 to size -1 ) {
if(Arr(i)<min) {
min = Arr(i)
}
}
return min
}
}
```

**Output**:

```
Enter the size of array: 5
Enter Elements of Array:
4
7
9
2
5
Max element is: 9
Min element is: 2
```

**2) Write a program to calculate transpose of a matrix.**

**Ans**:

```
object TransposeOfMatrix
{
def main(args: Array[String]) ={
var Matrix1 = Array.ofDim[Int](10, 10)
print("Enter no. of rows: ")
val rows = scala.io.StdIn.readInt()
print("Enter No. of columns: ")
val cols = scala.io.StdIn.readInt()
print("Enter values matrix: \n")
getMatrix(rows, cols, Matrix1)
println("The given first matrix is: ")
display(rows, cols, Matrix1)
transpose(rows, cols, Matrix1)
}
def getMatrix(rows: Int, cols: Int, Arr:
Array[Array[Int]]) {
for (i <- 0 to rows - 1) {
for (j <- 0 to cols - 1) {
Arr(i)(j) = scala.io.StdIn.readInt()
}
}
}
def display(rows: Int, cols: Int, Arr:
Array[Array[Int]]) {
for (i <- 0 to rows - 1) {
for (j <- 0 to cols - 1) {
print(Arr(i)(j) + " ")
}
print("\n")
}
}
def transpose(rows: Int, cols: Int, Arr:
Array[Array[Int]])
{
var Matrix2 = Array.ofDim[Int](10, 10)
for (i <- 0 to rows - 1) {
for (j <- 0 to cols - 1) {
Matrix2(i)(j) = Arr(j)(i)
}
}
println("The transpose of given first matrix is: ")
display(rows, cols, Matrix2)
}
}
```

**Output**:
```
Enter no. of rows: 2
Enter No. of columns: 2
Enter values matrix:
4
8
3
6
The given first matrix is:
4 8
3 6
The transpose of given first matrix is:
4 3
8 6
```

**3) Write a program to calculate determinant of a matrix**.

**Ans**:
```
object Determinant
{
def main(args: Array[String]) =
{
val matrix= Array.ofDim[Int](10,10)
print("Enter Row :")
val r=scala.io.StdIn.readInt()
print("Enter column :")
val c=scala.io.StdIn.readInt()
if(r!=c)
{
println("Rows and columns should be equal , please run once again")
}
else if(r==2 && c==2 || r==3 && c==3)
{
printf("\n\nCalculate the determinant of a " + r + " x " + c + " matrix :\n")
print("------------------------------------------------\n")
printf("Enter the values in the matrix :\n")
for(i<-0 to r-1)
{
for(j<-0 to c-1)
{
matrix(i)(j)=scala.io.StdIn.readInt()
}
}
det(r,c,matrix)
}
else
{
print("This Program calculate only for 2x2 and 3x3 matrix , please run once
again and enter 2 or 3 rows and columns")
}
}
def det(r:Int,c:Int,matrix:Array[Array[Int]])=
{
if(r==2 && c==2)
{
printf("The matrix is :\n")
for(i<-0 to r-1)
{
for(j<-0 to c-1)
print(matrix(i)(j)+" ");
println( )
}
val determinant= matrix(0)(0)*matrix(1)(1)-matrix(1)(0)*matrix(0)(1)
println("Determinant = "+determinant)
}
else if(r==3 && c==3)
{
printf("The matrix is :\n")
for(i<-0 to r-1)
{
for(j<-0 to c-1)
print(matrix(i)(j)+" ");
println( )
}
var x=matrix(0)(0)*(matrix(1)(1)*matrix(2)(2)-matrix(1)(2)*matrix(2)(1))
var y=matrix(0)(1)*(matrix(1)(0)*matrix(2)(2)-matrix(1)(2)*matrix(2)(0))
var z=matrix(0)(2)*(matrix(1)(0)*matrix(2)(1)-matrix(1)(1)*matrix(2)(0))
var Determinant= x - y + z
println("Determinant = "+Determinant)
}
}
```

```
}
```

**Output**:

```
Enter Row :2
Enter column :2


Calculate the determinant of a 2 x 2 matrix :
----------------------------------------------
Enter the values in the matrix :
1
7
9
4
The matrix is :
1 7
9 4
Determinant = -59
```

## String :

**1) Write program to count uppercase letters in a string and convert it to lowercase and display the new string.**

**Ans:**

```
object StringDemo {
def main(args: Array[String]) {
var count = 0
print("\nEnter the string: ")
val string = scala.io.StdIn.readLine()
for(i<-string){
if(i.isUpper == true) {
count = count+1
}
}
val str = for (c <- string) yield c.toLower
println("\nThis string is '" + string + "'")
println("Number of Uppercase letters in the string are: " +count)
println("\nThe new string is '" + str + "'")
}
}
```

Output:

```
Enter the string: I Love INDIA

This string is 'I Love INDIA'
Number of Uppercase letters in the string are: 7

The new string is 'i love india'
```

**2) Write a program to read a character from user and count the number of occurrences of that character.**

**Ans**:

```
object CountDemo {
def main(args: Array[String]) ={
print("\nEnter the string: ")
val string = scala.io.StdIn.readLine()
print("Enter the character you want to count: ")
val c = scala.io.StdIn.readChar()
val count = string.count(_ == c)
println("\nThis string is '" + string + "'")
println(s"Count of '$c' in the string : " + count)
}
}
object Count_Demo {
def main(args: Array[String]) {
var count = 0
print("\nEnter the string: ")
val string = scala.io.StdIn.readLine()
print("Enter the character you want to count: ")
val c = scala.io.StdIn.readChar()
for(i<-string){
if(i==c)
count = count+1
}
println("\nThis string is '" + string + "'")
println(s"Count of '$c' in the string :  " + count)
}
}
```

**Output**:

```
Enter the string: Hello World
Enter the character you want to count: l

This string is 'Hellpo World'
Count of 'l' in the string : 3
```

**3) Write a program to read two strings. Remove the occurrence of second string in first string.**

**Ans**:

```
object Scala_String {
def test(str1: String, str2: String): String = {
if (str1.length == str2.length)
return str1 + str2;
if (str1.length > str2.length) {
var diff = str1.length - str2.length;
str1.substring(diff, str1.length) + str2;
} else {
var diff = str2.length - str1.length;
str1 + str2.substring(diff, str2.length);
}
}
def main(args: Array[String]): Unit = {
var str1 = "Welcome";
var str2 = "home";
println("The given strings is: " + str1 + " and " + str2);
println("The new string is: " + test(str1, str2));
str1 = "Scala";
str2 = "Python";
println("The given strings is: " + str1 + " and " + str2);
println("The new string is: " + test(str1, str2));
}
}
```

**Output:**

```
The given strings is: Welcome and home
The new string is: comehome
The given strings is: Scala and Python
The new string is: Scalaython
```

# Classes and Objects :

**1) Define a class CurrentAccount (accNo, name, balance, minBalance). Define appropriate constructors and operations withdraw(), deposit(), viewBalance(). Create an object and perform operations.**

**Ans**:

```
import scala.io.StdIn._
import scala.util._
class CurrentAccount(var ano:Int,var nam:String,var minBal:Float)
{
var accNo:Int=ano
var name:String=nam
var balance:Float=minBal
var minBalance:Float=minBal

def withdraw()
{
println("Enter the amount to be withdraw:")
var amt:Float=scala.io.StdIn.readFloat()
if((balance-amt)>=minBalance)
{
println("Balance withdraw successfully:")
balance=balance-amt
println("Remaining Balance="+balance)
}
else
{
println("you can only withdraw amount greater than minBalance
i.e."+minBalance)
}
}

def deposit()
{
println("Balance before deposit is="+balance)
println("Enter the amount to deposit:")
var amt=scala.io.StdIn.readFloat()
balance=balance+amt
println("Balance after deposit="+balance)
}

def viewBalance()
{
println("Account Balance="+balance)
}
}
object CurrentAccountDemo
{
def main(args:Array[String])
{
println("Create New Account For Customer:")
println("Enter the account number:")
var ano=scala.io.StdIn.readInt()
println("Enter the account holder name:")
var nam=scala.io.StdIn.readLine()
println("Enter the account minimum balance:")
var min=scala.io.StdIn.readFloat()

var obj=new CurrentAccount(ano,nam,min)
var op=4
do
{
println("1.withdraw")
println("2.deposit")
println("3.viewBalance")
println("4.exit")
println("Enter the operation you want to perform:")
op=scala.io.StdIn.readInt()
op match {
case 1 =>obj.withdraw()
case 2 =>obj.deposit()
case 3 =>obj.viewBalance()
case whoa =>println("Unexpected case:"+whoa.toString)
}
}while(op!=4);
}
}
```

**Output**:
```
Create New Account For Customer:
Enter the account number:
101
Enter the account holder name:
Akash
Enter the account minimum balance:
0
1.withdraw
2.deposit
3.viewBalance
4.exit
Enter the operation you want to perform:
2
Balance before deposit is=0.0
Enter the amount to deposit:
5000
Balance after deposit=5000.0
1.withdraw
2.deposit
3.viewBalance
4.exit
Enter the operation you want to perform:
1
Enter the amount to be withdraw:
200
Balance withdraw successfully:
Remaining Balance=4800.0
1.withdraw
2.deposit
3.viewBalance
4.exit
Enter the operation you want to perform:
3
Account Balance=4800.0
1.withdraw
2.deposit
3.viewBalance
4.exit
```

**2) Define a class Employee (id, name, salary). Define methods accept() and display(). Display details of employee having maximum salary**.

**Ans**:
```
 class Employee {
var id: Int = 0
var name: String = _
var salary: Int = 0

