

LAB CYCLE-1

AMAL T S
20MCA011

EXPERIMENT 1 : SIMPLE JAVA PROGRAM

AIM

Java Program to Accept the Marks of a Student into a 1D Array and find total marks and percentage.

ALGORITHM

1. Define class StudentMarks with members n(int), rollno(int), total(int), name (String) and percentage(float).
2. Get the data from the user including name of student, roll no., number of subjects and marks secured in each out of 100.
3. Store each marks in an 1D array i.e marks[i]
4. Do the summation and return the value .

PROGRAM

```
import java.util.*;
public class SumPercentage
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        int n;
        System.out.println("Enter the total subjects:");
        n=sc.nextInt();
        int arr[] = new int[n];
        System.out.println("Enter the marks:");
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }

        int tot=0;
        for(int i=0;i<n;i++)
        {
            tot=tot+arr[i];
        }
        System.out.println("Total marks:"+tot);
        float per;
```

```
        per= (tot / (float)n);  
        System.out.println( "Total Percentage:"+ per + "%");  
    }  
}
```

OUTPUT

```
C:\Users\HP\Desktop>java SumPercentage  
Enter the total subjects:  
4  
Enter the marks:  
82  
56  
98  
75  
Total marks:311  
Total Percentage:77.75%
```

EXPERIMENT 2 : COUNT NUMBER OF OCCURRENCES

AIM

Java Program to Count the Number of Occurrence of an Element in an Array.

ALGORITHM

1. Define class Count with members n(int), x(int), count(int), i (int) .
2. Get the data from user including number of elements and those element.
3. Store each element in an 1D array i.e a[i].
4. For each element in a[i] if a[i]==x then increment count i.e count++.
5. Print the count.

PROGRAM

```
import java.util.Scanner;  
  
class Count  
{  
    public static void main(String[] args)  
    {  
        int n,x,count=0,i=0;
```

```

Scanner s = new Scanner(System.in);
System.out.print("enter number of elements : "); n =
s.nextInt(); int a[]=new int[n];
System.out.println("enter the elements : ");
for(i=0;i < n;i++)
{
a[i] = s.nextInt(); }
System.out.print("enter the element to count the number of occurrences ? "); x =
s.nextInt();
for(i = 0; i < n; i++)
{
if(a[i] == x)
{
count++;
}
}
System.out.println("number of occurrences : "+count);
}
}

```

OUTPUT

```

C:\Users\HP\Desktop>java Count
Enter no. of elements you want in array:5
Enter all the elements:
2
2
5
2
8
Enter the element of which you want to count number of occurrences:2
Number of Occurrence of the Element:3

```

EXPERIMENT 3 : MATRIX ADDITION

AIM

Java Program to Add Two MXN Matrix from User Input.

ALGORITHM

1. If both matrices are of the same size then only we can add the matrices.
2. Use the double dimensional array to store the matrix elements.

3. Read row number, column number and initialize the double dimensional arrays a[],b[],c[] with same row number, column number.
4. Store the first matrix elements into the two-dimensional array a[] using two for loops. i indicates row number, j indicates column index. Similarly matrix 2 elements in to b[].
5. Add the two matrices using for loop for i=0 to i;row for j=0 to j;col a[i][j] + b[i][j] and store it in to the matrix res at c[i][j] .

PROGRAM

```
import java.util.Scanner;
```

```
class MatrixAdd
{
public static void main(String args[])
{
int p, q, m, n;
Scanner s = new Scanner(System.in);
System.out.println("enter number of rows and columns in first matrix :"); p =
s.nextInt(); q = s.nextInt();
System.out.println("enter number of rows and columns in second matrix :"); m =
s.nextInt(); n = s.nextInt();
if (p == m && q == n)
{
int a[][] = new int[p][q]; int
b[][] = new int[m][n]; int
c[][] = new int[m][n];
System.out.println("enter the elements of first matrix :"); for (int
i = 0; i < p; i++)
{
for (int j = 0; j < q; j++)
{
a[i][j] = s.nextInt();
}
}
System.out.println("enter elements of second matrix :");
for (int i = 0; i < m; i++)
{
for (int j = 0; j < n; j++)
{
b[i][j] = s.nextInt();
}
}
}
```

```
System.out.println("first matrix :"); for
(int i = 0; i < p; i++)
{
for (int j = 0; j < q; j++)
{
System.out.print(a[i][j]+" ");
}
System.out.println("");
}
System.out.println("second matrix :");
for (int i = 0; i < m; i++)
{
for (int j = 0; j < n; j++)
{
System.out.print(b[i][j]+" ");
}
System.out.println("");
}
for (int i = 0; i < p; i++)
{
for (int j = 0; j < n; j++)
{
for (int k = 0; k < q; k++)
{
c[i][j] = a[i][j] + b[i][j];
}
}
}
System.out.println("matrix after addition :");
for (int i = 0; i < p; i++)
{
for (int j = 0; j < n; j++)
{
System.out.print(c[i][j]+" ");
}
System.out.println("");
}
}
else
{
System.out.println("Addition is not be possible !");
}
}
```

