B.M.S. COLLEGE OF ENGINEERING

(Autonomous college under VTU)

Bull Temple Rd, Basavanagudi, Bengaluru, Karnataka 560019
2023-2025
Department of Computer Applications

Report is submitted for fulfillment of Lab Task in the subject

"JAVA PROGRAMMING"
(22MCA2PCJP)

By

Manohar T H
1BM23MC051

Under the Guidance

Prof. R.V. Raghavendra Rao
(Assistant Professor)

B. M. S. COLLEGE OF ENGINEERING, BANGALORE – 19

(Autonomous Institute, Affiliated to VTU)

Department of Computer Applications

(Accredited by NBA for 5 years 2019-2024)

LABORATORY CERTIFICATE

This is to certify that **Manohar T H** (1BM23MC001) has satisfactorily Completed the course of practical in "**Java Programming - 22MCA2PCJP**" Laboratory prescribed by BMS College of Engineering (Autonomous college under VTU) 2nd Semester MCA Course in this college during the year 2023-2024.

Signature of Batch Incharge

Prof. P. Lakshminarayan

Signature of HoD

Dr. Ch. Ram Mohan Reddy

Examiner:

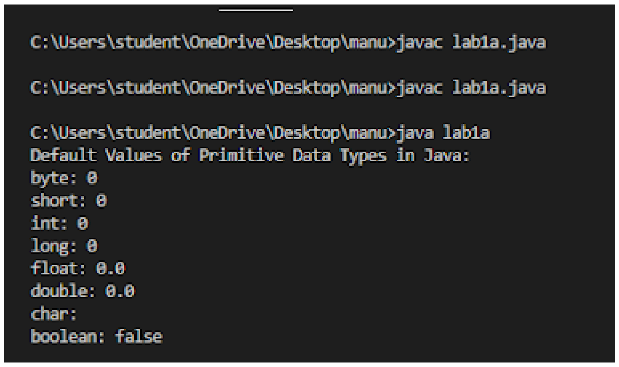
CONTENTS

1

a). Write a JAVA program to display default value of all primitive data type of JAVA

```
public class lab1a{
    public static void main(String[] args) {
        System.out.println("Default Values of Primitive Data Types in Java:");
        byte byteValue = 0;
        System.out.println("byte: " + byteValue);
        short shortValue = 0;
        System.out.println("short: " + shortValue);
        int intValue = 0;
        System.out.println("int: " + intValue);
        long longValue = 0;
        System.out.println("long: " + longValue);
        float floatValue = 0.0f;
        System.out.println("float: " + floatValue);
        double doubleValue = 0.0;
        System.out.println("double: " + doubleValue);
        char charValue = '\u0000';
        System.out.println("char: " + charValue);
        boolean booleanValue = false;
        System.out.println("boolean: " + booleanValue);
    }
}
```

OUTPUT:

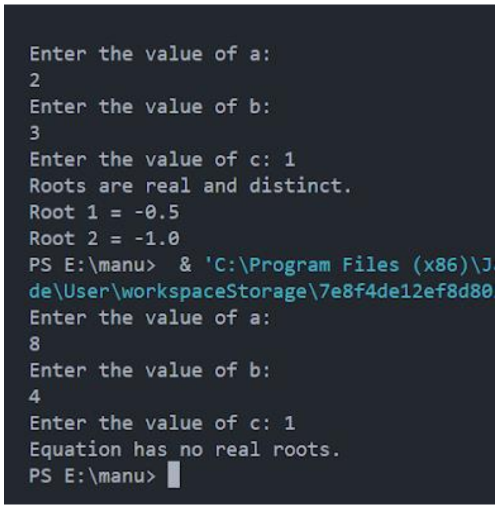


```
C:\Users\student\OneDrive\Desktop\manu>javac lab1a.java

C:\Users\student\OneDrive\Desktop\manu>javac lab1a.java

C:\Users\student\OneDrive\Desktop\manu>java lab1a
Default Values of Primitive Data Types in Java:
byte: 0
short: 0
int: 0
long: 0
float: 0.0
double: 0.0
char:
boolean: false
```

b). Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.

```
import java.util.Scanner;
public class lab1b{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the value of a:");
        double a = scanner.nextDouble();
        System.out.println("Enter the value of b:");
        double b = scanner.nextDouble();
        System.out.print("Enter the value of c: ");
        double c = scanner.nextDouble();
        double D = b * b - 4 * a * c; // D D = b^2 - 4ac
        if (D > 0) {
            double root1 = (-b + Math.sqrt(D)) / (2 * a);
            double root2 = (-b - Math.sqrt(D)) / (2 * a);
            System.out.println("Roots are real and distinct.");
            System.out.println("Root 1 = " + root1);
            System.out.println("Root 2 = " + root2);
        } else if (D == 0) {
            double root = -b / (2 * a);
            System.out.println("Roots are real and equal.");
            System.out.println("Root = " + root);
        } else {
            // No real roots (complex roots)
            System.out.println("Equation has no real roots.");
        }
        scanner.close();
    }
}
```

Output:

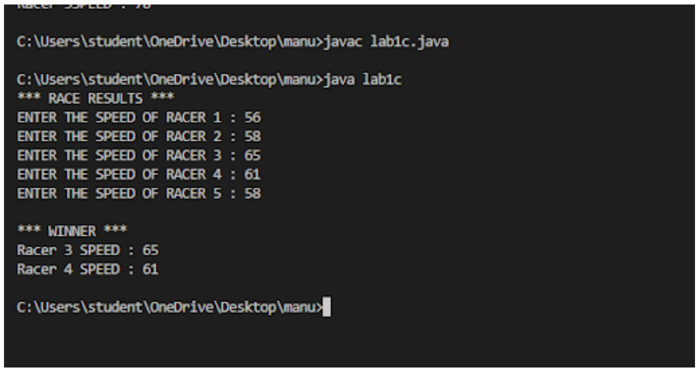


```
Enter the value of a:
2
Enter the value of b:
3
Enter the value of c: 1
Roots are real and distinct.
Root 1 = -0.5
Root 2 = -1.0
PS E:\manu>  & 'C:\Program Files (x86)\J
de\User\workspaceStorage\7e8f4de12ef8d80
Enter the value of a:
8
Enter the value of b:
4
Enter the value of c: 1
Equation has no real roots.
PS E:\manu>
```

c). Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers.Take as input the speed of each racer and print back the speed of qualifying racers.

```
import java.util.Scanner;
public class lab1c {
    public static void main(String args[]){
        Scanner inp = new Scanner(System.in);
        int arr[] = new int[10];
        int sum =0;
        System.out.println("*** RACE RESULTS ***");
        for(int i=0;i<5;i++){
            int a=i+1;
            System.out.print("ENTER THE SPEED OF RACER " + a + " : ");
            int val =inp.nextInt();
            arr[i]=val;
        }
        for(int n : arr){
            sum += n;
        }
        int avg =sum / 5 ;
        System.out.println("");
        System.out.println("*** WINNER ***");
        int i=0;
        for( int m : arr){
            i +=1;
            if(m > avg){
                System.out.println("Racer "+ i + " SPEED : " + m);
            }
        }
}
}
```

OUTPUT:



```
C:\Users\student\OneDrive\Desktop\manu>javac lab1c.java

C:\Users\student\OneDrive\Desktop\manu>java lab1c
*** RACE RESULTS ***
ENTER THE SPEED OF RACER 1 : 56
ENTER THE SPEED OF RACER 2 : 58
ENTER THE SPEED OF RACER 3 : 65
ENTER THE SPEED OF RACER 4 : 61
ENTER THE SPEED OF RACER 5 : 58

*** WINNER ***
Racer 3 SPEED : 65
Racer 4 SPEED : 61

C:\Users\student\OneDrive\Desktop\manu>
```

d) Write a case study on public static void main(250 words)

The public static void main method is the entry point of a Java program. It is the first method that is executed when a Java program is run. This method is declared in the public access modifier, which means it can be accessed from outside the class. The static keyword indicates that the method does not require an instance of the class to be executed. The void keyword specifies that the method does not return any value.

public: The main method must be public because it is called by the Java runtime system, which is outside the application’s scope. The Java Virtual Machine (JVM) needs to access it to start the program.

static: The main method is static because it needs to be called without creating an instance of the class. The JVM does not instantiate the class before calling the main method; it calls it directly on the class itself.

void: The main method does not return any value. It’s designed to be a starting point for the program, not a method that produces a result.

String[] args: This parameter allows the method to accept command-line arguments. These arguments are passed to the program when it is run, enabling dynamic input from the user.

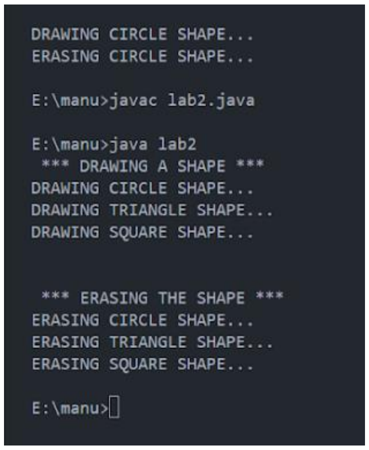
2.

a). Write a program to create a class named shape. In this class we have three sub classes circle, triangle and square each class has two member function named draw () and erase (). Create these using polymorphism Concepts.

Lab2.java

```
class shape{
    void draw(){
        System.out.println(" *** DRAWING A SHAPE ***");
    }
    void erase(){
        System.out.println(" *** ERASING THE SHAPE ***");
    }
}
class circle extends shape{
    @Override
    void draw(){
        System.out.println("DRAWING CIRCLE SHAPE...");
    }
    @Override
    void erase(){
        System.out.println("ERASING CIRCLE SHAPE...");
    }
}
class triangle extends shape{
    @Override
    void draw(){
        System.out.println("DRAWING TRIANGLE SHAPE...");
    }

    @Override
    void erase(){
        System.out.println("ERASING TRIANGLE SHAPE...");
    }
}


class square extends shape{
    @Override
    void draw(){
        System.out.println("DRAWING SQUARE SHAPE...");
    }
    @Override
    void erase(){
        System.out.println("ERASING SQUARE SHAPE...");
    }
}

public class lab2{
    public static void main(String[] args) {
        shape shp=new shape();
        shape cir=new circle();
        shape tri=new triangle();
        shape sqr=new square();
        shp.draw();
        cir.draw();
        tri.draw();
        sqr.draw();
        System.err.println("\n");
        shp.erase();
        cir.erase();
        tri.erase();
        sqr.erase();
    }
}
```

