MODULE: I	
Ex. No: 1.1	**Printing the output in the console**
Date:	

Aim:

To write a Java program to print the output in the console

Algorithm:

Step 1: Start the program

Step 2: Create a class and main method

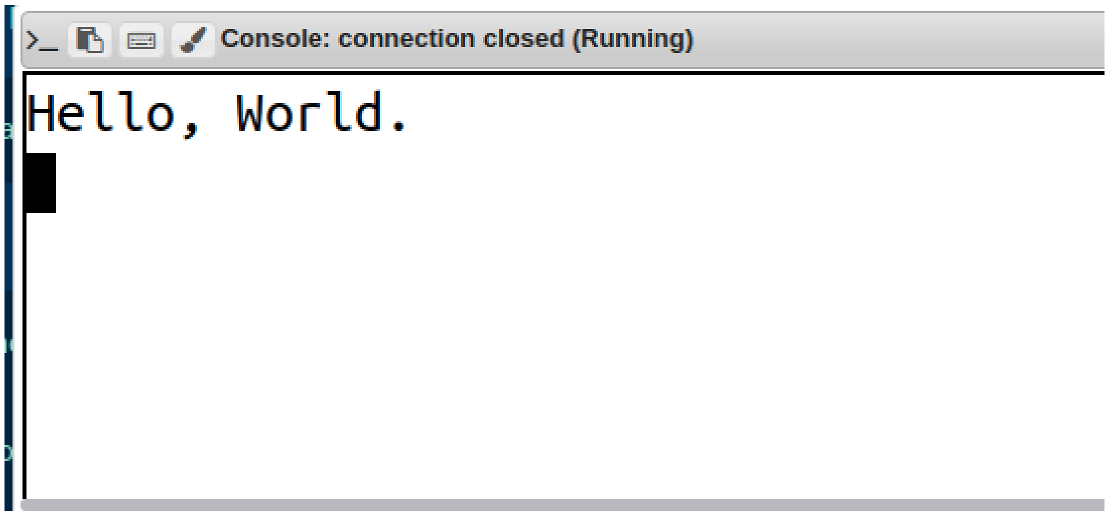
Step 3: Print the output

Step 4: Stop the program

Source Code:

```
class Main

{

    public static void main(String argsp[])

      {

                      System.out.println("Hello, World.");

      }

}
```

Output:



```
Hello, World.
```

Result:

Thus, the above program was executed successfully.

Ex. No: 1.2	Getting the input from the user using Scanner object
Date:	

Aim:

To write a Java program to get the input from the user using Scanner object as integers and print the output.

Algorithm:

Step 1: Start the program

Step 2: Import the header files

Step 3: Create a class and main method

Step 4: Get the input from the user using Scanner object

Step 5: Print the output

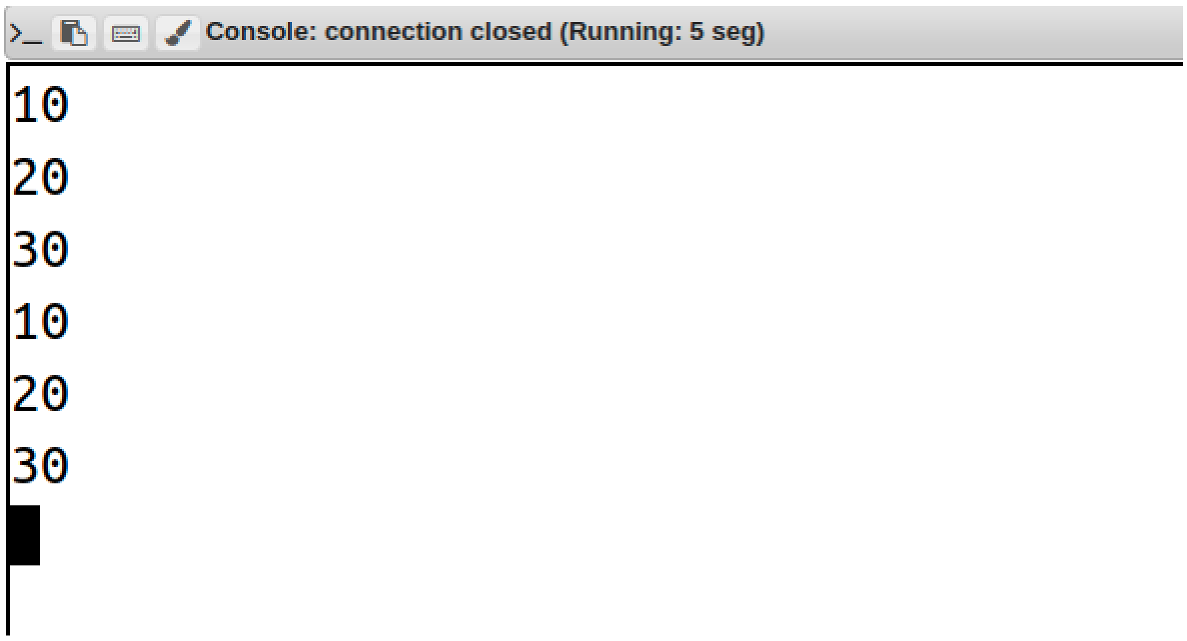
Step 6: Stop the program

Source Code:

```
import java.util.*;
public class Main
{
        public static void main(String[] args)
{
        Scanner scan = new Scanner(System.in);
        int a = scan.nextInt();
        int b = scan.nextInt();
        int c = scan.nextInt();
        System.out.println(a);
```

```
                System.out.println(b);

                System.out.println(c);

            }

        }
```

Output:



Console: connection closed (Running: 5 seg)

```
10
20
30
10
20
30
```

Result:

Thus, the above program was executed successfully.

Ex. No: 1.3	Sorting using Primitive data type
Date:	

Aim:

To write a Java program to determine which primitive data types are capable of properly storing that input.

Algorithm:

Step 1: Start the program

Step 2: Import header files

Step 3: Create a class and main method

Step 4: Get the input from the user using Scanner object

Step 5: Use exception handling to escape from exception

Step 6: Use for loop to print the required output

Step 7: In the for loop, use if statement for certain conditions

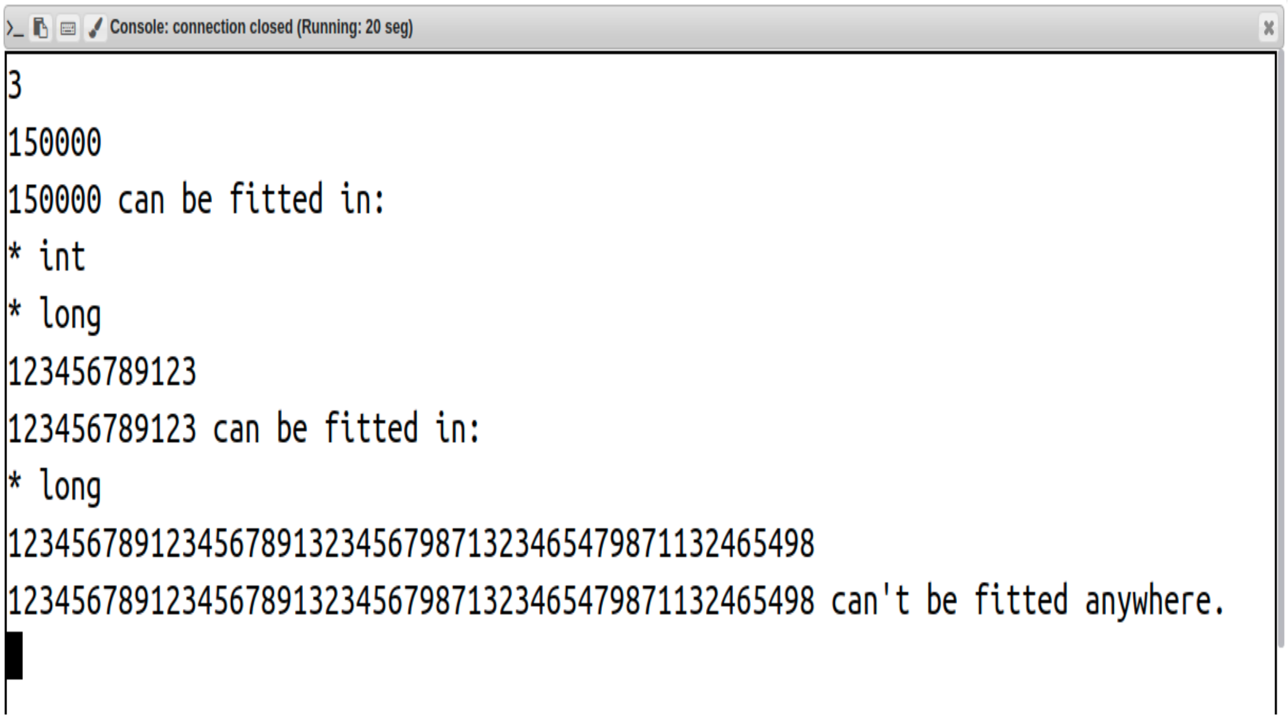
Step 8: Print the output

Step 9: Stop the program

Source Code:

```
import java.util.*;
import java.lang.Math;
public class Main
{
    public static void main (String[] args)
{
        Scanner sc=new Scanner(System.in);
        int num=sc.nextInt();
        for(int i=1;i<=num;i++)
        {
            try
            {
                long num1=sc.nextLong();
                System.out.println(num1+" can be fitted in:");
                if(num1>=-128 && num1<=127)
                    System.out.println("* byte");
                if(num1>=-32768 && num1<=32767)
                    System.out.println("* short");
                if(num1>=Math.pow(-2,31) && num1<=Math.pow(2,31)-1)
                    System.out.println("* int");
                if(num1>=Long.MIN_VALUE && num1<=Long.MAX_VALUE)
                    System.out.println("* long");
            }
            catch(Exception error)
            {
                System.out.println(sc.next()+" can't be fitted anywhere.");
            }
        }
    }
}
```

Output:



```
3
150000
150000 can be fitted in:
* int
* long
123456789123
123456789123 can be fitted in:
* long
123456789123456789132345679871323465479871132465498
123456789123456789132345679871323465479871132465498 can't be fitted anywhere.
```

Result:

Thus, the above program was executed successfully.

Ex. No: 1.4	Converting Integer to String
Date:	

Aim:

To write a Java program to convert an integer into a string.

Algorithm:

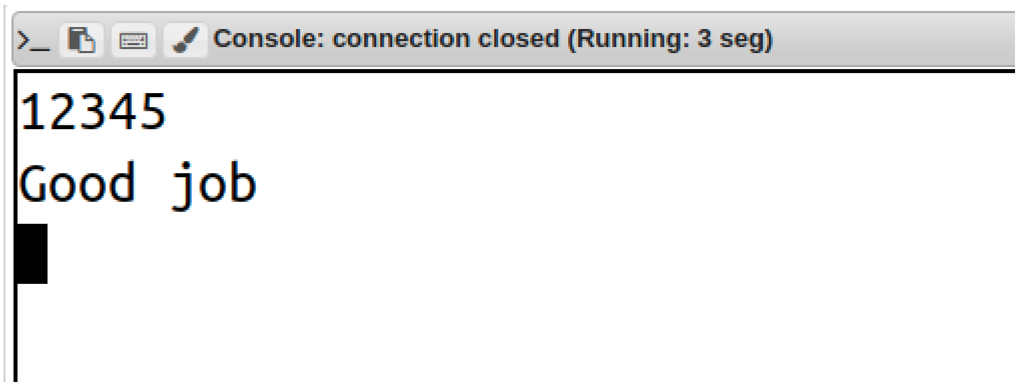
```
Step 1: Start
Step 2: Import header files
Step 3: Create a class and main method
Step 4: Get the input from the user
Step 5: Use if else statement
            if(num==Integer.parseInt(str))
             {
            }
             else{}
Step 6: Print the output
Step 7: Stop
```

Source Code:

```
import java.util.Scanner;
class Main
{
    public static void main (String[] args)
    {
       Scanner sc=new Scanner(System.in);
```

```
                int num=sc.nextInt();
                String str=Integer.toString(num);
                if(num==Integer.parseInt(str))
                {
                    System.out.println("Good job");
                }
                else
                {
                    System.out.println("Wrong Answer");
                }
            }
        }
```

Output:


Console: connection closed (Running: 3 seg)

```
12345
Good job
```

Result:

Thus, the above program was executed successfully.

Ex. No: 5	Reading file content until reaching End Of File
Date:	

Aim:

To write a Java program to read ***n*** lines of input until to reach End Of File, then number and print all ***n*** lines of content.

Algorithm:

Step 1: Start

Step 2: Import header files

Step 3: Create a class

Step 4: Get the input from the user

Step 5: Use while loop & if statement

Step 6: In while loop use if statement

```
while (true){
        if(!input.hasNext())
        {
                break;
        }}
```

Step 7: Print the output

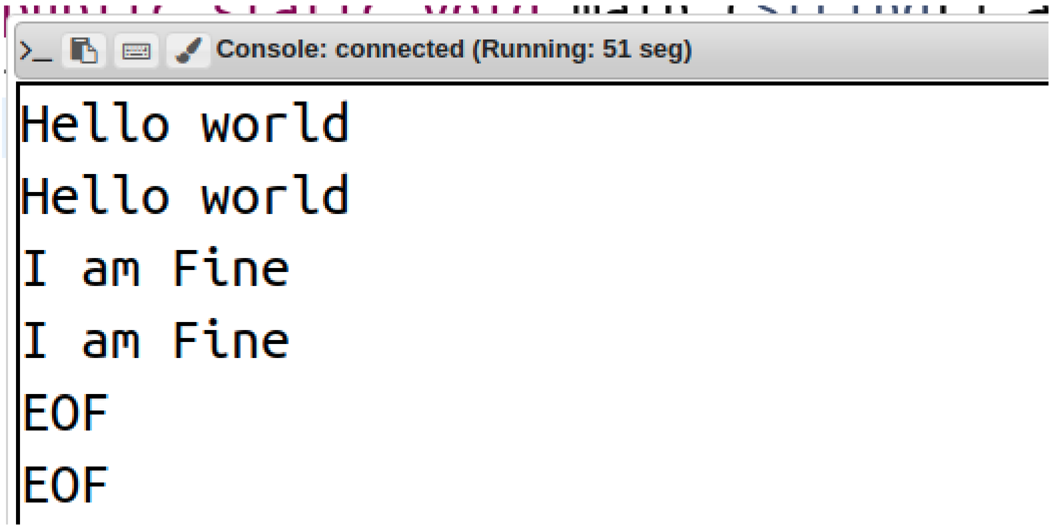
Step 8: Stop

Source Code:

```
import Java.util.*;
class Main
{
```

```
            public static void main (String[] args)
             {
                Scanner input=new Scanner(System.in);
                while (true) {
                    String line = input.nextLine();
                    System.out.println(line);
                    if(!input.hasNext())
                    {
                        break;
                    }
            }
         }
        }
```

<u>Output:</u>



```
Hello world
Hello world
I am Fine
I am Fine
EOF
EOF
```

<u>Result:</u>

Thus, the above program was executed successfully.

