1

**1 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)**

**object** AverageAll

{

**def** main(args:Array[*String*])

{

println("Enter 2 Numbers : ")

**var** n1:Int=scala.io.StdIn.readInt()

**var** n2:Int=scala.io.StdIn.readInt()

**var** n:Int=(n2-n1)+1

**var** sum:Int=0

**for**(i<-n1 to n2)

{

sum=sum+i

}

println(sum)

**var** avg:Int=sum/n

println("Average of all numbers between "+n1+" to "+n2+" is "+avg)

}

}

Output

Enter 2 Numbers :

100

300

40200

Average of all numbers between 100 to 300 is 200

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

**object** Factorial

{

**def** main(args:Array[*String*])

{

println("Enter Number : ")

**var** x:Int=scala.io.StdIn.readInt()

println("Factorial of "+x+"="+factorial(x))

}

**def** factorial(n:Int):Int=

{

**if**(n==0)

**return** 1

**else**

**return** n\*factorial(n-1)

}

}

Output

Enter Number :

5

Factorial of 5=120



2

**3. Write a program to read five random numbers and check that**

**random numbers are perfect number or not.**

**object** PerfectNum

{

**def** main(args:Array[*String*])

{

//var Random=Array(6,15,28,67,496)

println("The 5 random numbers are ")

/\*for(r<-Random)

{

print(" "+r)

}\*/

print("\n")

**for**(no<- 1 to 5)

{

**var** sum:Int=0

**var** i:Int=1

**var** r=**new** scala.util.Random

**var** n=r.nextInt(50);

**while**(i<n)

{

**if**(n%i==0)

{

sum=sum+i

}

i=i+1

}

**if**(sum==n)

{

println(n+" is a perfect number ")

}

**else**

{

println(n+" is not a perfect number ")

}

}

}

}

Output

The 5 random numbers are

7 is not a perfect number

6 is a perfect number

34 is not a perfect number

18 is not a perfect number

1 is not a perfect number



3

**4. Write a program to find second maximum number of four given**

**numbers**

**object** SecondMax

{

**def** main(args:Array[*String*])

{

println("Enter the size of array ")

**var** n:Int=scala.io.StdIn.readInt()

**var** Arr=**new** Array[Int](n)

**var** max1,max2:Int=0

println("Enter the "+n+" Numbers of Array : ")

**for**(i<- 0 to n-1)

{

Arr(i) = scala.io.StdIn.readInt()

}

println("The Array elements are : ")

**for**(a<-Arr)

{

print(" "+a)

}

**for**(i<- 0 to n-1)

{

**if**(Arr(i)>max1)

{

max2=max1

max1=Arr(i)

}

**else if**(Arr(i)>max2 && Arr(i)!=max1)

{

max2=Arr(i)

}

}

println("\nThe second largest number is : "+max2)

}

}

Output

Enter the size of array

4

Enter the 4 Numbers of Array :

98

5412

543

435

The Array elements are :

98 5412 543 435

The second largest number is : 543



4

**5. Write a program to calculate sum of prime numbers between 1**

**to 100**

**object** PrimeSum

{

**def** main(args:Array[*String*])

{

**var** i=1

**var** sum:Int=0

println("Enter value of n : ")

**var** n:Int=scala.io.StdIn.readInt()

//var n=100

**var** Prime=**new** Array[Int](n)

**var** count=0

println("Prime numbers between 1 to "+n+" are : ")

**for**(num<- 2 to n)

{

count=0

**for**(i<- 1 to num)

{

**if**(num%i==0)

{

count=count+1

}

}

**if**(count==2)

{

Prime:+=num

print(" "+num)

}

}

**var** a:Int=0

**while**(a<Prime.length)

{

sum+=Prime(a)

a+=1

}

println("\nSum of all Prime numbers between 1 to "+n+" is "+sum)

}

}

Output

Enter value of n :

100

Prime numbers between 1 to 100 are :

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Sum of all Prime numbers between 1 to 100 is 1060



5

**6. Write a program to read an integer from user and convert it**

**to binary and octal using user defined functions.**

**object** DecToBinaryOctal

{

**def** main(args:Array[*String*])

{

println("Enter a Numbers ")

**var** no:Int=scala.io.StdIn.readInt()

binary(no)

octal(no)

}

**def** binary(no:Int)

{

**var** n=no

print("Binary number of "+n+" is ")

**var** Binary=**new** Array[Int](32)

**var** i:Int=0

**while**(n>0)

{

Binary(i)=n%2

n=n/2

i=i+1

}

**for**(j<- i to 0 by -1)

{

print(Binary(j))

}

}

**def** octal(no:Int)

{

**var** m=no

print("\nOctal number of "+m+" is ")