OUTPUT:



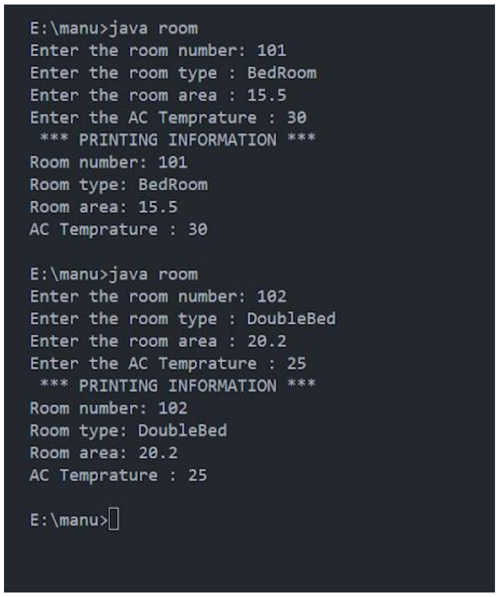
```
DRAWING CIRCLE SHAPE...
ERASING CIRCLE SHAPE...

E:\manu>javac lab2.java

E:\manu>java lab2
 *** DRAWING A SHAPE ***
DRAWING CIRCLE SHAPE...
DRAWING TRIANGLE SHAPE...
DRAWING SQUARE SHAPE...


 *** ERASING THE SHAPE ***
ERASING CIRCLE SHAPE...
ERASING TRIANGLE SHAPE...
ERASING SQUARE SHAPE...

E:\manu>
```

b). Write a program to create a room class, the attributes of this class is roomno, roomtype, roomarea and AC machine. In this class the member functions are setdata and displaydata.

Room.java

```
import java.lang.*;
import java.util.*;
class logivc{
    int roomno,ac;
    double roomarea;
    String roomtype;
    Scanner inp=new Scanner(System.in);
    void getdata(){
        System.out.print("Enter the room number: ");
        roomno=inp.nextInt();
        System.out.print("Enter the room type : ");
        roomtype=inp.next();
        System.out.print("Enter the room area : ");
        roomarea=inp.nextDouble();
        System.out.print("Enter the AC Temprature : ");
        ac=inp.nextInt();
    }
    void displaydata(){
        System.out.println(" *** PRINTING INFORMATION *** ");
        System.out.println("Room number: "+ roomno);
        System.out.println("Room type: "+ roomtype);
        System.out.println("Room area: "+ roomarea);
        System.out.println("AC Temprature : "+ ac);
    }
}


public class room{
    public static void main(String[] args) {
        logivc obj=new logivc();
        obj.getdata();
        obj.displaydata();
    }
}
```

OUTPUT:



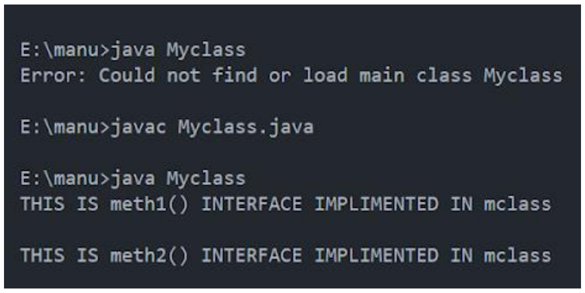
```
E:\manu>java room
Enter the room number: 101
Enter the room type : BedRoom
Enter the room area : 15.5
Enter the AC Temprature : 30
 *** PRINTING INFORMATION ***
Room number: 101
Room type: BedRoom
Room area: 15.5
AC Temprature : 30

E:\manu>java room
Enter the room number: 102
Enter the room type : DoubleBed
Enter the room area : 20.2
Enter the AC Temprature : 25
 *** PRINTING INFORMATION ***
Room number: 102
Room type: DoubleBed
Room area: 20.2
AC Temprature : 25

E:\manu>
```

3.a). Write a program to create interface A in this interface we have two method meth1 and meth2. Implements this interface in another class named MyClass.

MyClass.java

```
interface A{
   void meth1();
   void meth2();
}
class mclass implements A{
   public void meth1(){
      System.out.println("THIS IS meth1() INTERFACE IMPLIMENTED IN mclass \n");
   }
   public void meth2(){
      System.out.println("THIS IS meth2() INTERFACE IMPLIMENTED IN mclass");
   }
}
public class Myclass{
   public static void main(String[] args) {
      mclass obj = new mclass();
      obj.meth1();
      obj.meth2();
   }
}
```

OUTPUT :

