Term work

of

Java Programming Lab (PCS-408)

Submitted in partial fulfillment of the requirement for the IV semester

Bachelor of Technology

By

Rohan kumar

University Roll No

2061953

Under the Guidance of

Mr. Ravindra Koranga

Assistant Professor

Deptt. of CSE

Graphic Era
HILL UNIVERSITY
University under section 2(f) of UGC Act, 1956
Bhimtal Campus

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI

DISTRICT- NAINITAL-263132

2022-2023

CERTIFICATE

The term work of **Java Programming Lab (PCS-408),** being submitted by **Rohan kumar**, Roll no **20619333** to Graphic Era Hill University Bhimtal Campus for the award of bona fide work carried out by him/her. He/She has worked under my guidance and supervision and fulfilled the requirement for the submission of this work report.

(……)

Ravindra Koranga

ACKNOWLEDGEMENT

I take immense pleasure in thanking **Mr. Ravindra Koranga** (**Assistant Professor,** CS, GEHU Bhimtal Campus) for allowing us to carry out this project work under his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that have helped me in developing my subject concepts as a student.

I want to extend thanks to our President **"Prof. (Dr.) Kamal Ghanshala"** for providing us all infrastructure and facilities to work in need without which this work would not be possible.

ROHAN KUMAR

STUDENT'S DECLARATION

I, **Rohan Kumar** hereby declare the work, which is being presented in the report, entitled **Term work of Java Programming Lab (PCS-408)** in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (Computer Science)** in the session **2022-2023** for **semester IV**, is an authentic record of my own work carried out under the supervision of **Mr. Ravindra Koranga** (Assistant professor, Graphic Era Hill University, Bhimtal)

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date: …… …….

(Full signature of student)

Graphic Era
HILL UNIVERSITY
University under section 2(f) of UGC Act, 1956
Bhimtal Campus

Computer Science and Engineering Department
Java Programming LAB (PCS-408)

Index/List of Experiments

1.	WAP to show use of Command Line Arguments
2.	WAP to show the use of Scanner class to take input from user. Use Scanner class to take one int, one char and one line as input from user and print the output
3.	WAP to create and use function in java. Create an add function to add two variables and print them.
4.	WAP to show the use of class, object and constructor in java. Create a box class with variables length, width, height and functions set(), get(), volume().
5.	WAP to show function overloading in java.
6.	WAP to show the use of static keyword in java.
7.	WAP to show inheritance in java.
8.	WAP to show function overriding in java.
9.	WAP to show constructor chaining in java.
10.	WAP to show the use of super keyword in java.
11.	WAP to show the use of super() method in java.
12.	WAP to show the use of abstract class in java.
13.	WAP to show the use of interface in java.
14.	WAP to show how exception handling is done using try and catch block.
15.	WAP to show the use of throw and throws keyword in java.
16.	WAP to show to create a thread in java.
17.	WAP to show producer consumer problem in java.
18.	WAP to write some text to a file and then read this file and print it.
19.	WAP to create a generic class in java.
20.	WAP to show the use of ArrayList in java Collections.
21.	WAP to show event handling. Handle the mouse events using anonymous inner class.
22.	WAP to create a Swing GUI application. Take two input from user and print the difference.
23.	WAP to show the Database connectivity using jdbc driver in java. Connect to a Database and print the table in the database.

1. **WAP to show use of Command Line Arguments.**

```
class demo{
public static void main(String ar[]){



System.out.println("The first command line argument is "+ ar[0]);
System.out.println("The second command line argument is "+ ar[1]);


}
}
```

Output:-

2. **WAP to show the use of Scanner class to take input from user. Use Scanner class to take one int, one char and one line as input from user and print the output.**

```
//input one int and then one line from user
import java.util.*;
class demoscanintstring{
public static void main(String ar[]){


int a;
char c;
String s1;


Scanner scobj=new Scanner(System.in);


a=scobj.nextInt();
System.out.println(a);
```

```
            c=scobj.nex().charAt(0);
            System.out.println(c);

            s1=scobj.nextLine();
            s1=scobj.nextLine();//to remove new line char
            System.out.println(s1);



            //a=scobj.nextInt();
            //System.out.println(a);

        }
}
```

