srivenkateshwaraa
College of Engineering & Technology
(Approved by AICTE, New Delhi & Affiliated to Pondicherry University, Puducherry)
13-A, Pondy - Villupuram Main Road, Ariyur, Puducherry - 605 102.
ASPIRE TO EXCEL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

STAFF NAME : Ms.S.PAVITHRA

DESIGNATION : AP / CSE

DEPT HANDLED : CSE

YEAR/SEM : II / IV

SUBJECT CODE : CS P43

SUBJECT : OBJECT ORIENTED PROGRAMMING LAB

STAFF INCHARGE **HOD** **PRINCIPAL**

CS P43 OBJECT ORIENTED PROGRAMMING LABORATORY

LIST OF EXPERIMENTS

.

1. Program to implement classes and objects.
2. Program to implement constructors and destructors with array of objects.
3. Program to demonstrate function overloading.
4. Program to implement different types of inheritances like multiple, Multilevel and hybrid.
5. I/O Program to demonstrate the use of abstract classes.
6. Program to demonstrate I/O streams and functions.
7. Program to perform all possible type conversions.
8. Program to demonstrate exception handling technique.
9. Program to implement networking concepts.
10. Program to implement RMI concepts.
11. Program to implement AWT concepts.
12. Program to implement swing concepts.
13. Program to design and implement applet.
14. Program to design and implement JDBC
15. Program to design an event handling event for simulating a simple calculator.

Ex. No: 1 **CLASSES AND OBJECTS**

AIM:

To write a C++ program to generate Employee Pay Slip using Class & Objects

ALGORITHM:

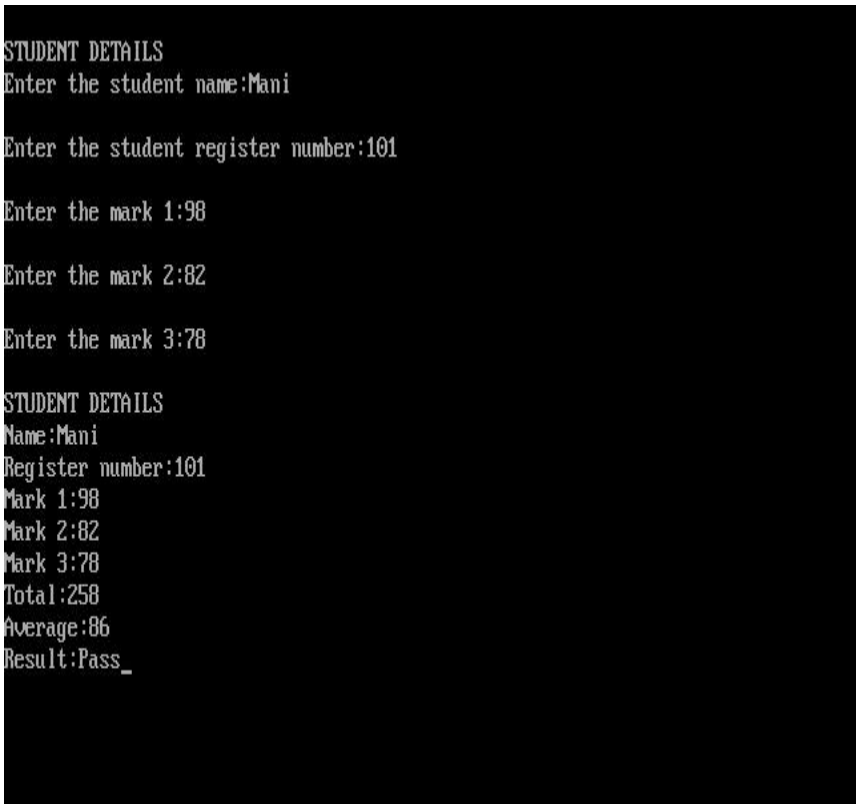
1. Create a class for the Employee with its data member and member function
2. Get number of employee's n from the user
3. Create object e for the class employee
4. Form a loop for n employees
5. Get the details of the employees
6. Calculate gross pay and net pay
7. Display the employee details
8. Repeat the step 5 until the condition i>n fails
9. Stop the process

CODING:

```
#include<iostream.h>
#include<conio.h>
class employee //Class Declaration
{
private:
char ename[20],designation[15]; //Data Member Declaration
float basic,hra,da,ta,pf,gp,net;
public:
void getdetails() //Function to get employee details
{
cout<<"\n\tEnter the Employee name\t\t\t:";
cin>>ename;
cout<<"\n\tEnter the Designation of the Employee :";
cin>>designation;
cout<<"\n\tEnter the Basic pay\t\t\t:";
cin>>basic;
}
void calculation() //Function to calculate salary
{
da= 0.4*basic;
hra=0.25*basic;

 pf=0.1*basic;

ta=0.5*basic;
gp=basic+da+hra+ta;
net=gp-pf;
}
void display() //Function to display employee details
{
cout<<"\n\t\t\tEMPLOYEE DETAILS";
cout<<"\n-----------------------------------------------------";
cout<<"\n\tEmployee name\t:"<<ename;
cout<<"\n\tEmployeeDesig :"<<designation;
cout<<"\n\tBasic pay\t:"<<basic;
```

```
cout<<"\n\tDA\t\t:"<<da;
cout<<"\n\tHRA\t\t:"<<hra;
cout<<"\n\tTA\t\t:"<<ta;
cout<<"\n\tPF\t\t:"<<pf;
cout<<"\n\tGross pay\t:"<<gp;
cout<<"\n\tNetpay\t\t:"<<net;
cout<<"\n-------------------------------------------------------";
}
};
void main()
{
clrscr();
employee e; //Object declaration
e.getdetails();
e.calculation();
e.display();
getch();
}
```

OUTPUT:



RESULT:

Thus the program class and objects executed and verified successfully

Ex. No: 2 **CONSTRUCTOR AND DESTRUCTOR**

AIM:

To write a C++ program to demonstrate Constructor and Destructor.

ALGORITHM:

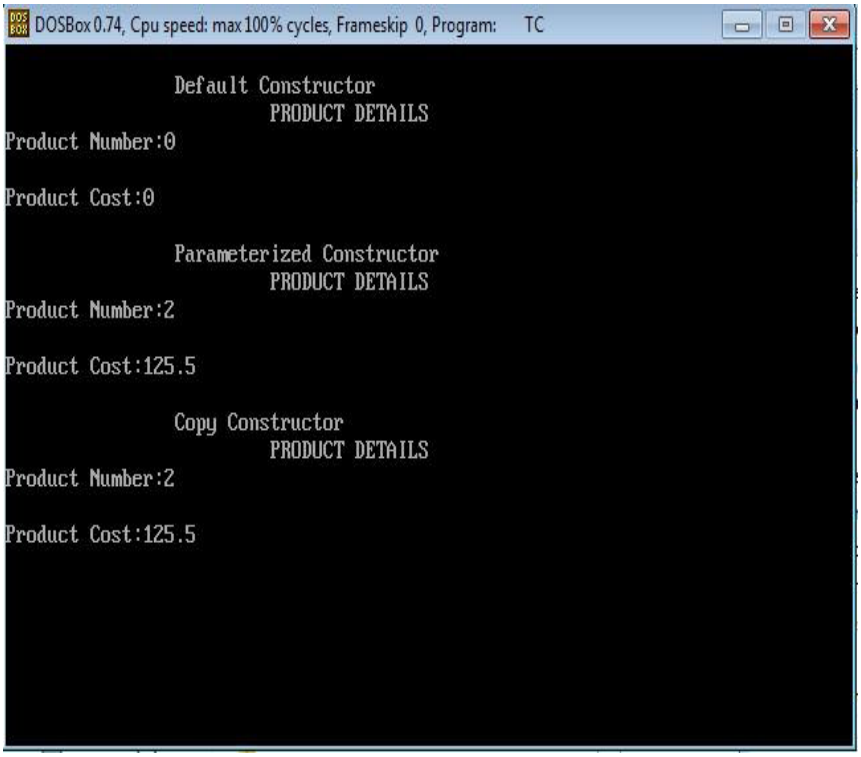
1. Start the program.
2. Declare the class name as product with data members and member functions.
3. The default constructor product() is assigned with the product number as 0 and the cost as 0.0
4. The parameterized constructor product() with argument is assigned with the product number as a and the cost as b.
5. Copy constructor with argument to assign the value.
6. Destructor is used to destroy the product.
7. Display function for displaying the product details.
8. In the main function, p1 object invoke default constructor.
9. p2 object invoke parameterized constructor.
10. P3 object invoke copy constructor.
11. Stop the program