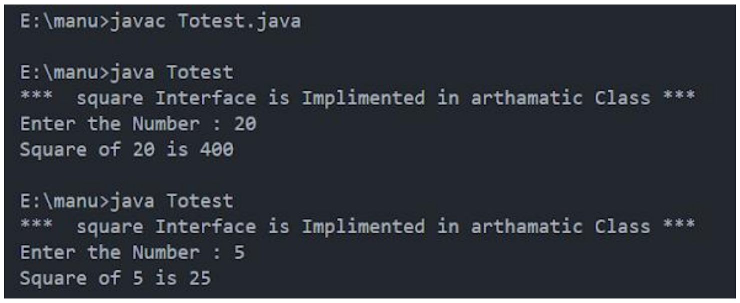


```
E:\manu>java Myclass
Error: Could not find or load main class Myclass

E:\manu>javac Myclass.java

E:\manu>java Myclass
THIS IS meth1() INTERFACE IMPLIMENTED IN mclass

THIS IS meth2() INTERFACE IMPLIMENTED IN mclass
```

b). Write a program to create interface named test. In this interface the member function is square. Implement this interface in arithmetic class. Create one new class called ToTestInt in this class use the object of arithmetic class.c). Create an outer class with a function display, again create another class inside the outer class named inner with a function called display and call the two functions in the main class.

```
import java.util.*;
interface test{
    void square();
}
class arthametic implements  test{
    public void square() {
        Scanner inp=new Scanner(System.in);
        int num;
        System.out.println("***  square Interface is Implimented in arthamatic Class ***");
        System.out.print("Enter the Number : ");
        num=inp.nextInt();
        System.out.println("Square of "+num+" is "+num*num);
    }}
public class Totest {
    public static void main(String[] args) {
        arthametic a = new arthametic();
        a.square();
    }
}
```

OUTPUT:



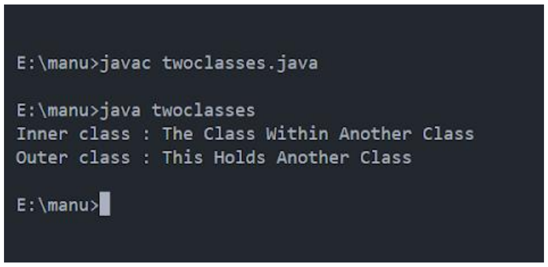
```
E:\manu>javac Totest.java

E:\manu>java Totest
***  square Interface is Implimented in arthamatic Class ***
Enter the Number : 20
Square of 20 is 400

E:\manu>java Totest
***  square Interface is Implimented in arthamatic Class ***
Enter the Number : 5
Square of 5 is 25
```

c). Create an outer class with a function display, again create another class inside the outer class named inner with a function called display and call the two functions in the main class.

twoclasses.java

```
class outerclass{
    void display(){
        System.out.println("Outer class : This Holds Another Class");
    }
    class innerclass{
        void display1(){
            System.out.println("Inner class : The Class Within Another Class");
        }
    }
}

public class twoclasses {
    public static void main(String[] args) {
        outerclass obj = new outerclass();
        outerclass.innerclass obj1 = obj.new innerclass();
        obj1.display1();
        obj.display();
    }
}
```

OUTPUT:



```
E:\manu>javac twoclasses.java

E:\manu>java twoclasses
Inner class : The Class Within Another Class
Outer class : This Holds Another Class

E:\manu>
```

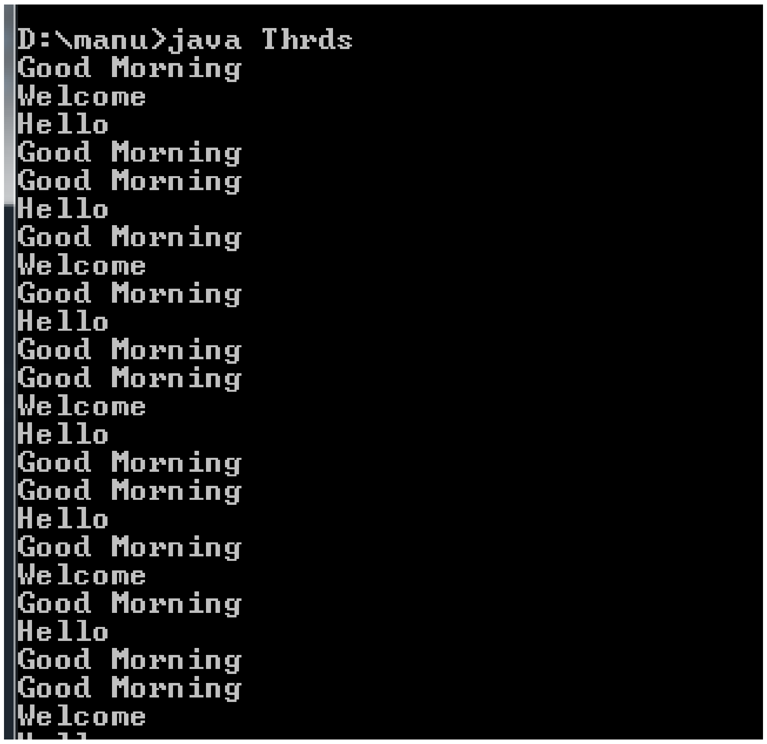
4.

a). Write a JAVA program that creates threads by extending Thread class .First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable)

```
class Thread1 extends Thread {
    public void run() {
        while (true) {
            System.out.println("Good Morning");
            try {
                Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

class Thread2 extends Thread {
    public void run() {
        while (true) {
            System.out.println("Hello");
            try {
                Thread.sleep(2000); // Sleep for 2 seconds
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}
```

```
class Thread3 extends Thread {
    public void run() {
        while (true) {
            System.out.println("Welcome");
            try {
                Thread.sleep(3000); // Sleep for 3 seconds
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

// Main.java
public class Thrds{
    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        Thread2 t2 = new Thread2();
        Thread3 t3 = new Thread3();

        t1.start();
        t2.start();
        t3.start();
    }
}
```