3. WAP to create and use function in java. Create an add function to add two variables and print them.

```
 //create an add fun to add two numbers
import java.util.Scanner;
class demoadd{
        static void add(int x, int y){ //it should be static and inside demo class
            int sum=x+y;
            System.out.println("The sum is "+ sum);
        }



        public static void main(String ar[]){

            int a,b;
            Scanner scobj=new Scanner(System.in);
```

```
        a=scobj.nextInt();
        b=scobj.nextInt();
        add(a,b);
    }
}
```

4. WAP to show the use of class, object and constructor in java. Create a box class with variables length, width, height and functions set(), get(), volume().

```
class box{
private int l,w,h;

box(int x,int y,int z){
        l=x;
        w=y;
        h=z;
    }

    void set(int x, int y, int z){
        l=x;
        w=y;
        h=z;
    }

    void get(){
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);
    }
```

```
    void vol(){
        System.out.println(l*w*h);
    }
}

class demobox{
    public static void main(String ar[]){

        box obj1, obj2;
        obj1=new box(10,20,30);

        obj1.get();
        obj1.vol();

    }
}
```

5. WAP to show function overloading in java.

```
import java.util.Scanner;
class demooverloading{
    static void add(int x, int y){
        System.out.println(x+y);
    }

    static void add(int x, int y, int z){
        System.out.println(x+y+z);
```

```
    }

    public static void main(String ar[]){

        int a,b,c;
        Scanner scobj=new Scanner(System.in);
        a=scobj.nextInt();
        b=scobj.nextInt();
        c=scobj.nextInt();
        add(a,b);
        add(a,b,c);
    }
}
```

6. WAP to show the use of static keyword in java.

```
 class box{
    private static String color;
    private static int l, w, h;

    static {
        System.out.println("static block created");
        color="Red";
    }
    public static void set(int x, int y, int z){
        l=x;w=y;h=z;
    }

    public static void get(){
        System.out.println(l);
```

```
            System.out.println(w);
            System.out.println(h);
            System.out.println(color);
        }




}
class demo{
        public static void main(String ar[]){

        //box obj1=new box();
        box.get();

        box obj1=new box();
        obj1.get();

        }
}
```

7. WAP to show inheritance in java.

```
 class box{
        protected I       nt l,w,h;
        public void set(int x, int y, int z){
              l=x;w=y;h=z;
        }

        public void get(){
              System.out.println(l);
```

```
            System.out.println(w);
            System.out.println(h);
        }
        public void vol(){
            System.out.println(l*w*h);
        }
}
class d1 extends box{
        protected int wt;
        public void set(int x, int y, int z, int u){
            l=x;w=y;h=z;wt=u;
        }

        public void get(){
            System.out.println(l);
            System.out.println(w);
            System.out.println(h);
            System.out.println(wt);
        }

        public void density(){
            System.out.println(wt/(l*w*h));
        }

}
class d2 extends d1{
        protected String col;
        public void set(int x, int y, int z, int u, String s){
            l=x;w=y;h=z;wt=u;col=s;
        }
```

```
    public void get(){
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);

        System.out.println(wt);
        System.out.println(col);
    }
}
class demo{
    public static void main(String ar[]){
        box obj_b=new box();
        d1 obj_d1=new d1();
        d2 obj_d2=new d2();

        obj_d2.set(1,2,3,4,"red");
        obj_d2.get();
        obj_d2.set(4,5,6);
        obj_d2.get();
        obj_d2.vol();
        obj_d2.density();
    }
}
```

8. WAP to show function overriding in java.

```
 class box{
    protected int l,w,h;
    public void set(int x, int y, int z){
        l=x;w=y;h=z;
    }
```

```
	public void get(){
		System.out.println(l);
		System.out.println(w);
		System.out.println(h);
	}
	public void vol(){
		System.out.println(l*w*h);
	}
}
class d1 extends box{
	protected int wt;
	public void set(int x, int y, int z, int u){
		l=x;w=y;h=z;wt=u;
	}

	public void get(){
		System.out.println(l);
		System.out.println(w);
		System.out.println(h);
		System.out.println(wt);
	}

	public void density(){
		System.out.println(wt/(l*w*h));
	}

}
class d2 extends d1{
	protected String col;
```

```
	public void set(int x, int y, int z, int u, String s){
		l=x;w=y;h=z;wt=u;col=s;
	}
	public void get(){
		System.out.println(l);
		System.out.println(w);
		System.out.println(h);

		System.out.println(wt);
		System.out.println(col);
	}
}
class demo{
	public static void main(String ar[]){
		box obj_b=new box();
		d1 obj_d1=new d1();
		d2 obj_d2=new d2();

		box ref;

		ref=obj_d2;
		ref.set(1,2,3);
		ref.get();
	}
}
```

