Module 1

Ex.No.1.1 SUM OF INDIVIDUAL DIGITS OF A POSITIVE INTEGER

AIM : To write a java program to find the sum of individual digits of a positive integer.

Algorithm:

1. Read or initialize an integer N.
2. Declare a variable (sum) to store the sum of numbers and initialize it to 0.
3. Find the remainder by using the modulo (%) operator. It gives the last digit of the number (N).
4. Add the last digit to the variable sum.
5. Divide the number (N) by 10. It removes the last digit of the number.
6. Repeat the above steps (3 to 5) until the number (N) becomes 0.

Program:

```
import java.util.Scanner;
public class SumOfDigits
{
public static void main(String args[])
{
int number, digit, sum = 0;
Scanner sc = new Scanner(System.in);
System.out.print("Enter the number: ");
number = sc.nextInt();
while(number > 0)
{
digit = number % 10;
sum = sum + digit;
number = number / 10;
}
System.out.println("Sum of Digits: "+sum);
}
}
```

Output:

Enter the number: 7896
Sum of Digits: 30

Result:

Thus the java program to find the sum of individual digits of a positive integer is executed and verified successfully.

Ex.No.1.2

GENERATE THE FIRST N TERMS OF THE SEQUENCE

AIM: To write a Java program to generate the first n terms of the sequence.

ALGORITHM

1. Start the program
2. Read a number n using Scanner class
3. Iterate for loop using variable i from 1 to n
4. Print i

Program

```
import java.util.*;
public class  Sequence {
public static void main(String[] args)
{
System.out.println("Enter Number: ");
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
for(int i = 1; i <= n; i++)
System.out.print(i+" ");
}
}
```

Output

Enter Number:
10
1 2 3 4 5 6 7 8 9 10

Result:

Thus the Java program to generate the first n terms of the sequence is executed and verified successfully.

Ex.No.1.3.
GENERATE ALL THE PRIME NUMBERS BETWEEN 1 AND N

AIM: To write a Java program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

Algorithm

Step1- Start

Step 2- Declare an integer : n

Step 3- Prompt the user to enter an integer value

Step 4- Read the value using Scanner class

Step 5- Using a for loop from 1 to n, check if the 'i' value is divisible by any number from 2 to i.

Step 6- If yes, check the next number

Step 7- If no, store the number as a prime number

Step 9- Stop

Program:

```
import java.util.Scanner;
public class PrimeNumbers{
  public static void main(String arg[]){
    int i,n,counter, j;
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the n value : ");
    n=scanner.nextInt();
```

```
    System.out.print("Prime numbers between 1 to n are ");
    for(j=2;j<=n;j++){
      counter=0;
      for(i=1;i<=j;i++){
        if(j%i==0){
          counter++;
        }
      }
      if(counter==2)
      System.out.print(j+" ");
    }
  }
}
```

Output:

Enter the n value : 32

Prime numbers between 1 to n are 2 3 5 7 11 13 17 19 23 29 31

Result:

Thus the Java program to generate all the prime numbers between 1 and n is executed and verified successfully

Ex. No.1.4FINDLARGEST AND SMALLEST NUMBER IN A LISTOF INTEGERS

AIM: To write a Java program to find both the largest and smallest number in a list of integers.

ALGORITHM:

1. Start
2. Read the number of values “n” in array
3. Declare an array and read values using Scanner class
4. Initialise two variablesmin and max with arr[0]
5. Iterate over array using for loop
6. If current element is greater than max, then assign current element to max.
7.If current element is smaller than min, then assign current element to min.
8.Print smallest and largest element

Program

```
import java.util.*;
public class MinMax
{
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
System.out.print("Enter array size: ");
int n = sc.nextInt();
int arr[]=new int[n];
System.out.print("Enter array elements: ");
for(int i=0;i<n;i++)
arr[i]=sc.nextInt();
System.out.println("Entered Array: ");
for(int i=0;i<n;i++)
System.out.print(arr[i]+ " ");
int min=arr[0],max=arr[0];
for(int i=0;i<n;i++)
{
if(min>arr[i])
min=arr[i];
if(max<arr[i])
max=arr[i];
}
System.out.println("\nMaximum is : "+max);
System.out.println("Minimum is : "+min);
```

```
}
}
```

Output:

```
Enter array size: 5
Enter array elements: 67
45
89
80
3
Entered Array:
67 45 89 80 3
Maximum is : 89
Minimum is : 3
```