def accept(): Unit = {
println("Enter employee's id")
id = scala.io.StdIn.readInt()
println("Enter employee's name")
name = scala.io.StdIn.readLine()
println("Enter employee's salary")
salary = scala.io.StdIn.readInt()
}

def display(): Unit = {
println("The employee id: " + id+ ", name: " +name + " and salary: " +salary )
}
}

object EmployeeObj {
def main(args: Array[String]): Unit = {
println("Enter the no of employees")
val emp_no = scala.io.StdIn.readInt()
val emp = new Array[Employee](emp_no)

for (i <- 0 until emp_no) {
  emp(i) = new Employee
  emp(i).accept()
  emp(i).display()
}

var max: Employee = emp(0)

for (j <- 0 until emp_no){
if( emp(j).salary > max.salary ){
max = emp(j)
}
}
println("The employee with maximum salary is: id: " + max.id+ ", name: "
+max.name + " and salary: " + max.salary )
}
}
```

**Output**:

```
Enter the no of employees
3
Enter employee's id
10
Enter employee's name
Akash
Enter employee's salary
100000
The employee id: 10, name: Akash and salary: 100000
Enter employee's id
11
Enter employee's name
Sohail
Enter employee's salary
50000
The employee id: 11, name: Sohail and salary: 50000
Enter employee's id
12
Enter employee's name
Sachin
Enter employee's salary
75000
The employee id: 12, name: Sachin and salary: 75000
The employee with maximum salary is: id: 10, name: Akash and salary: 100000
```

**3) Create abstract class Order (id, description). Derive two classes PurchaseOrder&amp; SalesOrder with members Vendor and Customer. Create object of each PurchaseOrder and SalesOrder. Display the details of each account.**

**Ans**:

```
abstract class Order
```

```
{
var id:Int=101
var desc:String="sweets"
var price:Int=500
}
class PurchaseOrder extends Order
{
var sweet_name:String="jalebi"
def purchesinfo()
{
println("order id : " +id)
println("description : " +desc)
println("name : " +sweet_name)
}}
class SalesOrder extends Order
{
var customer:String="Rushi dixit"
var vendor:String="Aditya Mohol"
def salesinfo()
{
println("price:" +price)
println("customer_name : " +customer)
println("vendor_name : " + vendor)
}
}
object Demo3
{
def main(args:Array[String])
{
var obj=new PurchaseOrder()
var obj1=new SalesOrder()
obj.purchesinfo()
obj1.salesinfo()
}
}
```

**Output**:

```
order id : 101
description : sweets
name : jalebi
```

```
price : 500
customer_name : Rushi dixit
vendor_name : Aditya Mohol
```

## <u>List :</u>

**1) Create Lists using five different methods( Lisp style , Java style, fill, range and tabulate methods).**

**Ans**:

```
object Scala_List
{
def main(args: Array[String]): Unit =
{
println("Scala List:")
println("Lisp style:")
val lisp_list = 101 :: 226 :: 30 :: Nil
println(lisp_list)
println("Java style:")
val num = List(1,2,3,4,5,6,7)
println(num)
println("Range List:")
val b = List.range(1, 15)
println(b)
val c = List.range(0, 20, 2)
println(c)
println("Uniform  List:")
val d = List.fill(3)("PPL")
println(d)
println("Tabulated List:")
val e = List.tabulate(5)(n => n * n * n)
println(e)
}
}
```

**Output**:

```
Scala List:
Lisp style:
List(101, 226, 30)
```

```
Java style:
List(1, 2, 3, 4, 5, 6, 7)
Range List:
List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
List(0, 2, 4, 6, 8, 10, 12, 14, 16, 18)
Uniform  List:
List(PPL, PPL, PPL)
Tabulated List:
List(0, 1, 8, 27, 64)
```

**2) Create two Lists and Merge it and store the sorted in ascending order.**

**Ans**:

```
import scala.collection.immutable._
object MainObject{
def main(args:Array[String]){
var list1 = List(1,8,5,6,9,58,23,15,4)
var list2 = List(88,100)
print("Elements in list 1: ")
list1.foreach((element:Int) => print(element+" "))
print("\nElements in list 2: ")
list2.foreach((element:Int) => print(element+" "))
var list3 = list1 ++ list2
print("\nElement after merging list1 and list2: ")
list3.foreach((element:Int)=>print(element+" "))
var list5 = list3.sorted
print("\nElements in ascending order of list: ")
list5.foreach((element:Int)=>print(element+" "))
}
}
```

**Output**:

```
Elements in list 1: 1 8 5 6 9 58 23 15 4
Elements in list 2: 88 100
Element after merging list1 and list2: 1 8 5 6 9 58 23 15 4 88 100
Elements in ascending order of list: 1 4 5 6 8 9 15 23 58 88 100
```

**3) Create a list of integers divisible by 3 from List containing numbers from 1 to 50.**

**Ans:**

```
object List_int{
def main(args:Array[String])
{
val x=List.range(1,50)
println("Number divided by 3")
for
{
i<-x
if i%3==0
}
println(i)
}
}
```

**Output**:

```
Number divided by 3
3
6
9
12
15
18
21
24
27
30
33
36
39
42
45
48
```

## Map :

**1) Write a user defined functions to convert lowercase letter to uppercase and call the function using Map.**

**Ans**:
```
object map1
{
def main(args:Array[String])
{
println("Enter the string :")
val lower = scala.io.StdIn.readLine()
val upper = lower.map(c=>c.toUpper)
println("The Upper case string is : "+upper)
}
}
```

**Output**:
```
Enter the string :
hello world
The Upper case string is : HELLO WORLD
```

**2) Write a program to create map with Rollno and FirstName. Print all student information with same FirstName**.

**Ans**:
```
object Slip1
{
def main(a:Array[String])
{
var map=Map(1->"mayuri",2->"pooja",3->"pooja",4->"mayuri",5->"puja");
for((k1,v1) <- map)
{
for((k2,v2)<-map)
{
if(v1==v2 && k1!=k2)
{
println(" Roll No: "+ k1+" FirstName:"+v1);
}
}
}
}
}
```

**Output**:

```
Roll No: 1 FirstName:mayuri
Roll No: 2 FirstName:pooja
Roll No: 3 FirstName:pooja
Roll No: 4 FirstName:mayuri
```

## Set :

**1) Write a program to create two sets and find common elements between them**.

**Ans**:

```
import scala.io.StdIn.{readInt}
import scala.collection.mutable.Set
object Set_p2{
def main(args:Array[String]){
val set1=Set[Int]()
println("how many numbers enter in a set :")
var n=readInt()
println("Enter number of elements in set :")
for(i<-0 to n-1){
set1 += readInt()
}
println("Largest element of the set is: "+set1.max)
println("Smallest element of the set is: "+set1.min)
}
}
```

**Output**:

```
how many numbers enter in a set :
3
Enter number of elements in set :
2
9
5
```

```
Largest element of the set is: 9
Smallest element of the set is: 2
```

**2) Write a program to display largest and smallest element of the Set.**

**Ans:**

```
import scala.io.StdIn.{readInt}
import scala.collection.mutable.Set
object Set_p1{
def main(args:Array[String]){
val set1=Set[Int]()
val set2=Set[Int]()
println("how many numbers enter in a set 1 :")
var n=readInt()
println("Enter number of elements in set :")
for(i<-0 to n-1){
set1 += readInt()
}
println("how many numbers enter in a set 2 :")
n=readInt()
println("Enter number of elements in set :")
for(i<-0 to n-1){
set2 += readInt()
}
println("Common elements between set1 and set2 are: "+set1.&(set2))

println("Common elements between set1 and set2 are: "+set1.intersect(set2))

}
}
```

**Output**:

```
how many numbers enter in a set 1 :
2
Enter number of elements in set :
4
6
how many numbers enter in a set 2 :
4
Enter number of elements in set :
```

```
7
4
6
3
Common elements between set1 and set2 are: HashSet(4, 6)
Common elements between set1 and set2 are: HashSet(4, 6)
```

**3) Write a program to merge two sets and calculate product and average of all elements of the Set.**

**Ans**:

```
object set {
def main(args: Array[String]): Unit = {
val IntSet1 = Set(1, 2, 3, 4)
println("Set 1 is = "+IntSet1)
val IntSet2 = Set(5, 6, 7)
println("Set 2 is = "+IntSet2)
val MergeSet = IntSet1 ++ IntSet2
println("Set after Merging Two Sets = "+MergeSet)
val ProductOfSet = MergeSet.product
println("Product of the Merge Set is = "+ProductOfSet)
val LengthOfSet = MergeSet.size
println("Length of Merge Set = "+LengthOfSet)
val SumOfSet = MergeSet.sum
println("Sum of the Merge Set = "+SumOfSet)
val AverageOfSet = SumOfSet/LengthOfSet
println("Average of the Merge Set is = "+AverageOfSet)
}
}
```

**Output**:

```
Set 1 is = Set(1, 2, 3, 4)
Set 2 is = Set(5, 6, 7)
Set after Merging Two Sets = HashSet(5, 1, 6, 2, 7, 3, 4)
Product of the Merge Set is = 5040
Length of Merge Set = 7
Sum of the Merge Set = 28
Average of the Merge Set is = 4
```