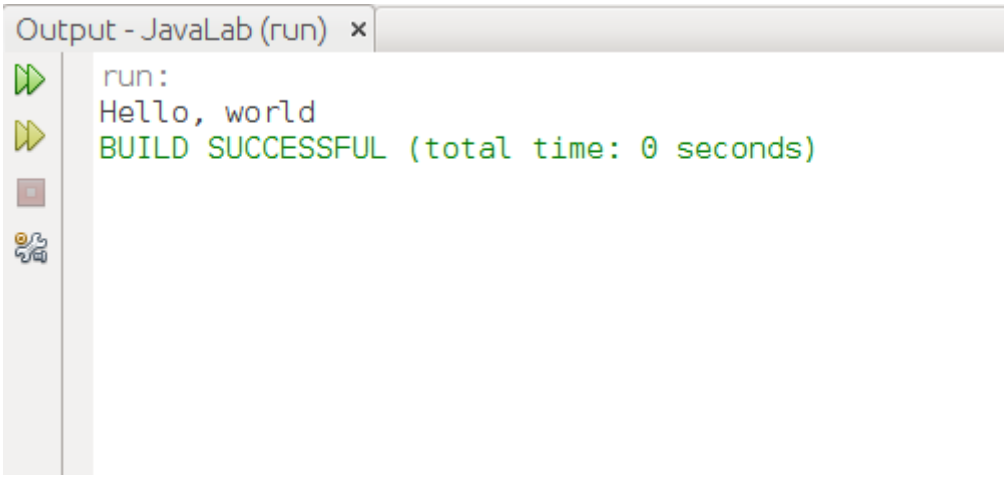
Experiment 1.1- WAP to display “Hello, World”

```
public class Exp1A {

    public static void main(String[] args) {

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

    }


}
```

OUTPUT


Output - JavaLab (run)
run:
Hello, world
BUILD SUCCESSFUL (total time: 0 seconds)

Experiment 1.2 – WAP to evaluate the following expressions:

1. (a/c)z/a
2. c++ + b/x -y

```
package javalab;

public class Exp1B {

    public static void main(String[] args) {

        int a=6,b=2,c=2,x=6,y=4,z=12;

        int sol;

        sol=(a/c)*z/a;

        System.out.println("(a/c)z/a :"+sol);


        sol=c++ + b/x-y;

        System.out.println("c++ + b/x-y :"+sol);

    }


}
```

OUTPUT

Output - JavaLab (run)

```
run:
(a/c)z/a :6
c++ + b/x-y :-2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 1.3-WAP to display the values by taking input from console using Scanner class.

```
import java.util.Scanner;


public class Exp1C {

    public static void main(String[] args) {

        Scanner input=new Scanner(System.in);

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

        String name=input.next();

        System.out.println("Hello, "+name);

    }

}
```

OUTPUT

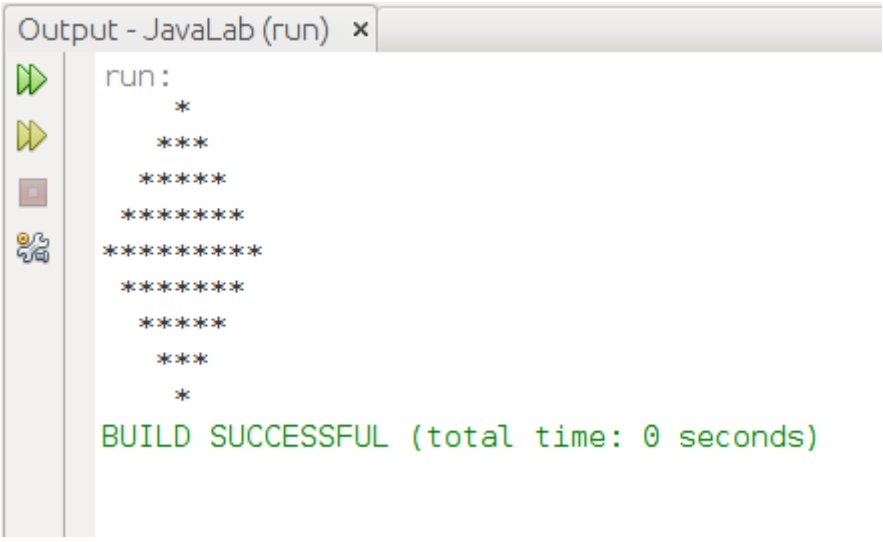
Output - JavaLab (run)

```
run:
Enter name:
SHAKTIMAN
Hello, SHAKTIMAN
BUILD SUCCESSFUL (total time: 1 minute 11 seconds)
```

Experiment 1.4- WAP to display “diamond” pattern

```
public class Exp1D {
    public static void main(String[] args) {
            for (int i = 0; i < 4; i++) {
            for (int j = 4; j > i; j--) {
                System.out.print(" ");
            }
            for (int k = 0; k < (i * 2 + 1); k++) {
                System.out.print("*");
            }
            System.out.println();
        }
        for (int i = 0; i < 9; i++) {
            System.out.print("*");
        }
        System.out.println();
        for (int i = 4; i > 0; i--) {
            for (int j = 4; j >= i; j--) {
                System.out.print(" ");
            }
            for (int k = (2 * (i - 1) + 1); k > 0; k--) {
                System.out.print("*");
            }

            System.out.println();
        }

    }
}
```

OUTPUT


Output - JavaLab (run)
run:
*
***
*****
****
****
****
*****
***
*
BUILD SUCCESSFUL (total time: 0 seconds)

Experiment 2.1-Find the largest and smallest element in an araay after getting values from the console.