CODING:

```
#include<iostream.h>
#include<conio.h>
class product //Class Declaration
{
private: //Data Member Declaration
int productno;
float cost;
public:
product() //Default Constructor to initialize data member
{
productno=0;
cost=0.0;
}
product(int a,float b) //Parameterized Constructor to initialize data member
{
productno=a;
cost=b;}
product(product &x) //Copy Constructor to initialize data member
{
productno=x.productno;
cost=x.cost;
}
product() //Destructor
{
cout<<"\nProduct "<<productno<<"is destroyed\n";
}
void display(void) //Function to display product details
{
cout<<"\n\t\t\t PRODUCT DETAILS";
cout<<"\nProduct Number:"<<productno<<"\n";
cout<<"\nProduct Cost:"<<cost<<"\n";
```

```
}
};
int main()
{
clrscr();
product p1; //Object declaration & Default constructor invoked
cout<<"\n\t\tDefault Constructor ";
p1.display();
cout<<"\n\t\tParameterized Constructor ";
product p2(2,125.5);//Object declaration & Parameterized constructor invoked
p2.display();
cout<<"\n\t\tCopy Constructor ";
product p3(p2); //Object declaration & Copy constructor invoked
p3.display();
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus the constructors and destructors program executed and verified successfully.

Ex. No: 3 **FUNCTION OVERLOADING**

AIM:

To write a C++ program to demonstrate Function Overloading

ALGORITHM:

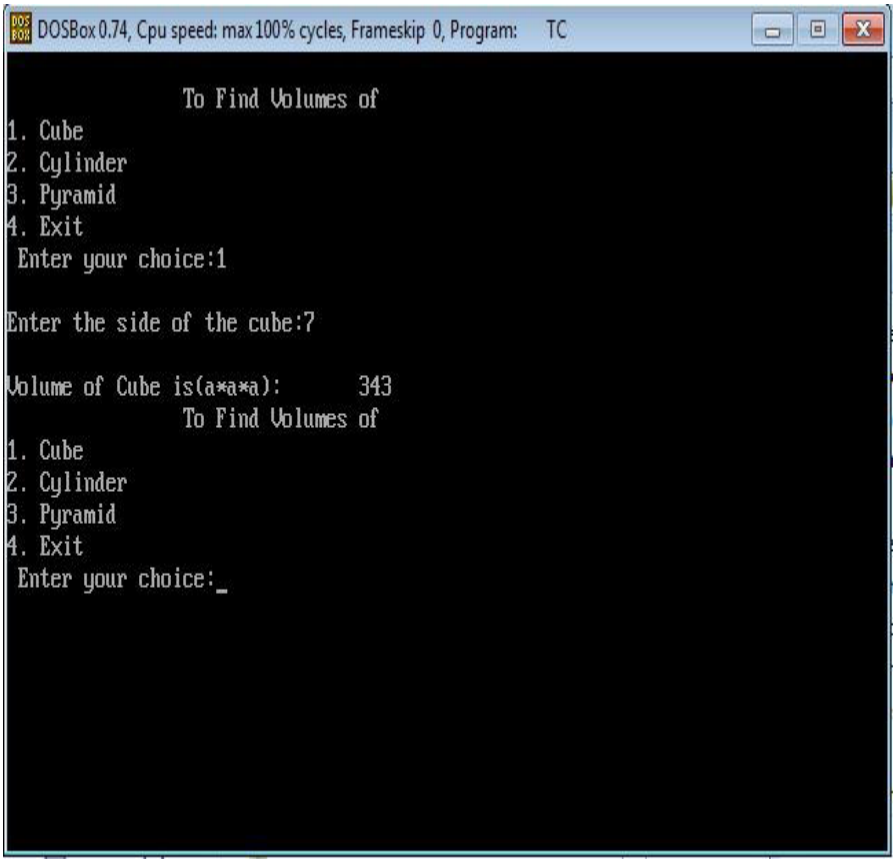
1. Start the program.
2. Declare the class name as volume with data members and member functions.
3. Read the choice from the user.
4. Choice=1 then go to the step 5.
5. The function calculate() to find the side of the cube with one integer argument.
6. Choice=2 then go to the step 7.
7. The function calculate() to find the radius and height of the cylinder with two integer argument.
8. Choice=3 then go to the step 9.
9. The function calculate () to find the base and height with two integer argument.
10. Choice=4 then stop the program

CODING:

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
class volume //Class Declaration
{
public:
void calculate(int a) //Overloaded Function to find volume of cube
{
cout<<"\nVolume of Cube is(a*a*a):\t"<<pow(a,3);
}
void calculate(float r,int h) //Overloaded Function to find volume of cylinder
{
cout<<"\nVolume of cylinder is(3.14*r*r*h):\t"<<3.14*r*r*h;
}
void calculate(int b,int h) //Overloaded Function to find volume of pyramid
{
cout<<"\nVolume of pyramid is((1/3)*b*h):\t"<<0.33*b*h;
}
};
int main()
{
clrscr();
volume v; //Object declaration
int a,b,h,ch; //Variable Declaraion
float r;
do
{
cout<<"\n\t\tTo Find Volumes of";
cout<<"\n1. Cube";
cout<<"\n2. Cylinder";
cout<<"\n3. Pyramid";
cout<<"\n4. Exit";
```

```
cout<<"\n Enter your choice:";
cin>>ch;
switch(ch)
{
case 1:
cout<<"\nEnter the side of the cube:";
cin>>a;
v.calculate(a);
break;
case 2:
cout<<"\nEnter the radius of the cylinder:";
cin>>r;
cout<<"\nEnter the height of the cylinder:";
cin>>h;
v.calculate(r,h);
break;
case 3:
cout<<"\nEnter the base of the pyramid:";
cin>>b;
cout<<"\nEnter the height of the pyramid:";
cin>>h;
v.calculate(b,h);
break;
case 4:
exit(0);
break;

default:
cout<<"\nEnter the correct choice";
}
}while(ch<=3);
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus the function overloading program executed and verified successfully

Ex. No: 4(a) **MULTIPLE INHERITANCE**

AIM:

To write a C++ program to implement multiple inheritance using Student information system

ALGORITHM:

1. Start the program.
2. Declare the base class student.
3. Declare and define the function getdata() to get the student details.
4. Declare the other class marks.
5. Declare and define the function getmarks() to get the student marks.
6. Create the class result derived from student and marks.
7. Declare and define the function calculaton() to find out the total and average of the student.
8. Declare the function display() to display the student details.
9. Declare the derived class object and call the functions getdata(), getmarks(), getcalculation() and display().
10. Stop the program

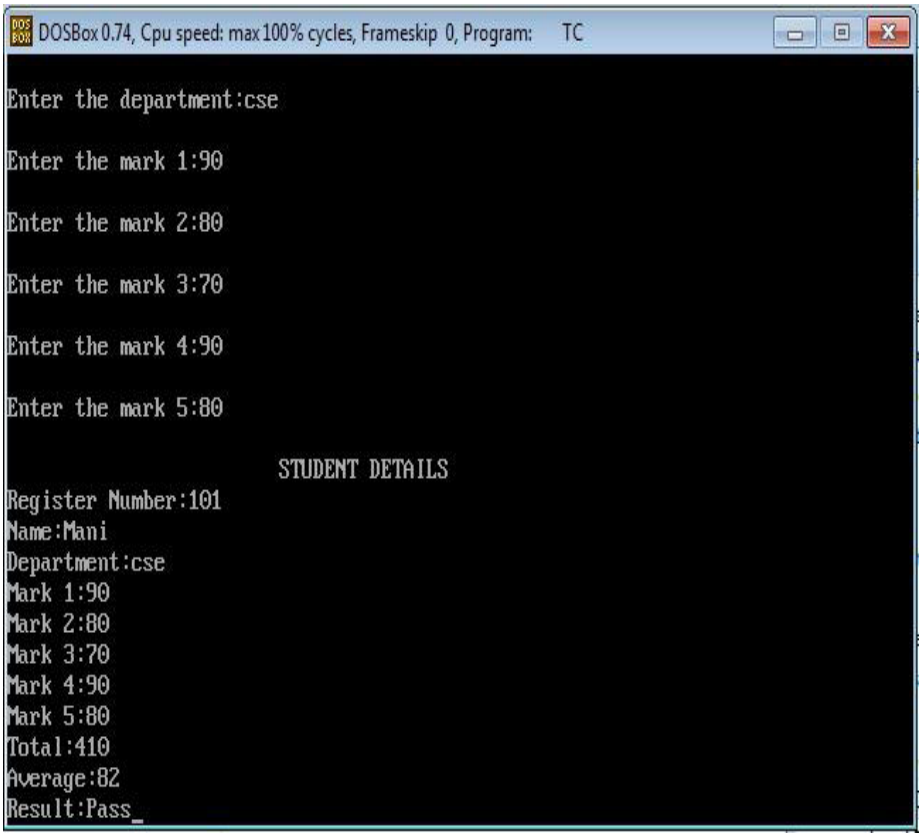
CODING:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class student //Class Declaration
{
protected:
int regno; //Data Member Declaration
char name[20],dept[5];
public:
void getdata() //Function to get student details
{
cout<<"\nEnter the register number:";
cin>>regno;
cout<<"\nEnter the name:";
cin>>name;
cout<<"\nEnter the department:";
cin>>dept;
}
};

class marks //Class Declaration
{
protected:
int m1,m2,m3,m4,m5; //Data Member Declaration
public:
void getmarks() //Function to get student marks
{
cout<<"\nEnter the mark 1:";
cin>>m1;
```

```
cout<<"\nEnter the mark 2:";
cin>>m2;
cout<<"\nEnter the mark 3:";
cin>>m3;
cout<<"\nEnter the mark 4:";
cin>>m4;
cout<<"\nEnter the mark 5:";
cin>>m5;
}
};
class result:public student,public marks //Deriving new class from more two base class
{
private:
int total; //Data Member Declaration
float average;
char result[6];
public:
void calculation() //Function to calculate result of the student
{
total=m1+m2+m3+m4+m5;
average=float(total)/5;
if(m1>49 && m2>49 && m3>49 && m4>49 && m5>49)
{
strcpy(result,"Pass");
}
else
{
strcpy(result,"Fail");
}
}

void display() //Function to display student details
{
cout<<"\n\t\t\tSTUDENT DETAILS";
cout<<"\nRegister Number:"<<regno;
cout<<"\nName:"<<name;
cout<<"\nDepartment:"<<dept;
cout<<"\nMark 1:"<<m1;
cout<<"\nMark 2:"<<m2;
cout<<"\nMark 3:"<<m3;
cout<<"\nMark 4:"<<m4;
cout<<"\nMark 5:"<<m5;
cout<<"\nTotal:"<<total;
cout<<"\nAverage:"<<average;
cout<<"\nResult:"<<result;
}
};
int main()
{
clrscr();
result r; //Object declaration for derived class
cout<<"\n\t\t\t STUDENT INFORMATION";
r.getdata(); //Calling Base class function using derived object
```

```
r.getmarks(); //Calling Base class function using derived object
r.calculation();
r.display();
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus the student information system using multiple inheritance executed and verified successfully

Ex. No: 4(b) **MULTILEVEL INHERITANCE**

AIM:

To write a C++ program to implement Multilevel inheritance using Student information system

ALGORITHM:

1. Start the program.
2. Declare the base class student.
3. Declare and define the function getdata() to get the student details.
4. Declare the other class marks.
5. Declare and define the function getmarks() to get the student marks.
6. Create the another class result from the derived new class marks.
7. Declare and define the function calculaton() to find out the total and average of the student.
8. Declare the function display() to display the student details.
9. Declare the derived class object and call the functions getdata(), getmarks(), getcalculation() and display().
10. Stop the program

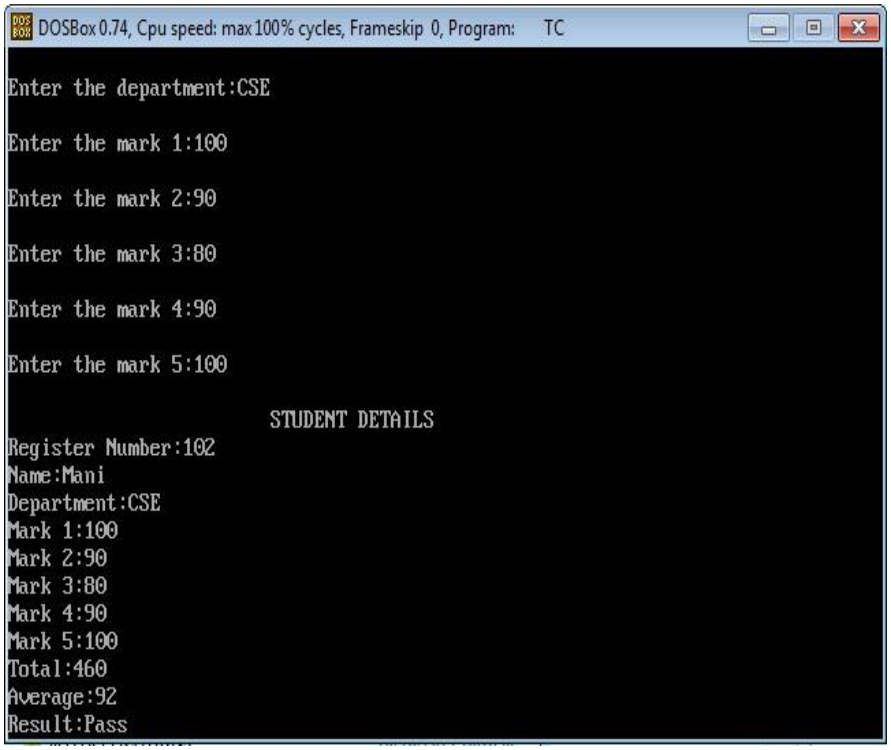
CODING:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class student //Class Declaration
{
protected:
int regno; //Data Member Declaration
char name[20],dept[5];
public:
void getdata() //Function to get student details
{
cout<<"\nEnter the register number:";
cin>>regno;
cout<<"\nEnter the name:";
cin>>name;
cout<<"\nEnter the department:";
cin>>dept;
}
};class marks: public student //Deriving new class from base class
{
protected:
int m1,m2,m3,m4,m5; //Data Member Declaration
public:
void getmarks() //Function to get student marks
{
cout<<"\nEnter the mark 1:";
cin>>m1;
cout<<"\nEnter the mark 2:";
```

```
cin>>m2;
cout<<"\nEnter the mark 3:";
cin>>m3;
cout<<"\nEnter the mark 4:";
cin>>m4;
cout<<"\nEnter the mark 5:";
cin>>m5;
}
};
class result: public marks //Deriving new class from another derived class
{
private:
int total; //Data Member Declaration
float average;
char result[6];
public:
void calculation() //Function to calculate result of the student
{
total=m1+m2+m3+m4+m5;
average=float(total)/5;
if(m1>49 && m2>49 && m3>49 && m4>49 && m5>49)
{
strcpy(result,"Pass");
}
else
{
strcpy(result,"Fail");
}
}
void display() //Function to display student details
{
cout<<"\n\t\t\tSTUDENT DETAILS";

cout<<"\nRegister Number:"<<regno;
cout<<"\nName:"<<name;
cout<<"\nDepartment:"<<dept;
cout<<"\nMark 1:"<<m1;
cout<<"\nMark 2:"<<m2;
cout<<"\nMark 3:"<<m3;
cout<<"\nMark 4:"<<m4;
cout<<"\nMark 5:"<<m5;
cout<<"\nTotal:"<<total;
cout<<"\nAverage:"<<average;
cout<<"\nResult:"<<result;
}
int main()
{
clrscr();
result r; //Object declaration for derived class
cout<<"\n\t\t\t STUDENT INFORMATION";
r.getdata(); //Calling Base class function using derived object
r.getmarks(); //Calling Base class function using derived object
r.calculation();
```

```
r.display();
getch();
return 0;
} };
```

OUTPUT:



RESULT:

Thus the student information system using multilevel inheritance executed and verified successfully

Ex. No: 4(c) **HYBRID INHERITANCE**

AIM:

To write a C++ program to implement Hybrid inheritance using Student information system

ALGORITHM:

1. Start the program.
2. Declare the base class student.
3. Declare and define the function getdata() to get the student details.
4. Declare and define the function showdata() to display the student details.
5. Declare the other class marks as virtual.
6. Declare and define the function getmarks() to get the student marks.
7. Declare and define the function showmark() to display mark
8. Create the another class attendance and make it virtual .
9. Declare and define the function gettotalhour() to find out the totalhour.
10. Declare and define the function getabsent() to find out the total absenthour.
11. Create the another class result which publically inherited classes marks and attendance
12. Declare the member function showcalculation() which show the total marks and percentage
13. Declare the member function member function showdisplay() to display the student details
14. Declare the function result().
15. Declare the derived class object, and call the functions getdata(), getmarks(), getattendance(), getcalculation() and getdisplay().
16. Stop the program