OUTPUT:

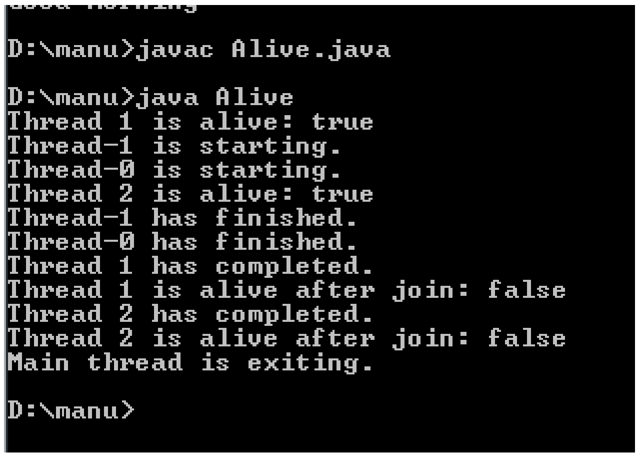


```
D:\manu>java Thrds
Good Morning
Welcome
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Welcome
```

b). Write a program illustrating isAlive and join ()

```
class MyThread extends Thread {
    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName() + " is starting.");
        try {
            // Simulate some work with sleep
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread().getName() + " was interrupted.");
        }
        System.out.println(Thread.currentThread().getName() + " has finished.");
    }
}

public class Alive {
    public static void main(String[] args) {
        MyThread thread1 = new MyThread();
        MyThread thread2 = new MyThread();

        // Start the threads
        thread1.start();
        thread2.start();

        // Check if the threads are alive
        System.out.println("Thread 1 is alive: " + thread1.isAlive());
        System.out.println("Thread 2 is alive: " + thread2.isAlive());

        try {
            // Wait for thread1 to finish
            thread1.join();
            System.out.println("Thread 1 has completed.");
            // Check if thread1 is still alive
            System.out.println("Thread 1 is alive after join: " + thread1.isAlive());

            // Wait for thread2 to finish
            thread2.join();
            System.out.println("Thread 2 has completed.");

            // Check if thread2 is still alive
            System.out.println("Thread 2 is alive after join: " + thread2.isAlive());
        } catch (InterruptedException e) {
            System.out.println("Main thread was interrupted.");
        }

        System.out.println("Main thread is exiting.");
    }
}
```

OUTPUT:



```
D:\manu>javac Alive.java

D:\manu>java Alive
Thread 1 is alive: true
Thread-1 is starting.
Thread-0 is starting.
Thread 2 is alive: true
Thread-1 has finished.
Thread-0 has finished.
Thread 1 has completed.
Thread 1 is alive after join: false
Thread 2 has completed.
Thread 2 is alive after join: false
Main thread is exiting.

D:\manu>
```

5

a). Write a Java program to perform employee payroll processing using packages. In the java file, Emp.java creates a package employee and creates a class Emp. Declare the variables name,empid, category, bpay, hra, da, npay, pf, grosspay, incometax, and allowance. Calculate the values in methods. Create another java file emppay.java. Create an object e to call the methods to perform and print values.

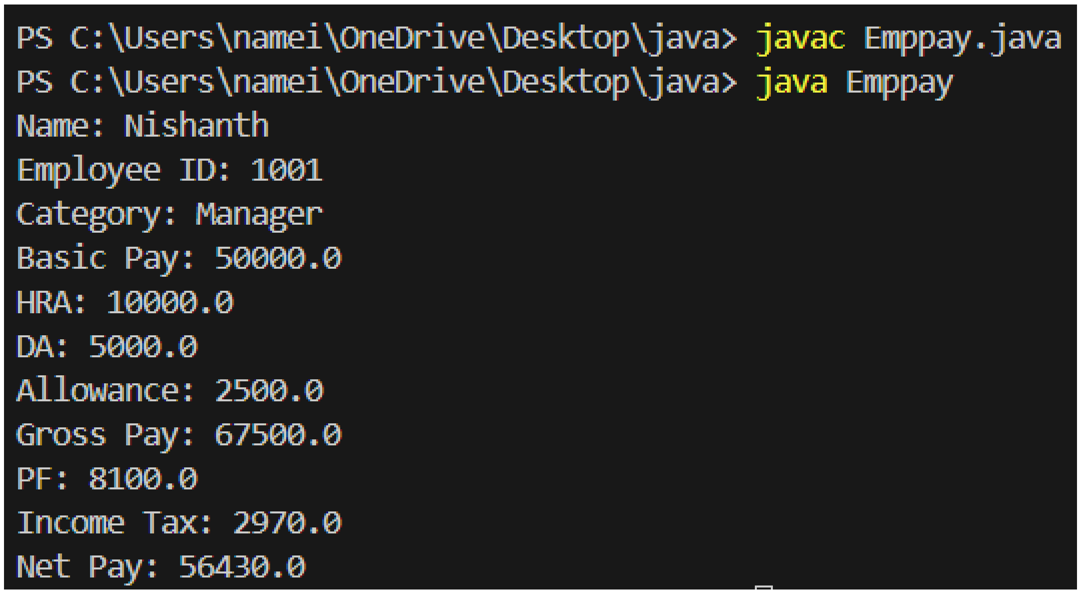
emp.java

```
package employee;
public class emp {
    String name;
    int empid;
    String category;
    double bpay;
    double hra;
    double da;
    double npay;
    double pf;
    double grosspay;
    double incometax;
    double allowance;
    public emp(String name, int empid, String category, double bpay) {
        this.name = name;
        this.empid = empid;
        this.category = category;
        this.bpay = bpay;
        logic();
    }
    public void logic() {
        hra = 0.20 * bpay;
        da = 0.10 * bpay;
        allowance = 0.05 * bpay;
        grosspay = bpay + hra + da + allowance;
        pf = 0.12 * bpay;
        incometax = calculateIncomeTax(grosspay);
        npay = grosspay - (pf + incometax);
    }
    public double calculateIncomeTax(double grosspay) {
        if (grosspay <= 250000) {
            return 0;
        } else if (grosspay <= 500000) {
            return 0.05 * (grosspay - 250000);
        } else {
            return 0.1 * (grosspay - 500000) + 12500;
        }
    }
}
```