```
import java.util.Scanner;

public class Exp2A {


    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int size = input.nextInt();

        int arr[] = new int[size];

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

            System.out.print("arr[" + i + "]:");

            arr[i] = input.nextInt();

        }


        int large = largest(arr, size);

        int small = smallest(arr, size);

        System.out.println("Largest:" + large + "\nSmallest:"
+ small);

    }


    private static int largest(int[] arr, int size) {


        int temp = arr[0];

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

            if (temp < arr[i]) {

                temp = arr[i];

            }

        }

        return temp;

    }
```

```
    private static int smallest(int[] arr, int size) {


        int temp = arr[0];
        for (int i = 0; i < size; i++) {
            if (temp > arr[i]) {
                temp = arr[i];
            }
        }
        return temp;
    }
}
```

OUTPUT

Output - JavaLab (run)

```
run:
5
arr[0]:21
arr[1]:3
arr[2]:43
arr[3]:111
arr[4]:54
Largest:111
Smallest:3
BUILD SUCCESSFUL (total time: 19 seconds)
```

Experiment 2.2 – Write a menu driven program to implement String and StringBuffer operations.

```
public class Exp2B {


    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);


        System.out.print("Enter String:");
        String str = input.nextLine();
        do {
            initMenu();
            System.out.println("Enter the choice:");
            int choice = input.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Length:" +
str.length());
                    break;
                case 2:
                    str = str.toLowerCase();
                    System.out.println("ToLowerCase:" + str);
                    break;
                case 3:
                    str = str.toUpperCase();
                    System.out.println("ToUpperCase:" + str);
                    break;
                case 4:
                    System.out.print("Enter String:");
                    String two = input.nextLine();
```

```
                    str = str + two;
                    //using stringbuilder
                    System.out.println("Concatinated string:"
+ str);
                    break;
                case 5:
                    str = str.trim();
                    System.out.println("Trim:" + str);
                    break;
                case 6:
                    System.out.println("Enter beg and end
index");
                    int beg = input.nextInt();
                    int end = input.nextInt();
                    System.out.println("Substring(" + beg +
"," + end + "):" + str.substring(beg, end));
                    break;
                case 7:
                    System.out.println("Enter Index:");
                    int index = input.nextInt();
                    System.out.println("char At:" +
str.charAt(index));
                    break;
                case 8:
                    System.out.println("Enter String:");


                case 9:
                    System.exit(0);
                default:
                    System.out.println("wrong choice!!\nTry
Again!");


            }
        } while (true);
```

```
    }

    private static void initMenu() {
        System.out.println("1.Length");
        System.out.println("2.ToLowerCase");
        System.out.println("3.ToUpperCase");
        System.out.println("4.Concatination");
        System.out.println("5.Trim");
        System.out.println("6.SubString");
        System.out.println("7.CharAt");
        System.out.println("8.Change String");
        System.out.println("9.exit");
        System.out.println();

    }

}
```

OUTPUT

```
Output - JavaLab (run)
run:
Enter String:valar moghulis
1.Length
2.ToLowerCase
3.ToUpperCase
4.Concatination
5.Trim
6.SubString
7.CharAt
8.Change String
9.exit

Enter the choice:
1
```

```
Enter the choice:
5
Trim:valar moghulis
```

```
Enter the choice:
6
Enter beg and end index
0
5
Substring(0,5):valar
```

```
Enter the choice:
2
ToLowerCase:valar moghulis
```

```
Enter the choice:
3
ToUpperCase:VALAR MOGHULIS
```

```
Enter the choice:
7
Enter Index:
3
char At:A
```

Experiment 2.3- Imlement Stack and Queue operations using ArrayList and Iterator class

STACK

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Iterator;
import java.util.ListIterator;

public class Exp2C {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        Stack stack=new Stack();

       do{

            System.out.println("1.push");
            System.out.println("2.pop");
            System.out.println("3.exit");
            System.out.println("Enter choice:");
            int choice=input.nextInt();
            switch(choice){
                case 1:  System.out.println("Enter value:");
                    int value=input.nextInt();
                    stack.push(value);
                    break;
                case 2:int pop=stack.pop();
                         System.out.println("Popped
element:"+pop);
                         break;
                case 3:System.exit(0);
                default: System.out.println("wrong choice!!");
            }
```

```
        }while(true);



    }
    }

    public class Stack {
        private ArrayList<Integer> stack=new ArrayList<>();
        private int top=-1;
        private static final int SIZE=10;

        public void push(int value){

            if(top==SIZE){
                System.out.println("OVERFLOW --->>");
            }else{
                top++;
                stack.add(value);
                System.out.println("PUSHED --->>");
                show();
            }
        }

        public int pop() {
            int value=-1;
            if(top==-1){
                System.out.println("UNDERFLOW --->>");
            }else{
                value=stack.get(top);

                stack.remove(top);
                top--;
                System.out.println("POPPED--->>");
                show();
            }
```

```
            return value;
        }

        public void  show(){
            ListIterator itr=stack.listIterator(stack.size());

            while(itr.hasPrevious()){
                    System.out.println("--->>"+itr.previous());
            }

        }


    }
```