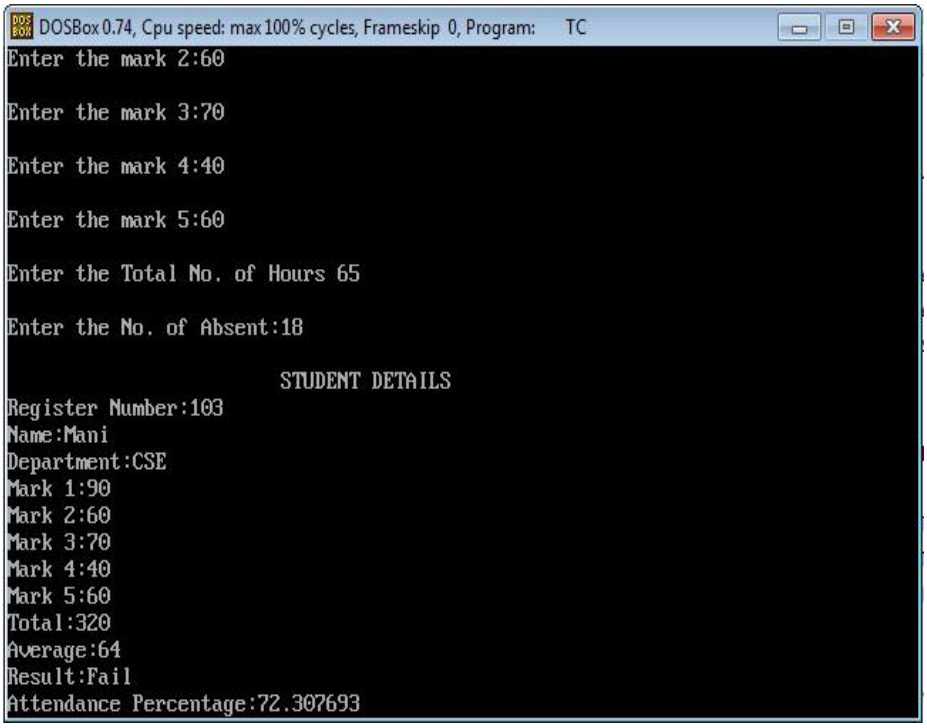
CODING:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class student //Class Declaration
{
protected:
int regno; //Data Member Declaration
char name[20],dept[5];
public:
void getdata() //Function to get student details
{
cout<<"\nEnter the register number:";
cin>>regno;
cout<<"\nEnter the name:";
cin>>name;
cout<<"\nEnter the department:";
cin>>dept;
}
};
class marks: public student //Deriving new class from another derived class
{
```

```
protected:
int m1,m2,m3,m4,m5; //Data Member Declaration
public:
void getmarks() //Function to get student marks
{
cout<<"\nEnter the mark 1:";
cin>>m1;
cout<<"\nEnter the mark 2:";
cin>>m2;
cout<<"\nEnter the mark 3:";
cin>>m3;
cout<<"\nEnter the mark 4:";
cin>>m4;
cout<<"\nEnter the mark 5:";
cin>>m5;
}
};
class attendance //Class Declaration
{
protected:
int totalhours,absent,present; //Data Member Declaration
float percentage;
public:
void getattendance() //Function to get student attendance details
{
cout<<"\nEnter the Total No. of Hours ";
cin>>totalhours;
cout<<"\nEnter the No. of Absent:";
cin>>absent;
}
};
class result: public marks,public attendance
{
private:
int total; //Data Member Declaration
float average;
char result[6];
public:
void calculation() //Function to calculate result of the student
{
total=m1+m2+m3+m4+m5;
average=float(total)/5;
if(m1>49 && m2>49 && m3>49 && m4>49 && m5>49)
{
strcpy(result,"Pass");
}
else
{
strcpy(result,"Fail");
}
present=totalhours-absent;
percentage=float(present)/totalhours;
```

```
percentage=percentage*100;
}
void display() //Function to display student details
{
cout<<"\n\t\t\tSTUDENT DETAILS";
cout<<"\nRegister Number:"<<regno;
cout<<"\nName:"<<name;
cout<<"\nDepartment:"<<dept;
cout<<"\nMark 1:"<<m1;
cout<<"\nMark 2:"<<m2;
cout<<"\nMark 3:"<<m3;
cout<<"\nMark 4:"<<m4;
cout<<"\nMark 5:"<<m5;
cout<<"\nTotal:"<<total;
cout<<"\nAverage:"<<average;
cout<<"\nResult:"<<result;
cout<<"\nAttendance Percentage:"<<percentage;
}
};

int main()
{
clrscr();
result r; //Object declaration
cout<<"\n\t\t\t STUDENT INFORMATION";
r.getdata();
r.getmarks();
r.getattendance();
r.calculation();
r.display();
getch();
return 0;
}
```

OUTPUT:



RESULT:

Thus the student information system using hybrid inheritance executed and verified successfully

BASIC JAVA PROGRAMS-SEARCH

AIM:

To write a java program to search an element in an array using loop and conditional statement

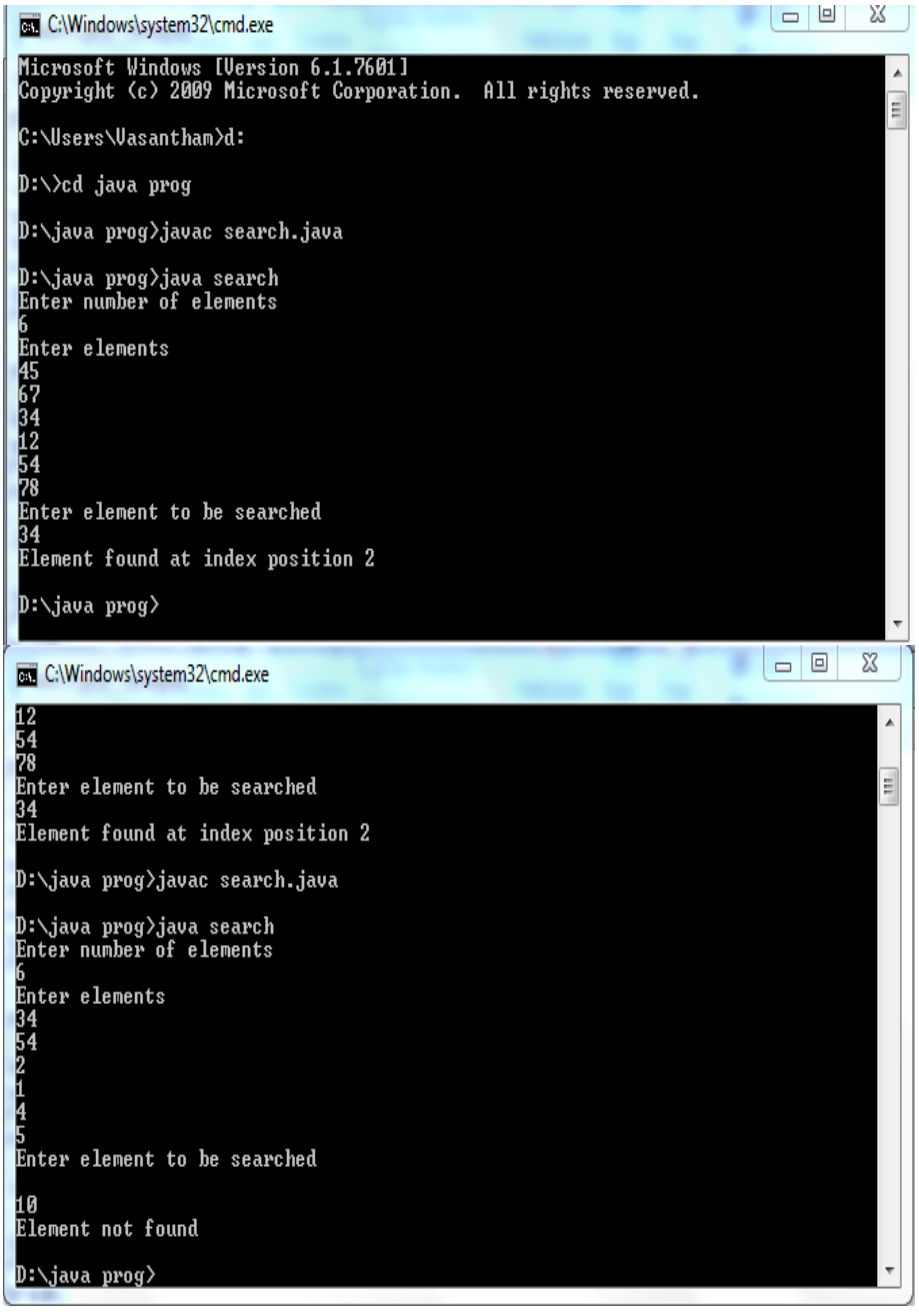
ALGORITHM:

1. Start the program
2. Create class search.
3. Create objects for Scanner and get array numbers
4. Get the number to be searched
5. Display the result
6. Stop the program

CODING:

```
import java.util.Scanner;
class search
{
public static void main(String arg[])
{
int a[]=new int[10];
int i,n,x,flag;
Scanner in = new Scanner(System.in);
System.out.print("Enter number of elements ");
n=in.nextInt();
System.out.println("Enter elements ");
for(i=0;i<n;i++)
{
a[i]=in.nextInt();
}
System.out.println("Enter element to be searched ");
x=in.nextInt();
flag = 0;
for(i=0;i<n;i++)
{
if(a[i] == x)
{
System.out.println("Element found at index position " + i);
flag=1;
break;
}
}
if(flag==0)
{
System.out.println("Element not found");
}
}
}
```

OUTPUT:



RESULT:

Thus the search program executed and verified successfully

BASIC JAVA PROGRAMS- FACTORIAL

AIM:

To write a java program to find factorial of a number using functions

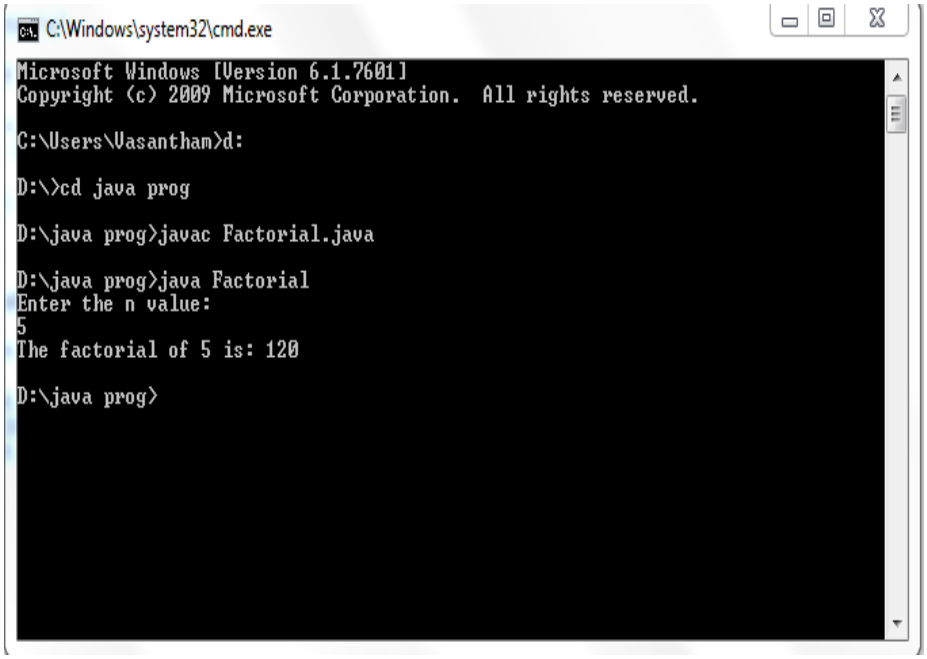
ALGORITHM:

1. Start the program
2. Create class Factorial with fact () method
3. Create objects for Scanner
4. Get input from the console and call the method fact to find factorial
5. Display the result
6. Stop the program

CODING:

```
import java.util.Scanner;

class Factorial
{
static int fact ( int input )
{
int i, f = 1;
for ( i = 1; i <= input; i++)
{
f = f * i;
}
return f;
}
public static void main(String arg[])
{
Scanner in = new Scanner(System.in);
int n;
System.out.println("Enter the n value: ");
n=in.nextInt();
System.out.println("The factorial of "+n+" is: " + fact(n));
}
}
```

OUTPUT:



RESULT:

Thus the factorial using functions program executed and verified successfully

Ex. No: 5 **ABSTRACT CLASS**

AIM:

To write a java program to demonstrate the use of abstract classes.

ALGORITHM:

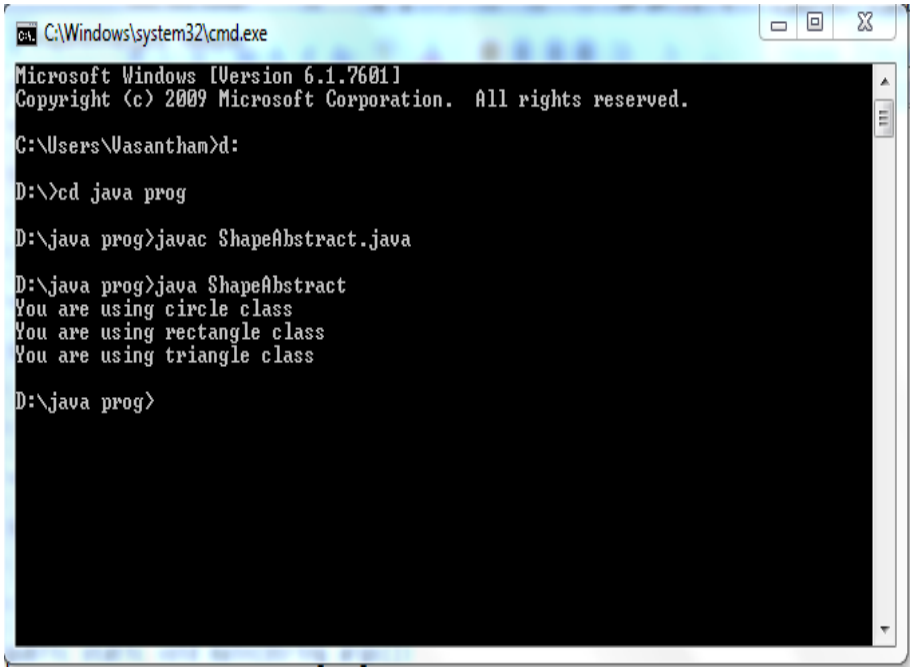
1. Create an abstract class Shape
2. Declare display() as abstract method
3. Create circle class by extending shape class and override the display() method
4. Create rectangle class by extending shape class and override the display() method
5. Create triangle class by extending shape class and override the display() method
6. Create circle object and assign it to shape object reference and call display method
7. Create rectangle object and assign it to shape object reference and call display method
8. Create triangle object and assign it to shape object reference and call display method
9. Stop the program

CODING:

```
abstract class Shape
{
abstract void display();
}
class Circle extends Shape
{
void display()
{
System.out.println("You are using circle class");
}
}
class Rectangle extends Shape
{
void display()
{
System.out.println("You are using rectangle class");
}
}
class Triangle extends Shape
{
void display()
{
System.out.println("You are using triangle class");
}
}
class ShapeAbstract
{
public static void main(String args[])
{
Shape s = new Circle();
s.display();
s = new Rectangle();
```

```
s.display();
s = new Triangle();
s.display();
}
}
```

OUTPUT:



RESULT:

Thus the abstract class program executed and verified successfully

Ex. No: 6(a) **I/O STREAMS**

AIM:

To write a java program to demonstrate simple i/o streams and its methods

ALGORITHM:

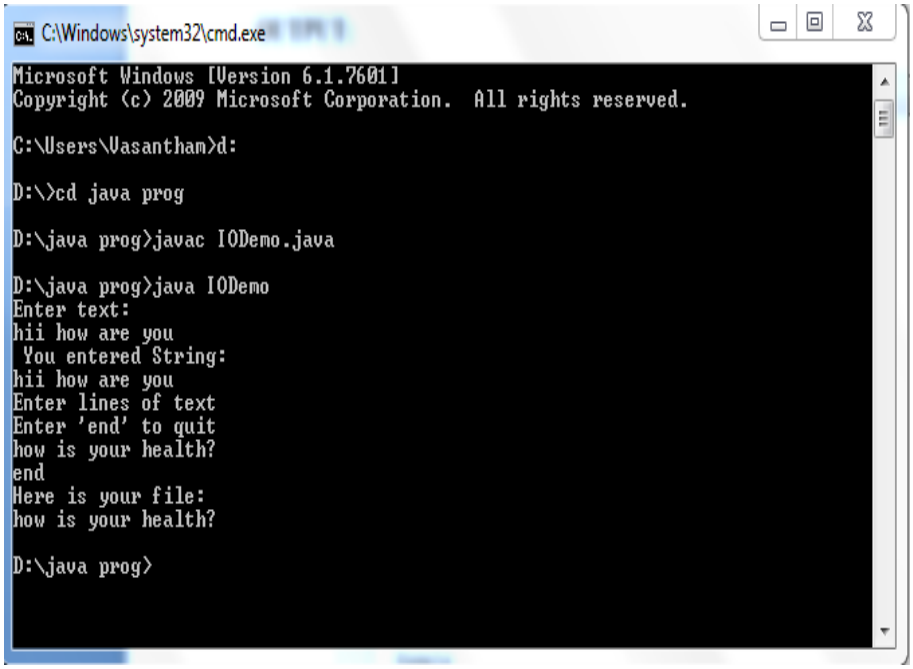
1. Create class IODemo
2. Create objects for InputStreamReader and BufferedReader
3. With readLine() method, get input from the console and store it in string str
4. Declare String array str1 with size 5. Using for loop, get multiple lines as input using readLine() method
6. The loop will terminate if the string matches with “end”
7. Display the str1 array
8. Stop the program

CODING:

```
import java.io.*;
public class IODemo
{
public static void main(String[] args)
{
try
{
InputStreamReader ir=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(ir);
//single line as input
System.out.println("Enter text: ");
String str= br.readLine();
System.out.println(" You entered String: ");
System.out.println(str);
//multiple lines as input
String str1[]=new String[100];
System.out.println("Enter lines of text");
System.out.println("Enter 'end' to quit");
for(int i=0; i<100; i++)
{
str1[i]=br.readLine();
if(str1[i].equals("end"))
break;
}

System.out.println("Here is your file:");

for(int i=0; i<100; i++)
{
if(str1[i].equals("end"))
break;
System.out.println(str1[i]);
}
}
catch(IOException e)
{
```

```
System.out.println(e);
}
}
}
```

OUTPUT:



RESULT:

Thus the I/O stream demonstration program executed and verified successfully

Ex. No: 6(b) **I/O STREAMS - FILE**

AIM:

To write a java program to demonstrate file i/o streams and its methods

ALGORITHM:

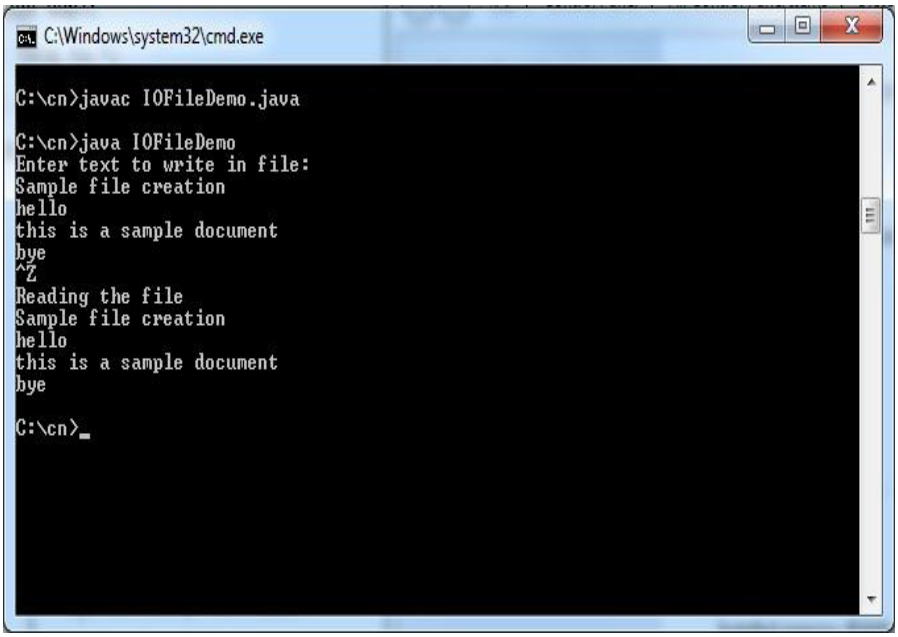
1. Create class IOFileDemo
2. Declare objects for InputStreamReader and BufferedReader
3. Declare an object for FileOutputStream to perform write operation in a file
4. Using br.read() method, get input from the console
5. Then write the input to the file using fout object
6. While completes writing the file, close the fout stream
7. Declare an object for FileInputStream to perform read operation in a file
8. Declare objects for InputStreamReader and BufferedReader associated with the file object
9. Using readLine() method, read all the contents in the file and display it on the console
10. Close the fis stream
11. Stop the program

CODING:

```
import java.io.*;
public class IOFileDemo
{
public static void main(String args[])
{
InputStreamReader ir=new InputStreamReader(System.in);
BufferedReader br=new BufferedReader(ir);
System.out.println("Enter text to write in file:");
int i;
//To perform write operation in a file
try
{
FileOutputStream fout=new FileOutputStream("sample.txt");
do
{
i=br.read();
if(i!=-1)
fout.write(i);
}while(i!=-1);

fout.close();
}
catch(IOException e)
{
System.out.println(e);
}
//to perform read operation in a file
try
{
FileInputStream fis=new FileInputStream("sample.txt");
DataInputStream dis=new DataInputStream(fis);
BufferedReader br1=new BufferedReader(new InputStreamReader(dis));
```

```
String str;
System.out.println("Reading the file");
while((str=br1.readLine())!=null)
{
System.out.println(str);
}
dis.close();
}
catch(IOException e)
{
System.out.println(e);
}
}
}
```

OUTPUT:



RESULT:

Thus the demonstration of file I/O stream program executed and verified successfully

Ex. No: 7 **TYPE CONVERSION**

AIM:

To write a java program to perform all possible type conversions.

ALGORITHM:

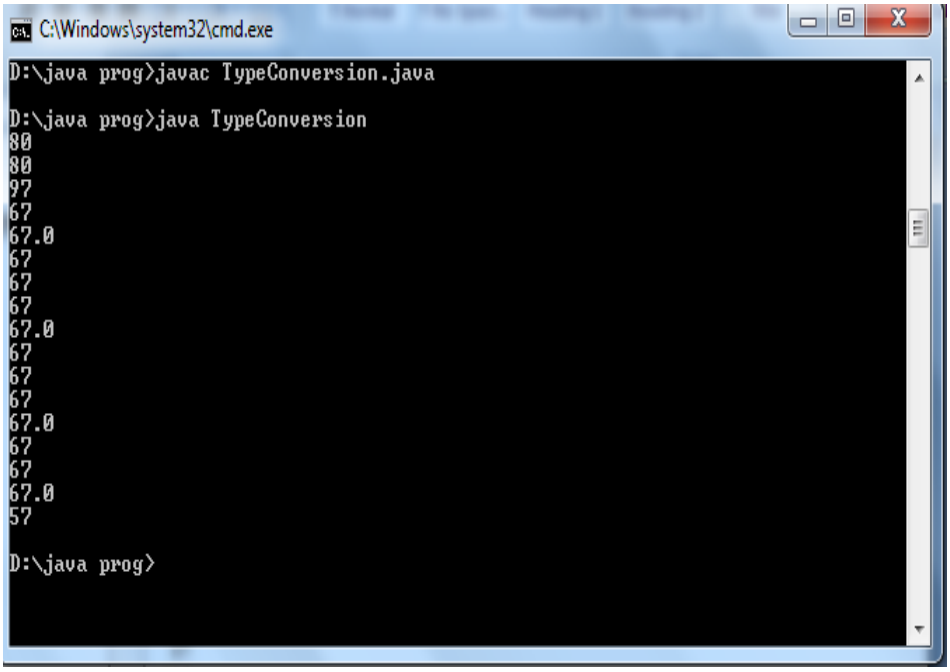
1. Create a class Type Conversion
2. Import java.io package to use io streams and methods
3. Created object for scanner class to get input from console
4. Convert string to integer using Integer.parseInt() method
5. Convert integer to string using Integer.toString() method
6. Convert xxx datatype to string using xxx.toString() method
7. Convert string to xxx datatype using xxx.parsexxx() method
8. Convert string to other types using valueOf() method
9. Convert char to int using external type casting
10. Stop the program

CODING:

```
import java.io.*;
import java.lang.*;
import java.util.*;
public class TypeConversion
{
public static void main(String arg[])
{
Scanner in=new Scanner(System.in);

//String to Integer
String s=in.next();
int i=Integer.parseInt(s);
System.out.println(i);

//Integer to String
int a=97;
String s3=Integer.toString(a);
System.out.println(s3);

//Double to String
double d=in.nextDouble();
s=Double.toString(d);
System.out.println(s);

//Long to String
long l=in.nextLong();
s=Long.toString(l);
System.out.println(s);


//Float to String
float f=in.nextFloat();
s=Float.toString(f);
System.out.println(s);
```

```
//String to Integer
s=in.next();
i=Integer.valueOf(s).intValue();
System.out.println(i);
i = Integer.parseInt(s);
System.out.println(i);

//String to Double
d=Double.valueOf(s).doubleValue();
System.out.println(d);

//String to Long
long lng=Long.valueOf(s).longValue();
System.out.println(lng);
lng=Long.parseLong(s);
System.out.println(lng);

//String to Float
f=Float.valueOf(s).floatValue();
System.out.println(f);