**var** Octal=**new** Array[Int](100)

**var** i:Int=0

**while**(m!=0)

{

Octal(i)=m%8

m=m/8

i=i+1

}

**for**(j<- i to 0 by -1)

{

print(Octal(j))

}

}

}

Output

Enter a Numbers

44

Binary number of 44 is 0101100

Octal number of 44 is 054



6

7

**2 Arrays**

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

**object** MinMaxArray

{

**def** main(args:Array[*String*])

{

println("How many elements : ")

**val** n:Int=scala.io.StdIn.readInt()

**var** marks = **new** Array[Int](n)

println("Enter the Array elements : ")

**for**(i<- 0 to n-1)

{

marks(i) = scala.io.StdIn.readInt()

}

print("Array elements are : ")

**for**(i<- 0 to (marks.length-1))

{

print(" "+marks(i))

}

**var** min:Int= marks(0)

**var** max:Int= marks(0)

**for**(i <- 1 to n-1)

{

**if**(marks(i)< min)

min=marks(i)

**else if** (marks(i)>max)

max = marks(i)

}

println("\nMinimum = "+min)

println("Maximum = "+max)

}

}

Output

How many elements :

6

Enter the Array elements :

34

877

12

7855

4

345

Array elements are : 34 877 12 7855 4 345

Minimum = 4

Maximum = 7855



8

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

**object** TransposeMatrix

{

**def** main(args:Array[*String*])

{

**var** A=Array.ofDim[Int](10,10)

**var** B=Array.ofDim[Int](10,10)

println("How many rows and columns in the matrix : ")

**var** r:Int=scala.io.StdIn.readInt()

**var** c:Int=scala.io.StdIn.readInt()

/\* Accepting Array Elements \*/

println("Enter the element of matrix : ")

**for**(i<- 1 to r)

{

**for**(j<- 1 to c)

{

A(i)(j)=scala.io.StdIn.readInt()

}

}

println("Matrix Before Transpose ")

**for**(i<- 1 to r)

{

**for**(j<- 1 to c)

{

print(" "+A(i)(j))

}

print("\n")

}

/\* Transpose of Matrix A and Store it to Matrix B \*/

**for**(i<- 1 to r)

{

**for**(j<- 1 to c)

{

B(j)(i)=A(i)(j)

}

}

println("Matrix After Transpose ")

**for**(i<- 1 to r)

{

**for**(j<- 1 to c)

{

print(" "+B(i)(j))

}

print("\n")

}

}

}

Output

How many rows and columns in the matrix :

3

3

Enter the element of matrix :

1

2

3

4

5

6



9

7

8

9

Matrix Before Transpose

1 2 3

4 5 6

7 8 9

Matrix After Transpose

1 4 7

2 5 8

3 6 9

10

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

**object** DeterminantMatrix

{

**def** main(args: Array[*String*])

{

println("Specify a size for your n \* n matrix:")

**val** size = scala.io.StdIn.readInt

**val** m = randomSqMatrix(size, 10)

println("Random Matrix:")

println(toString(m))

println("Determinant of the matrix above:")

println(getDeterminant(m))

println("---------------------\n")

main(Array());

}

**type** *Matrix* = *List*[*List*[Int]]

// sqMatrix(size, value) = a square matrix of size\*size, all cells filled

with value

**def** sqMatrix(size: Int, value: Int) : *Matrix* =

List.fill(size)(List.fill(size)(value))

// randomSqMatrix(size, limit) = a square matrix of size\*size, cells

filled with random numbers up to limit

**def** randomSqMatrix(size: Int, limit: Int) : *Matrix* =

List.tabulate(size)(n => List.fill(size)(util.Random.nextInt(limit)))

// getMinor(m)(i, j) = the minor of the matrix m for (i,j)

**def** getMinor(m: *Matrix*)(i: Int, j: Int) : *Matrix* =

{

// without(xs, n) = xs without nth element (n starting from 1)

**def** without[A](xs: *List*[A], n: Int) : *List*[A] =

xs.zipWithIndex.filterNot { **case** (\_, i) => i + 1 == n}.unzip.\_1

without(m, i).map(r => without(r, j))

}

// getFirstRowPairs(m) = the list of (row, col) for the first row of the

matrix

**def** getFirstRowPairs(m: *Matrix*) : *List*[(Int, Int)] =

(1 to (m.length)).toList.map(y => (1, y))

// getCell(m)(i, j) = the cell in the matrix m in (i,j)

**def** getCell(m: *Matrix*)(i: Int, j: Int) : Int =

m(i - 1)(j - 1)