9. WAP to show constructor chaining in java.

```
//constructor chaining with non parameterized constructor
//Non parameterized constructors are created
class box {
```

```
        protected int l, w, h;

        //Non parameterized Constructor
        box(){
            System.out.println("box called");
        }

        public void set(int x, int y, int z) {
            l=x;
            w=y;
            h=z;
        }

        public void get() {
            System.out.println(l);
            System.out.println(w);
            System.out.println(h);

        }

}

class d1 extends box{
        protected int wt;

        //non param constructor in d1
        //d1 will automatically call box()
        d1(){
            System.out.println("d1 called");
        }
```

```
    public void set(int x, int y, int z, int u) {
        l=x;
        w=y;
        h=z;
        wt=u;
    }

    public void get() {
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);
        System.out.println(wt);

    }


}

class d2 extends d1{
    protected String col;

    //non parameterized in d2
    //d2 will automatically call d1()
    d2(){
        System.out.println("d2 called");
    }
    public void set(int x, int y, int z, int u, String s) {
        l=x;
```

```
        w=y;
        h=z;
        wt=u;
        col=s;
    }

    public void get() {
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);
        System.out.println(wt);
        System.out.println(col);

    }


}

class demo{

    public static void main(String args[]) {


        //When obj of d2 is created,non parameterized constructor d2() is called.
        //d2() calls d1()
        //d1() calls box()
        //order: top to bottom

        d2 obj_d2=new d2();//all constr will be called
        obj_d2.set(1,2,3,4,"red");
```

```
            obj_d2.get();
        }
}
```

10. WAP to show the use of super keyword in java.

```
 //super keyword is used when base and derived class has variable or functions with same name.
//super is used to access members of base class with same name
//set function has same name in all classes. so set function of base class can be called by using super.set()
class box {
        protected int l, w, h;


        //set in box class
        public void set(int x, int y, int z) {
                l=x;
                w=y;
                h=z;
        }

        public void get() {
                System.out.println(l);
                System.out.println(w);
                System.out.println(h);

        }

}

class d1 extends box{
        protected int wt;
```

```
        //set in d1 class.
        //To call set function of base class use super keyword
        public void set(int x, int y, int z, int u) {
            super.set(x,y,z);//call set function of box class
            wt=u;//initialize own variables only
        }

        public void get() {
            System.out.println(l);
            System.out.println(w);
            System.out.println(h);
            System.out.println(wt);

        }


}

class d2 extends d1{
        protected String col;

        //set in d2 class
        public void set(int x, int y, int z, int u, String s) {
            super.set(x,y,z,u);//use super to call set in d1 class
            col=s;//initialize own variable
        }

        public void get() {
```

```
            System.out.println(l);
            System.out.println(w);
            System.out.println(h);
            System.out.println(wt);
            System.out.println(col);

        }


}

class demo{

        public static void main(String args[]) {




            d2 obj_d2=new d2();
            obj_d2.set(5,6,7,8,"red");// all constructor will be called
            obj_d2.get();
        }
}
```