//Character to Integer
char c='9';
i=(char)c;
System.out.println(i);
}
}
```

OUTPUT:



RESULT:

Thus the type conversion program executed and verified successfully

Ex. No: 8(a) **EXCEPTION HANDLING – DIVIDE BY ZERO**

AIM:

To write a java program to demonstrate exception handling technique.

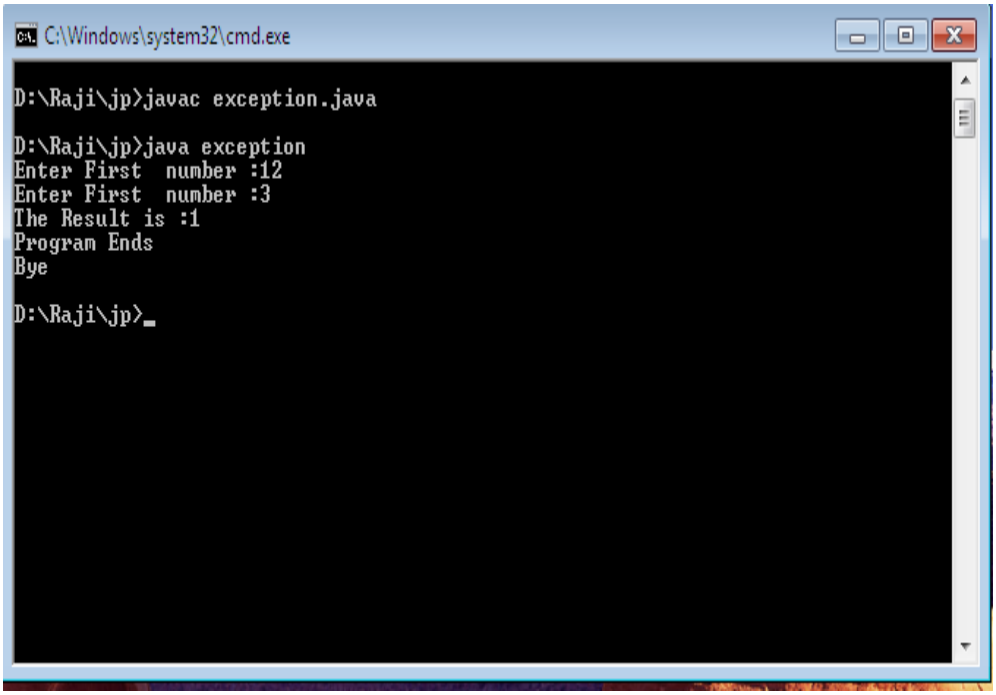
ALGORITHM:

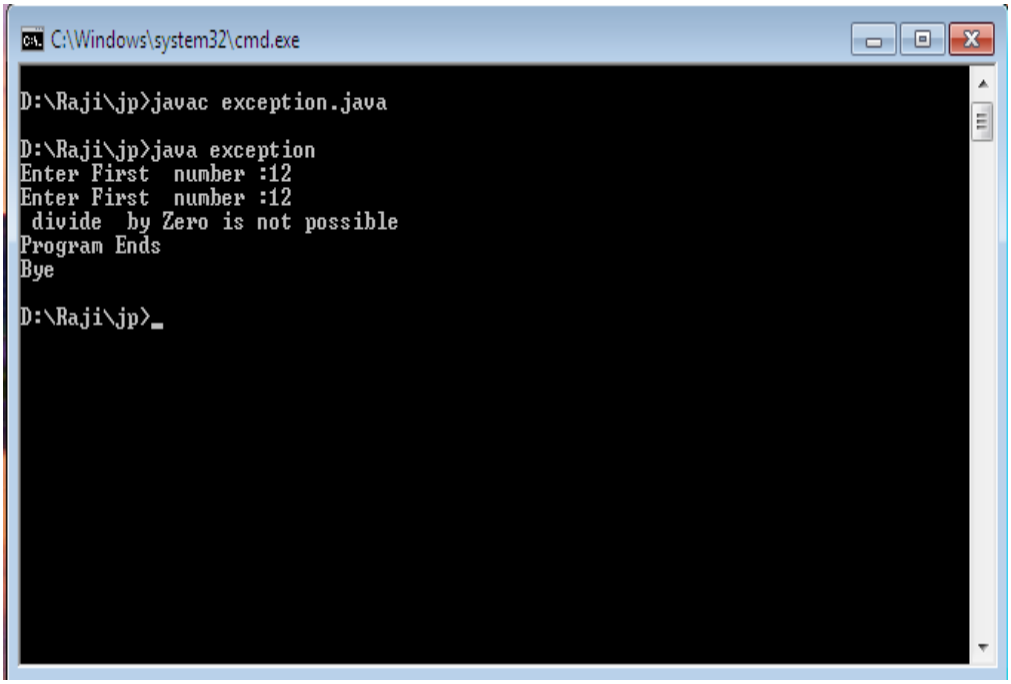
1. Create a class Exception.
2. Try dividing a numeric value by 0 in the try block
3. Catch the exception as Arithmetic exception and print the result as “Divide by Zero error”.
4. Finally compile and run the program to verify the result.

CODING:

```
//Program to demonstrate Exception handling in java
import java.util.*;
class exception
{
public static void main(String args[ ])
{
int a=10,b,c;
Scanner sin=new Scanner(System.in));
System.out.println(“ Enter First  number :”);
b=sin.nextInt();
System.out.println(“ Enter First  number :”);
c=sin.nextInt();

int x,y;
try
{
x = a/ (b-c);
}
catch(ArithmeticException e)
{
System.out.println("U can’t divide anything by Zero");
}
System.out.println("Bye");
}
}
```

OUTPUT:





RESULT:

Thus the exception handling program executed and verified successfully

Ex. No: 8(b) **EXCEPTION HANDLING –MULTIPLE CATCH BLOCK**

AIM:

To write a java program to demonstrate multiple catch block in exception handling technique.

ALGORITHM:

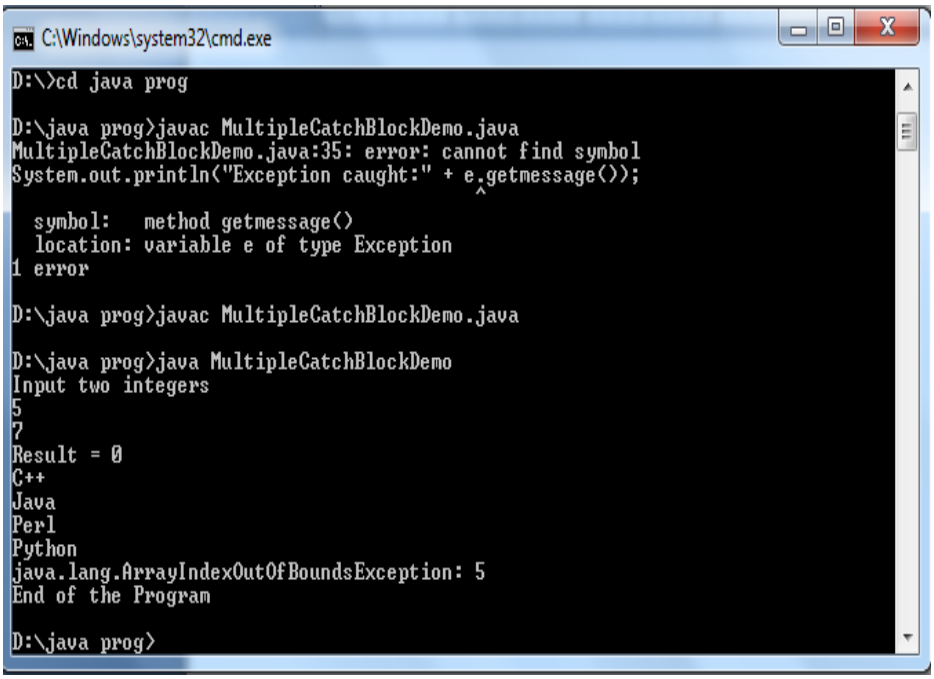
1. Create a class MultipleCatchBlockDemo with the method checkbound()
2. Create a string array 'Language' and initialize it with the values.
3. Try printing the values exceeding the array bounds, the compiler displays the array index out of bounds exception if the exception is caught.
4. Similarly try dividing a value by 0.
5. Multiple catch statements can be used to catch the exceptions such as Arithmetic exception, Divide by zero exception, etc.
6. Compile and run the program to verify the exception concept

CODING:

```
import java.util.Scanner;
class MultipleCatchBlockDemo
{
static void checkbound() throws ArrayIndexOutOfBoundsException
{
String languages[] = { "C", "C++", "Java", "Perl", "Python" };
for (int c = 1; c <= 5; c++)
{
System.out.println(languages[c]);
}
}
public static void main(String[] args)
{
int a, b, result;
Scanner input = new Scanner(System.in);
System.out.println("Input two integers");
a = input.nextInt();
b = input.nextInt();
try                                    // try block
{result = a / b;
System.out.println("Result = " + result);
checkbound();
}
//Multiple catch blocks
catch (ArithmeticException e)
{
System.out.println("Exception caught: Division by zero.");
}
catch (ArrayIndexOutOfBoundsException e)
{
System.out.println(e);
}
catch (Exception e)
{
System.out.println("Exception caught:" + e.getMessage());
}
```

```
System.out.println("End of the Program");
}
}
```

OUTPUT:



RESULT:

Thus the exception handling with multiple catch blocks program executed and verified successfully

Ex. No:9 **NETWORKING – TCP ONE WAY COMMUNICATION**

AIM:

To write a java program for one way communication using TCP

ALGORITHM:

Server Program:

1. Start.
2. Import the java.net and java.io packages.
3. Declare a new class called “TcpOneWayChatServer”.
4. Within class “TcpOneWayChatServer”, create a ServerSocket object called “ss” with the port number 8000.
5. Create a Socket object called “s” by using the accept() method, which listens for a connection to be made to this server socket and accepts it.
6. Create a new BufferedReader object, which acts as an input stream to the server from the client.
7. Create a new PrintStream object, which acts as an output stream to the client from the server.
8. Display the message “Server ready...” on the server window.
9. Repeat the following steps:

i. Prompt for the message to be sent from server to client.
ii. Read in the message to be sent from the user.
iii. Send the message to the client using the PrintStream object.
iv. If the message equals the string “end”, then close the input and output streams and exit the loop.

10. Stop

Client Program:

1. Start.
2. Import the java.net, java.io, and java.util packages.
3. Create a new class called “TcpOneWayClient”.
4. Inside “TcpOneWayClient”, create a new Socket object called “s” with port number 8000.
5. Create a new BufferedReader object which acts as the input stream to the client from the server.
6. Repeat the following steps:

i. Read in the input from the server to the client using the BufferedReader object.
ii. Display the message received.
iii. If the string received equals “end”, then close the input and output streams and exit the loop.

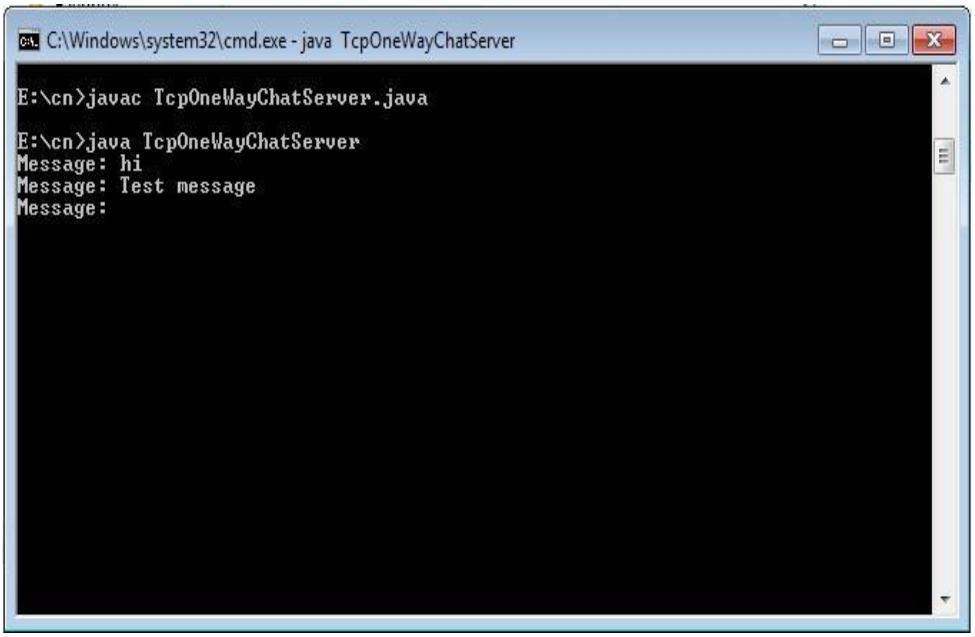
7. Stop.

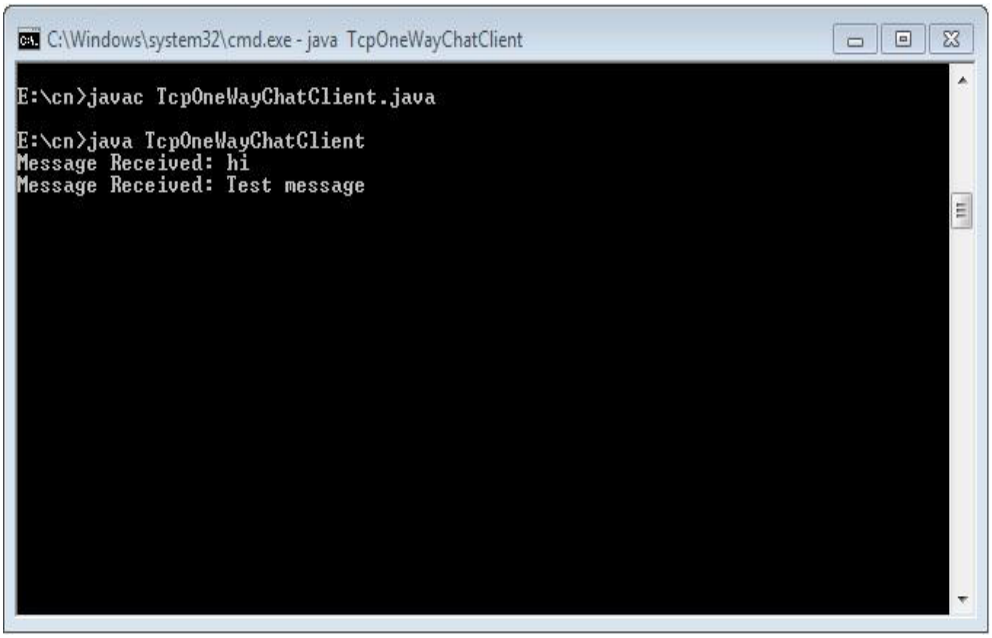
CODING:

```
//TcpOneWayChatServer
import java.io.*;
import java.net.*;
class TcpOneWayChatServer
{
public static void main(String a[])throws IOException
{
ServerSocket ss = new ServerSocket(8000);
Socket s=ss.accept();
BufferedReader keyIn = new BufferedReader(new InputStreamReader(System.in));
PrintStream socOut = new PrintStream(s.getOutputStream());
while(true)
{
System.out.print("Message: ");
```

```
String str = in.readLine();
if(str.equals("bye"))
{
break;
}
socOut.println(str);
}
}
}
//TcpOneWayChatClient
import java.io.*;
import java.net.*;
class TcpOneWayChatClient
{
public static void main(String args[])throws IOException
{
Socket c = new Socket("localhost", 8000);
BufferedReader socIn=new BufferedReader(new InputStreamReader(c.getInputStream()));
String str;
while(true)
{
str = socIn.readLine();
System.out.println("Message Received: " + str);
}
} }
```

OUTPUT:





RESULT:

Thus the one way communication using TCP program executed and verified successfully

Ex. No: 10 **REMOTE METHOD INVOCATION**

AIM:

To write a java program to design and implement RMI

ALGORITHM:

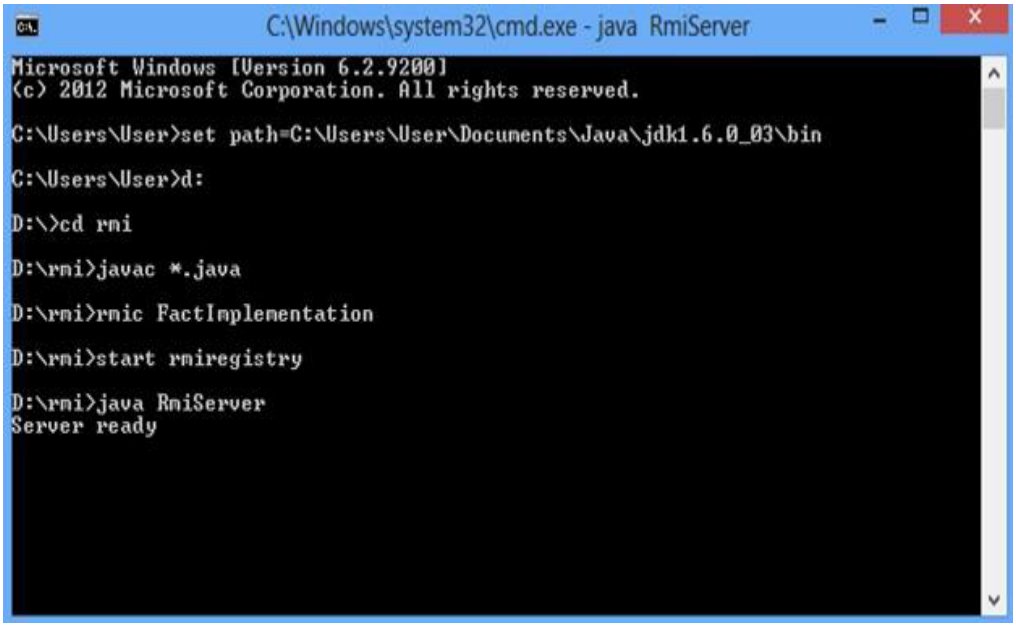
1. Create an interface named FactInterface
2. Declare fact method in the FactInterface
3. Create FactImplementation class by implementing FactInterface interface and define fact method
4. Create RmiServer class and create an object for FactImplementation class
5. Bind that object in the rmiregistry using Naming.rebind() method and display server ready
6. Create RmiClient class and lookup into the server's registry for the object using Naming.lookUp() method
7. Get the object from server and typecasting it as reference
8. Using that object, invoke fact method and display the output
9. Stop the program

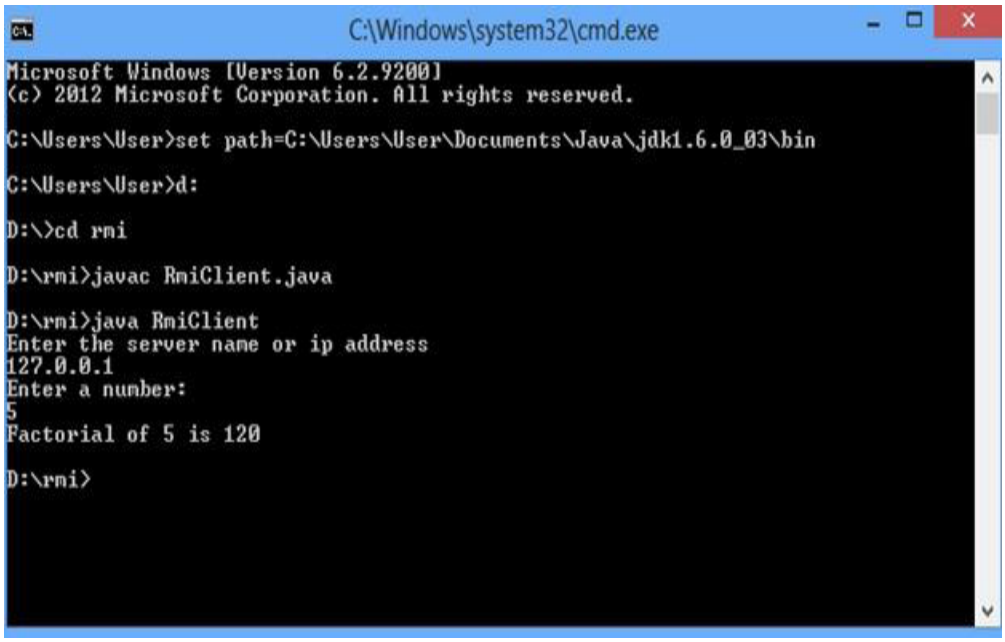
CODING:

```
//Fact Interface
import java.rmi.*;
import java.rmi.server.*;
public interface FactInterface extends Remote
{
public int fact(int n) throws RemoteException;
}
//Fact Implementation
import java.rmi.*;
import java.rmi.server.*;
public class FactImplementation extends UnicastRemoteObject implements FactInterface
{
public FactImplementation() throws RemoteException
{ }
public int fact(int n) throws RemoteException
{
int f = 1;
for(int i = 1; i <= n; i++)
{
f = f * i;
}
return f;
}
}

//RMI Server
import java.rmi.*;
import java.net.*;
public class RmiServer
{
public static void main(String args[]) throws RemoteException
{
try
{
```

```
FactImplementation fi = new FactImplementation();
Naming.rebind("Server", fi);
System.out.println("Server ready");
}
catch(Exception e)
{
System.out.println("Exception:" + e);
}
}
}//RMI client
import java.rmi.*;
import java.io.*;
public class RmiClient
{
public static void main(String args[]) throws RemoteException
{
try
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the server name or ip address");
String url = br.readLine();
String host="rmi://" + url + "/server";
FactInterface serv = (FactInterface) Naming.lookup(host);
System.out.println("Enter a number:");
int n = Integer.parseInt(br.readLine());
int fact = serv.fact(n);
System.out.println("Factorial of " + n + " is " + fact);
}
catch(Exception e)
{

System.out.println("Error " + e);

}

 }

 }
```

OUTPUT:





RESULT:

Thus the RMI using factorial method program executed and verified successfully

Ex. No: 11 **AWT CONCEPTS**

AIM:

To write a java program to implement AWT concepts.

ALGORITHM:

1. Create a AwtControl class which extends the Frame class
2. Set the grid layout with labels ,text boxes, dropdown boxes and check boxes
3. Add two buttons, Show and Exit with the addActionListener() method.
4. Then add the components such as labels, textboxes, dropdown boxes and check boxes.
5. Perform the actions in the actionPerformed() method.
6. We can set the size of the layout using setSize() method.
7. Create the main() function and the object for the AwtControl class and call the methods using the objects.
8. Finally compile and run the program

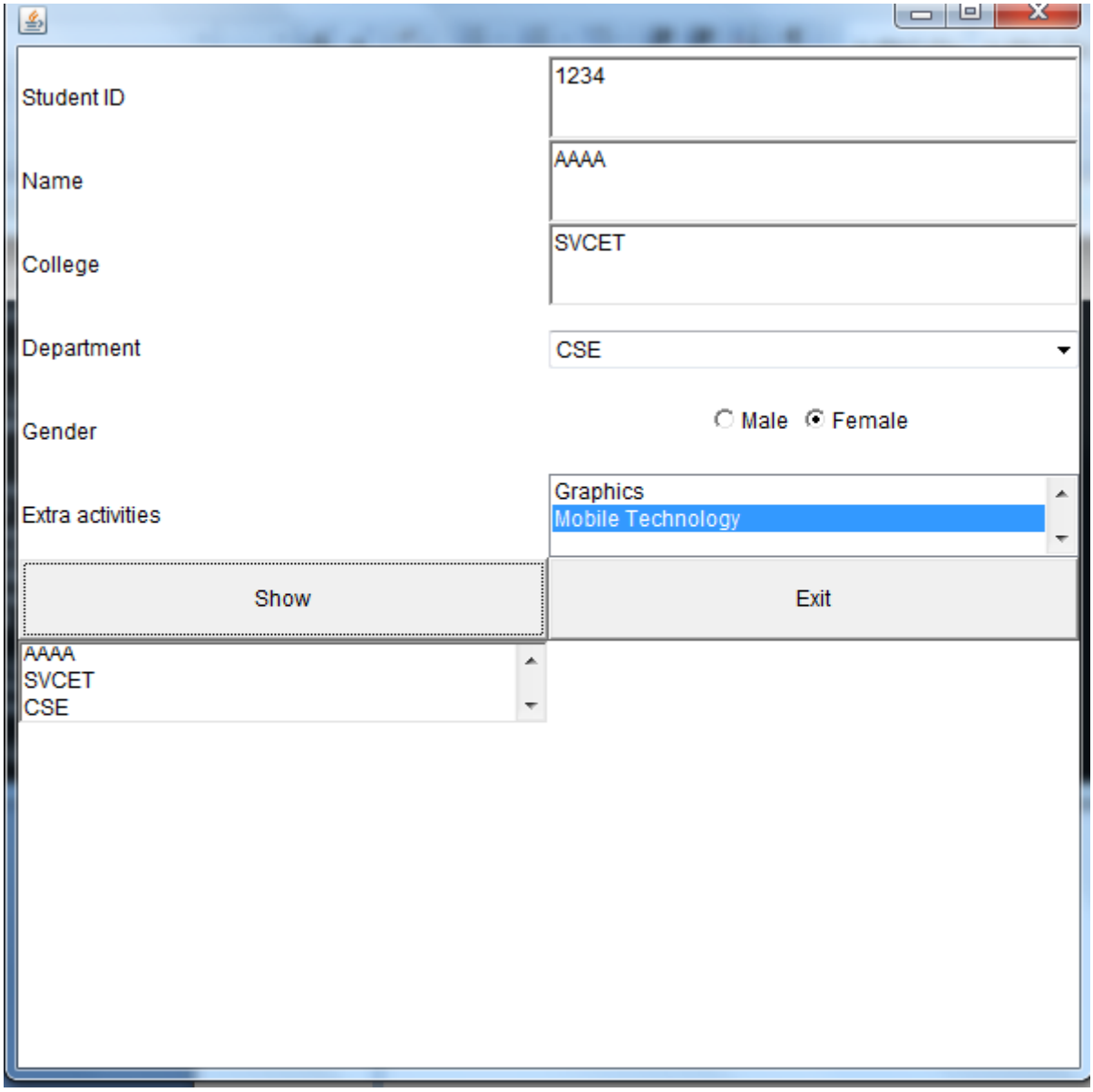
CODING:

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
public class AwtControl extends Frame implements ActionListener
{
Label l1, l2, l3, l4, l5, l6, l7;
TextField t1, t2, t3, t4;
Choice ch1;
Checkbox cb1, cb2;
CheckboxGroup cbg;
List It;
Button b1, b2;
Panel p1;
TextArea ta;
public AwtControl()
{
setLayout(new GridLayout(12, 2));
ta = new TextArea( 10, 20);
l1 = new Label("Student ID ");
l2 = new Label("Name ");

l3 = new Label("College");
l4 = new Label("Department");
l5 = new Label("Gender ");
l6 = new Label("Extra activities");
t1 = new TextField(20);
t2 = new TextField(20);
t3 = new TextField(20);
ch1 = new Choice();
ch1.addItem("CSE");
ch1.addItem("IT");
ch1.addItem("MCA");
ch1.addItem("ECE");
cbg = new CheckboxGroup();
cb1 = new Checkbox("Male", cbg, true);
```

```
cb2 = new Checkbox("Female", cbg, true);
p1 = new Panel();
p1.add(cb1);
p1.add(cb2);
It = new List();
It.addItem("Sports");
It.addItem("Graphics");
It.addItem("Mobile Technology");
b1 = new Button("Show");
b1.addActionListener(this);
b2 = new Button("Exit");
b2.addActionListener(this);
add(l1);
add(t1);
add(l2);
add(t2);
add(l3);
add(t3);
add(l4);
add(ch1);
add(l5);

add(p1);
add(l6);
add(It);
add(b1);
add(b2);
add(ta);
}

public void actionPerformed(ActionEvent e)
{
if(e.getSource()==b2)
{
System.exit(0);
}
ta.setText(""); //Clear
ta.append(t1.getText() + "\n");
ta.append(t2.getText()+ "\n");
ta.append(t3.getText()+ "\n");
ta.append(ch1.getSelectedItem()+ "\n");
ta.append(cbg.getSelectedCheckbox().getLabel()+ "\n");
ta.append(It.getSelectedItem()+ "\n");
}
public void windowClosing(WindowEvent e)
{
dispose();
System.exit(0);
}
public static void main(String args[])
{
AwtControl control = new AwtControl();
control.setSize(600,600);
```

```
control.setVisible(true);
}
}
```

OUTPUT:



RESULT:

Thus the AWT program executed and verified successfully

Ex. No: 12 **SWING CONCEPTS**

AIM:

To write a java program to implement swing concepts.