// evensNegative(xs) = the same list with the elements with even indices

negated indices start from 1

**def** evensNegative(xs: *List*[Int]) : *List*[Int] =

{

xs.zipWithIndex.map { **case** (x, n) => (x, n % 2 != 0) }.map { **case** (x,

isEven) => **if** (isEven) -x **else** x }

}

// getDeterminant(m) = calculates determinant for matrix m

**def** getDeterminant(m: *Matrix*) : Int =

{

**if** (m.length == 1) getCell(m)(1, 1)

**else**



11

{

**val** list = getFirstRowPairs(m).map(p =>(getCell(m) \_).tupled(p) \*

getDeterminant((getMinor(m) \_).tupled(p)))

evensNegative(list).sum

}

}

// toString(m) = string representation of m

**def** toString(m: *Matrix*) : *String* =

{

**def** listToString[A](l : *List*[A]) : *String* =

"[" + l.mkString(",") + "]"

listToString(m.map(listToString))

}

}

Output

Specify a size for your n \* n matrix:

2

Random Matrix:

[[8,1],[5,6]]

Determinant of the matrix above:

43

---------------------

Specify a size for your n \* n matrix:

3

Random Matrix:

[[7,3,7],[7,7,2],[7,7,2]]

Determinant of the matrix above:

0

---------------------

Specify a size for your n \* n matrix:

3

Random Matrix:

[[6,0,4],[3,7,5],[6,9,8]]

Determinant of the matrix above:

6

---------------------



12

**4. Write a program to check if the matrix is upper triangular**

**or not.**

**object** UpperTriMatrix

{

**def** main(args:Array[*String*])

{

**var** M=Array(Array(1, 3, 5, 3),

Array(0, 4, 6, 2),

Array(0, 0, 2, 5),

Array(0, 0, 0, 6)

)

**var** obj=**new** Tri()

**if**(obj.isUpperTriangularMatrix(M))

println("Yes")

**else**

println("No")

}

**class** Tri

{

**final var** N:Int=4

**def** isUpperTriangularMatrix(M:Array[Array[Int]]):Boolean=

{

**for**(i<- 1 to N-1)

{

**for**(j<- 0 to i-1)

{

**if**(M(i)(j)!=0)

**return false**

}

}

**return true**

}

}

}

Output

Yes



13

**5. Write a program to sort the matrix using insertion sort.**

**object** SortMatrixInsertion

{

**def** main(args:Array[*String*])

{

**var** a:Array[Int]=Array(82,54,32,227,99,2)

println("Array Befor Sorting ")

**for**(i<- 0 to a.length-1)

{

print(" "+a(i))

}

print("\n")

insert\_sort(a)

printarray(a)

}

**def** insert\_sort(a:Array[Int])

{

**for**(i<- 1 to a.length-1)

{

**var** key:Int=a(i)

**var** j=i-1

**while**(j>=0&&a(j)>key)

{

a(j+1)=a(j)

j=j-1

}

a(j+1)=key

}

}

**def** printarray(a:Array[Int])

{

println("Array After Sorting ")

**for**(i<- 0 to a.length-1)

{

print(" "+a(i))

}

print("\n")

}

}

Output

Array Befor Sorting

82 54 32 227 99 2

Array After Sorting

2 32 54 82 99 227



14

**6. Write a program for multiplication of two matrices(Validate**

**number of rows and columns before multiplication and give**

**appropriate message)**

**object** MatrixMult

{

**final var** N:Int=2

**def** main(args:Array[*String*])

{

**var** A=Array(Array(2, 4),Array(2, 5))

**var** B=Array(Array(4, 2),Array(5, 2))

**var** C:Array[Array[Int]]=Array.ofDim[Int](N,N)

println("Multiplication of Matrix A & B is : ")

**for**(i<- 0 to N-1)

{

**for**(j<- 0 to N-1)

{

C(i)(j)=0

**for**(k<- 0 to N-1)

{

C(i)(j)+=A(i)(k)\*B(k)(j)

}

print(" "+C(i)(j))

}

println()

}

}

}

Output

Multiplication of Matrix A & B is :

28 12

33 14



15

**With Validate row and columns**

**object** MatrixMult

{

**var** N:Int=100

**def** main(args:Array[*String*])

{

**var** A=Array.ofDim[Int](N,N)

**var** B=Array.ofDim[Int](N,N)

**var** C:Array[Array[Int]]=Array.ofDim[Int](N,N)

println("How many rows and columns in the matrix A : ")

**var** r1:Int=scala.io.StdIn.readInt()

**var** c1:Int=scala.io.StdIn.readInt()

println("Enter the element of matrix A: ")

**for**(i<- 1 to r1)

{

**for**(j<- 1 to c1)

{

A(i)(j)=scala.io.StdIn.readInt()

}

}

println("How many rows and columns in the matrix B : ")

**var** r2:Int=scala.io.StdIn.readInt()

**var** c2:Int=scala.io.StdIn.readInt()

println("Enter the element of matrix B: ")

**for**(i<- 1 to r2)

{

**for**(j<- 1 to c2)

{

B(i)(j)=scala.io.StdIn.readInt()

}

}

print(" Matrix A : \n")