11. WAP to show the use of super() method in java.

```
//constructor chaining with parameterized constructor
//parameterized constructors are created
//When base class has a parameterized constructor, derived class constructor must call super()
class box {
        protected int l, w, h;
```

```
        //parameterized Constructor
        box(int x, int y, int z){
                l=x;
                w=y;
                h=z;
        }

        public void set(int x, int y, int z) {
                l=x;
                w=y;
                h=z;
        }

        public void get() {
                System.out.println(l);
                System.out.println(w);
                System.out.println(h);

        }

}

class d1 extends box{
        protected int wt;

        //parameterized constructor in present in d1
        //d1 must call super because base class box has a parameterized constructor
        d1(int x, int y, int z, int u){
                super(x, y, z);//call and pass values to base constructor box()
                wt=u;//initialize own variable
```

```
    }


    public void set(int x, int y, int z, int u) {
        l=x;
        w=y;
        h=z;
        wt=u;
    }

    public void get() {
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);
        System.out.println(wt);

    }


}

class d2 extends d1{
    protected String col;

    //parameterized is present in d2
    //d2() must call super(), because base class has a parameterized constructor
    d2(int x, int y, int z, int u, String s){
        super(x,y,z,u);//call and pass values to d1() constructor
        col=s;//initialize own variable
```

```
    }
    public void set(int x, int y, int z, int u, String s) {
        l=x;
        w=y;
        h=z;
        wt=u;
        col=s;
    }

    public void get() {
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);
        System.out.println(wt);
        System.out.println(col);

    }


}

class demo{

    public static void main(String args[]) {


        //When obj of d2 is created,parameterized constructor d2() is called.
        //d2() calls d1() through super()
        //d1() calls box() through super()
        //order: top to bottom
```

```
        d2 obj_d2=new d2(10,20,30,40,"red");//all constr will be called
        obj_d2.get();
    }
}
```

12. WAP to show the use of abstract class in java.

```
//abstract class are classes with atleast one abstract functions.
//it can have abstract as well as concrete functions.
//abstract classes are meant to be inherited.
//abstract classes are used to provide abstraction. The derived class must define the abstract functions in abstract class

//show the use of abstract, concrete, final, static, constructor in abstract class

abstract class base{

   //abstract function in base
   abstract public void get(); // it will not be defined here. box class must define it

   //concrete(defined) function in base. It is declared as final, static
   final public static void fun(){
      System.out.println("Stati Fun called");
   }

   //constructor of abstract class
   base(){
      System.out.println("constructor of abstract class called");
   }
```

```
}

class box extends base{

    protected int l,w,h;

    public void set(int x, int y, int z){
        l=x;w=y;h=z;
    }
    //since box extends base, it must define this function.
    public void get(){
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);
    }
}

class demo{
    public static void main( String args[]){
        //create an obj of box class. Obj of base class can't be created
        box obj=new box();//when obj of derived class is created base constructor is called because of constructor chaining.
        obj.set(1,2,3);
        obj.get();

        //fun is static function.
        base.fun();

    }
```

}

13. WAP to show the use of interface in java.

```
 // how to create and use an interface
interface myinterface{
    //interface contains undefined functions
    public void set(int x, int y , int z);
    public void get();
}

class box implements myinterface{

    //class must implement/define set and get.
    private int l,w,h;

    //set must be public
    public void set(int x, int y , int z){
        l=x;w=y;h=z;
    }
    //get must be public
    public void get(){
        System.out.println(l);
        System.out.println(w);
        System.out.println(h);

    }

    public void vol(){
        System.out.println(l*w*h);
    }
```

```
}

class demo{
    public static void main(String args[]){
        //create obj of class and call set(),get()
        box obj=new box();
        obj.set(1,2,3);
        obj.get();
    }
}
```

14. **WAP to show how exception handling is done using try and catch block.**

```
 //example to show Exception Handling
import java.util.Scanner;
class demo{
        public static void main(String args[]){

                int a;
                Scanner sc=new Scanner(System.in);
                a=sc.nextInt();
                try{
                        //Write the statement inside try block.
                        //where exception may occur.
                        System.out.println(5/a);
                }

                //specify the type of exception inside catch statement.
                catch(ArithmeticException e){
                        //Handle exception here
                        System.out.println("divide by zero caught "+e);
```

```
            }
        }
}
```

15. WAP to show the use of throw and throws keyword in java.

```
 import java.util.Scanner;
//Exception inside functions
//In this case , function throws two types of exceptions.
//ArithmeticException is unchecked exception so it may not be specified by throws
//IllegalAccessException is checked exception so it must be specified by function by throws statement.

class demo{

    public static void fun(int x)
    throws IllegalAccessException{
        if(x==0){
            //This is unchecked exception. So it may not be specified in the throws statement
            throw new ArithmeticException("msg");
        }
        if(x==1){
            //this is checked exception. so it must be specified in the throws statement.
            throw new IllegalAccessException("msg");
        }

    }
    public static void main(String args[]){

        int a;
        Scanner sc=new Scanner(System.in);
        a=sc.nextInt();
```

```
        try{
            fun(a);
        }

        catch(ArithmeticException e){
            System.out.println("arithmetic exception caught");
        }

        catch(IllegalAccessException e){
            System.out.println("Illegal access exception caught");
        }
    }
}
```