OUTPUT

Output - **JavaLab (run) #2**

```
run:
1.push
2.pop
3.exit
Enter choice:
1
Enter value:
21
PUSHED --->>
--->>21
1.push
2.pop
3.exit
Enter choice:
1
Enter value:
45
PUSHED --->>
--->>45
--->>21
1.push
2.pop
3.exit
Enter choice:
2
POPPED--->>
--->>21
Popped element:45
1.push
2.pop
3.exit
Enter choice:
2
POPPED--->>
Popped element:21
1.push
2.pop
3.exit
Enter choice:
2
UNDERFLOW --->>
```

QUEUE

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.ListIterator;

public class Exp2CB {
     public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        Queue queue=new Queue();

       do{

           System.out.println("1.Insert");
           System.out.println("2.Remove");
           System.out.println("3.exit");
           System.out.println("Enter choice:");
           int choice=input.nextInt();
           switch(choice){
               case 1:  System.out.println("Enter value:");
                   int value=input.nextInt();
                   queue.insert(value);
                   break;
               case 2:int pop=queue.remove();
                        if(pop!=-1)
                            System.out.println("Popped
element:"+pop);
                        break;
               case 3:System.exit(0);
               default: System.out.println("wrong choice!!");
           }
       }while(true);
    }
}
```

```
public class Queue {

    private static final int SIZE = 10;
    private ArrayList<Integer> queue = new ArrayList<>(SIZE);
    private int rear = -1, front = -1;

    public void insert(int value) {

        if ((rear + 1) == SIZE) {
            System.out.println("OVERFLOW --->>");
        } else if (rear == -1) {
            front = 0;
            rear = 0;
            queue.add(value);
            System.out.println("INSERTED --->>");
            show();

        } else {
            rear++;
            queue.add(value);
            System.out.println("INSERTED --->>");
            show();
        }
    }

    public int remove() {
        int value = -1;
        if (front == -1) {
            System.out.println("UNDERFLOW --->>");
        } else if (front == rear) {
            front = -1;
            rear = -1;
            queue.remove(0);
```

```
            System.out.println("QUEUE IS EMPTY!!");
        } else {

            value = queue.get(0);
            queue.remove(0);
            front++;
            System.out.println("REMOVED--->>");
            show();
        }
        return value;
    }

    public void show() {

        Iterator itr = queue.iterator();
        while (itr.hasNext()) {
            System.out.println("-->>" + itr.next());
        }
    }

}
```

OUTPUT

```
run:
1.Insert
2.Remove
3.exit
Enter choice:
1
Enter value:
21
INSERTED --->>
-->>21
1.Insert
2.Remove
3.exit
Enter choice:
1
Enter value:
34
INSERTED --->>
-->>21
-->>34
1.Insert
2.Remove
3.exit
Enter choice:
2
REMOVED--->>
-->>34
Popped element:21
1.Insert
2.Remove
3.exit
Enter choice:
2
QUEUE IS EMPTY!!
1.Insert
2.Remove
3.exit
Enter choice:
```

Experiment 3- (a) Calculate Volume of a box using classes.

(b)-Modify Part (a) by adding constructors and “this” reference.

(c)- Add the member function to the existing class to show the demo of passing objects as a value and returning objects from class.

(d)- Add member function to give a demo of varargs (variable arguments).

```
import java.util.Scanner;

public class Exp3A {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        Box box1=new Box();

        System.out.println("Enter dimensions of Box1");
        System.out.print("Length:");
        box1.setLength(input.nextFloat());
        System.out.print("Breadth:");
        box1.setBreadth(input.nextFloat());
        System.out.print("Height:");
        box1.setHeight(input.nextFloat());

        System.out.println("Volume:"+box1.volume());


        System.out.println("Enter dimensions of Box2");
        System.out.print("Length:");
        float l=input.nextFloat();
        System.out.print("Breadth:");
        float b=input.nextFloat();
        System.out.print("Height:");
```

```
        float h=input.nextFloat();

        Box box2=new Box(l,b,h);

        System.out.println("Adding box 1 to Box 2");
        Box box3=box1.addBox(box2);
        System.out.println(box3.toString());
        System.out.println("Display all boxes");