PrintMatrix(A,r1,c1)

print(" Matrix B : \n")

PrintMatrix(B,r2,c2)

**if**(r2!=r2 || c1!=c2)

{

print("Multiplication not possible because No. of rows & columns not

Equal")

}

**else**

{

println("Multiplication of Matrix A & B is : ")

**for**(i<- 0 to r1)

{

**for**(j<- 0 to c2)

{

**for**(k<- 0 to r2)

{

C(i)(j)+=A(i)(k)\*B(k)(j)

}

}

}

PrintMatrix(C,r1,c2)

}//else

}//main



16

**def** PrintMatrix(m:Array[Array[Int]],r:Int,c:Int)

{

**for**(i<- 1 to r)

{

**for**(j<- 1 to c)

{

print(" "+m(i)(j))

}

print("\n")

}

}

}//obj

Output

How many rows and columns in the matrix A :

2

2

Enter the element of matrix A:

2

4

2

5

How many rows and columns in the matrix B :

2

2

Enter the element of matrix B:

4

2

5

2

Matrix A :

2 4

2 5

Matrix B :

4 2

5 2

Multiplication of Matrix A & B is :

28 12

33 14



17

**3 String**

**1. Write a program to count uppercase letters in a string and**

**convert it to lowercase and display the new string.**

**object** UpToLowString

{

**def** main(args:Array[*String*])

{

**var** Str:*String*="Shubham Shyamrao Kulkarni"

**var** cnt:Int=0

**for**(i<- 0 to Str.length()-1)

{

**var** ch:Char=Str.charAt(i)

**if**(ch.isUpper)

{

cnt+=1

}

}

**var** lowercase:*String* = Str.toLowerCase()

println("\n '"+cnt+"' Uppercase letters in the String '"+Str+"' \n")

println(" String after converting to lowercase '"+lowercase+"' ")

}

}

Output

'3' Uppercase letters in the String 'Shubham Shyamrao Kulkarni'

String after converting to lowercase 'shubham shyamrao kulkarni'

**2. Write a program to read a character from user and count the**

**number of occurrences of that character.**

**object** StringOccure

{

**def** main(args:Array[*String*])

{

**var** Str:*String*="Shubham Shyamrao Kulkarni"

println("Enter a character ")

**var** Chr:Char=scala.io.StdIn.readChar()

**var** Count:Int=Str.count(\_==Chr)

println(" '"+Chr+"' occured in '"+Str+"' "+Count+" times")

}

}

Output

Enter a character

a

'a' occured in 'Shubham Shyamrao Kulkarni' 4 times



18

**3. Write a program to read two strings. Remove the occurrence**

**of second string in first string.**

**object** RemoveString

{

**def** main(args:Array[*String*])

{

println("Enter a First String ")

**var** Str1:*String*=scala.io.StdIn.readLine()

println("Enter a Second String ")

**var** Str2:*String*=scala.io.StdIn.readLine()

**var** Str3 = Str1.replaceAll(Str2, "")

print(" First String will beome after removing '"+Str2+"' = "+Str3)

}

}

Output

Enter a First String

shubham shubham

Enter a Second String

bh

First String will beome after removing 'bh' = shuam shuam

**4. Create array of strings and read a string from user.**

**Display all the elements of array containing given string.**

**object** StringArray

{

**def** main(args:Array[*String*])

{

**var** S=Array("shubham","rocky","hockey","london","monkey","india")

println(" Enter a String ")

**var** Str:*String*=scala.io.StdIn.readLine()

println(" Only those array elements contains '"+Str+"' in it")

**for**(i<- 0 to S.length-1)

{

**var** temp:*String*=S(i)

**if**(temp.contains(Str))

{

print(" "+temp)

}

}

}

}

Output

Enter a String

ke

Only those array elements contains 'ke' in it

hockey monkey



19

**4 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.**

**class** CA

{

**var** accno:Int=121

**var** name:*String*="Ambani"

**var** bal:Float=10000

**var** min\_bal:Float=500

**def** withdraw(wd:Float)

{

**if**(bal>wd)

{

bal-=wd

}

**else**

{

println(" Insufficient balance ")

}

}

**def** deposit(dp:Float)

{

bal+=dp

}

**def** view\_bal()

{

println(" Balance : "+bal)

}

}

**object** CurrentAccount

{

**def** main(args:Array[*String*])

{

**var** ca=**new** CA()

println("--------------------------------------------------------------

")

")

println("\n\tAccno\tName\tBalance\tMin\_Bal")

println("\t"+ca.accno+"\t"+ca.name+"\t"+ca.bal+"\t"+ca.min\_bal)

println("--------------------------------------------------------------

**var** ch:Int=0

**while**(ch!=4)

{

println(" Enter your choice : ")

println("1 : Withdraw\n2 : Deposit\n3 : View Balance\n4 : Exit")

ch=scala.io.StdIn.readInt()

ch **match**

{

**case** 1=>

println(" Enter amount to withdraw ")