16. WAP to show to create a thread in java.

```
 //29a Creating threads of different types.
//one thread will print 1 to 100
//other thread will print a to z



//task class contains two different functions
class task{
    //thread1 obj will call this run function to print 1 to 100
    public void run() {
        for(int i=0;i<100;i++){
            System.out.println("thread1:"+i+" ");

        }
    }
```

```
        //thread2 obj will call this run2 function to print a to z
        public void run2(){
                for(char c='a';c<'z';c++){
                        System.out.println("thread2 : "+c);
                }


        }
}


//create thread1 to define one type of thread.
class thread1 extends Thread{
        //create thread obj and task obj
        Thread obj_thread;
        task obj_task;

        thread1(task obj_task){
                //initialize task obj
                this.obj_task=obj_task;

                //create thread obj and call start()
                obj_thread=new Thread(this);
                obj_thread.start();

        }
        public void run() {

                //thread1 will call run function in task class
```

```
            obj_task.run();

        }
}

//create thread2 to define other type of thread.
class thread2 extends Thread{
        //create thread and task obj
        Thread obj_thread;
        task obj_task;



        thread2(task obj_task){
                //initialize task obj
                this.obj_task=obj_task;

                //initialize thread obj and call start fun
                obj_thread=new Thread(this);
                obj_thread.start();

        }
        public void run() {

                //call run2 function of task class
                obj_task.run2();

        }
}
```

```
class demo{
	public static void main(String args[]){
		task obj=new task();//create a task obj
		thread1 obj1=new thread1(obj);//same task obj passed to both threads
		thread2 obj2=new thread2(obj);//same task obj passed to both threads



	}
}
```

17. WAP to show producer consumer problem in java.

```
//29f producer and consumer problem
//there are two threads - producer and consumer
//producer will assign values 1 to 1000 to variable x
//consumer will print the values of x

//Both threads must run alternatively.
class q{
	//shared variable n
	int x;
	int turn;// to check

	q(){
		turn='p';
	}

	synchronized public void set(int i) {

		while(turn=='c') {
```

```
            try{
                wait();
            }
            catch(InterruptedException e){

            }


        }
            System.out.println("\n producer sets "+i+" ");
            x=i;
            turn='c';
            notify();

}

synchronized public void get(){
            while(turn=='p') {

                try{
                    wait();
                }
                catch(InterruptedException e){

                }


            }
```

```
                System.out.println("Consumer gets "+x+" ");
                turn='p';
                notify();

        }

}

class thread1 extends Thread{
        Thread t;
        q obj_q;
        thread1(q po){
                t=new Thread(this);
                obj_q=po;
                t.start();
        }

        public void run(){
                for(int i=0;i<1000;i++){

                        obj_q.set(i);
                }
        }
}

class thread2 extends Thread{
        Thread t;
        q obj_q;
        thread2(q po){
```

```
        t=new Thread(this);
        obj_q=po;
        t.start();
    }

    public void run(){
        for(int i=0;i<1000;i++){

            obj_q.get();
        }
    }
}

class demo{
    public static void main(String args[]){
        q obj=new q();
        thread1 obj1=new thread1(obj);
        thread2 obj2=new thread2(obj);

        for(int i=0;i<100;i++){
            System.out.print("main ");
        }
    }
}
```

18. WAP to write some text to a file and then read this file and print it.

```
//Program to demonstrate reading and writing to a file
import java.io.*;
class demo {
```

```
    //these are Checked Exceptions specified after throws keyword.
    //FileNotFoundException is thrown while opening file
    //IOException is thrown while using read(), write() function
    public static void main(String args[])throws FileNotFoundException, IOException{

        //write this string to the file
        String s="this is line\nthis is line2";

        //create output stream to write to the file
        FileOutputStream fout=new FileOutputStream("test.txt");

        //use write function to write one char at a time
        for(int i=0;i<s.length();i++){
            fout.write(s.charAt(i));
        }

        fout.close();

        //create input stream to read the file
        FileInputStream fin=new FileInputStream("test.txt");
        int c;

        //the stream returns -1 when EOF is reached
        //read return char in ascii value
        while( (c=fin.read())!=-1 ){
            //cast to char to convert c to char.
            System.out.print((char)c);
        }
    }
}
```