Module II: Object-Oriented Mechanisms	
Ex. No: 2.1	**Sum of two numbers using Constructor**
Date:	

Aim:

To create a constructor in person.java and invoke the constructor in Main.java to find the sum of the two numbers

Algorithm:

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2 and assign the result to sum.
                sum=num1+num2
Step 5: Display sum
Step 6: Stop
```

Source Code:

Main.Java

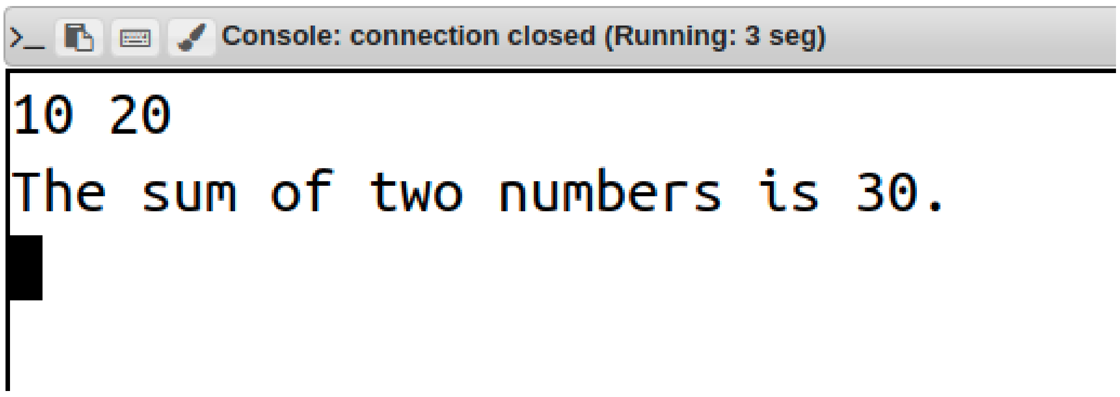
```
import java.util.Scanner;
class Main
{
	public static void main (String[] args)
	{
		Scanner sc=new Scanner(System.in);
		int num1=sc.nextInt();
```

```
                int num2=sc.nextInt();

                Person person = new Person(num1, num2);

        }

}
```

<u>Person.Java</u>

```
class Person

{

        Person(int num1,int num2)

        {

                System.out.println("The    sum    of    two    numbers    is

        "+(num1+num2)+".");

        }

}
```

<u>Output:</u>


Console: connection closed (Running: 3 seg)

```
10 20
The sum of two numbers is 30.
```

<u>Result:</u>

Thus, the program is executed successfully.

Ex. No: 2.2	Inheritance
Date:	

Aim:

To write a Java program for the given schema

Algorithm:

Step -1: Start the Program

Step -2: Declare the parent class Animal

Step -3: Declare a String name and define the method eat which prints "Hai "+name.

Step -4: Declare the derived class Dog which is derived from Animal class

Step -5: Declare the String n and get the value from user and define the method eat which prints "Hai"+n.

Step -5: Create object for Dog class

Step -6: Call the eat() method with created object

Step -7: Stop the Program

Source Code:

Main.Java

```
class Main
{        public static void main(String[] args)
       {
            Dog obj = new Dog();
            obj.eat();
        }
}
```

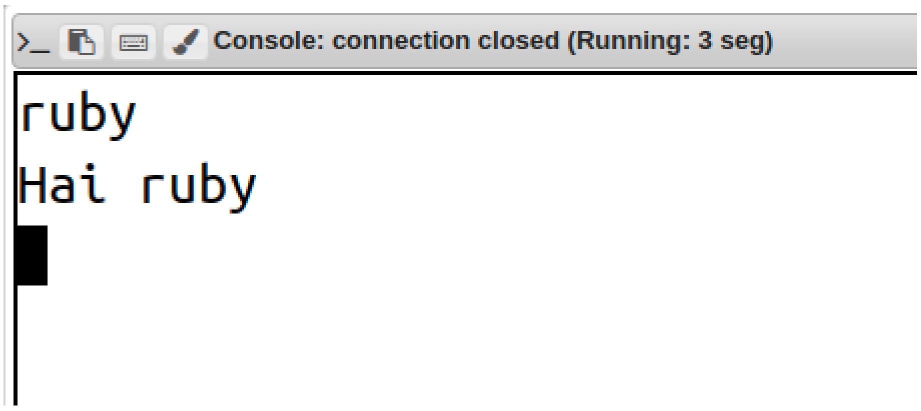
Animal.java

```
class Animal
{
    String name;
    void eat()
    {
        System.out.println("Hai "+name);
    }
}
```

Dog.java

```
import java.util.*;
class Dog extends Animal
{
        Scanner sc = new Scanner(System.in);
        String n = sc.next();
        void eat()
        {
            System.out.println("Hai "+n);
        }
}
```

Output:



```
ruby
Hai ruby
```

Result:

Thus, the program was executed successfully.

Ex. No: 2.3	**Displaying Bank account details using Inheritance**
Date:	

Aim:

To create a class named account with the given private attributes and to create a class named currentAccount with the given private attributes that extends the class account.

Algorithm:

Step 1: Start the program

Step 2: Declare a class Account

Step 3: Declare the strings accName, accNo, bankname and define a protected method display which prints account Name, account Number and bank Name.

Step 4: Declare a class SavingsAccount which should be derived from Account class

Step 5: Declare a string and orgName and define a method display which call it's overridden method and prints the orgName.

Step 6: Declare a class CurrentAccount which should be derived from Account class

Step 7: Declare a string and tinNumber and define a method display which call it's overridden method and prints the tinNumber.

Step 8: Declare a variable choice and get the choice from user

Step 9: if choice is 1 then get accName,accNo,bankName,orgName from user.

Step 10: Create object for savings account and initialize it and it's parents member with the input values and call display method

Step 11: Else create object for CurrentAccount get the tinNumber and call display method.

Step 12: Stop the program

Source Code:

Main.Java

```
import java.util.*;

class Main

{

    public static void main (String[] args)

{

     System.out.println("choose Account Type\n1.Savings

     Account\n2.Current Account");

        int choice,i;

        String[] a= new String[4];

        Scanner sc=new Scanner(System.in);

        choice=sc.nextInt();

        if(choice==1)

        {

            System.out.println("Enter Account details in comma

separated(Account Name,Account Number,Bank Name,Organisation Name)");

            sc.nextLine();

            for(i=0;i<4;i++){

            a[i]=sc.nextLine();}

            SavingsAccount o=new SavingsAccount();

            o.accName=a[0];

            o.accNo=a[1];

            o.bankName=a[2];

            o.orgName=a[3];

            o.display();
        }
        else
        {
            CurrentAccount o=new CurrentAccount();
        }
    }
}
```

Account.java

```
class Account
{
    String accName;
    String accNo;
    String bankName;
    protected void display()
    {
        System.out.println("Account  Name:"+accName+"\nAccount  Number:"
+accNo+"\nBank Name:"+bankName);
    }
}
```

SavingsAccount.java

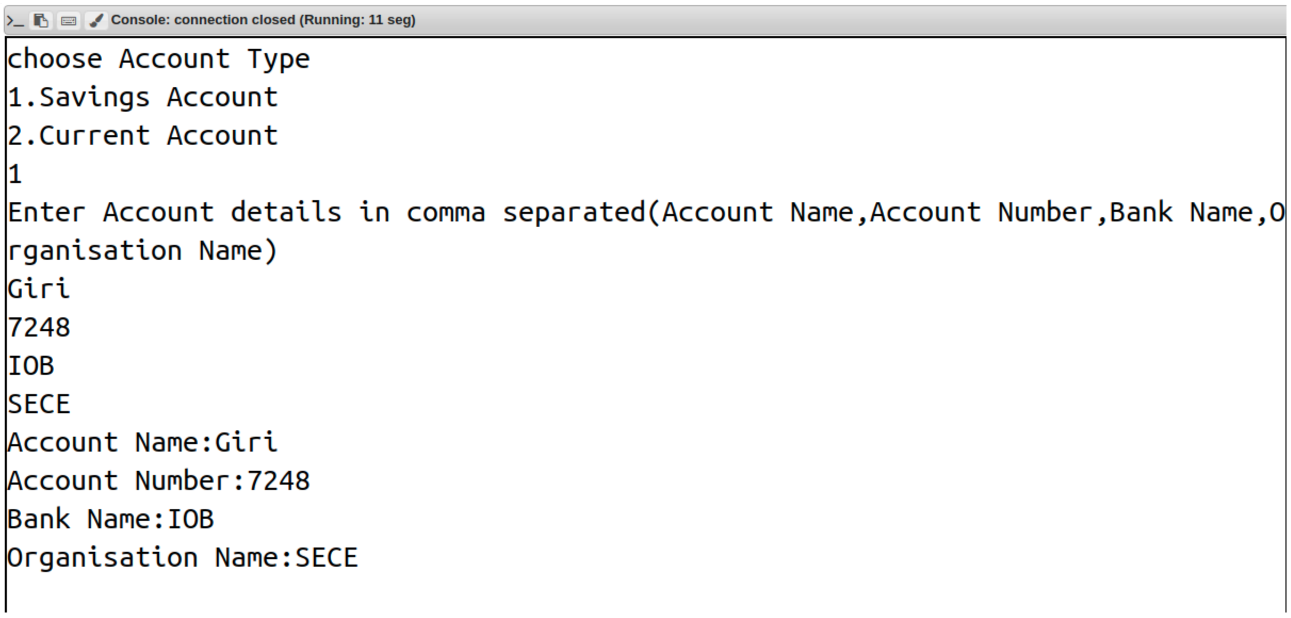
```
     class SavingsAccount extends Account
     {
         String orgName;
         public void display()
```

```
        {
            super.display();
            System.out.println("Organisation Name:"+orgName);
        }
    }
```

CurrentAccount.java

```
    class CurrentAccount extends Account
    {
        String tinNumber;
        public void display()
        {
            super.display();
            System.out.println("\nTin Number:"+tinNumber);
        }
    }
```

Output:


Console: connection closed (Running: 11 seg)

```
choose Account Type
1.Savings Account
2.Current Account
1
Enter Account details in comma separated(Account Name,Account Number,Bank Name,O
rganisation Name)
Giri
7248
IOB
SECE
Account Name:Giri
Account Number:7248
Bank Name:IOB
Organisation Name:SECE
```

Result:

Thus, the program was executed successfully.

Ex. No: 2.4	Displaying details of Cricket players using Inheritance (IS-A) Relationship
Date:	

Aim:

To create the constructor for the cricket class, to create object for the cricket and to invoke display() method in main method

Algorithm:

Step 1: Start the program

Step 2: Declare a class Cricket

Step 3: Declare the variables name, age, weight, height, runs and define a method display which prints the player details

Step 4: Create a object for Cricket class

Step 5: Get the player details and pass it to Cricket class and call display method.

Step 6: Stop the program

Source Code:

Main.java

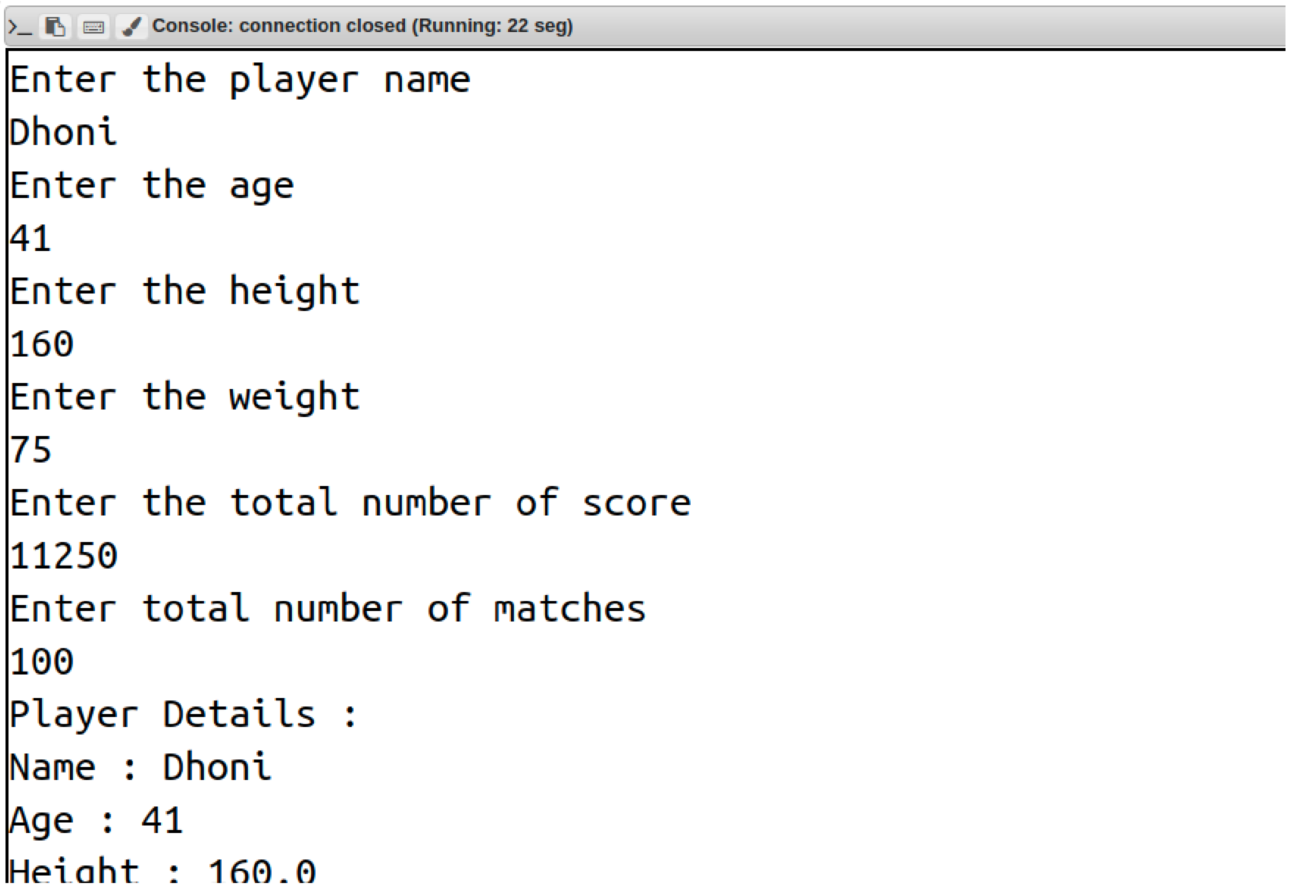
```
import java.util.*;
class Main
{
    public static void main (String[] args)
{
        Scanner sc=new Scanner(System.in);
```