}

OUTPUT

```
C:\Users\HP\Desktop>java AddTwoMatrix
Enter the number of rows in the first matrix:3
Enter the number of columns in the first matrix:
3
Enter the number of rows in the second matrix:3
Enter the number of columns in the second matrix:3
Enter all the elements of first matrix:
2
5
4
5
4
2
8
7
5
Enter all the elements of second matrix:
7
6
1
2
8
4
6
3
8
First Matrix:
2 5 4
5 4 2
8 7 5
Second Matrix:
7 6 1
2 8 4
6 3 8
Matrix after addition:
9 11 5
7 12 6
14 10 13
```

EXPERIMENT 4 : COMPLEX NUMBER ADDITION

AIM

Program to add complex numbers.

ALGORITHM

1. Define class Complex with members real, img of int datatype for the real and imaginary part of the complex number.
2. Define a constructor to initialise real and img.
3. Create a temporary complex number to hold the sum of two numbers
4. Do the summation and return the value.

PROGRAM

```
import java.util.Scanner;
class Complex
{
float real1,image1,real2,image2;
void get()
{
Scanner op=new Scanner(System.in);
System.out.print("\n enter the real part of the first complex number");
real1=op.nextFloat();
System.out.print("\n enter the imaginary part of the first complex number");
image1=op.nextFloat();
System.out.print("\n enter the real part of the second complex number");
real2=op.nextFloat();
System.out.print("\n enter the imaginary part of the second complex number");
image2=op.nextFloat();
}
void display()
{
System.out.println("\n Sum of complex numbers"+real1+"+"+image1+"i
and"+real2+"+"+image2+"i is "+(real1+real2)+"+"+(image1+image2)+"i");
}
public static void main(String args[])
{
Complex cmp=new Complex();
cmp.get();
```



```
cmp.display();  
}  
}
```

OUTPUT

```
C:\Users\HP\Desktop>javac complex.java  
C:\Users\HP\Desktop>java Complex  
  
enter the real part of the first complex number3  
enter the imaginary part of the first complex number5  
enter the real part of the second complex number6  
enter the imaginary part of the second complex number9  
Sum of complex numbers3.0+5.0i and6.0+9.0i is9.0+14.0i  
C:\Users\HP\Desktop>
```

EXPERIMENT 5 : MATRIX IS SYMMETRIC OR NOT

AIM

Read a matrix from the console and check whether it is symmetric or not.

ALGORITHM

1. Read the no of rows and columns into integer variables rows and cols respectively.
2. Check If(rows != cols), then print given matrix is not symmetric and End. Otherwise, continue with step
3. Declare and initialise an integer array matrix[rows][cols].
4. Read the elements of the matrix[r][c].
5. Display the input matrix[r][c].
6. Check if the matrix equal to transpose matrix i.e., if(matrix[r][c] == matrix[c][r]) Print symmetric and End.
7. Print matrix is symmetric accordingly.

PROGRAM

```
import java.util.*;  
  
class SymmetricMatrix
```

```

{
public static void main(String args[])
{
Scanner s=new Scanner(System.in); int r,c;
int count=0;
System.out.print("enter number of rows in the matrix : "); r=s.nextInt();
System.out.print("enter number of columns in the matrix : ");
c=s.nextInt(); int x[][]=new int[r][c]; int y[][]=new int[r][c];
System.out.println("enter the elements of the matrix : "); for(int
i=0;i<r;i++)
{
for(int j=0;j<c;j++)
{
x[i][j]=s.nextInt();
}
}
System.out.println("the given matrix is:");
for(int i=0;i<r;i++)
{
for(int j=0;j<c;j++)
{
System.out.print(x[i][j]+" ");
}
System.out.println("");
}
if(r==c)
{
for(int i=0;i<r;i++)
{
for(int j=0;j<c;j++)
{
y[i][j]=x[j][i];
}
}
}
System.out.println("transpose of given matrix is : ");
for(int i=0;i<r;i++)
{
for(int j=0;j<c;j++)
{
System.out.print(y[i][j]+" ");
}
System.out.println("");
}

```

```

    }
    for(int i=0;i<r;i++) for(int
    j=0;j<c;j++)
    if(x[i][j]==y[i][j])
    {
    count++;
    }
    if(count==r*c)
    System.out.println("yes,the given matrix is symmetric !"); else
    System.out.println("No,the given matrix is assymmetric !");
    }
    }

```

OUTPUT

```

C:\Users\HP\Desktop>javac SymmetricMatrix.java

C:\Users\HP\Desktop>java SymmetricMatrix
enter number of rows in the matrix : 3
enter number of columns in the matrix : 3
enter the elements of the matrix :
1
2
3
2
4
5
3
5
8
the given matrix is:
1 2 3
2 4 5
3 5 8
transpose of given matrix is :
1 2 3
2 4 5
3 5 8
yes,the given matrix is symmetric !