ALGORITHM:

1. Create the class StudentSwing which extends the the class JFrame and implements the interface ActionListener.
2. Create a panel and add the label, text fields, combo boxes and buttons to it.
3. Obtain the input for roll number, name, department, and marks from the user.
4. Check if the marks are greater than 50 in the calculate() method.
5. If it is true, print “You have Cleared” else print “Better Luck next time”.
6. Create an object for the class and call the methods in the main() function.
7. Finally compile and run the program to verify the result

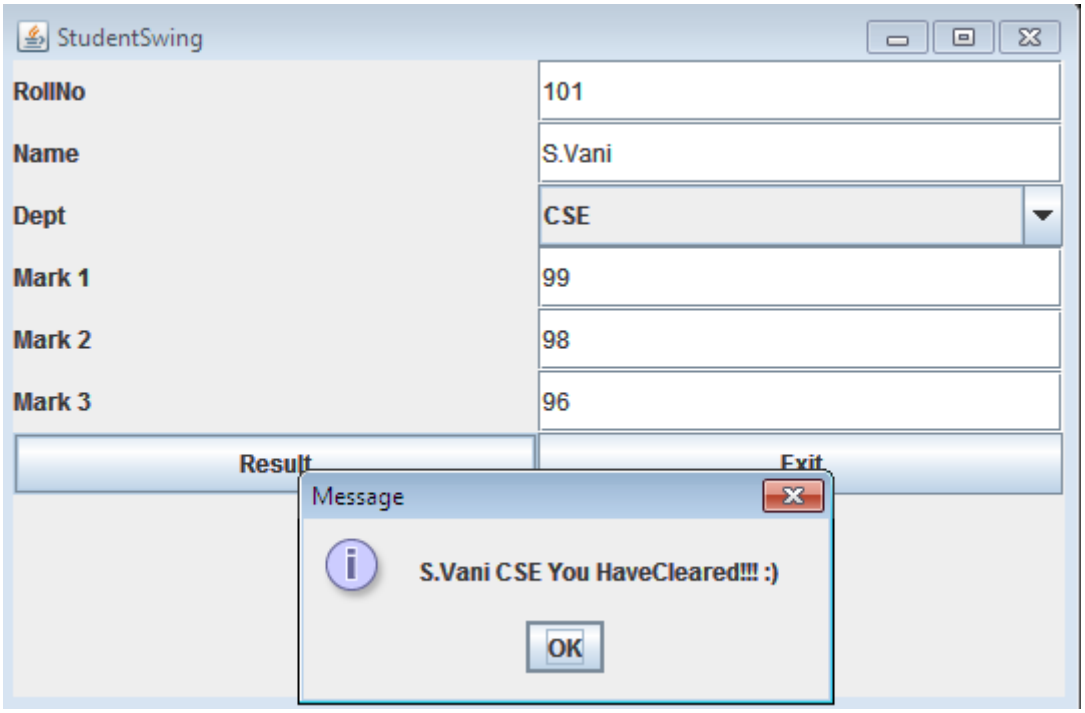
CODING:

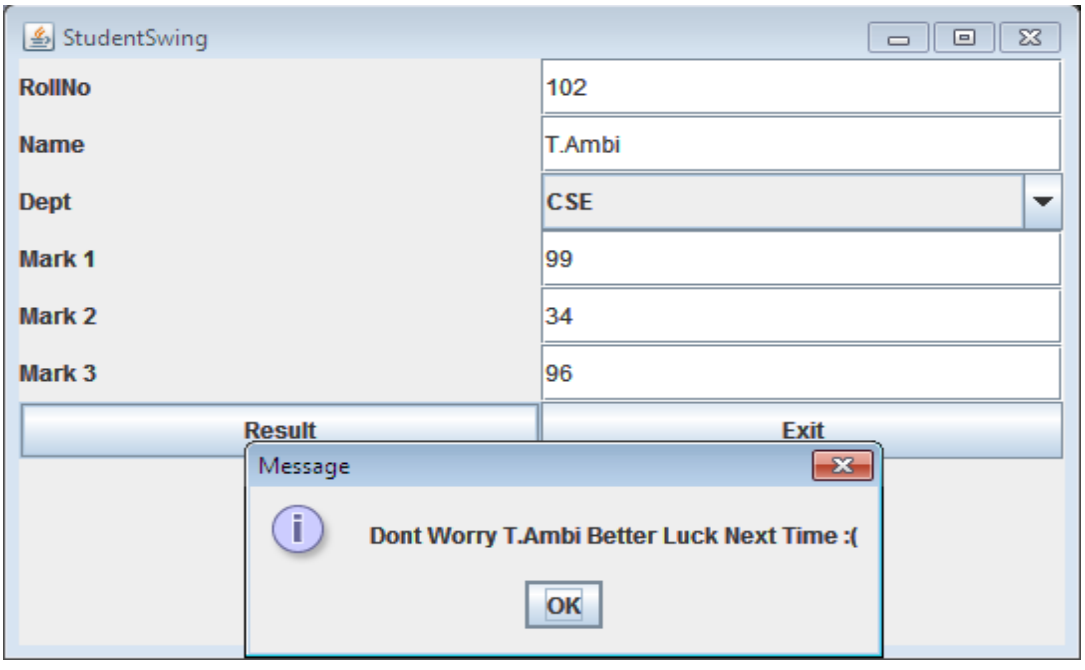
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class StudentSwing extends JFrame implements ActionListener
{
JLabel lblRollno,lblName,lblDept,lblMark1,lblMark2,lblMark3;
JComboBox comboDept;
JTextField txtRollno,txtName,txtDept,txtMark1,txtMark2,txtMark3;
JButton result,exit;
StudentSwing()
{
super("StudentSwing");
JPanel panel = new JPanel(new GridLayout(10, 2));
lblRollno= new JLabel("RollNo");
panel.add(lblRollno);
txtRollno= new JTextField();
panel.add(txtRollno);
lblName= new JLabel("Name");
panel.add(lblName);
txtName= new JTextField();
panel.add(txtName);
lblDept= new JLabel("Dept");

panel.add(lblDept);
String[] dept={"CSE","IT"};
comboDept= new JComboBox(dept);
comboDept.addItem("ECE");
comboDept.addItem("EEE");
panel.add(comboDept);
lblMark1= new JLabel("Mark 1");
panel.add(lblMark1);
txtMark1= new JTextField();
panel.add(txtMark1);

lblMark2= new JLabel("Mark 2");
panel.add(lblMark2);
txtMark2= new JTextField();
```

```
panel.add(txtMark2);
lblMark3= new JLabel("Mark 3");
panel.add(lblMark3);
txtMark3= new JTextField();
panel.add(txtMark3);
result = new JButton("Result");
panel.add(result);
result.addActionListener(this);

exit = new JButton("Exit");
panel.add(exit);
exit.addActionListener(this);
add(panel);
setSize(800, 600);
this.setVisible(true);
}public void actionPerformed(ActionEvent e)
{
if(e.getSource()==result)
{
Calculate();
}
else if(e.getSource()==exit)
{
this.dispose();

}
}
public void Calculate()
{
try
{
int rollno = Integer.parseInt(txtRollno.getText());
String name = txtName.getText();
String department = comboDept.getSelectedItem().toString();
int mark1 = Integer.parseInt(txtMark1.getText());
int mark2 = Integer.parseInt(txtMark2.getText());
int mark3 = Integer.parseInt(txtMark3.getText());
if(mark1 >=50 && mark2 >=50 && mark3 >=50)
{
JOptionPane.showMessageDialog(this, name +" " + department + " You HaveCleared!!! :) ")
}
else
{
JOptionPane.showMessageDialog(this, "Dont Worry " + name + " Better Luck Next Time :(");
}
}
catch (Exception ex)
{
JOptionPane.showMessageDialog(this, ex);
}
}
public static void main(String[] args)
{
```

```
StudentSwing ss = new StudentSwing();
ss.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
} }
```

OUTPUT:





RESULT:

Thus the swing application program executed and verified successfully

Ex. No: 13 **APPLET**

AIM:

To write a java program to design and implement applet.

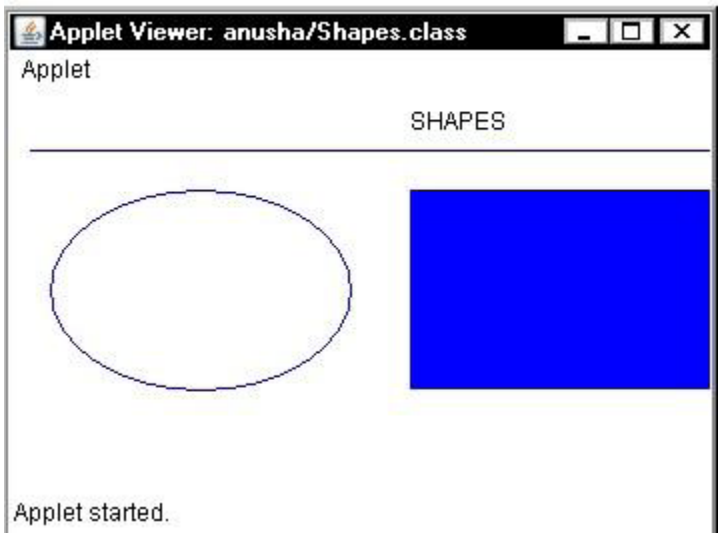
ALGORITHM:

1. Create a class Shape which extends the class Applet.
2. Initialize the thread using init() method and start it using start() method.
3. Draw different shapes such as oval, line and rectangle in the paint() method.
4. The stop() method is used to stop the thread
5. The destroy() method is used to destroy the thread.
6. Compile and run the program to verify the result

CODING:

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Shapes" width=450 height=250>
</applet>
*/
public class Shapes extends Applet
{
public void init()
{
System.out.println("Initialize");
}
public void start()
{
System.out.println("Start");
}
public void paint(Graphics g)
{
g.drawString("SHAPES",200,20);
g.setColor(Color.blue);
g.drawLine(10,30,400,30);
g.drawOval(20,50,150,100);
g.fillRect(200,50,200,100);
}
public void stop()
{
System.out.println("Stop");
}
public void destroy()
{
System.out.println("Destroy");
}
}
```

OUTPUT:



RESULT:

Thus the applet program executed and verified successfully

Ex. No: 14 **JDBC**

AIM:

To write a java program to design and implement JDBC

ALGORITHM:

1. Create a class StudentSwingJdbc which extends the class JFrame and implements the ActionListener.
2. Create the panel using JPanel.
3. Use textboxes and labels for rollno, name,department, mark1, mark2, mark3.
4. Add the buttons „Save‟ and „Exit‟.
5. Use the JDBC database connection in the AddStudent() method to add the new student details by using insert command.
6. Check if the marks are greater than 50 and if it is true, print the result as “PASS” and if it is false, print the result as “FAIL”.
7. Create the object for the class in the main() method.
8. Now compile and run the program to see the result.