```
            System.out.println("Enter the player name");
            String n=sc.nextLine().trim();
            System.out.println("Enter the age");
            int a=sc.nextInt();
            System.out.println("Enter the height");
            float h=sc.nextFloat();
            System.out.println("Enter the weight");
            int w=sc.nextInt();
            System.out.println("Enter the total number of score");
            int r=sc.nextInt();
            System.out.println("Enter total number of matches");
            float m=sc.nextInt();
            Cricket cricket=new Cricket(n,a,h,w,r,m);
            cricket.display();
        }
    }
```

Cricket.Java

```
    class Cricket
    {
        String name;
        int age;
        float height;
        int weight;
        int runs;
        float matches;
        Cricket(String   name,int   age,float   height,int   weight,int 
    runs,float matches)
        {
            this.name=name;
            this.age=age;
            this.height=height;
            this.weight=weight;
            this.runs=runs;
            this.matches=matches;
        }
        public void display()
        {
         System.out.println("Player Details : ");
         System.out.println("Name : "+this.name);
         System.out.println("Age : "+this.age);
         System.out.println("Height : "+this.height);
         System.out.println("Weight : "+this.weight);
         double ar=this.runs/this.matches;
         System.out.printf("Average Run : %.2f",ar);
        }
    }
```

Output:



```
Enter the player name
Dhoni
Enter the age
41
Enter the height
160
Enter the weight
75
Enter the total number of score
11250
Enter total number of matches
100
Player Details :
Name : Dhoni
Age : 41
Height : 160.0
```

Result:

Thus, the program is executed successfully.

Ex. No: 2.5	Encapsulation
Date:	

Aim:

To implement encapsulation by creating person class with private variables and to create object in Main class

Algorithm:

Step 1: Start the program

Step 2: Declare the class Person

Step 3: Declare the private members name, empNo, age.

Step 4: Declare the setters and getters for each member.

Step 5: Create object for person class

Step 6: Get the person details from user and call the appropriate set methods

Step 7: Call the get method of each member

Step 8: Stop the program

Source Code:

Main.Java

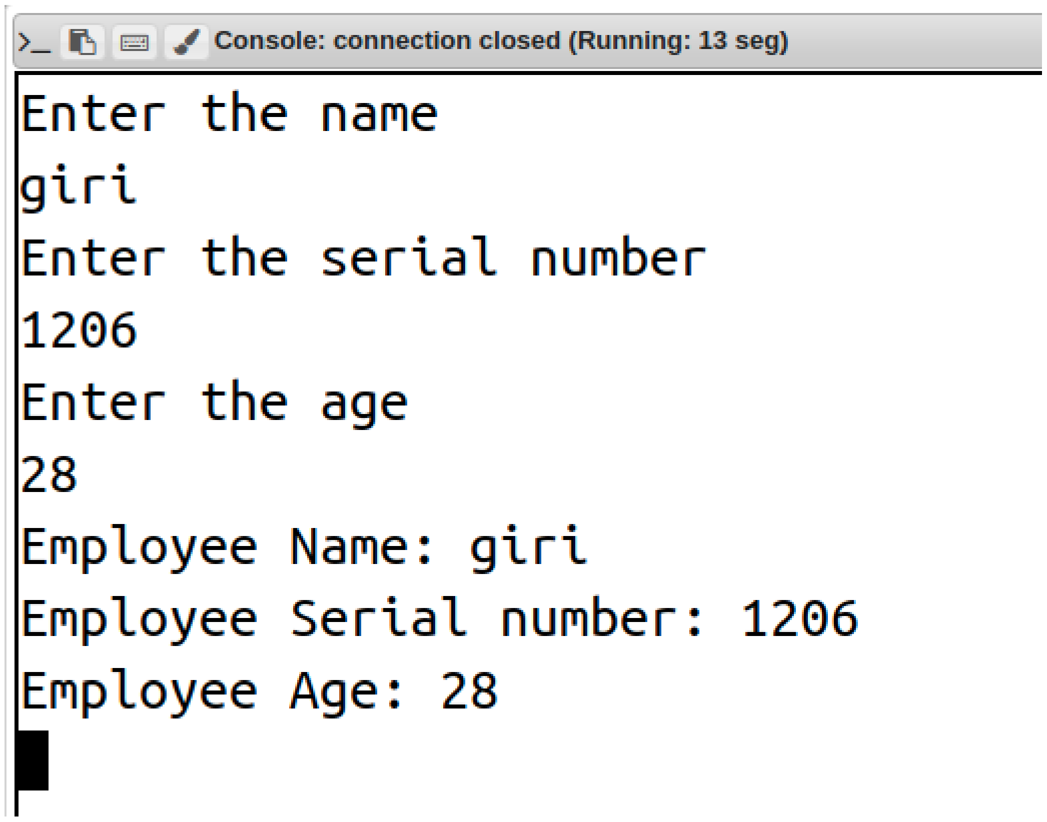
```
import java.util.*;
class Main
{
    public static void main (String[] args)
    {
        Scanner scan = new Scanner(System.in);
```

```
            String name,sno;

            int ag;

            System.out.println("Enter the name");

            name = scan.nextLine();

            System.out.println("Enter the serial number");

            sno = scan.nextLine();

            System.out.println("Enter the age");

            ag = scan.nextInt();

            Person p = new Person();

            p.setemployeeName(name);

            p.setserialNumber(sno);

            p.setage(ag);

            System.out.println("Employee Name: "+p.getemployeeName());

            System.out.println("Employee Serial number:
    "+p.getserialNumber());

            System.out.println("Employee Age: "+p.getage());

        }

    }
```

<u>**Person.Java**</u>

```
    class Person

    {

        private String employeeName;

        private String serialNumber;

        private int age;

        public void setemployeeName(String name)
        {
            employeeName = name;
        }
        public void setserialNumber(String sno)
        {
            serialNumber = sno;
        }
        public void setage(int ag)
        {
            age = ag;
        }
        public String getemployeeName()
        {
            return employeeName;
        }
        public String getserialNumber()
        {
            return serialNumber;
        }
        public int getage()
        {
            return age;
        }
    }
```

Output:



```
Enter the name
giri
Enter the serial number
1206
Enter the age
28
Employee Name: giri
Employee Serial number: 1206
Employee Age: 28
```

Result:

Thus, the program was executed successfully.

Ex. No: 2.6	Abstract Class
Date:	

Aim:

To create an abstract class named book and to create an instance for the same

Algorithm:

Step 1: Start the program.

Step 2: Declare a abstract class Book.

Step 3: Declare a String title and method setTitle.

Step 4: Declare a class Mybook which inherits from abstract class Book

Step 5: Define the abstract method setTitle which prints title of the book

Step 6: Create object for Book class

Step 7: Get Title from user and call setTitle method

Step 8: Stop the program

Source Code:

Main.java

```
import java.util.Scanner;
class Main{
    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
        String title=sc.nextLine();
        Book b=new MyBook();
        b.setTitle(title);
    }}
```

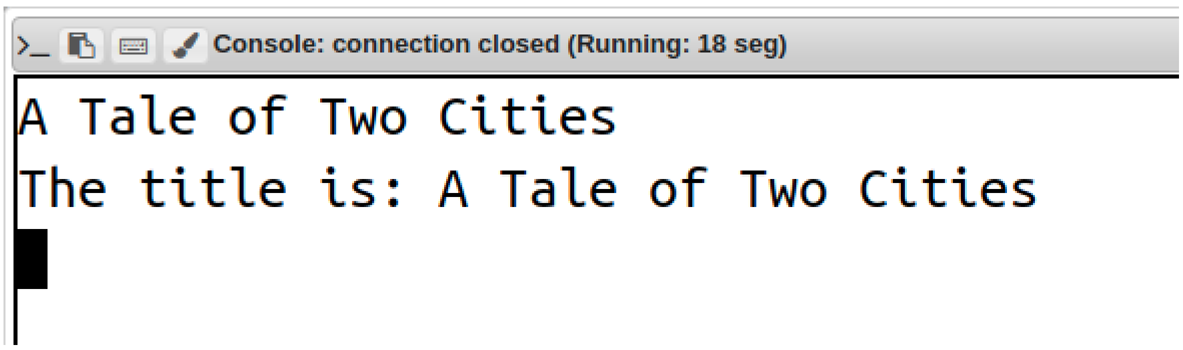
Book.java

```
abstract class Book

{   String title;

    abstract void setTitle(String s);

}
```

MyBook.java

```
class MyBook extends Book

{         void setTitle(String s)

        {

              System.out.println("The title is: "+s);        }

}
```

Output:



Console: connection closed (Running: 18 seg)

```
A Tale of Two Cities
The title is: A Tale of Two Cities
```

Result:

Thus, the program is executed successfully.

Ex. No: 2.7	Interface
Date:	

Aim:

To create an interface MyCalculator which implements interface

Algorithm:

```
Step 1: Start the program
Step 2: Declare a interface AdvancedArithmetic and declare a method divisor_sum
Step 3: Declare a class My Calculator which implements AdvancedArithmetic interface and
        define the method divisor_sum which returns divisor of given number
Step 4: Create a object for MyCalculator class
Step 5: Get a integer from user
Step 6: Call the divisor_sum method by passing input value
Step 7: Stop the program
```

Source Code:

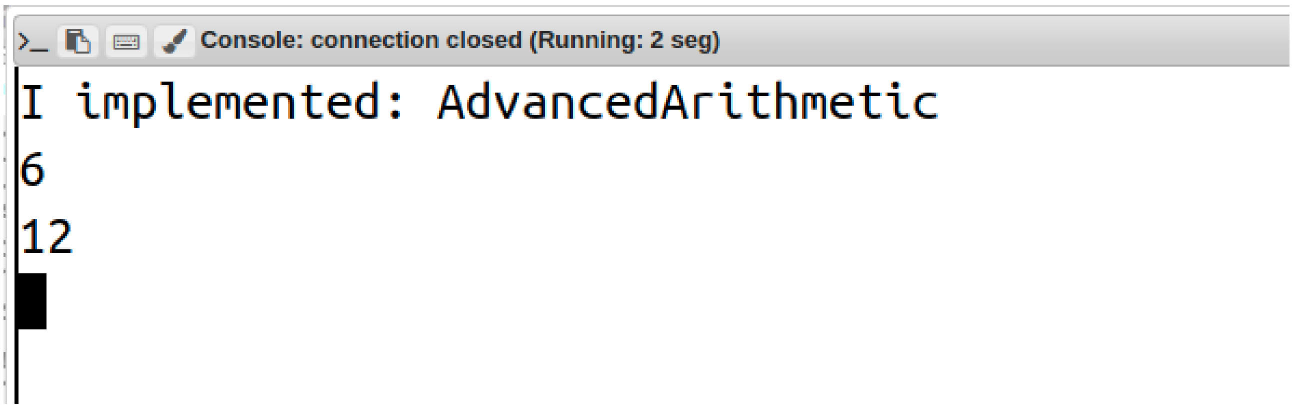
Main.Java

```
import java.util.Scanner;
interface AdvancedArithmetic
{
  int divisor_sum(int n);
}
class MyCalculator implements AdvancedArithmetic
{
 public int divisor_sum(int n)
```

```
{
    int sum = 0, i = 1;
    while (n != 0 && i <= n)
{
      if (n % i == 0)
{
        sum += i;
      }
      i++;
    }
    return sum;
  }
}
public class Main
{
public static void main(String[] args)
{
    MyCalculator my_calculator = new MyCalculator();
    System.out.print("I implemented: ");
    ImplementedInterfaceNames(my_calculator);
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    System.out.print(my_calculator.divisor_sum(n) + "\n");
    sc.close();
  }
  static void ImplementedInterfaceNames(Object o)
```

```
        {
            Class[] theInterfaces = o.getClass().getInterfaces();
            for (int i = 0; i < theInterfaces.length; i++)
        {
              String interfaceName = theInterfaces[i].getName();
              System.out.println(interfaceName);
            }
          }
        }
```

Output:



```
I implemented: AdvancedArithmetic
6
12
```

Result:

Thus, the program is executed successfully.

MODULE III: String	
Ex. No: 3.1	String Operations
Date:	

Aim:

To write a Java program, to print the sum of two strings, to check if the string, is greater than string 2 lexicographically and to capitalize the first letter of the two string.

Algorithm:

Step 1: Start

Step 2: Create a class 'Main'

Step 3: Instantiate an Input stream reader class by passing Inputstream object as a perameter

Step 4: Create a buffer reader, by passing the above Inputstreamreader

Step 5: Read data from the current reader as string using the readline() or read() method

Step 6: Then, find the length of given two strings

Step 7: Add the length of 2 Strings and print the result

Step 8: Compare two strings, if string, is lexicographically greater than string 2, the print "Yes" or else "No"

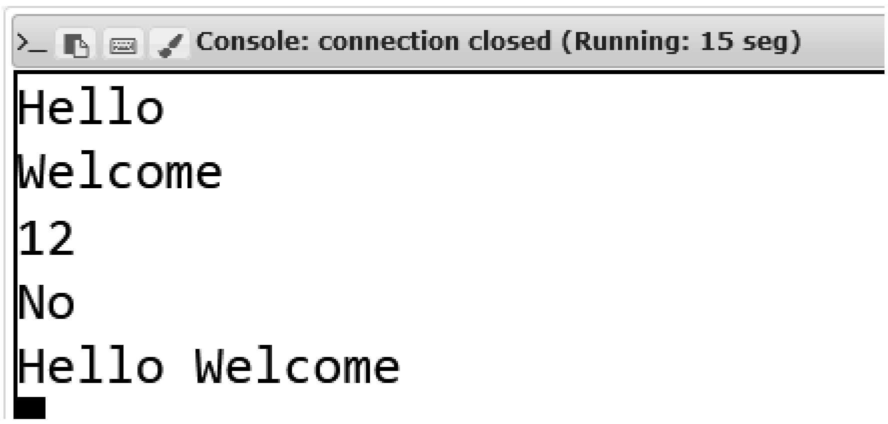
Step 9: Capitalize the first character of two strings and print the result

Step 10: Stop

Source Code:

```
import java.io.*;
class Main
{
    public static void main (String[] args) throws IOException
{
        BufferedReader br = new BufferedReader(new In-
putStreamReader(System.in));
        String str1 = br.readLine();
        String str2 = br.readLine();
        int l1 = str1.length();
        int l2 = str2.length();
        System.out.println(l1+l2);
        if(str1.compareTo(str2)>=0)   {
            System.out.println("Yes");
        }
        else       {
            System.out.println("No");
        }
        char c1 = str1.charAt(0);
        char c2 = str2.charAt(0);
        c1 = Character.toUpperCase(c1);
        c2 = Character.toUpperCase(c2);
        System.out.println(c1+str1.substring(1)+"     "+c2+str2.sub-
string(1));
    }
}
```

Output:



```
Hello
Welcome
12
No
Hello Welcome
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.2	String Concatenation
Date:	

Aim:

To write a Java program to perform the concatenation of given two strings using string library functions.