**var** wd:Float=scala.io.StdIn.readFloat()

ca.withdraw(wd)

**case** 2=>

println(" Enter amount to deposit ")

20

**var** dp:Float=scala.io.StdIn.readFloat()

ca.deposit(dp)

**case** 3=>

ca.view\_bal()

**case** 4=>

println(" Exit successful")

}

}

}

}

Output

--------------------------------------------------------------

Accno Name Balance

121 Ambani 10000.0

Min\_Bal

500.0

--------------------------------------------------------------

Enter your choice :

1 : Withdraw

2 : Deposit

3 : View Balance

4 : Exit

3

Balance : 10000.0

Enter your choice :

1 : Withdraw

2 : Deposit

3 : View Balance

4 : Exit

1

Enter amount to withdraw

6050

Enter your choice :

1 : Withdraw

2 : Deposit

3 : View Balance

4 : Exit

3

Balance : 3950.0

Enter your choice :

1 : Withdraw

2 : Deposit

3 : View Balance

4 : Exit

2

Enter amount to deposit

550

Enter your choice :

1 : Withdraw

2 : Deposit

3 : View Balance

4 : Exit

3

Balance : 4500.0

Enter your choice :

1 : Withdraw

2 : Deposit

3 : View Balance

4 : Exit

4

Exit successful

21

**2. Define a class Employee (id, name, salary). Define methods**

**accept() and display(). Display details of employee having**

**maximum salary.**

**class** Employee

{

**var** id:Int=0

**var** name:*String*=**null**

**var** salary:Float=0

**def** accept()

{

println("\nEnter id : ")

id=scala.io.StdIn.readInt()

println("\nEnter name : ")

name=scala.io.StdIn.readLine()

println("\nEnter salary : ")

salary=scala.io.StdIn.readFloat()

}

**def** display()

{

println("\t"+id+"\t"+name+"\t"+salary)

println("----------------------------------------------------")

}

}

**object** EmployeeClass

{

**def** main(args:Array[*String*])

{

println("How many records you want to insert : ")

**var** n:Int=scala.io.StdIn.readInt()

**var** emp=**new** Array[Employee](n)

**for**(i<- 0 to n-1)

{

emp(i)=**new** Employee()

emp(i).accept()

//emp(i).display()

}

**for**(i<- 0 to n-1)

{

println("----------------------------------------------------")

println("\n\tId\tName\tSalary")

emp(i).display()

}

**var** m:Int=0

**var** max:Float=emp(0).salary

**for**(i<- 1 to n-1)

{

**if**(emp(i).salary>max)

{

max=emp(i).salary

m=i

}

}

println("This Employee have maximum salary")

println("\tId\tName\tSalary")

emp(m).display()

}

}



22

Output

How many records you want to insert :

3

Enter id :

1

Enter name :

Ramesh

Enter salary :

900

Enter id :

2

Enter name :

Mukesh

Enter salary :

90

Enter id :

3

Enter name :

Rajesh

Enter salary :

9000

----------------------------------------------------

Id

1

Name Salary

Ramesh

900.0

----------------------------------------------------

----------------------------------------------------

Id

2

Name Salary

Mukesh

90.0

----------------------------------------------------

----------------------------------------------------

Id

3

Name Salary

Rajesh

9000.0

----------------------------------------------------

This Employee have maximum salary

Id

Name Salary

3

Rajesh

9000.0

----------------------------------------------------

23

**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.**

**object** Orders

{

**def** main(args:Array[*String*])

{

**val** p=**new** Purchase()

p.accept()

p.display()

**val** s=**new** Salesorder()

s.accept()

s.display()

}

**abstract class** Order

{

**var** id=0

**var** desc=" "

**def** accept()

**def** display()

}

**class** Purchase **extends** Order

{

**var** amt=0

**def** accept()

{

println("Enter purchase id:")

id=scala.io.StdIn.readInt()

println(" Enter the description:")

desc=scala.io.StdIn.readLine()

println("Enter the amount:")

amt=scala.io.StdIn.readInt()

}

**def** display()

{

println("\tId:"+id+"\tDescription:"+desc+"\tAmount:"+amt)

}

}

**class** Salesorder **extends** Order

{

**var** vendor=" "

**var** customer=" "

**def** accept()

{

println("Enter vendor name:")

vendor=scala.io.StdIn.readLine()

println("Enter customer name:")

customer=scala.io.StdIn.readLine()

}

**def** display()

{

println("\tVendor:"+vendor+"\tCustomer:"+customer)

}

}

}

24

Output

Enter purchase id:

12

Enter the description:

Mobile

Enter the amount:

15000

Id:12 Description:Mobile

Amount:15000

Enter vendor name:

Om

Enter customer name:

SK

Vendor:Om Customer:SK

25

**4. Create abstract class Shape with abstract functions**

**volume() and display(). Extend two classes Cube and Cylinder**

**from it. Calculate volume of each and display it.**

**abstract class** Shape(r:Double,h:Double,s:Double)

{

**def** volume()

**def** display()

}

**class** Cube(r:Double,h:Double,s:Double) **extends** Shape(r,h,s)

{

**var** vol:Double=0

**def** volume()

{

vol=s\*s\*s

}

**def** display()

{

volume()

println(" Volume of Cube : "+vol)

}

}

**class** Cylinder(r:Double,h:Double,s:Double) **extends** Shape(r,h,s)

{

**var** vol:Double=0

**def** volume()

{

vol=3.14\*r\*r\*h

}

**def** display()

{

volume()

println(" Volume of Cube : "+vol)

}

}

**object** AbstractClass

{

**def** main(args:Array[*String*])

{

**var** r:Double=0

**var** h:Double=0

**var** s:Double=0

println(" Enter the length of side for Cube ")

s=scala.io.StdIn.readDouble()

println(" Enter the radius and height of Cylinder ")

r=scala.io.StdIn.readFloat()

h=scala.io.StdIn.readFloat()

**var** c1=**new** Cube(r,h,s)

c1.volume()

c1.display()

**var** c2=**new** Cylinder(r,h,s)

c2.volume()

c2.display()

}

}

26

Output

Enter the length of side for Cube

4.4

Enter the radius and height of Cylinder

3.5

5.8

Volume of Cube : 85.18400000000003

Volume of Cube : 223.09700733661654

27

**6. Define a class Sports (id, name, description, amount).**

**Derive two classes Indoor and Outdoor. Define appropriate**

**constructors and operations. Create an object and perform**

**operations.**

**object** Sports

{

**def** main(args:Array[*String*])

{

println("Enter Details of Indoor Game :")

**var** i=**new** Indoor()

println("Enter Details of Outdoor Game :")

**var** o=**new** Outdoor()

}

}

**class** Sport

{

println("Enter sport id:")

**var** id=scala.io.StdIn.readInt()

println("Enter sport name:")

**var** name=scala.io.StdIn.readLine()

println("Enter sport description:")

**var** desc=scala.io.StdIn.readLine()

println("Enter sport amount:")

**var** amt=scala.io.StdIn.readFloat()

}

**class** Indoor **extends** Sport

{

println("\tSport id:"+id+"\tSport

name:"+name+"\tDescription:"+desc+"\tAmount"+amt)

}

**class** Outdoor **extends** Sport

{

println("\tSport id:"+id+"\tSport

name:"+name+"\tDescription:"+desc+"\tAmount"+amt)

}

Output

Enter Details of Indoor Game :

Enter sport id:

12

Enter sport name:

Chess

Enter sport description:

MindGame

Enter sport amount:

2000

Sport id:12 Sport name:Chess Description:MindGame

Amount2000.0

Enter Details of Outdoor Game :

Enter sport id:

15

Enter sport name:

Cricket

Enter sport description:

Coordination

Enter sport amount:

200000

Sport id:15 Sport name:Cricket

Description:Coordination

Amount200000.0

28

**7. Design abstract class Employee with computeSal() as**

**abstract function. Create two subclasses Worker and Manager.**

**Salary of worker should be calculated on hourly basis of work**

**and Salary of Manager should be calculated on monthly basis**

**with additional incentives.**

**abstract class** Employe

{

**def** computsal()

}

**class** Worker **extends** Employe

{

**def** computsal()

{

println("Total hours in a day:")

**var** h=scala.io.StdIn.readInt()

**var** sal=h\*50\*24

println("Monthly salary : "+sal)

}

}

**class** Manager **extends** Employe

{

**def** computsal()

{

println("Total days in the month:")

**var** d=scala.io.StdIn.readInt()

**var** sal=d\*1500

println("Monthly salary : "+sal)

}

}

**object** Salary

{

**def** main(args:Array[*String*])

{

println("worker information:")

**var** w=**new** Worker()

w.computsal()

println("manager information:")

**var** m=**new** Manager()

m.computsal()

}

}

Output

worker information:

Total hours in a day:

12

Monthly salary : 14400

manager information:

Total days in the month:

24

Monthly salary : 36000

29

**5 List**

**1. Create Lists using five different methods( Lisp style ,**

**Java style, fill, range and tabulate methods)**

**object** CreateLists

{

**def** main(args:Array[*String*])

{

**val** list1=10::20::30::**Nil**

println("\n Creating List using Lisp-style : "+list1)

**val** list2=List(11,22,33)

println("\n Creating List using Java-style : "+list2)

**val** list3=List.fill(3)("scala")

println("\n Creating List using fill : "+list3)