Emppay.java

```
import employee.*;
import java.util.*;
public class Emppay {
    public static void Main(String[] args) {
        Scanner inp = new Scanner(System.in);
        System.out.println("ENTER THE EMPLOY NAME : ");
        String name = inp.next();
        System.out.println("ENTER THE EMPLOYEE ID : ");
        int id = inp.nextInt();
        System.out.println("ENTER EMPLOY CATEGORY : ");
        String cat = inp.next();
        System.out.println("ENTER THE BASIC SALARY : ");
        double bpay = inp.nextDouble();
        emp e = new emp(name, id, cat, bpay);
    }
}
```

OUTPUT:



```
PS C:\Users\namei\OneDrive\Desktop\java> javac Emppay.java
PS C:\Users\namei\OneDrive\Desktop\java> java Emppay
Name: Nishanth
Employee ID: 1001
Category: Manager
Basic Pay: 50000.0
HRA: 10000.0
DA: 5000.0
Allowance: 2500.0
Gross Pay: 67500.0
PF: 8100.0
Income Tax: 2970.0
Net Pay: 56430.0
```

b). Write a Package MCA which has one class Student. Accept student detail through parameterized constructor. Write display () method to display details. Create a main class which will use package and calculate total marks and percentage.

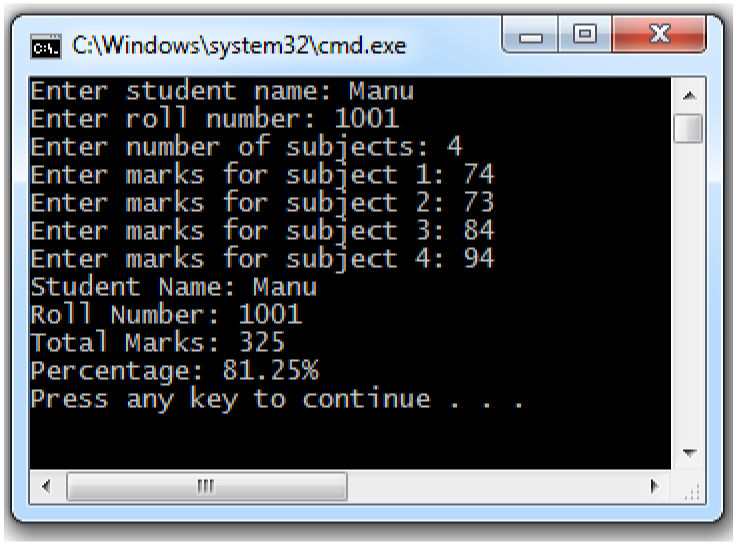
MCA/Student.java

```
package MCA;
public class Student {
private String name;
private int rollNumber;
private int[] marks;
private int totalMarks;
private double percentage;

public Student(String name, int rollNumber, int[] marks) {
this.name = name;
this.rollNumber = rollNumber;
this.marks = marks;
calculateTotalMarks();
calculatePercentage();
}

private void calculateTotalMarks() {
totalMarks = 0;
for (int mark : marks) {
totalMarks += mark;
}
}
private void calculatePercentage() {
if (marks.length &gt; 0) {
percentage = (totalMarks * 100.0) / (marks.length * 100);
} else {
percentage = 0;
}
}
public void display() {
System.out.println(&quot;Student Name: &quot; + name);
System.out.println(&quot;Roll Number: &quot; + rollNumber);
System.out.println(&quot;Total Marks: &quot; + totalMarks);
System.out.println(&quot;Percentage: &quot; + String.format(&quot;%.2f&quot;, percentage) +
&quot;%&quot;);
}
}
```

Main3.java

```
import MCA.Student;
import java.util.Scanner;
public class Main3 {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);

System.out.print(&quot;Enter student name: &quot;);
String name = scanner.nextLine();
System.out.print(&quot;Enter roll number: &quot;);
int rollNumber = scanner.nextInt();
System.out.print(&quot;Enter number of subjects: &quot;);
int numSubjects = scanner.nextInt();
int[] marks = new int[numSubjects];
for (int i = 0; i &lt; numSubjects; i++) {
System.out.print(&quot;Enter marks for subject &quot; + (i + 1) + &quot;: &quot;);
marks[i] = scanner.nextInt();
}
Student student = new Student(name, rollNumber, marks);
student.display();
scanner.close();
}
}
```