19. WAP to create a generic class in java.

```
class genclass<T>{
    T obj1, obj2;

    genclass(T po1, T po2){
        obj1=po1;
        obj2=po2;
    }

    T add(){
        return obj1+obj2;
    }
}
class demo{
    public static void main(String args[]){

        genclass<Integer> obj1=new genclass<Integer>(1,2);
        obj1.add();

        genclass<String> obj2=new genclass<String> ("abc", "def");
        obj2.add();
    }
}
```

20. WAP to show the use of ArrayList in java Collections.

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

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

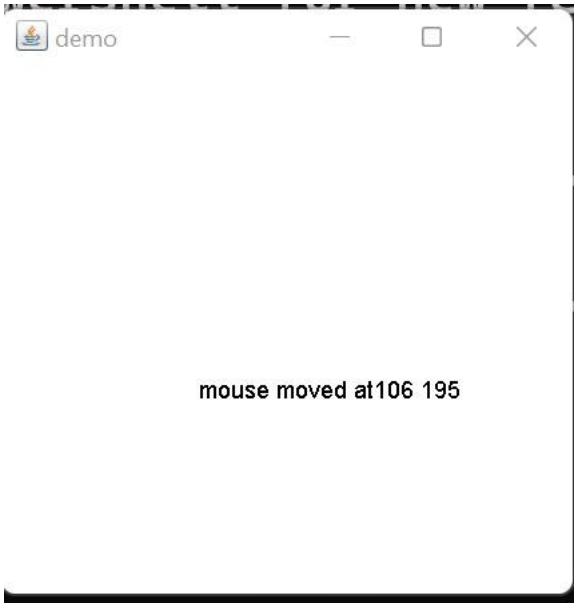
```
        al.add(10);
        al.add(20);
        al.add(30);
        al.add(40);

        System.out.println(al);

        al.remove(1);
        al.remove(Integer.valueOf(10) );

        System.out.println(al);

    }
}
```

21. WAP to show event handling. Handle the mouse events using anonymous inner class.

```
//Event handling
//using anonymous inner class
import java.awt.*;
import java.awt.event.*;

class demo extends Frame{
    String msg="";
    int x, y;

    demo(){
        //anonymous class is created inside this listener
        addMouseListener( new MouseAdapter() {
            //handler created inside anonymous inner class
            public void mouseClicked(MouseEvent me){
```

```
            x=10;
            y=100;
            msg="mouse clicked";

            repaint();

        }

});

//anonymous class is created inside this listener
addMouseMotionListener( new MouseMotionAdapter( ) {

        //handler created inside anonymous inner class
        public void mouseMoved(MouseEvent me){
            x=me.getX();
            y=me.getY();
            msg="mouse moved at"+x+" "+y;

            repaint();
        }
});

//anonymous class is created inside this listener
addWindowListener( new WindowAdapter(){
        public void windowClosing(WindowEvent we){
            System.exit(0);
        }

});
```

```
    }
    public void paint(Graphics g){
        g.drawString( msg, x,y );}

    public static void main(String args[]){
        demo app= new demo();
        app.setSize(new Dimension (300, 300));
        app.setTitle("demo");
        app.setVisible(true);
    }
}
```