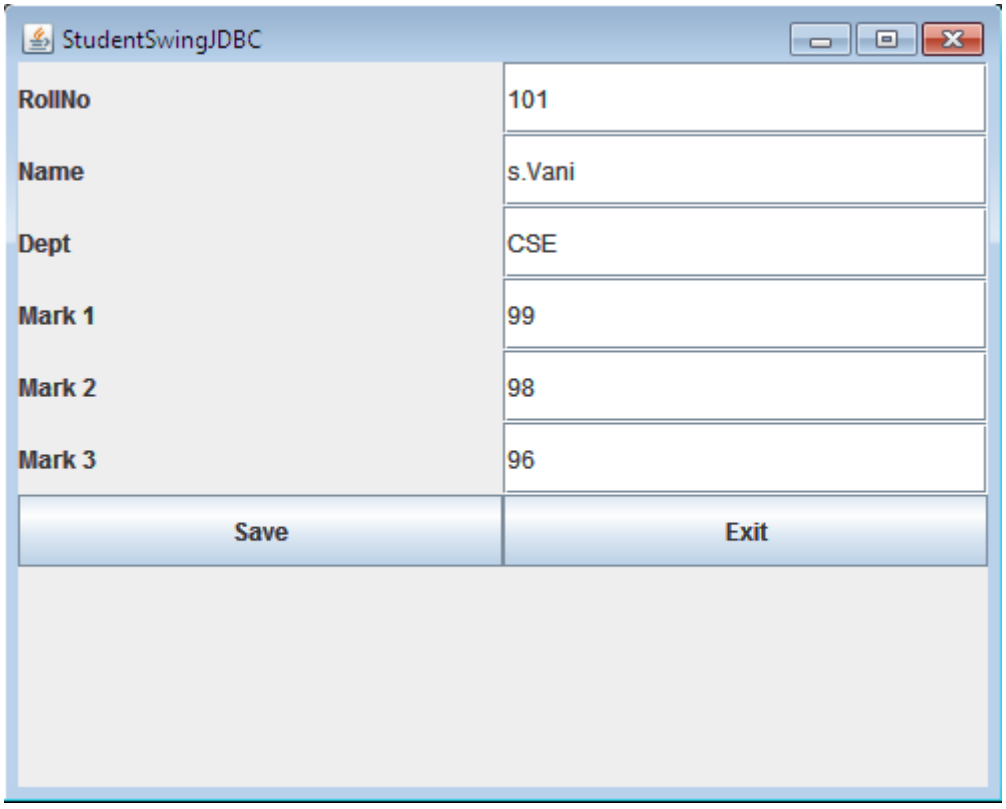
CODING:

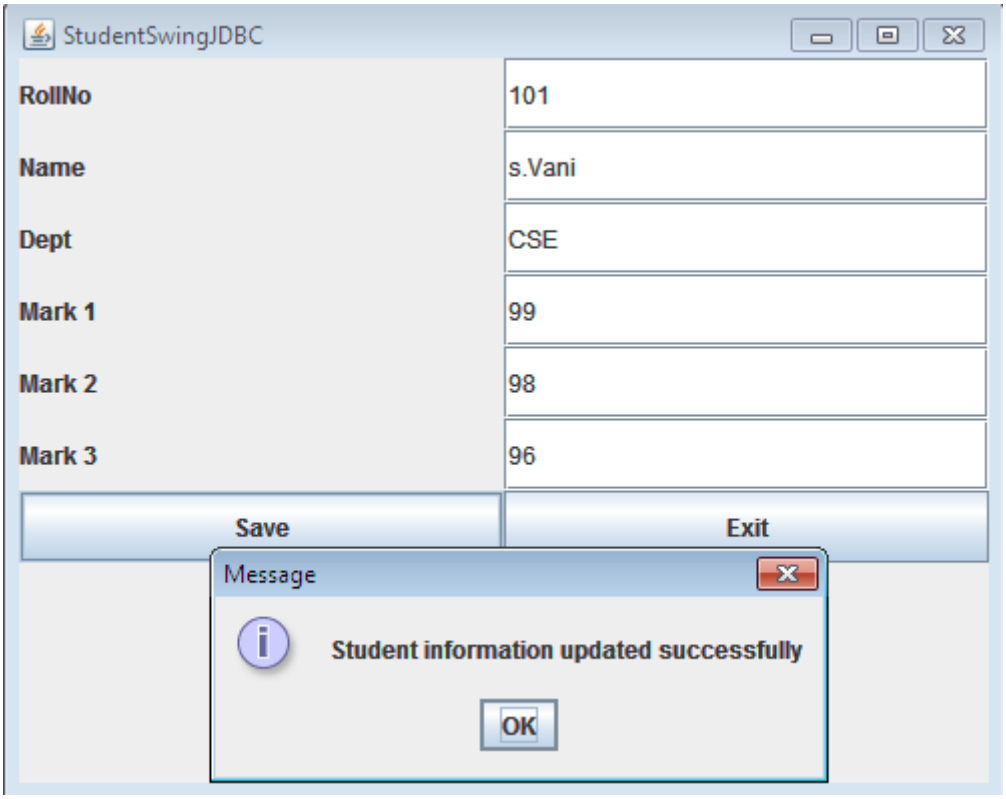
```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.sql.*;
public class StudentSwingJDBC extends JFrame implements ActionListener
{
JLabel lblRollno,lblName,lblDept,lblMark1,lblMark2,lblMark3;
JTextField txtRollno,txtName,txtDept,txtMark1,txtMark2,txtMark3;
JButton save,exit;
StudentSwingJDBC()
{
super("StudentSwingJDBC");
JPanel panel = new JPanel(new GridLayout(10, 2));
lblRollno= new JLabel("RollNo");
panel.add(lblRollno);
txtRollno= new JTextField();
panel.add(txtRollno);
lblName= new JLabel("Name");
panel.add(lblName);
txtName= new JTextField();
panel.add(txtName);

lblDept= new JLabel("Dept");
panel.add(lblDept);
txtDept= new JTextField();
panel.add(txtDept);
lblMark1= new JLabel("Mark 1");
panel.add(lblMark1);
txtMark1= new JTextField();
panel.add(txtMark1);
lblMark2= new JLabel("Mark 2");
panel.add(lblMark2);
```

```
txtMark2= new JTextField();
panel.add(txtMark2);
lblMark3= new JLabel("Mark 3");
panel.add(lblMark3);
txtMark3= new JTextField();
panel.add(txtMark3);
save = new JButton("Save");
panel.add(save);
save.addActionListener(this);
exit = new JButton("Exit");
panel.add(exit);
exit.addActionListener(this);
add(panel);
setSize(500, 400);
this.setVisible(true);
}

public void actionPerformed(ActionEvent e)
{
if(e.getSource()==save)
{
AddStudent();
JOptionPane.showMessageDialog(this, "Student information updated successfully");
}
else if(e.getSource()==exit)
{
this.dispose();
}
}public void AddStudent()
{
try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:raji", "", "");
int rollno = Integer.parseInt(txtRollno.getText());
String name = txtName.getText();
String department = txtDept.getText();
int mark1 = Integer.parseInt(txtMark1.getText());
int mark2 = Integer.parseInt(txtMark2.getText());
int mark3 = Integer.parseInt(txtMark3.getText());
int total = mark1 + mark2 + mark3;
String result;
if(mark1 >=50 && mark2 >=50 && mark3 >=50)
{ result = "Pass";}
else
{ result = "Fail"; }
PreparedStatement ps=con.prepareStatement("INSERT INTO stu (rollno, sname,
dept,mark1,mark2,mark3,total,result) VALUES (?,?,?,?,?,?,?,?)");
ps.setInt(1,rollno);
ps.setString(2,name);
ps.setString(3,department);
ps.setInt(4,mark1);
ps.setInt(5,mark2);
```

```
ps.setInt(6,mark3);
ps.setInt(7,total);
ps.setString(8,result);
ps.executeQuery();
ps.close();
con.close();
}
catch (Exception ex)
{
JOptionPane.showMessageDialog(this, ex);
}
}
public static void main(String[] args)
{
StudentSwingJDBC ss = new StudentSwingJDBC();
ss.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

OUTPUT:





RESULT:

Thus the java database connectivity program executed and verified successfully

Ex.No : 15 **EVENT HANDLING - CALCULATOR**

AIM:

To write a java program to implement event handling event for simulating a simple calculator.

ALGORITHM:

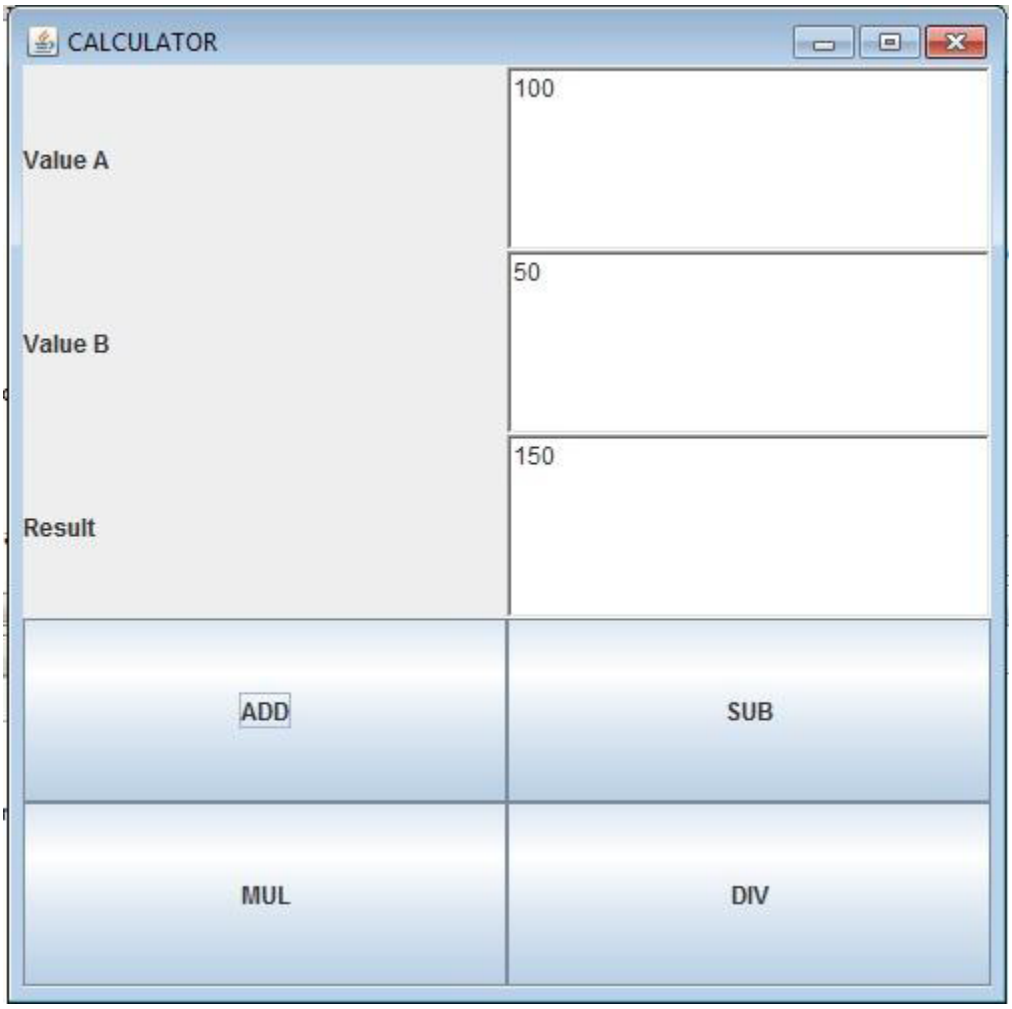
1. Create the calculator class
2. Create a frame using JFrame class and a panel using JPanel class
3. Create the labels ,text fields and buttons to the panel
4. Create the class Action that extends the class WindowAdapter and implements the interface ActionListener and TextListener.
5. Perform the addition operation in actionPerformed() method by overriding it.
6. Set the value for the second textbox if not entered in the textValueChanged() method.
7. Create the object in the main() method and call the other methods.
8. Finally compile and run the program

CODING:

```
package oopl;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Calculator
{
TextField t1,t2,t3;
JButton b1,b2,b3,b4;
JLabel l1,l2,l3;
public Calculator()
{
JFrame f=new JFrame("CALCULATOR");
f.setVisible(true);
f.setSize(500,500);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JPanel p=new JPanel();
l1=new JLabel("Value A");
l2=new JLabel("Value B");

l3=new JLabel("Result");
t1 = new TextField(10);
t2=new TextField(10);
t3=new TextField(10);
b1=new JButton("ADD");
b2=new JButton("SUB");
b3=new JButton("MUL");
b4=new JButton("DIV");
p.add(l1);
p.add(t1);
p.add(l2);
p.add(t2);
p.add(l3);
p.add(t3);
p.add(b1);
```

```
p.add(b2);
p.add(b3);
p.add(b4);
p.setLayout(new GridLayout(5,2));
f.setContentPane(p);
b1.addActionListener(new Action());
b2.addActionListener(new Action());
b3.addActionListener(new Action());
b4.addActionListener(new Action());
f.addWindowListener(new Action());
}
public static void main(String[] args)
{
Calculator c=new Calculator();
}
private class Action extends WindowAdapter implements ActionListener
{

public void actionPerformed(ActionEvent ae)

{
int a=Integer.parseInt(t1.getText());
int b=Integer.parseInt(t2.getText());
if(ae.getSource()==b1)
{
int c= a+b;
t3.setText(""+c);
}
if(ae.getSource()==b2)
{
int c= a-b;
t3.setText(""+c);
}
if(ae.getSource()==b3)
{
int c= a*b;
t3.setText(""+c);
}
if(ae.getSource()==b4)
{
float c= a/b;
t3.setText(""+c);
}
}

public void windowClosing(WindowEvent we)
{
System.exit(0);
}
}
}
```

OUTPUT:



RESULT:

Thus the simulation of calculator program executed and verified successfully