Algorithm:

Step-1: Start

Step-2: Create a class ‘Main’

Step-3: Use Scanner to get inputs from the user

Step-4: Get a String (a)

Step-5: Get a String (b)

Step-6: Concatenate two strings a and b

Step-7: Display the concatenated string

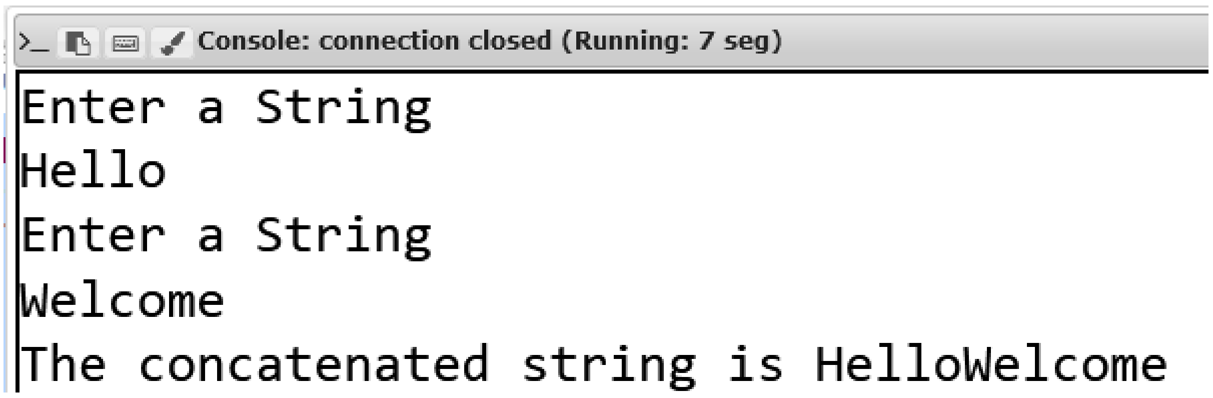
Step-8: Stop

Source Code:

```
import java.util.Scanner;

class Main

{

    public static void main (String[] args)

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a String");

        String a = sc.nextLine();
```

```
            System.out.println("Enter a String");
            String b = sc.nextLine();
            System.out.println("The concatenated string is "+a+b);
        }
    }
```

Output:



```
Enter a String
Hello
Enter a String
Welcome
The concatenated string is HelloWelcome
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.3	Converting String into uppercase
Date:	

Aim:

To write a program to change the Given string to uppercase without using string Library functions.

Algorithm:

Step-1: Start

Step-2: Create a class ‘Main

Step-3: Use Scanner to get input from the user

Step-4: Get a String named str

Step-5: Convert the string into Uppercase using for each loop

Step-6: Display the String

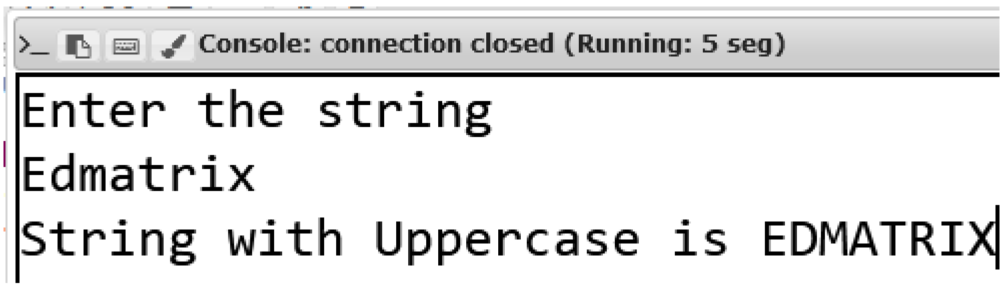
Step-7: Stop

Source Code:

```
import java.util.*;

class Main

{

    public static void main (String[] args)

{

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the string");

        String s = sc.nextLine();

        System.out.print("String with Uppercase is ");
```

```
                for(char c: s.toCharArray())
               {
                    System.out.print((c>96) ? (char)(c-32) : c);
                }
            }
        }
```

Output:



Console: connection closed (Running: 5 seg)

```
Enter the string
Edmatrix
String with Uppercase is EDMATRIX
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.4	Print Substring
Date:	

Aim:

To write a Java program to print the substring in the inclusive range from start to end

Algorithm:

Step 1: Start

Step 2: Create a class 'Main'

Step 3: Use a start value to get inputs from the user

Step 4: Get string input (s)

Step 5: Get the Integer value (n)

Step 6: Use the substring() method and store the result in another str (str)

Step 7: Declare the string called min and store the string (str)

Step 8: Declare the string called max and store the string (str)

Step 9: Use for loop and compare str with min, max

Step 10: If min>0, Store the (str) in min

Step 11: If max<0, Store the (str) in max

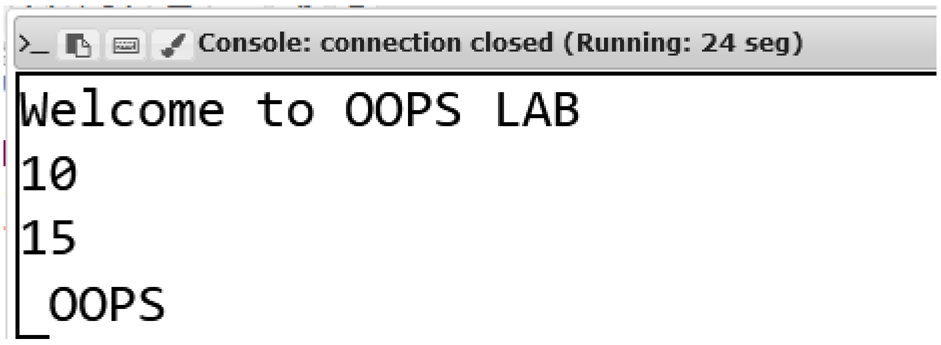
Step 12: Print min, max string

Step 13: Stop

Source Code:

```
import java.util.*;

class Main

{

    public static void main (String[] args)
{
        Scanner sc = new  Scanner(System.in);
        String a = sc.nextLine();
        int x = sc.nextInt();
        int y = sc.nextInt();
        System.out.println(a.substring(x,y));
    }
}
```

Output:



```
Welcome to OOPS LAB
10
15
 OOPS
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.5	Find the Lexicographically Smallest and Largest Substring
Date:	

Aim:

To write a Java program to find out the Lexicographically smallest and largest substring of length H.

Algorithm:

Step 1: Start

Step 2: Create a class ‘Main’

Step 3: Use Scanner to get inputs from the user

Step 4: Get String input (s)

Step 5: Get the Integer value (n)

Step 6: Use the substring() method and store the result in another str (str)

Step 7: Declare the string called min and store the string

Step 8: Declare the string called max and store the string

Step 9: Use for Loop and Compare str with min,max

Step 10: If min>0, Store the (str) in min

Step 11: If max<0, Store the(str) in max

Step 12: Print min and max string

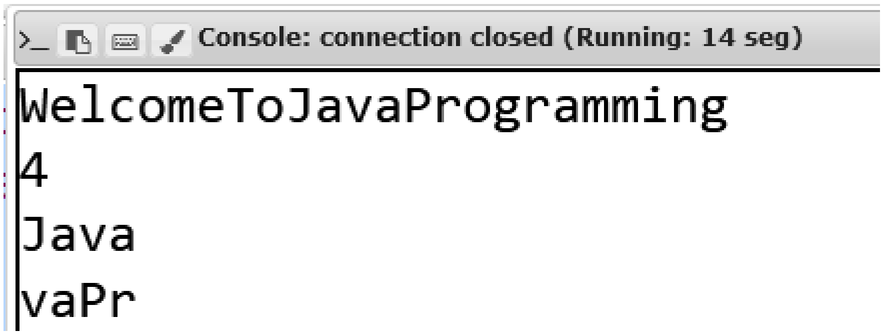
Step 13: Stop

Source Code:

```
import java.util.*;
class Main{
    public static void main (String[] args) {
```

```
                Scanner sc = new Scanner(System.in);
                String s = sc.nextLine();
                int n = sc.nextInt();
                String str = s.substring(0,n);
                String min = str;
                String max = str;
                for(int i=n;i<s.length();i++){
                    str = str.substring(1,n)+s.charAt(i);
                    if(min.compareTo(str)>0){min = str;}
                    if(max.compareTo(str)<0){max = str;}   }
                System.out.println(min+"\n"+max);
            }
        }
```

Output:



```
WelcomeToJavaProgramming
4
Java
vaPr
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.6	Check Whether the String is Palindrome or Not
Date:	

Aim:

To write program to check if the string is palindrome (or) Not

Algorithm:

Step 1: Start

Step 2: Create a class ‘Main’

Step 3: Use scanner to get an input from the user, get a string (str1)

Step 4: Create a function named **‘PalindromeCheck’**

Step 5: Create a String Buffer object by passing the required string as a Parameter

Step 6: Reverse the contents of the object using the reverse() method

Step 7: Convert the String buffer object to string using the tostring() method

Step 8: Compare the string and the Reversed one, If True, hen print **‘Yes’** or else print **‘No’**

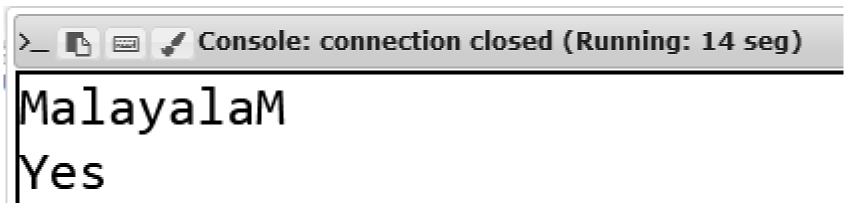
Step 9: Stop

Source Code:

```
import java.util.Scanner;
class Main{
  public static boolean palindromeCheck(String str){
        StringBuffer sb = new StringBuffer(str);
        StringBuffer rev = new StringBuffer(str).reverse();
        return (sb.toString()).equals(rev.toString());
    }
    public static void main (String[] args){
```

```
                Scanner sc = new Scanner(System.in);

                String str1 = sc.nextLine();

                if(palindromeCheck(str1)){

                    System.out.println("Yes");

                }

         else

        {

                    System.out.println("No");

                }

            }

        }
```

Output:



```
MalayalaM
Yes
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.7	Count the Number of Vowels in a String
Date:	

Aim:

To write a Java program to count the number of vowels in a given string

Algorithm:

Step 1: Start

Step 2: Create a class 'Main'

Step 3: Use Scanner to get inputs from the user

Step 4: Get a string input (str)

Step 5: Convert the Given string into lowercase and declare the count

Step 6: Initialize count = 0

Step 7: Use for loop to Compare each character in the sentence with Characters {'a', 'e','i','o','u'}

Step 8: If a match occurs, increment the count

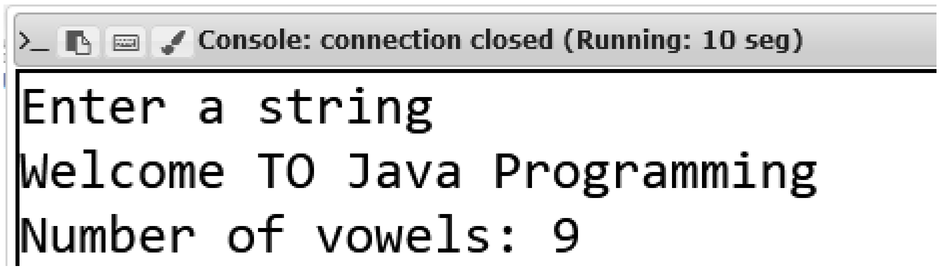
Step 9: Finally, print count

Step 10: Stop

Source Code:

```
import java.util.*;

class Main

{

    public static void main (String[] args) {

        Scanner scan = new Scanner(System.in);

        String str;
        System.out.println("Enter a string");
        str = scan.nextLine();
        str = str.toLowerCase();
        int count;
        count = 0;
        for(int i = 0 ; i < str.length() ; i++) {
            char ch = str.charAt(i);
            if(ch == 'a'||ch == 'e'||ch == 'i'||ch == 'o'||ch == 'u') {
                count++;
            }  }
        System.out.println("Number of vowels: "+count);   }
}
```

Output:



```
Console: connection closed (Running: 10 seg)
Enter a string
Welcome TO Java Programming
Number of vowels: 9
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.8	Count the Number of Alphabets in the Given String
Date:	

Aim:

To write a Java program to count the number of alphabets in the given string

Algorithm:

Step-1: Start

Step-2: Create a class ‘Main’

Step-3: Initialize counters array of 256 length

Step-4: Use Scanner to get input from the user

Step-5: Iterate our string and increase count by 1 at index based on Characters

Step-6: Iterate over counter array and print character and frequency if counter [i]!=0

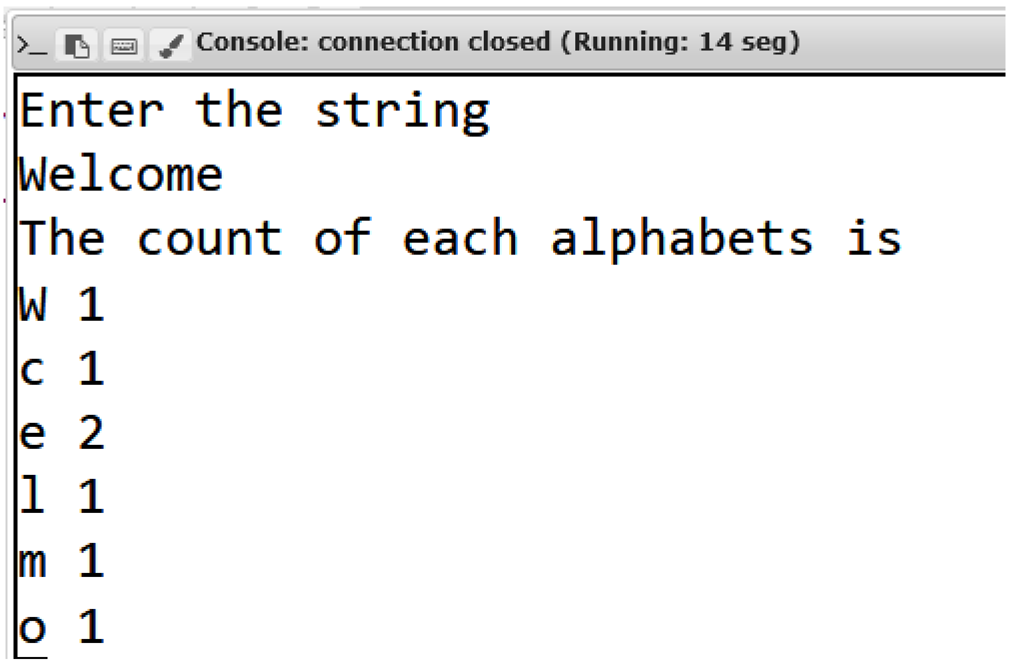
Step-7: Stop

Source Code:

```
import java.util.*;

class Main{

    public static void main (String[] args) {

        int counter[] = new int[256];

        Scanner scan = new Scanner(System.in);

        String str;

        System.out.println("Enter the string");

        str = scan.nextLine();

        int i;
```

```
            for(i = 0 ; i < str.length() ; i++)  {
                counter[(int) str.charAt(i)]++;
            }
            System.out.println("The count of each alphabets is");
            for(i = 0 ; i < 256 ; i++)  {
                if(counter[i] != 0)    {
                    System.out.println((char)i+" "+counter[i]);
                }
            }
        }       }
```