        Box.displayBoxes(box1,box2);
    }

}

class Box{
    float length,breadth,height;

    public Box() {
    }

   //using constructor with this reference  PART (B)

    public Box(float length, float breadth, float height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }

    public float getLength() {
        return length;
    }

    public void setLength(float length) {
        this.length = length;
    }
```

```
    public float getBreadth() {
        return breadth;
    }

    public void setBreadth(float breadth) {
        this.breadth = breadth;
    }

    public float getHeight() {
        return height;
    }

    public void setHeight(float height) {
        this.height = height;
    }

    public float volume(){
        return length*breadth*height;
    }

    //method to take and return arguments as objects PART (C)
    public Box addBox(Box box2){

        float len=box2.getLength()+this.length;
        float bre=box2.getBreadth()+this.breadth;
        float hei=box2.getHeight()+this.height;

        return new Box(len,bre,hei);
    }

    @Override
    public String toString() {
        return "Length:"+length+" Breadth:"+breadth
                + " Height:"+height; //To change body of
generated methods, choose Tools | Templates.
    }
```

```
    // method to demonstrate varargs
    public static void displayBoxes(Box...arg){
        int i=1;
        for(Box box:arg){
            System.out.println("Box:"+i);
            System.out.println(box.toString());
            i++;
        }
    }


}
```

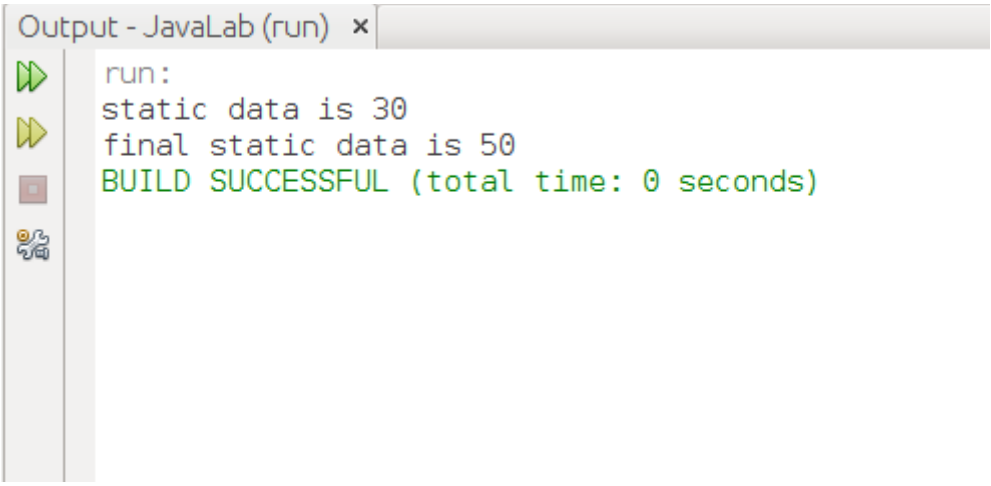
OUTPUT

```
Output ×
JavaLab (run) #3 ×   JavaLab (run) #4 ×
run:
Enter dimensions of Box1
Length:2
Breadth:3
Height:4
Volume:24.0
Enter dimensions of Box2
Length:3
Breadth:4
Height:5
Adding box 1 to Box 2
Length:5.0 Breadth:7.0 Height:9.0
Display all boxes
Box:1
Length:2.0 Breadth:3.0 Height:4.0
Box:2
Length:3.0 Breadth:4.0 Height:5.0
BUILD SUCCESSFUL (total time: 11 seconds)
```

Experiment 4-(a) Write a program to give the demo of ‘static’ and ‘final’ variable using Nested class concept.

```
package javalab;


public class Exp4A {
  public static void main(String args[]){
  TestOuter1.Inner obj=new TestOuter1.Inner();
  obj.msg();
  obj.msg2();
  }
}
class TestOuter1{
  static int data=30;
  static final int data2=50;

  static class Inner{
   void msg(){
       System.out.println("static data is "+data);
   }
   void msg2(){
      System.out.println("final static data is "+ data2 );
  }
  }



}
```

Output



Experiment 4-(b) Write a program to input command line arguments.

```
package javalab;
```

```
public class Exp4C {

    public static void main(String[] args) {

        if (args == null) {
            System.out.println("enter arguments at command
line");

        } else {

            System.out.println("Command line arguments are");

            for (String str : args) {
                System.out.println(str);
            }
        }
    }
}
```

Output