**val** list4=List.range(0,10,2)

println("\n Creating List using range : "+list4)

**val** list5=List.tabulate(5)(n=>n\*n)

println("\n Creating List using tabulate : "+list5)

}

}

Output

Creating List using Lisp-style : List(10, 20, 30)

Creating List using Java-style : List(11, 22, 33)

Creating List using fill : List(scala, scala, scala)

Creating List using range : List(0, 2, 4, 6, 8)

Creating List using tabulate : List(0, 1, 4, 9, 16)



30

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

**ascending order.**

**import** scala.collection.mutable.ListBuffer

**object** CombineList

{

**def** main(args:Array[*String*])

{

**var** list1=**new** ListBuffer[Int]

**var** list2=**new** ListBuffer[Int]

**var** num=0

println(" How many elements in List1 : ")

**var** n:Int=scala.io.StdIn.readInt()

println("Enter elements of List1 : ")

**for**(i<- 0 to n-1)

{

num=scala.io.StdIn.readInt()

list1+=num

}

println(" How many elements in List2 : ")

n=scala.io.StdIn.readInt()

println("Enter elements of List2 : ")

**for**(i<- 0 to n-1)

{

num=scala.io.StdIn.readInt()

list2+=num

}

println(" List 1 : "+list1.toList)

println(" List 2 : "+list2.toList)

**val** merge=List.concat(list1,list2)

println(" Merged List 3 : "+merge)

println("Sorted List is : "+merge.sorted)

}

}

Output

How many elements in List1 :

4

Enter elements of List1 :

9

3

1

7

How many elements in List2 :

5

Enter elements of List2 :

6

2

8

4

5

List 1 : List(9, 3, 1, 7)

List 2 : List(6, 2, 8, 4, 5)

Merged List 3 : List(9, 3, 1, 7, 6, 2, 8, 4, 5)

Sorted List is : List(1, 2, 3, 4, 5, 6, 7, 8, 9)



31

**3. Create a list of integers divisible by 3 from List**

**containing numbers from 1 to 50.**

**import** scala.collection.mutable.ListBuffer

**object** ListDiv

{

**def** main(args:Array[*String*])

{

**val** list=List.range(1,50)

**var** div=**new** ListBuffer[Int]

**for**(i<- list)

{

**if**(i%3==0)

{

div+=i

}

}

println("\n List of integers divisible by 3 from another list :

\n"+div)

}

}

Output

List of integers divisible by 3 from another list :

ListBuffer(3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48)

**4. Create a list of even numbers up to 10 and calculate its**

**product**.

**import** scala.collection.mutable.ListBuffer

**object** EvenListProd

{

**def** main(args:Array[*String*])

{

**val** list=1 to 10 toList

**var** even=**new** ListBuffer[Int]

**for**(i<-list)

{

**if**(i%2==0)

{

even+=i

}

}

println("\n List of even numbers upto 10 : "+even)

println(" Product of Elements of list even : "+even.product)

}

}

Output

List of even numbers upto 10 : ListBuffer(2, 4, 6, 8, 10)

Product of Elements of list even : 3840



32

**5. Write a program to create list with 10 members using**

**function 3n2+4n+6**

**import** scala.collection.mutable.ListBuffer

**object** ListFunc

{

**def** main(args:Array[*String*])

{

**var** func=**new** ListBuffer[Int]

**for**(n<- 1 to 10)

{

func+=3\*n\*n+4\*n+6

}

println(" List of 10 members usnig function 3n^2+4n+6 \n"+func)

}

}

Output

List of 10 members usnig function 3n^2+4n+6

ListBuffer(13, 26, 45, 70, 101, 138, 181, 230, 285, 346)

**6. Write a program to create a list of 1 to 100 numbers.**

**Create second list from first list selecting numbers multiple**

**of 10.**

**import** scala.collection.mutable.ListBuffer

**object** MultList

{

**def** main(args:Array[*String*])

{

**val** list=1 to 100 toList

**var** mult10=**new** ListBuffer[Int]

**for**(i<-list)

{

**if**(i%10==0)

{

mult10+=i

}

}

//println("\n List of numbers between 1 to 100 : "+list)

println("\n Second list contain only those elements of first list is

multiple of 10 \n"+mult10)

}

}

Output

Second list contain only those elements of first list is multiple of 10

ListBuffer(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)



33

**7. Create a list of 50 members using function 2n+3. Create**

**second list excluding all elements multiple of 7.**

**import** scala.collection.mutable.ListBuffer

**object** List2nplus3

{

**def** main(args:Array[*String*])

{

**var** func=**new** ListBuffer[Int]

**var** mult7=**new** ListBuffer[Int]

**for**(n<- 1 to 50)

{

func+=2\*n+3

}

**for**(i<-func)

{

**if**(i%7!=0)

mult7+=i

}

println(" List of 50 members usnig function 2n+3 \n"+func)

println("\n Second list excluding all elements multiple of 7 \n"+mult7)