Output:



```
Enter the string
Welcome
The count of each alphabets is
W 1
c 1
e 2
l 1
m 1
o 1
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.9	Return the Num Length Encoded String for the Input String
Date:	

Aim:

To write a Java program to return the num length encoded string for the input string

Algorithm:

Step-1: Start

Step-2: Create a class 'Main'

Step-3: Use Scanner to get inputs from the user

Step-4: Get a string input (s)

Step-5: Use for loop for iterating the characters in a string and append it to the compressed string

Step-6: Declare and initialize c =1

Step-7: Count the number if occurrences of the specific character and append it to the com pressed string

Step-8: Repeat this process for all the characters write the end of the string

Step-9: Display the result

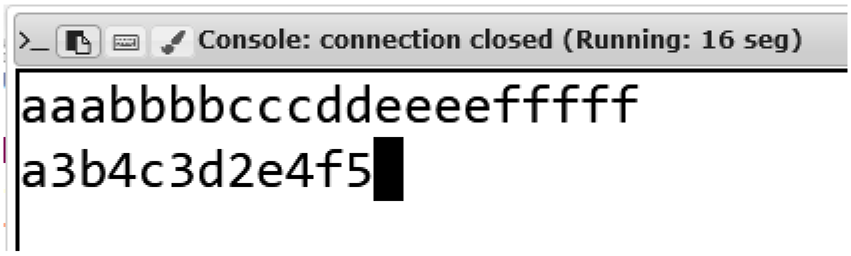
Step-10: Stop

Source Code:

```
import java.util.Scanner;
class Main
{
    public static void main (String[] args)     {
        Scanner sc = new Scanner(System.in);
```

```
                String s = sc.nextLine();
                for(int i=0;i<s.length();i++) {
                    int c=1;
                    while(i<s.length()-1 && s.charAt(i)==s.charAt(i+1))  {
                        c++;
                        i++;
                     }
                    System.out.print(s.charAt(i));
                    System.out.print(c);
                }
        }            }
```

Output:



```
aaabbbbcccddeeeefffff
a3b4c3d2e4f5
```

Result:

Thus, the above program was executed successfully.

Ex. No: 3.10	Check if the Given String is Pangram or Not
Date:	

Aim:

To write a Java program to check the given string is pangram (or) not

Algorithm:

Step 1: Start

Step 2: Create a class Main’

Step 3: Use scanner to get an input

Step 4: Get a string

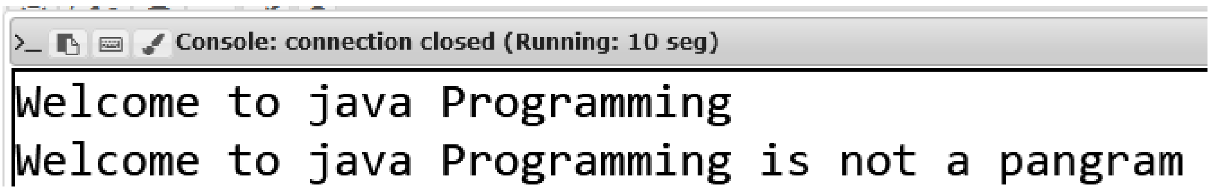
Step 5: If given string is empty, then print “Invalid Input”

Step 6: Else, Create a function named “Pangram check” and pass the string as a constructor

Source Code:

```
import java.util.*;

class Main

{

    public static boolean pangramCheck(String str)    {

        boolean[] x = new boolean[26];

        int index = 0;

        for(int i = 0; i < str.length(); i++){

            if('A' <= str.charAt(i) && str.charAt(i) <= 'Z'){

                index = str.charAt(i) - 'A';    }

            else if('a' <= str.charAt(i) && str.charAt(i) <= 'z'){

                index = str.charAt(i) - 'a';    }
            else{
             continue;     }
            x[index] = true;
        }
        for(int i=0;i<=25;i++){if(x[i] == false){
             return (false);     }     }
        return (true);
    }
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        if(s.length() == 0){
             System.out.println("Invalid Input");     }
        else {
            if(pangramCheck(s) == true)    {
             System.out.println(s + " is a pangram"); }
            else      {
             System.out.println( s + " is not a pangram"); }  }  }
```

Output:



```
Welcome to java Programming
Welcome to java Programming is not a pangram
```

Result:

Thus, the above program was executed successfully.

Module IV: Collection	
Ex. No: 4.1	**Extract Elements in the Subset of the Array**
Date:	

Aim:

To Write a Java program to extract elements in the subset of the array.

Algorithm:

Step 1: Declare and initiate BufferReader in the main method with InputStream as a parameter.

Step 2: Declare a list of string datatype.

Step 3: Get the number of inputs from the user.

Step 4: Appending the words of the sentence in an array using split method of BufferReader.

Step 5: Using the for each loop add the elements of array into the list.

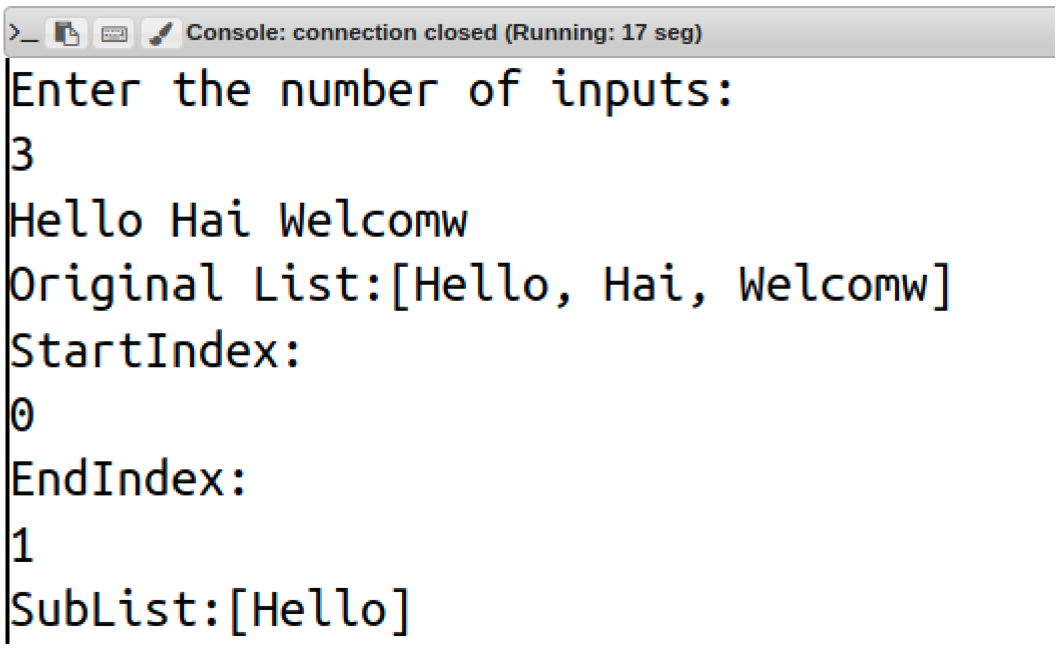
Step 6: Initially the original list is printed then a new list is declared for printing the subset.

Step 7: Get the start and end index from the user using which the subset of original list is printed.

Source Code:

```
import java.io.*;
import java.util.*;
class Binary {
    public static void main (String[] args) throws IOException {
        BufferedReader br=new BufferedReader(new
                                InputStreamReader(System.in));
        List<String> al = new ArrayList<String>();
        System.out.println("Enter the number of inputs:");
        int n = Integer.parseInt(br.readLine());

        String[] inp = br.readLine().split(" ");

        for(String s : inp)        {

            al.add(s);        }

        System.out.println("Original List:"+al);

        System.out.println("StartIndex:");

        int start = Integer.parseInt(br.readLine());

        System.out.println("EndIndex:");

        int end = Integer.parseInt(br.readLine());

        List<String> res = al.subList(start,end);

        System.out.println("SubList:"+res);    }     }
```

Output:



```
Enter the number of inputs:
3
Hello Hai Welcomw
Original List:[Hello, Hai, Welcomw]
StartIndex:
0
EndIndex:
1
SubList:[Hello]
```

Result:

Thus, the above program was executed successfully.

Ex. No: 4.2	Replace an Element of the Array List
Date:	

Aim:

To Write a Java program to replace an element of the ArrayList.

Algorithm:

Step 1: Declare an ArrayList.

Step 2: Using the scanner get the size of the ArrayList.

Step 3: Using the for loop get the elements of the ArrayList.

Step 4: Print the original ArrayList using for each loop.

Step 5: Get the position of the element to be replaced from the user.

Step 6: Get the replacement value from the user.

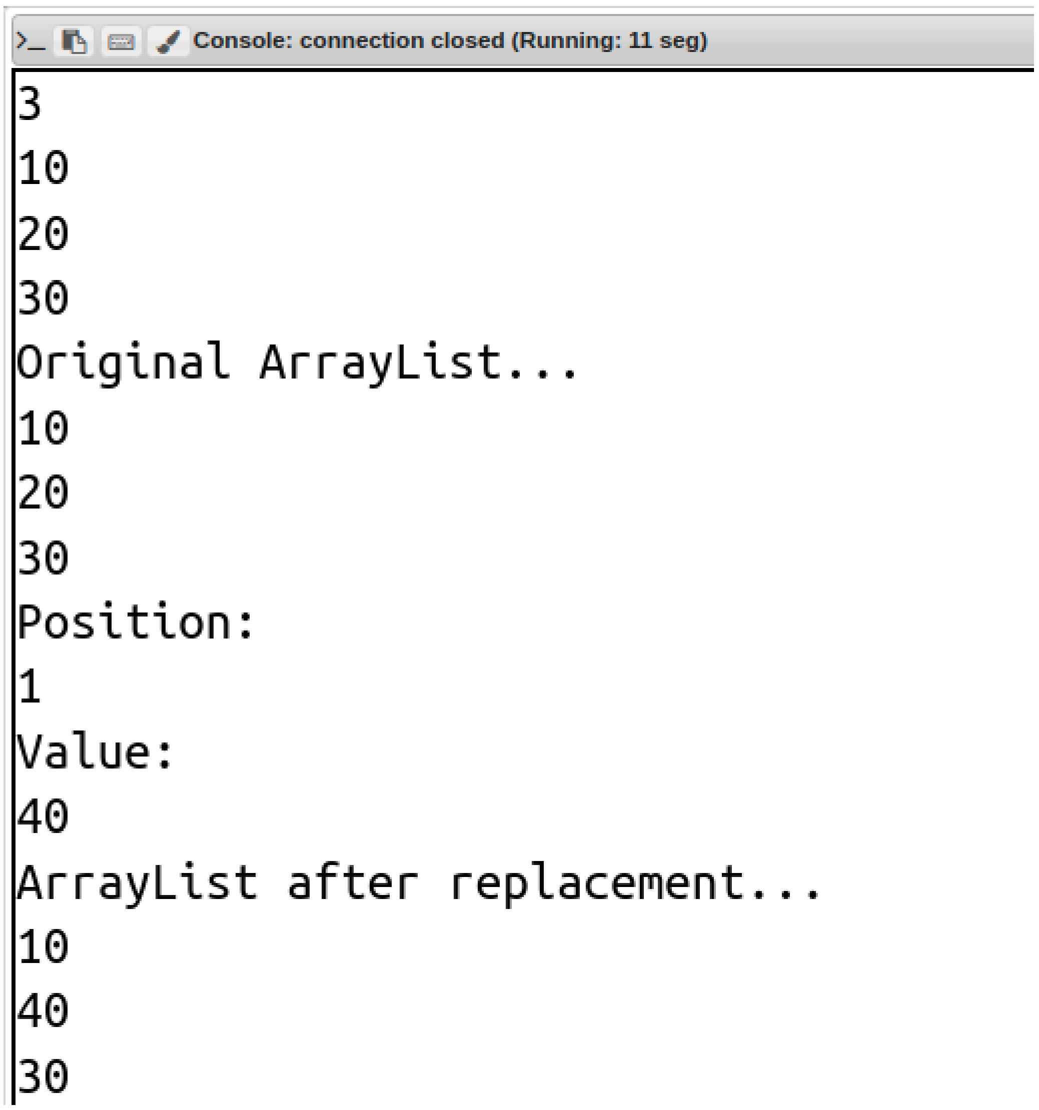
Step 7: Using set function replace the value and print the ArrayList after replacement.

Source Code:

```
import java.util.Scanner;

import java.util.List;

import java.util.ArrayList;

public class Binary

{

      public static void main(String[] args)

       {

          Scanner sc = new Scanner(System.in);

          List<Integer> al = new ArrayList<Integer>();

          int n = sc.nextInt();
```

```
            for(int i=0;i<n;i++)
            {
                al.add(sc.nextInt());
            }
            System.out.println("Original ArrayList...");
            for(Integer i : al)
            {
                System.out.println(i);
            }
            System.out.println("Position:");
            int pos = sc.nextInt();
            System.out.println("Value:");
            int val = sc.nextInt();
            al.set(pos,val);
            System.out.println("ArrayList after replacement...");
            for(Integer I : al)
            {
                System.out.println(I);
            }
        }
    }
```

Output:



```
3
10
20
30
Original ArrayList...
10
20
30
Position:
1
Value:
40
ArrayList after replacement...
10
40
30
```

Result:

Thus, the above program was executed successfully.

Ex. No: 4.3	Find the Index of the Specified Element Using List
Date:	

Aim:

To Write a Java program to find the index of the specified element.

Algorithm:

Step 1: Declare an ArrayList of integer datatype.

Step 2: Get the size of the ArrayList from the user.

Step 3: Get the elements of the ArrayList using add function in for loop.

Step 4: Get the key value from the user to find it’s index.

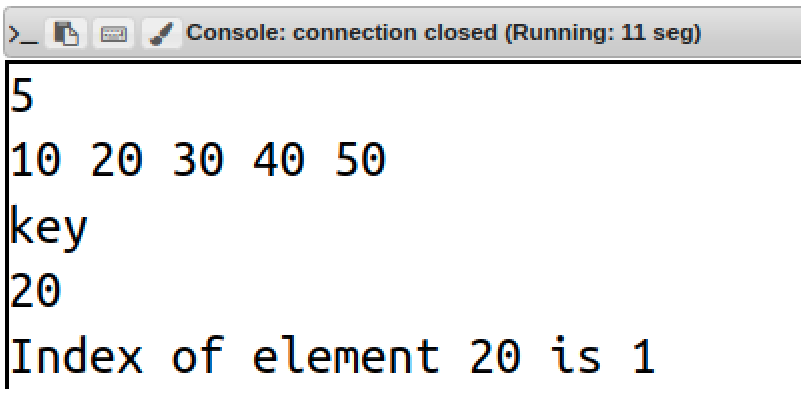
Step 5: Using the indexOf function the index of the specified element is found.

Step 6: If the specified element is not found, ”Element not found” message is displayed.

Source Code:

```
import java.util.Scanner;

import java.util.List;

import java.util.ArrayList;

class Binary

{

    public static void main (String[] args)

     {

        Scanner scanner = new Scanner(System.in);

        int n = scanner.nextInt();

        List<Integer> al = new ArrayList<Integer>();

        for(int index=0; index<n; index++)   {
            al.add(scanner.nextInt());
        }
        System.out.println("key");
        int key = scanner.nextInt();
        if(al.indexOf(key)>=0)     {
            System.out.println("Index of element "+key+" is
                        "+al.indexOf(key));
        }
        else {
            System.out.println("Element is not found in the list");
        }           }        }
```

Output:



```
5
10 20 30 40 50
key
20
Index of element 20 is 1
```

Result:

Thus, the above program was executed successfully

Ex. No: 4.4	Student Information Application Using Array List
Date:	

Aim:

To create an application which stores a student information.

Algorithm:

Step 1: Create Java class with names Student and Binary.

Step 2: In Student class declare three variables with private modifiers in which two are of string datatype and one as int datatype.

Step 3: Generate Setter method for all three variables.

Step 4: Create a void method as printStudent with Student as parameter where the rollnum, age and name are printed.

Step 5: Create another method for Student class named as getStudent.

Step 6: Declare and initiate the student class and get the values of roll,name,age.

Step 7: Set the values of rollno,name,age using student object.

Step 8: Return the student object.

Step 9: In Binary class create a main method.

Step 10: Declare and initiate an ArrayList and get the number of students from the user.

Step 11: Using for loop set the values for rollno,name and age.

Step 12: Using for each loop the details of students is printed by calling printStudent method.

Source Code:

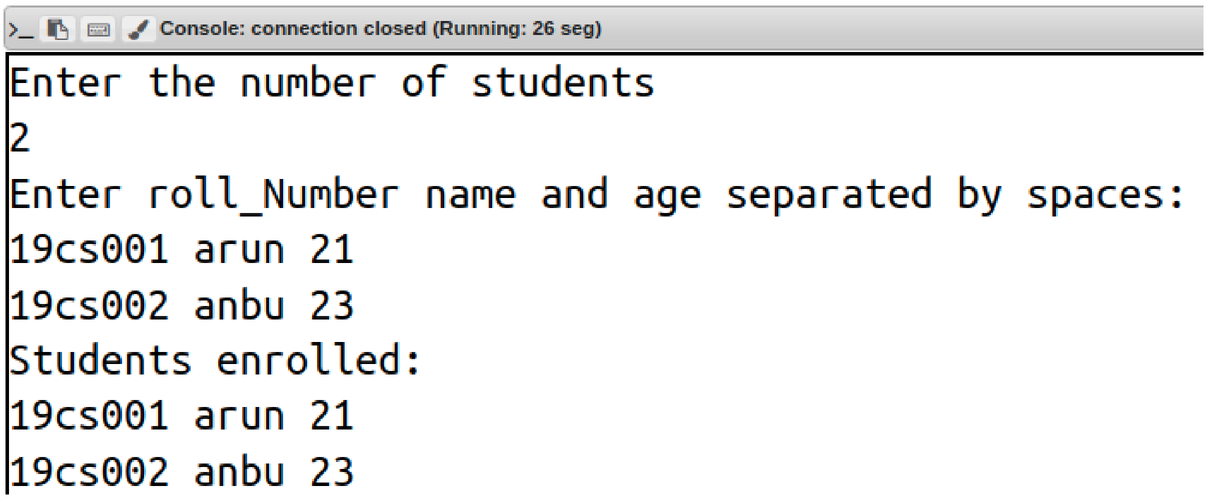
Student.java

```
import java.util.Scanner;
class Student   {
    private String roll_Number;
    private String name;
    private int age;
    public void setRoll(String roll)  {
        this.roll_Number=roll;
    }
    public void setName(String name)  {
         this.name=name;
    }
    public void setAge(int age)  {
        this.age=age;
    }
       public void printStudent(Student s) {
        System.out.println(s.roll_Number+" "+s.name+" "+s.age);
    }
    public Student getStudent()  {
        Scanner obj=new Scanner(System.in);
        Student s = new Student();
        String a=obj.next(),b=obj.next();
        int c=obj.nextInt();
        s.setRoll(a);
        s.setName(b);
        s.setAge(c);
        return s;
    }
}
```

Binary.Java

```
import java.util.*;

class Binary    {

    public static void main (String[] args) {

        Scanner obj=new Scanner(System.in);

        ArrayList<Student> students=new ArrayList<Student>();

        Student s=new Student();

        System.out.println("Enter the number of students");

        int n=obj.nextInt();

        System.out.println("Enter roll_Number name and age

                                   separated by spaces:");

        for(int i=0;i<n;++i)    {

            Student s1=new Student();

            String a=obj.next(),b=obj.next();

            int c=obj.nextInt();

            s1.setRoll(a);

            s1.setName(b);

            s1.setAge(c);

            students.add(s1);

        }

        System.out.println("Students enrolled:");

        for(Student i:students)

        s.printStudent(i);

    }

}
```

Output:



```
Enter the number of students
2
Enter roll_Number name and age separated by spaces:
19cs001 arun 21
19cs002 anbu 23
Students enrolled:
19cs001 arun 21
19cs002 anbu 23
```

Result:

Thus, the above program was executed successfully

Ex. No: 4.5	Find the First Repeating Character in the Given String using Set
Date:	

Aim:

To write a Java program to find first repeating character in the given string using Java program.

Algorithm:

Step 1: Get the input using scanner class.

Step 2: Create an object for LinkedHashSet.

Step 3: Find the length of the String

Step 4: Using for loop add the characters to the string and using if condition check for repeating character, if a character is repeated print the character and break the loop.

Step 5: If no character is repeated then add the string characters to the Set.

Step 6: Then print “there is no repeating characters in the string”.

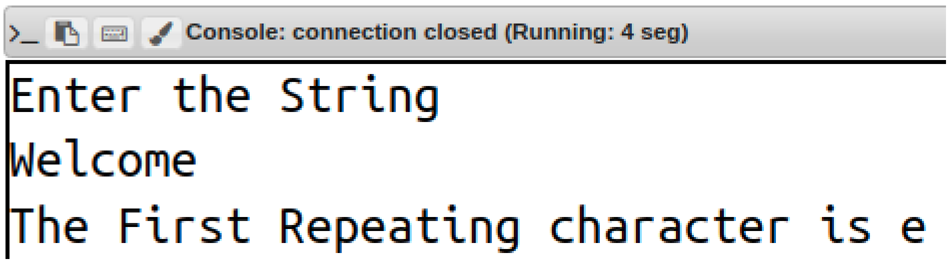
Step 7: Stop

Source Code:

```
import java.util.Scanner;
import java.util.Set;
import java.util.LinkedHashSet;
class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the String");
        String str = sc.next();
        Set<Character> set = new LinkedHashSet<Character>();
```

```
            int len = str.length();
            int i=0;
            for(i=0;i<len;i++)    {
                if(!(set.add(str.charAt(i))))    {
                    System.out.println("The First Repeating character
                                                            is "+str.charAt(i));
                    break;    }
               else  {
                    set.add(str.charAt(i));   }
            }
            if(i == len)    {
                System.out.println("There is no Repeating Character in
                                                                 the given string");
            }
          }
    }
```

Output:



```
Enter the String
Welcome
The First Repeating character is e
```

Result:

Thus, the program is executed successfully.

Ex. No: 4.6	Search a String in a Paragraph Using Set
Date:	

Aim:

To search the given string in the paragraph using Java program.

Algorithm:

Step 1: Get the input using scanner class.

Step 2: Create an object for HashSet.

Step 3: Declare a string array.

Step 4: Using for each loop add the string to the HashSet.

Step 5: Using scanner class get the string which is to be searched.

Step 6: Using contains function check whether the string is present in the Set or not.If present then print the string.

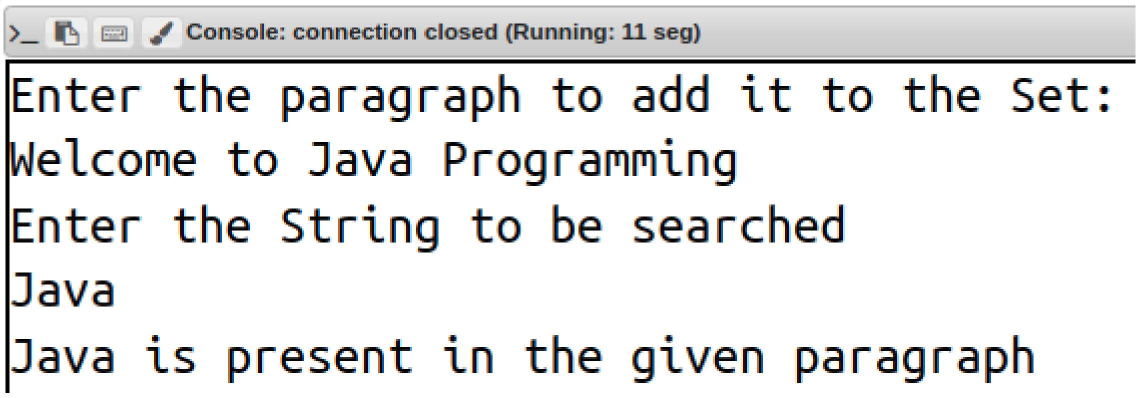
Step 7: Stop

Source Code:

```
import java.util.*;
class Main      {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        Set<String> hs = new HashSet<String>();
        System.out.println("Enter the paragraph to add it to the
                                                     Set:");
        String[] inp = sc.nextLine().split(" ");
        for(String str : inp)   {
```

```
                        hs.add(str);
                    }
                    System.out.println("Enter the String to be searched");
                    String str1=sc.nextLine();
                    if(hs.contains(str1))
                    System.out.println(str1+" is present in the given
                                                                paragraph");
                    else
                       System.out.println(str1+" is not present in the given
                                                                paragraph");
                }
            }
```

Output:



```
Enter the paragraph to add it to the Set:
Welcome to Java Programming
Enter the String to be searched
Java
Java is present in the given paragraph
```

Result:

Thus, the program is executed successfully.

Ex. No: 4.7	Add the Student Object to a Linked Hash Set
Date:	

Aim:

To Write a Java Program to add the student object to a linked hashset.

Algorithm:

Step 1: Get the input using scanner class.

Step 2: Create an object for HashSet.

Step 3: Declare a string array.

Step 4: Using for each loop add the string to the HashSet.

Step 5: Using scanner class get the string which is to be searched.

Step 6: Using contains function check whether the string is present in the Set or not.If present then print the string.

Step 7: Stop

Source Code:

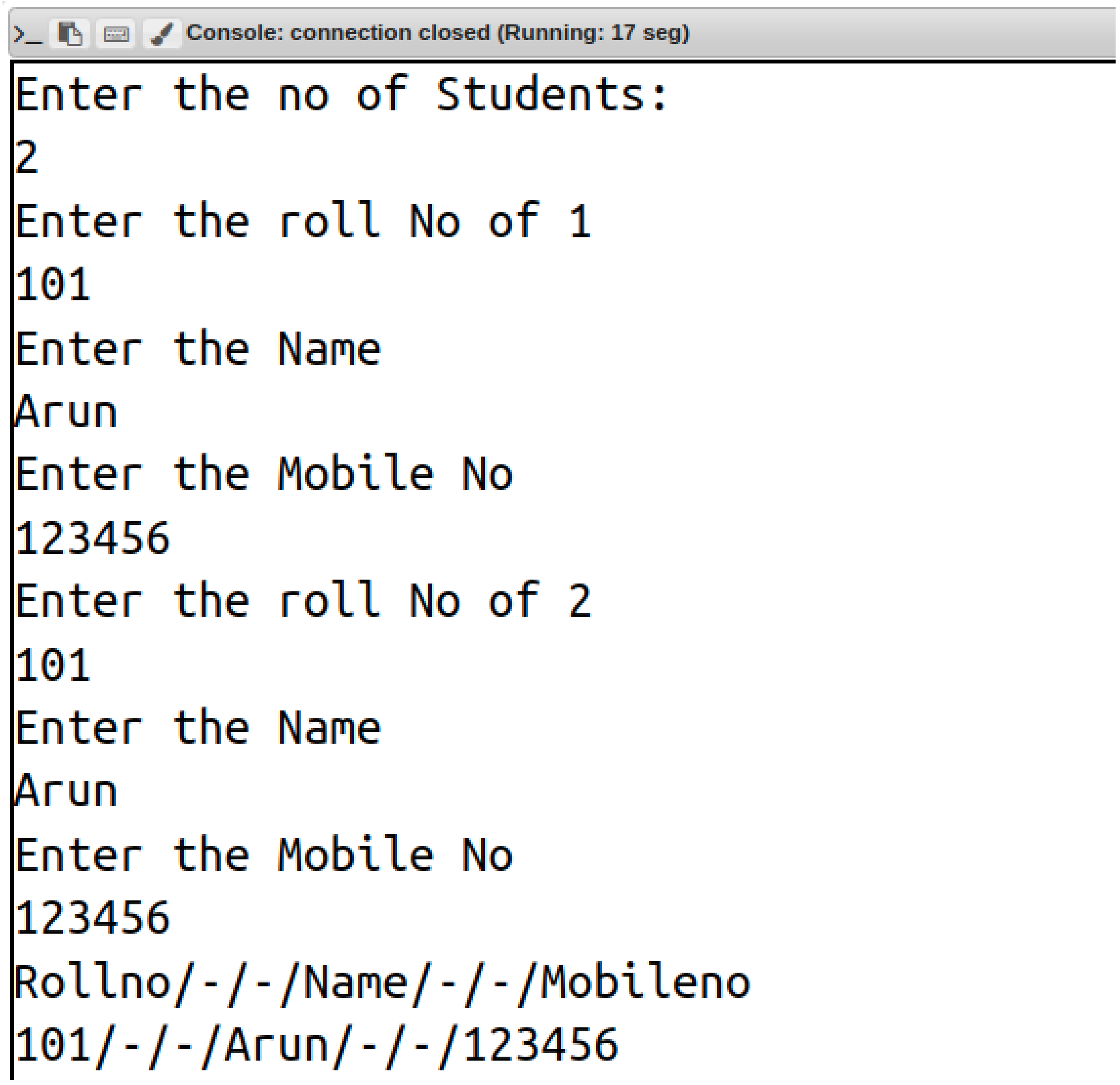
Main.java

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n;
        System.out.println("Enter the no of Students:");
}

        n = sc.nextInt();

        sc.nextLine();

        Student student=null;

        HashSet<Student> hs = new LinkedHashSet<Student>();

        for(int i=1;i<=n;i++)

        {

            String rno;

            String nam;

            String mobile;

            System.out.println("Enter the roll No of "+i);

            rno=sc.nextLine();

            System.out.println("Enter the Name");

            nam=sc.nextLine();

            System.out.println("Enter the Mobile No");

            mobile=sc.nextLine();

           student = new Student(Integer.parseInt(rno),nam,mobile);

            hs.add(student);

        }

        System.out.println("Rollno/-/-/Name/-/-/Mobileno");

        for(Student i:hs)

        {

            System.out.println(i);

        }

    }

}
```