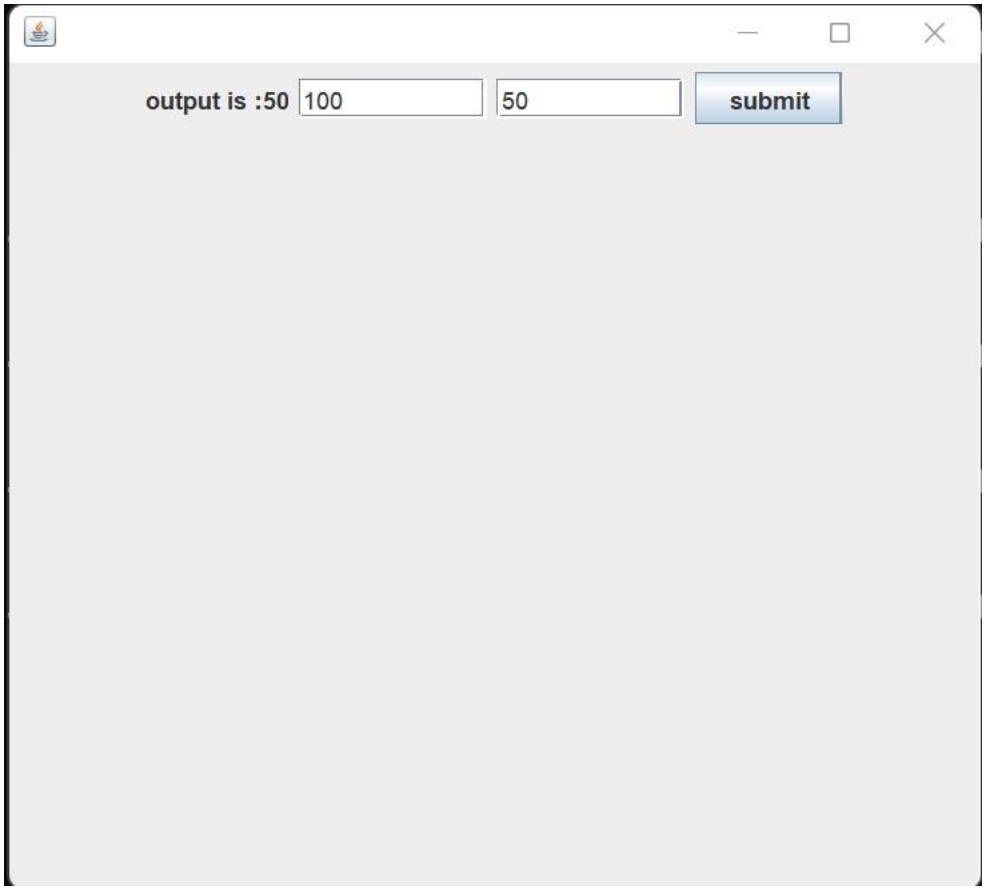


22. WAP to create a Swing GUI application. Take two input from user and print the difference.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class demo {
```

```
JLabel l;
JTextField a,b;
JButton bb;

demo(){
    JFrame jf=new JFrame();
    jf.setLayout(new FlowLayout() );
    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    jf.setSize(500,450);
    jf.setVisible(true);

    l=new JLabel("Label");
    a=new JTextField(10);
    b=new JTextField(10);
    bb=new JButton("submit");

    jf.add(l);
    jf.add(a);
    jf.add(b);
    jf.add(bb);

    bb.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){

            if( ae.getActionCommand().equals("submit") ){
                int x= Integer.parseInt(a.getText());
                int y=Integer.parseInt(b.getText() );
                int result= x-y;
```

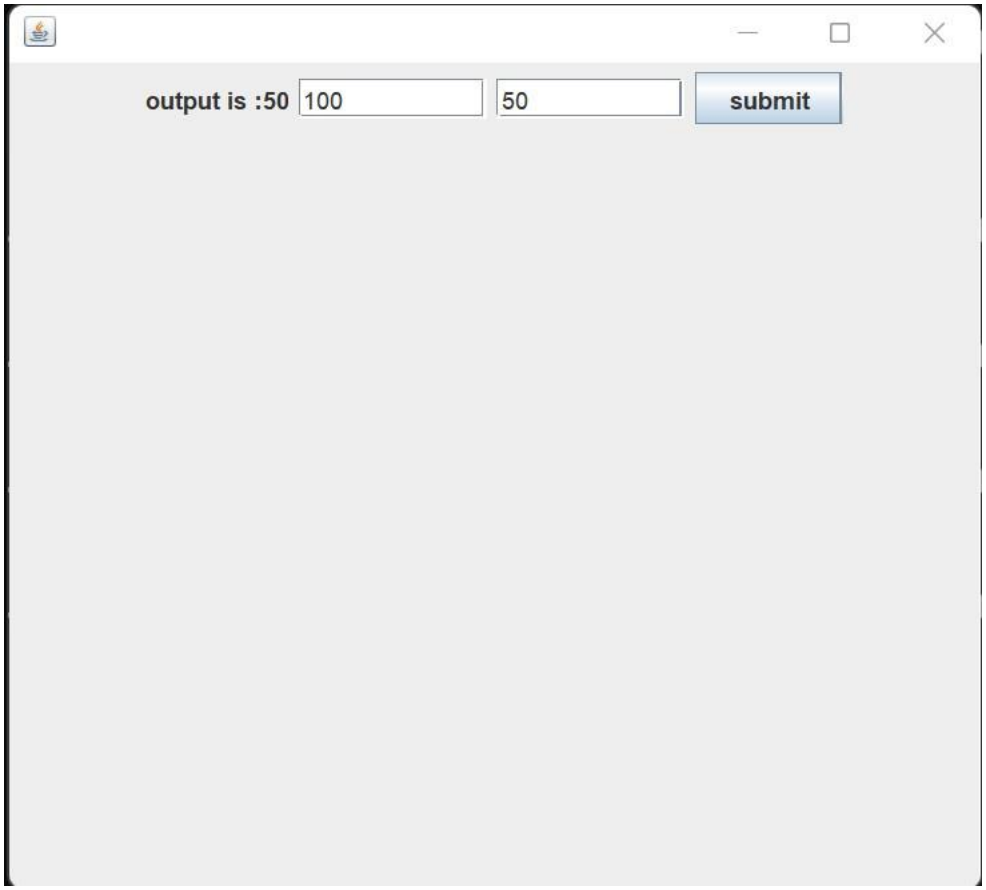
```
                    l.setText("output is :"+String.valueOf(result));
                }
            }
        }
        );