```

EXPERIMENT 6 : DESIGN AND USE PRODUCT CLASS

AIM

**Define a class 'product' with data members pcode, pname and price.
Create 3 objects of the class and find the product having the lowest price.**

ALGORITHM

1. Create a public class Product with members pname, pcode of String datatype and price of integer datatype.
2. Define both default and paramertized constructors.
3. Define getters and setters for members pname, pcode and price.
4. Define a used-defined method display() to display the product details.
5. End class Product
6. Create public class EXP1
7. Define main(String[] args)
8. Create object p1 using default constructor and objects p2 and p3 using overloaded constructors
9. For object p1 created using default constructor, assign values to p1.pname, p1.pcode, p1.price and call p1.display()
10. Get the product p which has the lowest price among p1, p2 and p3. (Use ternary operator for one line of code!)
11. Call p.display();
12. End main
13. End Class EXP1.

PROGRAM

```
import java.util.*;
class Product
{
String pname,pcode;
int price;
public String getPname()
{
return pname;
}
public Product()
{
}
public Product(String pname,String pcode,int price)
{
```

```
this.pname=pname;
this.pcode=pcode;
this.price=price;
}
public void setPname(String pname)
{
this.pname = pname;
}
public String getPcode()
{
return pcode;
}
public void setPcode(String pcode)
{
this.pcode = pcode;
}
public int getPrice()
{
return price;
}
public void display()
{
System.out.println("pcode : "+this.pcode);
System.out.println("pname : "+this.pname);
System.out.println("price : "+this.price);
}
}
public class EXP1
{
public static void main(String args[])
{
Product p1 = new Product();
p1.pcode = "p104";
p1.pname = "pen";
p1.price = 10;
System.out.println("*****Displaying p1*****");
p1.display();
Product p2 = new Product("car2015", "swift", 250000);
System.out.println("*****Displaying p2*****");
p2.display();
Product p3 = new Product("ph458", "samsung", 20000);
System.out.println("*****Displaying p3*****");
p3.display();
}
```

```
Product p = p3.getPrice()<(p1.price<p2.price?p1.price:p2.price)?  
p3:(p1.price<p2.price?p1:p2);  
System.out.println("\n*****Displaying product with lowest price*****");  
p.display();  
}  
}
```

OUTPUT

```
C:\Users\HP\Desktop>javac EXP1.java  
  
C:\Users\HP\Desktop>java EXP1  
*****Displaying p1*****  
pcode : p104  
pname : pen  
price : 10  
*****Displaying p2*****  
pcode : swift  
pname : car2015  
price : 250000  
*****Displaying p3*****  
pcode : samsung  
pname : ph458  
price : 20000  
  
*****Displaying product with lowest price*****  
pcode : p104  
pname : pen  
price : 10
```

EXPERIMENT 7 : CPU AND ITS DETAILS – USING INNER CLASS

AIM

Create CPU with attribute price. Create inner class Processor (no. of cores, manufacturer) and static nested class RAM (memory, manufacturer). Create an object of CPU and print information of Processor and RAM.

ALGORITHM

1. Define class CPU with member price of int datatype.
2. Define an inner class Processor with members cores (int) and manufacturer

(String).

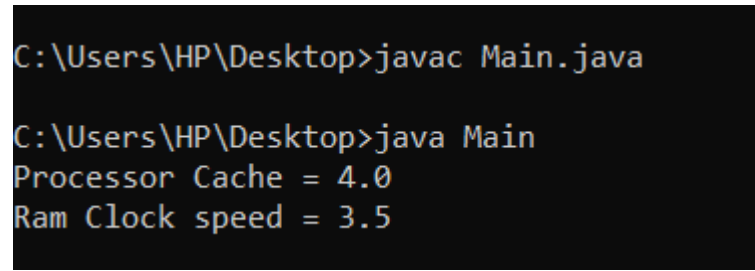
3. Define another nested protected class RAM with members memory and manufacturer of int and String data types respectively.
4. Define a public class PC
5. Create object of Outer class CPU i.e., CPU c = new CPU();
6. Create an object of inner class Processor using outer class CPU.Processor p = c.new Processor();
7. Create an object of inner class RAM using outer class CPU CPU.RAM r = c.new RAM();
8. Print processor cache : p.getCache();
9. Print ram memory type = r.getType ();
10. End class PC.

PROGRAM

```
class CPU
{
    double price;
    class Processor
    {
        double cores;
        String manufacturer;
        double getCache()
        {
            return 4;
        }
    }
    protected class RAM
    {
        double memory;
        String manufacturer;
        double getClockSpeed()
        {
            return 3.5;
        }
    }
}
public class Main
{
    public static void main(String[] args)
    {
        CPU cpu = new CPU();
```

```
CPU.Processor processor = cpu.new Processor();
CPU.RAM ram = cpu.new RAM();
System.out.println("Processor Cache = " + processor.getCache());
System.out.println("Ram Clock speed = " + ram.getClockSpeed());
}
}
```

OUTPUT



```
C:\Users\HP\Desktop>javac Main.java

C:\Users\HP\Desktop>java Main
Processor Cache = 4.0
Ram Clock speed = 3.5
```