Student.java

```
import java.util.*;
public class Student
{
    private Integer rollNo;
    private String name;
    private String mobileNo;
    Student(Integer rollNo,String name,String mobileNo)
    {
        this.rollNo=rollNo;
        this.name=name;
        this.mobileNo=mobileNo;
    }
    public void setRollNo(Integer rollNo)
    {
        this.rollNo = rollNo;
    }
    public Integer getRollNumber()
    {
        return this.rollNo;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public String getName()
    {
        return this.name;
    }
    public void setMobileNo(String mobileNo)
    {
        this.mobileNo = mobileNo;
    }
    public String getMobileNo()
    {
        return this.mobileNo;
    }
    @Override
    public String toString()
    {
     return  getRollNumber()+"/-/-/"+getName()+"/-/-/"+getMobileNo();
    }
    @Override
    public boolean equals(Object obj)
    {
        Student s = (Student)obj;
        return this.rollNo.equals(s.rollNo);
    }
    @Override
    public int hashCode()    {
        return this.rollNo;
    }       }
```

Output:



Console: connection closed (Running: 17 seg)

```
Enter the no of Students:
2
Enter the roll No of 1
101
Enter the Name
Arun
Enter the Mobile No
123456
Enter the roll No of 2
101
Enter the Name
Arun
Enter the Mobile No
123456
Rollno/-/-/Name/-/-/Mobileno
101/-/-/Arun/-/-/123456
```

Result:

Thus, the program is executed successfully.

Ex. No: 4.8	Find the Frequency in a String Using Linked Hash Map
Date:	

Aim:

To write a Java program to get a string from user and find the frequency in a string

Algorithm:

Step 1: Start

Step 2: Import Java util packages and create a class named as "Solution" with static method named as "frequency" with return type void with String and Integer as arguments(str,n)

Step 3: Create a reference for Map class (Character,Integer==generics) and name it as "map"

Step 4: Construct a For loop and initialize index as 0 and index is less than n and increment index by 1

Step 5: Initialize temporary variable count as 0

Step 6: Check If map contains a character key using containsKey method using character as parameter (map.containsKey(str.charAt(index)))

Step 7: If Step 6 is true then get the value in the corresponding key(character) using "get method" in map and store it in count and put the set(key and value) in map using "put" method

Step 8: If Step 6 is False put the key(character) and value as 1

Step 9: Continue Step-4 until the loop ends

Step 10: print "Character Frequency"

Step 11: Construct a foreach loop and print key and value in the map

Step 12: Construct a main method and create a reference for scanner class

Step 13: print "Enter the String" and get a string from user as input and store it in str

Step 14: Get a length of the string using length() method and store it in len variable

Step 15: Call the frequency method with 'str' & 'len' as parameters

Step 16: End

Source Code:

```
import java.util.Scanner;

import java.util.Map;

import java.util.LinkedHashMap;

class Solution

{

    static void frequency(String str,int n)

     {

        Map<Character,Integer>map= new

                                LinkedHashMap<Character,Integer>();

        for(int index=0;index<n;index++)   {

            int count =0;

            if(map.containsKey(str.charAt(index)))    {

                count = map.get(str.charAt(index));

                map.put(str.charAt(index),++count);

            }

           else  {

                map.put(str.charAt(index),1);

            }

        }

        System.out.println("Character Frequency:");

        for(Map.Entry<Character,Integer> entry : map.entrySet()){


           System.out.println(entry.getKey()+":"+entry.getValue());
```

```
            }
        }
        public static void main (String[] args)
       {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the String:");
            String str = sc.next();
            int len = str.length();
            frequency(str,len);
        }
    }
```

Output:

```
Character  Frequency:
h:1
e:1
l:2
o:1
```

Result:

Thus, the above program was executed successfully.

Ex. No: 4.9	Find Duplicate Element in String Using Linked Hash Map
Date:	

Aim:

To write a Java program to get a String there is one character which appears twice.

Algorithm:

Step 1: Start

Step 2: Import utility package and class named as solution.

Step 3: Create a static method named findDuplicate with return type character and parameter as string.

Step 4: Create a Map class and named it as map.

Step 5: Find the length of the string and stored it in the variable named as len using length method.

Step 6: Construct a for loop and initialize i as 0 untill i less than len and increment i to 1

Step 7: Initialize temporary count variable as 0.

Step 8: Check if map contains a character using containsKey method.

Step 9: If the condition is true assign the value of the specific key character to the variable count and increment the count variable and store it in map using put method.

Step 10: If STEP 8 is false store the value of the particular variable as 1 using put method.

Step 11: Initialize character variable result as null.

Step 12: Construct a for each loop using for map with reference as entry.

Step 13: Check if the value of the particular variable is greater than 1 using get method. If the condition is true initialize the particular character to the variable result.

Step 14: Break the for each loop using break statement and return the character.

Step 15: Construct a main method and define the scanner class with reference as sc to get input from the user.

Step 16: Print "Enter the String:" and get input from the string.

Step 17: Print the Duplicate character in the string.

Step 18: STOP.

Source Code:

```
    import java.util.*;

    class Solution

    {

         static char  findDuplicate(String str)

    {

            Map<Character,Integer> map = new

                                    LinkedHashMap<Character,Integer>();

            int len = str.length();

            for(int i=0;i<len;i++)  {

                int count=0;

                if(map.containsKey(str.charAt(i)))  {

                    count = map.get(str.charAt(i));

                    map.put(str.charAt(i),++count);

                }

               else {

                    map.put(str.charAt(i),1);

                }

            }

            char result='\0';

            for(Map.Entry<Character,Integer> entry : map.entrySet()){

                if(entry.getValue()>1) {
                    result = entry.getKey();
                    break;
                }
            }
            return result;
        }
        public static void main (String[] args)
         {
                Scanner sc = new Scanner(System.in);
               System.out.println("Enter the String:");
               String input = sc.next();
               System.out.println("Duplicate Character: " +
                                                  findDuplicate(input));
        }
    }
```

Output:

```
Enter the String: Hello
Duplicate Character: l
```

Result:

Thus, the above program was executed successfully.

Ex. No: 4.10	**Find the Character that Appeared Maximum Number of Time Using Linked Hash Map**
Date:	

Aim:

To write a Java program to find the character which appeared the maximum time.

Algorithm:

Step 1: START

Step 2: Import utility package and interface named as QuestionInterface and declare the method majorityElement with parameter string.

Step 3: Construct the class named as Answer and implement the QuestionInterface and override the above declared method in interface.

Step 4: Define the method majorityElement with return type character and parameter string. Create a map class with reference map.

Step 5: Calculate the length of the string and stored it in the len variable using length method.

Step 6: Construct a for loop and initialize i as 0 until i less than len and increment i to 1.

Step 7: Initialize temporary count variable as 0.

Step 8: Check if map contains a particular character using containsKey method.

Step 9: If the condition is true, assign a value of the specific key character to the variable count and increment count variable and store it in the map using put method.

Step 10: If the STEP 8 is false, store the value of the particular variable as 1 using put method.

Step 11: Calculate the maximum values of specific key character using max method in collection and assign it to the integer variable max.

Step 12: Initialize the character variable result as 0.

Step 13: Construct a for each loop using for map with reference as entry.

Step 14: If the value of the particular is equal to the max value the condition become true and particular key character get stored in the result variable.

Step 15: Break the loop using Break statement and return the character.

Step 16: Construct a main method and define the reference for the Answer class

Step 17: Define the scanner class with reference as scan to get input from the user.

Step 18: Print the majority first occurred element in the string.

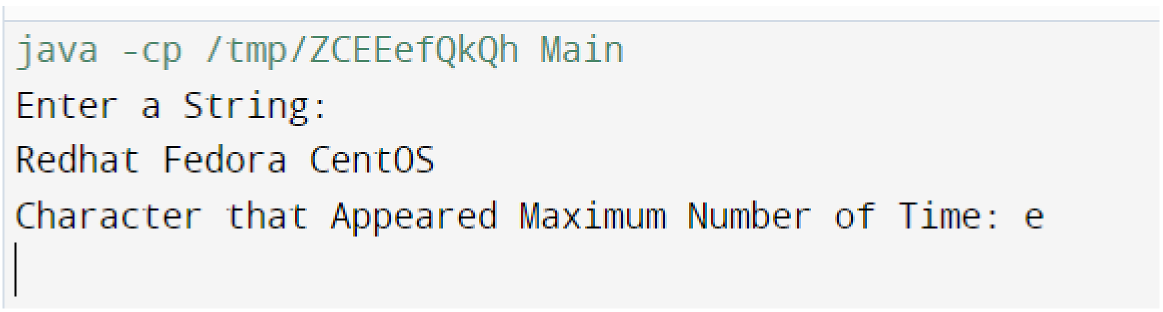
Step 19: STOP.

Source Code:

```
import java.util.Map;
import java.util.LinkedHashMap;
import java.util.Collections;
import java.util.Scanner;
class Main {
    public static void main(String[] args)
    {
        Answer obj = new Answer();
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a String: ");
        String str=scan.nextLine();
        System.out.println("Character that Appeared Maximum
                    Number of Time: " + obj.majorityElement(str));
    }
}
interface QuestionInterface     {
    char majorityElement(String str);     }
```

```
class Answer implements QuestionInterface  {
    @Override
    public char majorityElement(String str)
     {
        Map<Character,Integer> map = new
                                LinkedHashMap<Character,Integer>();
        int len = str.length();
        for(int index = 0;index<len;index++)    {
            int count = 0;
            if(map.containsKey(str.charAt(index)))    {
                count = map.get(str.charAt(index));
                map.put(str.charAt(index),++count);
            }
           else {
                map.put(str.charAt(index),1);
            }
        }
        int max = Collections.max(map.values());
        char result ='\0';
        for(Map.Entry<Character,Integer> entry : map.entrySet()) {
            if(entry.getValue()==max) {
                result = entry.getKey();
                break;          }
     }
        return result;    }
}
```

Output:



```
java -cp /tmp/ZCEEefQkQh Main
Enter a String:
Redhat Fedora CentOS
Character that Appeared Maximum Number of Time: e
```

Result:

Thus, the above program was executed successfully.

Ex. No: 4.11	**Find the Common Characters Between Two Given Strings Using Linked Hash Map**
Date:	

Aim:

To write a Java program to Find the Common Characters between two given string. Return the string which has character which appears in both string .

Algorithm:

Step 1: START

Step 2: Import Java util packages and create an interface named as QuestionInterface with method intersection with String as return type and 2 Strings as arguments

Step 3: Create a class Answer which implements QuestionInterface

Step 4: Override the interface method intersection

Step 5: Initialize a String res and assign none("") to it

Step 6: Create a reference for map and name it as hm with generics as Character(key) and integer(value)

Step 7: Construct a for-each loop for input2(which we will get from user and pass as argument)

Step 8: Check if value present in map(hm) or not using containskey() method

Step 9: If Step-8 is True than put that character and value of the character in map with 1 increment

Step 10: If Step-9 is False than put that character as key and value as 1

Step 11: Continue Step-8 until the for-each loop ends

Step 12: Construct an another foreach loop with input1(which we will get from user and pass as argument) as parameter

Step 13: Check if value present in map(hm) or not using containskey() method

Step 14: If Step 13 is True than append the character in res variable and put that character with value as character with decrement by 1

Step 15: Check if value of character is 0 or not

Step 16: If Step 15 is True than remove character from hm(map) and return res

Step 17: Construct a main class and a main method and create a reference for scanner class

Step 18: Create a reference for Answer class

Step 19: Get 2 String inputs from the User and pass those String in intersection method and print the desired output

Step 20: END

Source Code:

```
import java.util.*;
class Main
{
	public static void main(String[] args)
	{
		Answer obj = new Answer();
		Scanner scan = new Scanner(System.in);
		System.out.print(“String1:”);
		String str=scan.nextLine();
		System.out.print(“String2:”);
		String str2=scan.nextLine();
		System.out.println(“Common Characters are: “ +
		                                obj.intersection(str,str2));
	}
}
 interface QuestionInterface    {
```

```
          String intersection(String str,String str2);     }
     class Answer implements QuestionInterface  {
         @Override
         public String intersection(String input1,String input2)
         {
         String res="";
         Map<Character,Integer> hm = new HashMap<Character,Integer>();
         for(Character ch:input2.toCharArray())    {
             if(hm.containsKey(ch))        {
                 hm.put(ch,hm.get(ch)+1);
             }
             else     {
                 hm.put(ch,1);
             }
         }
         for(Character ch:input1.toCharArray())    {
             if(hm.containsKey(ch))        {
                 res+=ch;
                 hm.put(ch,hm.get(ch)-1);
                 if(hm.get(ch)==0)         {
                 hm.remove(ch);
                 }        }
        }
         return res;
     }      }
```

Output:

```
String1: Hello

String2: Halo

Common Characters are: lo
```

Result:

Thus, the above program was executed successfully.

Module V: Exception Handling	
Ex. No: 5.1	**Exception Handling Mechanism**
Date:	

Aim:

To write a Java program to compute x/y and check If x and y are not 32 bit signed integers or if x is zero, exception will occur and it has to be handled using try/catch mechanism.