    }

        public static void main(String args[]){
            SwingUtilities.invokeLater(
                new Runnable() {
                    public void run() {
                        new demo();

                    }
                }
            );
        }

}
```



23. WAP to show the Database connectivity using jdbc driver in java. Connect to a Database and print the table in the database.

```
import java.sql.*;

class JdbcSelectTest {

public static void main(String[] args) {

    try (


      Connection conn = DriverManager.getConnection(

           "jdbc:mysql://localhost:3306/ebookshop",

           "myuser", "xxxx");   // For MySQL only


      Statement stmt = conn.createStatement();

    ) {
```

```
      String strSelect = "select title, price, qty from books";
      System.out.println("The SQL statement is: " + strSelect + "\n"); // Echo For debugging

      ResultSet rset = stmt.executeQuery(strSelect);

      System.out.println("The records selected are:");
      int rowCount = 0;

      while(rset.next()) {   // Repeatedly process each row
        String title = rset.getString("title");  // retrieve a 'String'-cell in the row
        double price = rset.getDouble("price");  // retrieve a 'double'-cell in the row
        int    qty   = rset.getInt("qty");       // retrieve a 'int'-cell in the row
        System.out.println(title + ", " + price + ", " + qty);
        ++rowCount;
      }
      System.out.println("Total number of records = " + rowCount);

    } catch(SQLException ex) {
      ex.printStackTrace();
    }
  }
}
```



23. WAP to show the Database connectivity using jdbc driver in java. Connect to a Database and print the table in the database.

```
import java.sql.*;

class JdbcSelectTest {

  public static void main(String[] args) {

    try (

      Connection conn = DriverManager.getConnection( "jdbc:mysql://localhost:3306/ebookshop",

          "myuser", "xxxx");   // For MySQL only

      Statement stmt = conn.createStatement();

    ) {

      String strSelect = "select title, price, qty from books";

      System.out.println("The SQL statement is: " + strSelect + "\n"); // Echo For debugging

       ResultSet rset = stmt.executeQuery(strSelect);
```

```
        System.out.println("The records selected are:");
        int rowCount = 0;
        while(rset.next()) {   // Repeatedly process each row
          String title = rset.getString("title");  // retrieve a 'String'-cell in the row
          double price = rset.getDouble("price");  // retrieve a 'double'-cell in the row
          int    qty   = rset.getInt("qty");       // retrieve a 'int'-cell in the row
          System.out.println(title + ", " + price + ", " + qty);
          ++rowCount;
        }
        System.out.println("Total number of records = " + rowCount);
      } catch(SQLException ex) {
        ex.printStackTrace();
      }
    }
}
```

Output

The records Selected are:

The complete reference java 658.0 1

The complete reference c++ 578.00 2

The complete reference python 577.00 1

Total number of records : 3