```
Output - JavaLab (run)
run:
Command line arguments are
my
name
is
rishabh
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 5- (a) Calculate Volume of a box by implementing Simple Inheritance

```
import java.util.Scanner;

public class Exp5ABoxProperties extends Box1{
```

```
    public Exp5ABoxProperties(double length, double breadth,
double height) {
        super(length, breadth, height);
    }



    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);

        System.out.print("Enter Length:");
        double length=input.nextInt();

        System.out.print("Enter Breadth:");
        double breadth=input.nextInt();

        System.out.print("Enter Height:");
        double height=input.nextInt();



        Exp5ABoxProperties box=new Exp5ABoxProperties(length,
breadth, height);

        double volume=box.getVolume();
        System.out.println("Volume:"+volume);
    }

    double getVolume(){
        double vol=0;
        double length=getLength();
        double breadth=getBreadth();
        double height=getHeight();

        vol=length*breadth*height;

        return vol;
    }
```

```
}

public class Box1 {

    double length,breadth,height;

    public Box1(double length, double breadth, double height)
{
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }

    public double getLength() {
        return length;
    }

    public void setLength(double length) {
        this.length = length;
    }

    public double getBreadth() {
        return breadth;
    }

    public void setBreadth(double breadth) {
        this.breadth = breadth;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }
```

```
    @Override
    public String toString() {
        return "Box1{" + "length=" + length + ", breadth=" +
breadth + ", height=" + height + '}';
    }



}
```

OUTPUT

```
Output - JavaLab (run)
run:
Enter Length:4
Enter Breadth:5
Enter Height:6
Volume:120.0
BUILD SUCCESSFUL (total time: 6 seconds)
```

Experiment 5 -(b) Modify part (a) by using method overriding and 'super' keyword.

```
public class Exp5B extends Box1 {

    double vol = 0;
    public Exp5B(double length, double breadth, double height)
{
        super(length, breadth, height);
    }
```

```
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter Length:");
        double length = input.nextInt();

        System.out.print("Enter Breadth:");
        double breadth = input.nextInt();

        System.out.print("Enter Height:");
        double height = input.nextInt();

        Exp5B box = new Exp5B(length, breadth, height);
        box.vol = box.getVolume();
        System.out.println(box.toString());

    }

    @Override
    public String toString() {
        return super.toString() + "Volume:" + vol;
    }

    double getVolume() {
        double vol = 0;
        double length = getLength();
        double breadth = getBreadth();
        double height = getHeight();

        vol = length * breadth * height;

        return vol;
    }

}
```

```
//Box class is same as Exp1
```

Output

```
Output - JavaLab (run)
run:
Enter Length:3
Enter Breadth:4
Enter Height:2
Box1{length=3.0, breadth=4.0, height=2.0}Volume:24.0
BUILD SUCCESSFUL (total time: 8 seconds)
```

Experiment 5- (c) Implement the concept of dynamic method dispatch.

```
public class Exp5C extends Game{
```

```
    public void type()
 {  System.out.println("outdoor game"); }

 public static void main(String[] args)
 {
   Game gm = new Game();
   Exp5C ck = new Exp5C();
   gm.type();
   ck.type();
   gm=ck;
   gm.type();
 }
}

public class Game {
    public void type(){
        System.out.println("Indoor & outdoor");
    }
}
```

OUTPUT

```
Output - JavaLab (run)
run:
Indoor & outdoor
outdoor game
outdoor game
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 5 (d)- Create Abstract and final class concept.

```
abstract class Bank {
```

```
    abstract int getRateOfInterest();
}

class SBI extends Bank {

    int getRateOfInterest() {
        return 7;
    }
}

class PNB extends Bank {

    int getRateOfInterest() {
        return 8;
    }
}

class TestBank {

    public static void main(String args[]) {
        Bank b;
        b = new SBI();
        System.out.println("Rate of Interest is: " +
b.getRateOfInterest() + " %");
        b = new PNB();
        System.out.println("Rate of Interest is: " +
b.getRateOfInterest() + " %");
    }
}
```

OUTPUT

```
Output - JavaLab (run)
run:
Rate of Interest is: 7 %
Rate of Interest is: 8 %
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 6- (a) Create interface and implement it using class

```
public class Exp6A implements Animal{
    public static void main(String[] args) {
```

```
        Exp6A tiger=new Exp6A();
        tiger.eat();
        tiger.travel();
        tiger.type();
    }

    @Override
    public void eat() {
            System.out.println("Animal eats");
    }

    @Override
    public void travel() {
        System.out.println("Animal travels");
    }

    @Override
    public void type() {
        System.out.println("Animal Type: Tiger");
    }
}

public interface Animal {
    void eat();
    void travel();
    void type();
}
```

OUTPUT

```
Output - JavaLab (run)
run:
Animal eats
Animal travels
Animal Type: Tiger
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 6- (b)Extending an interface to the class.

```
public interface Printable {
    void print();
```

```
}
public interface Showable extends Printable{
    void show();

}

public class Exp6B implements Showable{

    public static void main(String[] args) {
        Exp6B obj=new Exp6B();
        obj.show();
        obj.print();
    }

    @Override
    public void show() {
        System.out.println("show function from Showable
interface!");
    }

    @Override
    public void print() {
        System.out.println("print function from Printable
interface!");
    }
}
```

OUTPUT

```
Output - JavaLab (run)
run:
show function from Showable interface!
print function from Printable interface!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 7- (a) WAP to create package by defining three classes.