Algorithm:

Step 1: Start

Step 2: Create a class(your file name) and main method

Step 3: Create a reference for the Scanner class to get inputs from the user

Step 4: Initialize 2 variables(integer type) as x,y or (any)

Step 5: Create a try-catch block

Step 6: Using scanner reference get an input from the user for dividend and store it in a variable (x)

Step 7: Repeat **Step 4** get input and store for divisor in a variable (y)

Step 8: In the try-block try to print the value of x/y

Step 9: If there is "InputMismatchException" error catch this exception in catch block and print the InputMismatchException

Step 10: If there is an "ArithmeticException" error catch this exception in catch block and print the ArithmeticException

Step 11: If there are other than these two exceptions then print out the specified exception

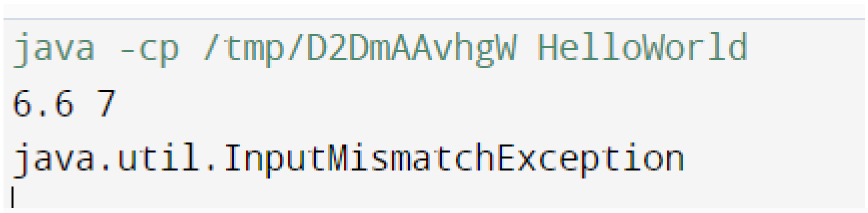
Step 12: End

}

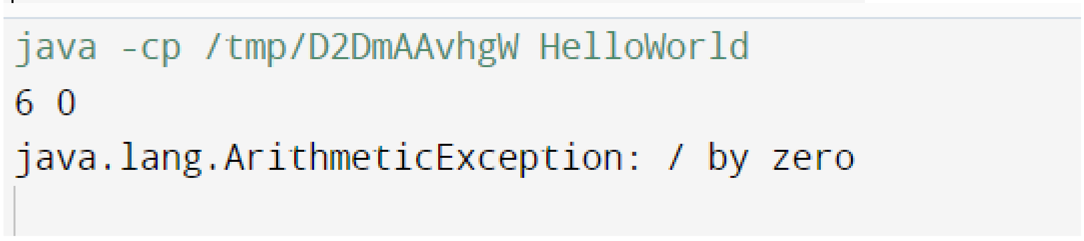
Source Code:

```
import java.io.*;

import java.util.*;

class Main {

    public static void main (String[] args)

     {

        Scanner sc = new Scanner(System.in);

        int x,y;

        try {

            x=sc.nextInt();

            y=sc.nextInt();

            System.out.println(x/y);

        }

         catch(InputMismatchException ie)   {

            System.out.println("java.util.InputMismatchException");

        }

       catch(ArithmeticException ae)     {

            System.out.println(ae);

        }

       catch(Exception e)       {

            System.out.println(e);

        }

        sc.close();

   }

}
```

Output:



```
java -cp /tmp/D2DmAAvhgW HelloWorld
6.6 7
java.util.InputMismatchException
```



```
java -cp /tmp/D2DmAAvhgW HelloWorld
6 0
java.lang.ArithmeticException: / by zero
```

Result:

Thus, the above program was executed successfully.

Ex. No: 5.2	Custom Exception Handling
Date:	

Aim:

To Write a Java program to create A class MyCalculator consists of a single method power(int,int). This method takes two integers, n and p; as parameters and finds np .If either n or p is negative, then the method must throw an exception which says "n and p should be non-negative".

Algorithm:

Step 1: Start

Step 2: Create a class named as MyCalculator with a method as power with int as its return type with two int parameters (i.e n,p) as arguments

Step 3: Assign a temporary variable res and assign a value 1 (i.e res=1)

Step 4: Construct a FOR loop with i=1 I less than or equal to p(limit) and increment i

Step 5: multiply res with n (res*=n)

Step 6: if i<=p continue Step-5 else goto Step-7

Step 7: return res

Step 8: Import scanner class and create a class(your file name) and main method

Step 9: Create an reference for scanner class to get inputs from the user

Step 10: Initialize 2 variables(integer type) as n,p or (any)

Step 11: Create a reference(mc) for MyCalculator class

Step 12: Using try block try to get integer inputs from user and store them in n and p

Step 13: Check if both the integer inputs are greater the 0 if not throw exception as "n and p should be non-negative"

Step 14: If both the inputs are not integer using catch block throw the specified exception to the user

Step 15: If both the inputs are greater than 0 call the power method with reference of MyCalculator i.e mc with n,p as parameters

Step 16: Print the desired output

Step 17: End

__**Source Code:**__

```
import java.util.Scanner;

class Main

{

public static void main (String[] args)

{

        Scanner sc = new Scanner(System.in);

        int n,p;

        MyCalculator mc = new MyCalculator();

        try {

            n=sc.nextInt();

            p=sc.nextInt();

            if(n<0 || p<0) {

                throw new Exception("n and p should be non-negative");

            }

            System.out.println(mc.power(n,p));

        }

       catch(Exception e) {

            System.out.println(e);

        }

       }        }
```

```
class MyCalculator

{

    public int power(int n,int p)

     {

        int res=1;

        for(int i=1;i<=p;i++)   {

            res *= n;

        }

        return res;

    }

}
```

Output:

```
java.lang.Exception: n and p should be non-negative
```

Result:

Thus, the above program was executed successfully.

Ex. No: 5.3	Exception Handling in Array
Date:	

Aim:

To Write a Java program to get an array ,print the array index element for the specified index. If the index value is incorrect return the exception class related to it

Algorithm:

Step 1: Import Java scanner class and Create a class(your file name) and main method

Step 2: Create a reference for scanner class to get inputs from the user

Step 3: Print "Enter the size of the array" and get size of array and store it in int n

Step 4: Print "Enter the array elements"

Step 5: Create an array as arr with size as n

Step 6: Using for loop get inputs from user and store it in array(arr) limit for the loop is less than n

Step 7: Print "Enter the specific index:"

Step 8: Using try block get an input (integer) from user and store it in "a" for index

Step 9: Try to print the desired integer in the specified index(i.e a)

Step 10: If there is any exceptions in the program then catch that exception using catch block and print out the exception which you have got!

Step 11: End

Source Code:

```
class Main {
    public static void main (String[] args) {
        Scanner scanner=new Scanner(System.in);
        System.out.println("Enter the size of the array:");
        int n=scanner.nextInt();

        System.out.println("Enter the array elements:");

        int[] arr=new int[n];

        for(int i=0;i<n;i++)    {

            arr[i]=scanner.nextInt();  }

        System.out.println("Enter the specific index:");

        Try     {

            int a=scanner.nextInt();

            System.out.println("The value at the specified index is

                                "+arr[a]);

        }

        catch(ArrayIndexOutOfBoundsException ae)     {

            System.out.println(ae);    }    }

}
```

Output:

```
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
```

Result:

Thus, the above program was executed successfully.

Ex. No: 5.4	Arithmetic Exception with Finally
Date:	

Aim:

To Write a Java program to execute the finally block in the exception handling and check the Arithmetic Exception.

Algorithm:

Step 1: START

Step 2: Construct the scanner class with reference as sc.

Step 3: Initialize and get the value for the variable x and y inside a try block.

Step 4: Also in try block, the division operation is performed by dividing x and y and if the value of y is zero (or) inappropriate, then the program flow will be passed onto catch block.

Step 5: Catch block will specify the type exception occurred in try block and display in the output.

Step 6: Then, the finally block will be explicitly displayed in the output - "Finally block is always executed".

Step 7: STOP.

Source Code:

```
import java.util.*;
import java.io.*;
class Main
{
    public static void main(String args[])      {
        Scanner sc = new Scanner(System.in);
        int x,y;
```

```
            try     {
                System.out.println("Enter a number");
                x = sc.nextInt();
                System.out.println("Enter the divident");
                y = sc.nextInt();
                System.out.println(x/y);
            }
            catch(ArithmeticException ae)       {
                System.out.println(ae);
            }
            finally        {
                System.out.println("finally block is always executed");
            }
        }
    }
```

Output:

```
java.lang.ArithmeticException: / by zero
finally block is always executed
```

Result:

Thus, the above program was executed successfully.

Ex. No: 5.5	Number Format Exception
Date:	

Aim:

To write a Java program to Get the input String from user and parse it to integer, if it is not a number it will throw number format exception Catch it and print "Entered input is not a valid format for an integer." or else print the square of that number.

Algorithm:

Step 1: START

Step 2: Construct the scanner class with reference as sc.

Step 3: Initialize the integer variable as i and get the input from the user inside the try block.

Step 4: Inside a try block, if i is greater than zero, it prints the squaring the value of i and the statement - “The work has been done successfully” will also be displayed.

Step 5: If the STEP 4 is false, the number format exception will be thrown to catch block.

Step 6: Then, the catch block will display as “Entered input is not a valid format for an integer”.

Step 7: STOP.

Source Code:

```
import java.util.*;
class Main{
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int i;
        try   {
            System.out.println("Enter an integer:");
            i=sc.nextInt();
            if(i>0)   {
                System.out.println("The square value is "+i*i);
                System.out.println("The work has been done successfully");
            }
            else    {
                throw new NumberFormatException();
            }        }
            catch(Exception e)       {
            System.out.println("Entered input is not a valid format for
                                                               an integer");
        }       }    }
```

Output:

```
The square value is 144
The work has been done successfully.
```

Result:

Thus, the above program was executed successfully.

Ex. No:5.6	Custom Exception
Date:	

Aim:

To write a Java program to create a custom exception.

Algorithm:

```
Step 1: START
Step 2: Initialize the string variable as s and get the input from user.
Step 3: Inside try block, check ‘s’ equals to “c++” using equalsIgnoreCase method and if the
        condition is true, throw an exception to catch block.
Step 4: Then, the catch block will be executed and prints “Inside the catch block”.
Step 5: If the STEP 3 is false, it prints “No Exception”.
Step 6: It print the statement string value with entered string outside the try-catch block.
Step 7: STOP.
```

Source Code:

```java
import java.util.*;
import java.util.*;
class Main {
 public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     System.out.println("Enter a string :");
     String s = sc.nextLine();
     try {
         if(s.equalsIgnoreCase("c++"))
```

```
                {
                    throw new Exception();
                }
                else
                {
                System.out.println("No Exception.");
                }
            }
            catch(Exception e)
            {
                System.out.println("Inside catch block.");
            }
            System.out.println("String val is "+s);
        }
    }
```

Output:

```
Inside catch block.
String val is C++
```

Result:

Thus, the above program was executed successfully.

Module VI: Threads	
Ex. No:6.1	**Implementation of Multi-Threading in Java**
Date:	

Aim:

To write a Java program to implement multi-threading concept

Algorithm:

Step 1: START

Step 2: Define two classes extending the Thread class.

Step 3: Inside the first class created, override the run() method by printing "Thread 1" five times with a 5-millisecond sleep between each iteration.

Step 4: Inside the second class created, overrides the run() method by printing "Thread 2" five times with a 5-millisecond sleep between each iteration.

Step 5: Define the Thread_Priority_Sleep class.

Step 6: In the main method of the Thread_Priority_Sleep class, An instance of Class1 (obj1) and an instance of Class2 (obj2) are created.

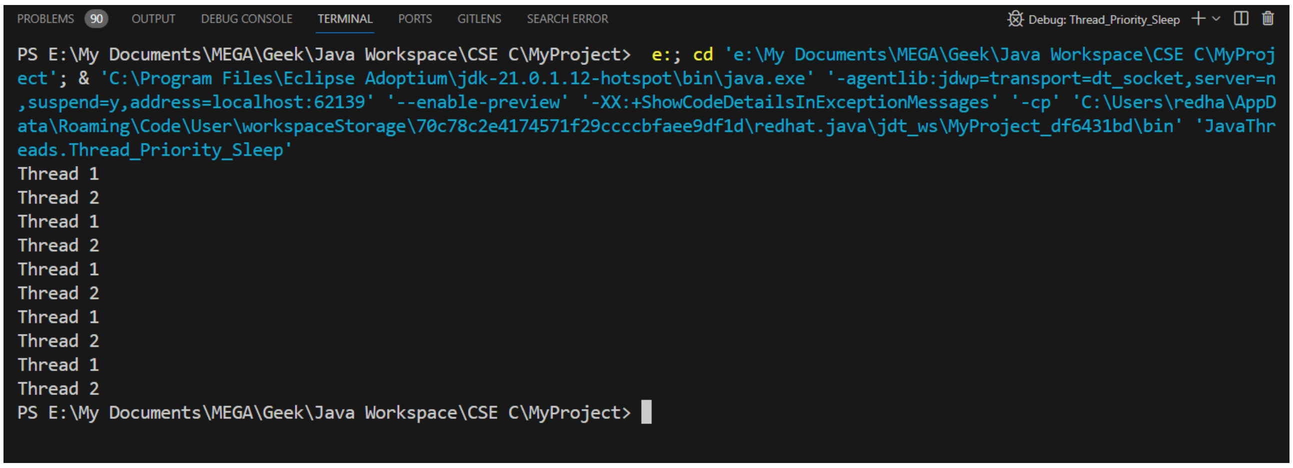
Step 7: The start() method is called on both obj1 and obj2. This results in executing the run() method of both AA and BB classes concurrently in separate threads.

Step 8: STOP.

Source Code:

```
class Class1 extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Thread 1");
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }
    }
}

class Class2 extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Thread 2");
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }
    }
}

public class Thread_Priority_Sleep {
    public static void main(String[] args) throws NumberFormatException {

        Class1 obj1 = new Class1();
        Class2 obj2 = new Class2();

        // obj1.show();
        // obj2.show();

        // 1 implies low priority and 10 implies high priority
        // obj2.setPriority(Thread.MAX_PRIORITY);
        // System.out.println(obj1.getPriority());
        // System.out.println(obj2.getPriority());
        // obj2.setPriority(6);

        obj1.start();
        // try {
        // Thread.sleep(5);
        // } catch (InterruptedException e) {
        // e.printStackTrace();
        // }
        obj2.start();
    }
}
```

Output:



```
PS E:\My Documents\MEGA\Geek\Java Workspace\CSE C\MyProject>  e:; cd 'e:\My Documents\MEGA\Geek\Java Workspace\CSE C\MyProj
ect'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.1.12-hotspot\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n
,suspend=y,address=localhost:62139' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\redha\AppD
ata\Roaming\Code\User\workspaceStorage\70c78c2e4174571f29ccccbfaee9df1d\redhat.java\jdt_ws\MyProject_df6431bd\bin' 'JavaThr
eads.Thread_Priority_Sleep'
Thread 1
Thread 2
Thread 1
Thread 2
Thread 1
Thread 2
Thread 1
Thread 2
Thread 1
Thread 2
PS E:\My Documents\MEGA\Geek\Java Workspace\CSE C\MyProject>
```

Result:

Thus, the above program was executed successfully.