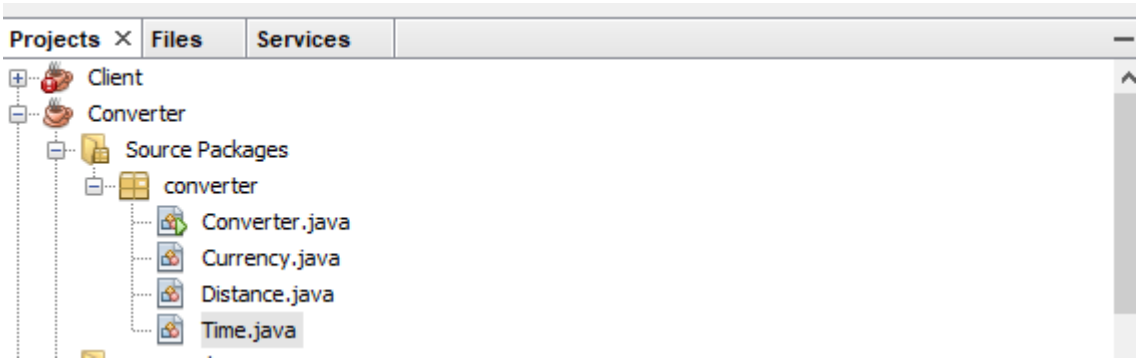
RESULT: Thus the Java program to find both the largest and smallest number in a list of integers is executed and verified successfully.

Ex.No.1.5 PROGRAM USING PACKAGES

Aim: To write a Java program to to implement a package for currency, distance and time converter.

ALGORITHM:

1. Create a package named converter
2. Inside that package create 3 classes named Currency, Distance and Time.
3. Define the methods for conversion in the created class
4. Create objects for the classes in Converter class
5. Call the methods from the Converter class using switch case statement
6. Print the result.



Program

Currency.java

```
package converter;

import java.util.Scanner;
import java.io.*;

public class Currency {

double inr,usd;
Scanner s=new Scanner(System.in);
public void dollartorupee( )
{
System.out.println("Enter dollars to convert into Rupees");
usd=s.nextDouble( );
inr=usd*67;
System.out.println("Dollar="+usd+"equal to INR="+inr);
}
public void rupeetodollar( )
{
System.out.println("Enter Rupee to convert into Dollars:");
inr =s.nextDouble();
usd=inr/67;
System.out.println("Rupee="+inr+"equal to Dollars="+usd);
}
}
```

Distance.java

```
package converter;
import java.util.*;
import java.io.*;

public class Distance
{
Scanner sc=new Scanner(System.in);
double m,km;
public void mtokm( )
{
System.out.print("Enter in meter");
m=sc.nextDouble();
km=(m/1000);
System.out.println(m+ "m" +"equal to" +km+ "kilometres");
}
public void kmtom( )
{
System.out.print("Enter in km");
km=sc.nextDouble( );
m=(km*10000);
System.out.println(km+"km"+"equal to"+ m +"metres");
}

}
```

Time.java

```
package converter;
import java.util.*;
import java.io.*;

public class Time {

int hours,seconds,minutes;
int input;
Scanner sc=new Scanner(System.in);
public void secondtohours( )
{
System.out.print("Enter the number of seconds");
input=sc.nextInt( );
```

```
hours=input/3600;
minutes=(input%3600) / 60;
seconds=(input%3600) % 60;
System.out.println("Hours."+hours);
System.out.println("Minutes."+minutes);
System.out.println("Seconds."+seconds);
}
public void minutestohours( )
{
System.out.print("Enter the number of minutes.");
minutes=sc.nextInt( );
hours=minutes / 60;
minutes=minutes%60;
System.out.println("Hours"+hours);
System.out.println("Minutes."+minutes);
}

}
```

Converter.java

```
package converter;

import java.util.*;
import java.io.*;
public class Converter {

    public static void main(String[] args) {
        // TODO code application logic here

        Scanner s=new Scanner(System.in);
int choice=1,ch;
Currency c=new Currency( );
Distance d=new Distance( );
Time t= new Time( );

do
{
System.out.println("1.dollar to rupee");
System.out.println("2. rupee to dollar");
System.out.println("3.Meter to kilometer");
System.out.println("4.kilometer to meter");
System.out.println("5.seconds to hours");
```

```
System.out.println("6.minutes to hours");

choice=s.nextInt();
switch(choice)
{
case 1:
{
c.dollartorupee( );
break;
}
case 2:
{
c.rupeetodollar( );
break;
   }
case 3 :
{
d.mtokm( );
break;
}
case 4:
{
d.kmtom( );
break;
}

case 5:
{
t.secondtohours( );
break;
}
case 6:
{
t.minutestohours( );
break;
}
   }
System.out.println("Enter 0 to quit and 1 to continue");
ch=s.nextInt( );
}while(ch==1);
}
}
```

Output

1.dollar to rupee
2. rupee to dollar
3.Meter to kilometer
4.kilometer to meter
5.seconds to hours
6.minutes to hours
1
Enter dollars to convert into Rupees
20
Dollar=20.0equal to INR=1340.0
Enter 0 to quit and 1 to continue
1
1.dollar to rupee
2. rupee to dollar
3.Meter to kilometer
4.kilometer to meter
5.seconds to hours
6.minutes to hours
4
Enter in km6
6.0kmequal to60000.0metres
Enter 0 to quit and 1 to continue

Result:

Thus the Java program to to implement a package for currency, distance and time converter has been executed and verified successfully.