(b) Implement the defined package in a class to access the classes defined in a package.

```
package mypackage;

public class A{

  public static int VAR=1;
  public void print(){
    System.out.println("hello, from class:A");
  }
}

package mypackage;

public class B{
  public static int VAR=2;

  public void print(){
    System.out.println("hello, from class:B");
  }
}

package mypackage;

public class C{
  public static int VAR=3;

  public void print(){
    System.out.println("hello, from class:C");
  }
}

// Part (b)
import mypackage.C;

class Test{
  public static void main(String[] args) {
```

```
        A aObj=new A();
        B bObj=new B();
        C cObj=new C();
        aObj.print();
        bObj.print();
        cObj.print();
    }
}
```

OUTPUT

```
run:
hello, from class:A
hello, from class:B
hello, from class:C
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 7 -(c)Implement execption handling using try, throw, finally, finalize keywords.

```
import java.io.*;

public class InsufficientFundException extends Exception {
   private double amount;

   public InsufficientFundException(double amount) {
      this.amount = amount;
   }

   public double getAmount() {
      return amount;
   }
}

import java.io.*;
public class Exp7C {

    private double balance;
   private int number;

   public Exp7C(int number) {
      this.number = number;
   }

   public static void main(String [] args) {
      Exp7C c = new Exp7C(101);
      System.out.println("Depositing $500...");
      c.deposit(500.00);

      try {
         System.out.println("\nWithdrawing $100...");
         c.withdraw(100.00);
```

```
            System.out.println("\nWithdrawing $600...");
            c.withdraw(600.00);
         }catch(InsufficientFundException e) {
            System.out.println("Sorry, but you are short $" +
e.getAmount());
            System.out.println("Error!! Rolling back
changes...");
            e.printStackTrace();
         }finally{

             System.gc();
         }

      }

       public void finalize(){
           System.out.println("cleaning garbage!");
       }

       public void deposit(double amount) {
         balance += amount;
      }

      public void withdraw(double amount) throws
InsufficientFundException {
         if(amount <= balance) {
            balance -= amount;
         }else {
            double needs = amount - balance;
            throw new InsufficientFundException(needs);
         }
      }

      public double getBalance() {
         return balance;
      }
```

```
    public int getNumber() {
        return number;
    }
}
```

OUTPUT

```
Output - JavaLab (run)
run:
Depositing $500...

Withdrawing $100...

Withdrawing $600...
Sorry, but you are short $200.0
Error!! Rolling back changes...
javalab.InsufficientFundException
        at javalab.Exp7C.withdraw(Exp7C.java:55)
        at javalab.Exp7C.main(Exp7C.java:30)
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment 8- (a) WAP to implement StringTokenizer class methods.

```
import java.util.StringTokenizer;


class Exp8A {
    public static void main(String[] args) {
        StringTokenizer st = new StringTokenizer("my name is
khan"," ");


        int count=st.countTokens();
        System.out.println("no. of times nextToken method can
be called:"+count);


        while (st.hasMoreTokens()) {
        System.out.println(st.nextToken());
     }
        // creating string tokenizer with delimeter
         StringTokenizer st2 = new
StringTokenizer("Come/to/learn");


         // checking next token
        System.out.println("Next token is : " +
st2.nextToken("/"));
    }


}
```

OUTPUT

Output - JavaLab (run)

```
run:
no. of times nextToken method can be called:4
my
name
is
jindal
Next token is : keep
BUILD SUCCESSFUL (total time: 0 seconds)
```

Experiment- 9 (a) WAP to implement multi-threading

```
import java.util.logging.Level;
import java.util.logging.Logger;

public class Exp9A {

    public static void main(String[] args) {

         Thread obj=new Thread(new Runner2());
        Thread obj2=new Thread(new Runner2());

        obj.start();
        obj2.start();
    }

}
class Runner2 implements Runnable{
    public void run(){
        for(int i=0;i<10;i++){
            System.out.println("Hello"+i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {

            }
        }
    }
}
```