}

}

Output

List of 50 members usnig function 2n+3

ListBuffer(5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37,

39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75,

77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103)

Second list excluding all elements multiple of 7

ListBuffer(5, 9, 11, 13, 15, 17, 19, 23, 25, 27, 29, 31, 33, 37, 39, 41,

43, 45, 47, 51, 53, 55, 57, 59, 61, 65, 67, 69, 71, 73, 75, 79, 81, 83, 85,

87, 89, 93, 95, 97, 99, 101, 103)



34

35

**6 Map**

**1. Write a user defined functions to convert lowercase letter**

**to uppercase and call the function using Map.**

**object** MapUpper

{

**def** main(args:Array[*String*])

{

**val** stud=List("a","b","c","d","e","f")

**var** new\_stud=stud.map(toupper)

println(new\_stud)

}

**def** toupper(s:*String*):*String*=

{

s.toUpperCase()

}

}

Output

List(A, B, C, D, E, F)

**2. Write a program to create map with Rollno and FirstName.**

**Print all student information with same FirstName.**

**object** MapStudent

{

**def** main(args:Array[*String*])

{

**val** map:*Map*[Int,*String*]=Map(101->"Rajesh",102->"Varun",103-

>"Swara",104->"Rajesh")

**val** values=map.values.toArray

**val** key=map.keys.toArray

**val** n=key.length

**val** v=scala.collection.mutable.Set[Int]()

**for**(i<-0 to n-1)

{

**for**(j<-i+1 to n-1)

{

**if**(values(i)==values(j))

{

v+=i

v+=j

}

}

}

**val** arr=v.toArray

**for**(a<-arr)

println(key(a)+" "+values(a))

}

}

Output

(Roll No : 101,First Name : Rajesh)

(Roll No : 104,First Name : Rajesh)



36

37

**7 Set**

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

**between them.**

**object** SetIntersect

{

**def** main(args:Array[*String*])

{

**val** num1=Set(5,6,9,20,30,45)

**val** num2=Set(50,60,9,20,35,55)

println("num1.&(num2) : "+num1.&(num2))

println("num1.intersect(num2) : "+num1.intersect(num2))

}

}

Output

num1.&(num2) : Set(20, 9)

num1.intersect(num2) : Set(20, 9)

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

**the Set**

**object** MaxMinSet

{

**def** main(args:Array[*String*])

{

**var** mset=scala.collection.mutable.Set[Int]()

**var** num=0

println("How many elements are in Set : ")

**var** n:Int=scala.io.StdIn.readInt()

println("Enter elements of Set : ")

**for**(i<- 1 to n)

{

num=scala.io.StdIn.readInt()

mset+=num

}

println("The Set is as fallows : "+mset)

println("Largest element from Set is "+mset.max)

println("Smallest element from Set is "+mset.min)

}

}

Output

How many elements are in Set :

5

Enter elements of Set :

43

12

786

65

65

The Set is as fallows : Set(12, 786, 43, 65)

Largest element from Set is 786

Smallest element from Set is 12



38

**3. Write a program to merge two sets and calculate product and**

**average of all elements of the Set**

**object** MergeSetProdAvg

{

**def** main(args:Array[*String*])

{

**var** mset1=scala.collection.mutable.Set[Int]()

**var** mset2=scala.collection.mutable.Set[Int]()

**var** num:Int=0

println("How many elements are in Set1 : ")

**var** n:Int=scala.io.StdIn.readInt()

println("Enter elements of Set : ")

**for**(i<- 1 to n)

{

num=scala.io.StdIn.readInt()

mset1+=num

}

println("How many elements are in Set2 : ")

n=scala.io.StdIn.readInt()

println("Enter elements of Set : ")

**for**(i<- 1 to n)

{

num=scala.io.StdIn.readInt()

mset2+=num

}

println("The Set1 is as fallows : "+mset1)

println("The Set2 is as fallows : "+mset2)

**val** mset3=mset1++mset2

println("The Merged Set is as fallows : "+mset3)

**var** prod:Int=1

mset3.foreach(prod\*=\_)

println("Product of set elements= "+prod)

**var** avg:Float=0

**var** sum:Int=0

mset3.foreach(sum+=\_)

avg=sum/mset3.size

println("Average of set elements = "+avg)

}

}

Output

How many elements are in Set1 :

3

Enter elements of Set :

1

2

3

How many elements are in Set2 :

3

Enter elements of Set :

4

5

6

The Set1 is as fallows : Set(1, 2, 3)

The Set2 is as fallows : Set(5, 6, 4)

The Merged Set is as fallows : Set(1, 5, 2, 6, 3, 4)

Product of set elements= 720

Average of set elements = 3.0