Output

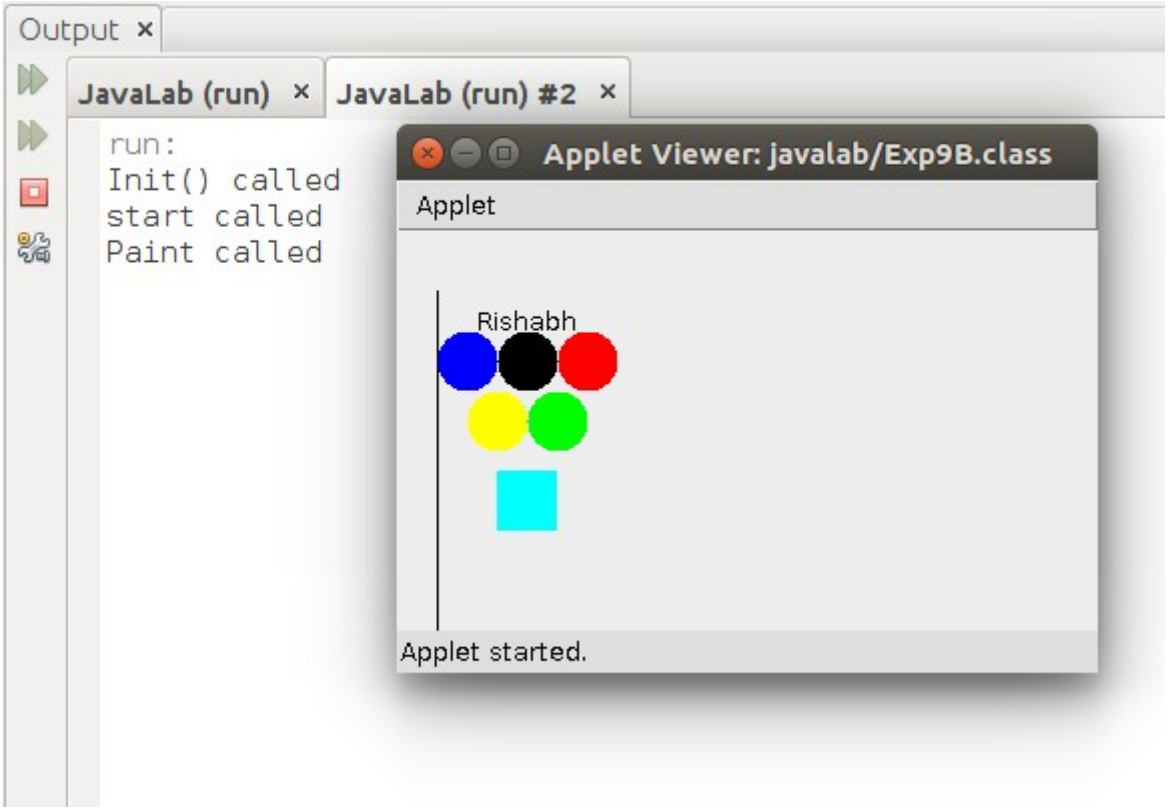
```
run:
Hello0
Hello0
Hello1
Hello1
Hello2
Hello2
Hello3
Hello3
Hello4
Hello4
Hello5
Hello5
Hello6
Hello6
Hello7
Hello7
Hello8
Hello8
Hello9
Hello9
BUILD SUCCESSFUL (total time: 10 seconds)
```

Experiment 9- (b) WAP to create an applet using all of its functions.

```
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JApplet;

public class Exp9B extends JApplet{



    @Override
    public void init() {
        super.init(); //To change body of generated methods,
choose Tools | Templates.
            System.out.println("Init() called");
       }

    @Override
    public void start() {
        super.start(); //To change body of generated methods,
choose Tools | Templates.
        System.out.println("start called");
    }



    public void paint(Graphics g){
        super.paint(g);
        System.out.println("Paint called");

      g.setColor(Color.BLACK);
g.drawString("Rishabh",40, 50);
g.drawLine(20,30,20,200);

g.setColor(Color.CYAN);
```

```
    g.fillRect(50,120,30,30);

    g.setColor(Color.BLUE);
    g.fillOval(20,50,30,30);

    g.setColor(Color.BLACK);
    g.fillOval(50,50,30,30);

    g.setColor(Color.RED);
    g.fillOval(80,50,30,30);

    g.setColor(Color.YELLOW);
    g.fillOval(35,80,30,30);

    g.setColor(Color.GREEN);
    g.fillOval(65,80,30,30);



    }

    @Override
    public void stop() {
        super.stop(); //To change body of generated methods,
    choose Tools | Templates.
        System.out.println("Stop called");
    }

    @Override
    public void destroy() {
        super.destroy(); //To change body of generated
    methods, choose Tools | Templates.
        System.out.println("Destroy called");
    }
}
```

Output


Output
JavaLab (run)
JavaLab (run) #2
run:
Init() called
start called
Paint called
Applet Viewer: javalab/Exp9B.class
Applet
Rishabh
Applet started.

Experiment 9- (c) Create a moving banner by setting background and forground color of the text and by setting status bar text.

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JApplet;

public class Exp9C extends JApplet{

    private Image image;

    @Override
    public void init() {
        super.init(); //To change body of generated methods,
choose Tools | Templates.
        image=getImage(getCodeBase(),"bg_main.jpg");
        setBackground(Color.YELLOW);
    }

    @Override
    public void paint(Graphics g) {

            super.paint(g); //To change body of generated
methods, choose Tools | Templates.
            System.out.println(getCodeBase());
            g.drawString("rishabh moving image", 10, 10);

            for(int i=0;i<500;i++){
                try {
                    g.drawImage(image, i,200,this);
```

```
                    Thread.sleep(10);
                } catch (InterruptedException ex) {

Logger.getLogger(Exp9C.class.getName()).log(Level.SEVERE,
null, ex);
                }


    }
  }
}
```

Output


Applet
rishabh moving image
Kapayapaan
Мир
Amani
Mau te rongo
Paz
Mir
Vrede
Síochána
Pace
Мир
Pacon
Mir
Pau
Ukuthula
Zaman lafiya
Paz
Applet started.

Experiment 10 – (a) WAP to use join() and isAlive() methods and setting thread priorities.

```
import java.util.logging.Level;
import java.util.logging.Logger;

public class Exp10A extends Thread{
    public void run()
    {
        System.out.println(Thread.currentThread().getName()+":
"+Thread.currentThread().getPriority());
        try {
            Thread.sleep(1000);
        }
        catch(InterruptedException ie) { }
        System.out.println("r2 ");
    }
    public static void main(String[] args)   {
        Exp10A t1=new Exp10A();
        Exp10A t2=new Exp10A();
        t1.setPriority(9);
        t2.setPriority(1);
        t1.start();
        t2.start();
        try {
            t1.join();
        } catch (InterruptedException ex) {
        }
        System.out.println(t1.isAlive());
        System.out.println(t2.isAlive());
    }
}
```

Output

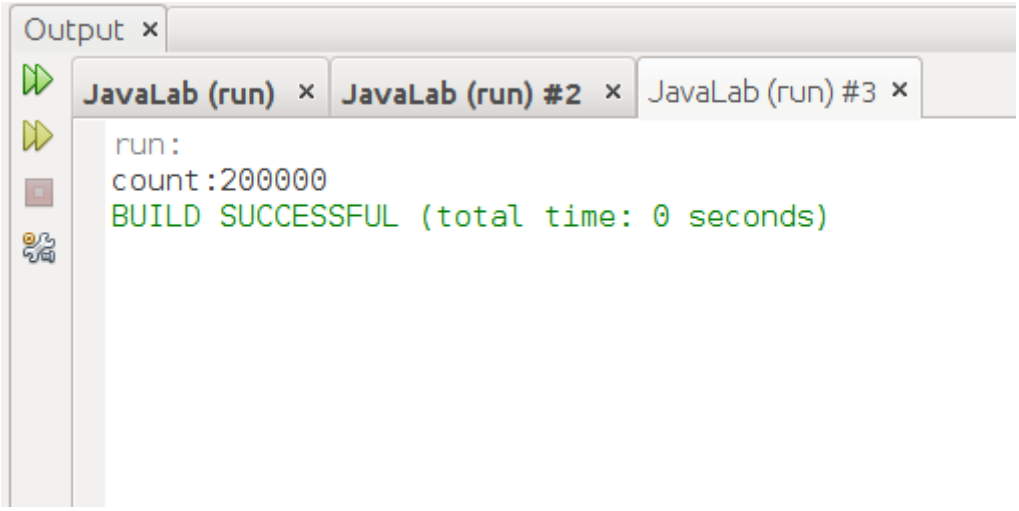
```
run:
Thread-0: 9
Thread-1: 1
r2
r2
false
false
BUILD SUCCESSFUL (total time: 1 second)
```

Experiment 10 -(b) WAP to implement synchronization.

```
import java.util.logging.Level;
import java.util.logging.Logger;

public class Exp10B {

    private int count=0;
    public static void main(String[] args) {

        Exp10B obj=new Exp10B();
        obj.doWork();
    }

    public void doWork(){
        Thread thread1=new Thread(new Runnable() {
            @Override
            public void run() {
                for(int i=0;i<100000;i++){
                    increment();
                }
            }
        });

        Thread thread2=new Thread(new Runnable() {
            @Override
            public void run() {
                for(int i=0;i<100000;i++){
                    increment();
                }
            }
        });
```

```
        thread1.start();
        thread2.start();


        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException ex) {

        }


        System.out.println("count:"+count);
    }

    public synchronized  void increment(){

        count++;
    }
}
```

Output


Output
JavaLab (run)
JavaLab (run) #2
JavaLab (run) #3
run:
count:200000
BUILD SUCCESSFUL (total time: 0 seconds)

Experiment 10- (c) WAP to show Deadlock operation.

```
public class Exp10C {

    public static void main(String[] args) {

    final String resource1 = "RIshabh JIndal";
    final String resource2 = "Ankit Jindal";

    Thread t1 = new Thread() {
      public void run() {
          synchronized (resource1) {
           System.out.println("Thread 1: locked resource 1");

           try { Thread.sleep(100);} catch (Exception e) {}

           synchronized (resource2) {
            System.out.println("Thread 1: locked resource 2");
           }
         }
      }
    };

    Thread t2 = new Thread() {
      public void run() {
        synchronized (resource2) {
          System.out.println("Thread 2: locked resource 2");

          try { Thread.sleep(100);} catch (Exception e) {}

          synchronized (resource1) {
            System.out.println("Thread 2: locked resource 1");
          }
```

```
            }
          }
        };


        t1.start();
        t2.start();
        }

    }
```

Output


Output
JavaLab (run)
JavaLab (run) #2
JavaLab (run) #3
run:
Thread 1: locked resource 1
Thread 2: locked resource 2