OUTPUT:



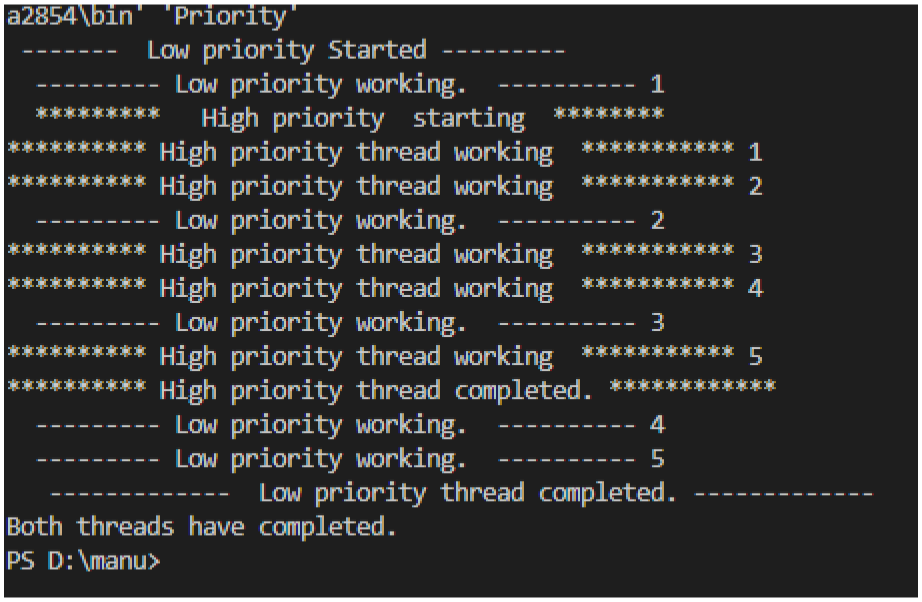
```
Enter student name: Manu
Enter roll number: 1001
Enter number of subjects: 4
Enter marks for subject 1: 74
Enter marks for subject 2: 73
Enter marks for subject 3: 84
Enter marks for subject 4: 94
Student Name: Manu
Roll Number: 1001
Total Marks: 325
Percentage: 81.25%
Press any key to continue . . .
```

6.

a). Write a complex program to illustrate how the thread priorities? Imagine that the first thread has just begun to run, even before it has a chance to do anything. Now comes the higher priority thread that wants to run as well. Now the higher priority thread has to do its work before the first thread starts.

```
class Low extends Thread {
    public void run() {
        System.out.println(" -------  Low priority Started ---------");
        for (int i = 0; i < 5; i++) {
            System.out.println("  --------- Low priority working.  ---------- " + (i + 1));
            try {
                // Simulating work with sleep
                Thread.sleep((long) (Math.random() * 500));
            } catch (InterruptedException e) {
                System.out.println("  ----------- Low priority thread interrupted  ---------");
            }
        }
        System.out.println("   -------------  Low priority thread completed. -------------");
    }
}

class High extends Thread {
    public void run() {
        System.out.println("  *********   High priority  starting  ********");
        for (int i = 0; i < 5; i++) {
            System.out.println("********** High priority thread working  *********** " + (i + 1));
            try {
                // Simulating work with sleep
                Thread.sleep((long) (Math.random() * 300));
            } catch (InterruptedException e) {
                System.out.println("************ High priority thread interrupted.  ************");
            }
        }
        System.out.println("********** High priority thread completed. ************");
    }
}

public class Priority {
    public static void main(String[] args) {
        // Create threads
        Low low = new Low();
        High high = new High();

        low.setPriority(Thread.MIN_PRIORITY); // Lower priority
        high.setPriority(Thread.MAX_PRIORITY); // Higher priority

        low.start();

        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            System.out.println("  !!!!!!   Main thread interrupted.  !!!!!");
        }

        high.start();

        try {
            low.join();
            high.join();
        } catch (InterruptedException e) {
            System.out.println("  !!!!!!!!!!!  Main thread interrupted while waiting for threads
to     finish. !!!!!!!!!!!");
        }
        System.out.println("Both threads have completed.");
    }
}
```

OUTPUT:



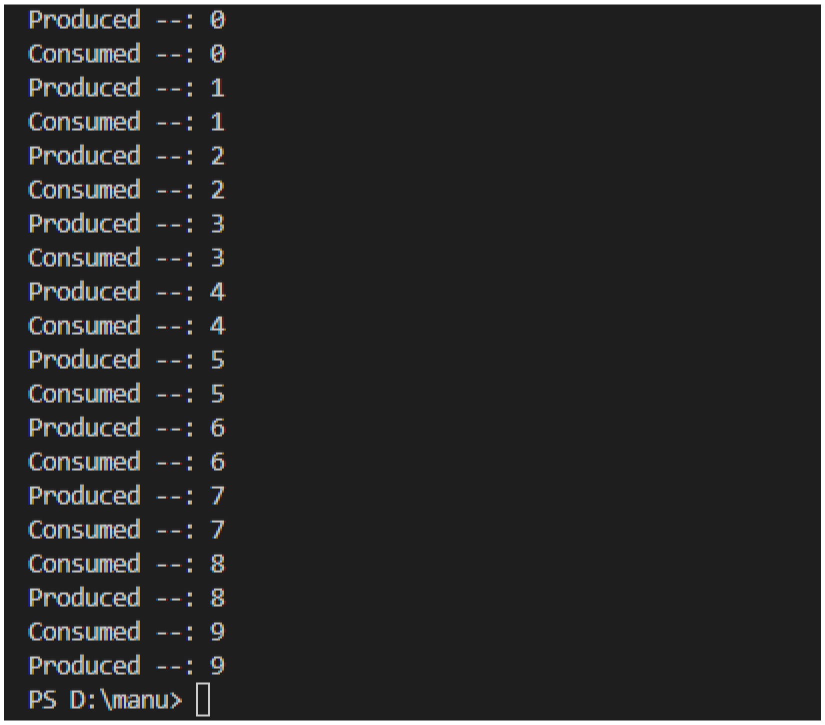
```
a2854\bin' 'Priority'
 -------  Low priority Started ---------
  --------- Low priority working.  ---------- 1
  *********   High priority  starting  ********
********** High priority thread working  *********** 1
********** High priority thread working  *********** 2
  --------- Low priority working.  ---------- 2
********** High priority thread working  *********** 3
********** High priority thread working  *********** 4
  --------- Low priority working.  ---------- 3
********** High priority thread working  *********** 5
********** High priority thread completed. ************
  --------- Low priority working.  ---------- 4
  --------- Low priority working.  ---------- 5
   -------------  Low priority thread completed. -------------
Both threads have completed.
PS D:\manu>
```

b). Write a Java program that implements producer consumer problem using the concept of inter thread communication.

```
class Buffer {
    private int data;
    private boolean available = false;
    public synchronized void put(int value) {
        while (available) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        data = value;
        available = true;
        notifyAll();
    }
    public synchronized int get() {
        while (!available) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        available = false;
        notifyAll();
        return data;
    }
}
```

```
class Producer extends Thread {
    private Buffer buffer;
    public Producer(Buffer buffer) {
        this.buffer = buffer;
    }
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            buffer.put(i);
            System.out.println("Produced --: " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}
class Consumer extends Thread {
    private Buffer buffer;
    public Consumer(Buffer buffer) {
        this.buffer = buffer;
    }
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            int data = buffer.get();
            System.out.println("Consumed --: " + data);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

// Main class
public class PCproblem {
    public static void main(String[] args) {
        Buffer buffer = new Buffer();
        Producer producer = new Producer(buffer);
        Consumer consumer = new Consumer(buffer);
        producer.start();
        consumer.start();
    }
}
```

OUTPUT:

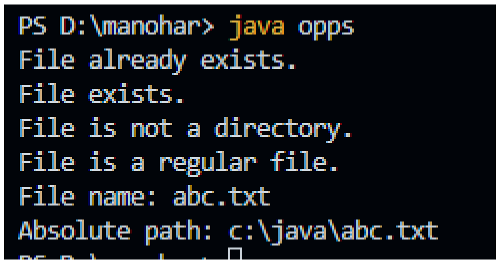


```
Produced --: 0
Consumed --: 0
Produced --: 1
Consumed --: 1
Produced --: 2
Consumed --: 2
Produced --: 3
Consumed --: 3
Produced --: 4
Consumed --: 4
Produced --: 5
Consumed --: 5
Produced --: 6
Consumed --: 6
Produced --: 7
Consumed --: 7
Consumed --: 8
Produced --: 8
Consumed --: 9
Produced --: 9
PS D:\manu>
```

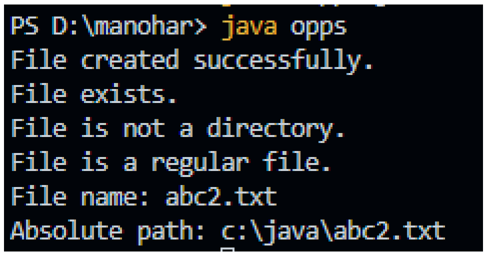
7 a). Write a program to create a text file in the path c:\java\abc.txt and check whether that file is exists. Using the command exists(), isDirectory(), isFile(), getName() and getAbsolutePath().

```
import java.io.File;
import java.io.IOException;
public class opps{
    public static void main(String[] args) {
        String filePath = "c:/java/abc.txt";
        File file = new File(filePath);
        try {
            boolean created = file.createNewFile();
            if (created) {
                System.out.println("File created successfully.");
            } else {
                System.out.println("File already exists. ");
            }
            if (file.exists()) {
                System.out.println("File exists.");
            } else {
                System.out.println("File does not exist.");
            }
            if (file.isDirectory()) {
                System.out.println("File is a directory.");
            } else {
                System.out.println("File is not a directory.");
            }
            if (file.isFile()) {
                System.out.println("File is a regular file.");
            } else {
                System.out.println("File is not a regular file.");
            }
            String fileName = file.getName();
            System.out.println("File name: " + fileName);
            String absolutePath = file.getAbsolutePath();
            System.out.println("Absolute path: " + absolutePath);
        } catch (IOException e) {
            System.out.println("An error occurred while creating the file.");
            e.printStackTrace();
        }
    }
}
```

OUTPUT :



```
PS D:\manohar> java opps
File already exists.
File exists.
File is not a directory.
File is a regular file.
File name: abc.txt
Absolute path: c:\java\abc.txt
```

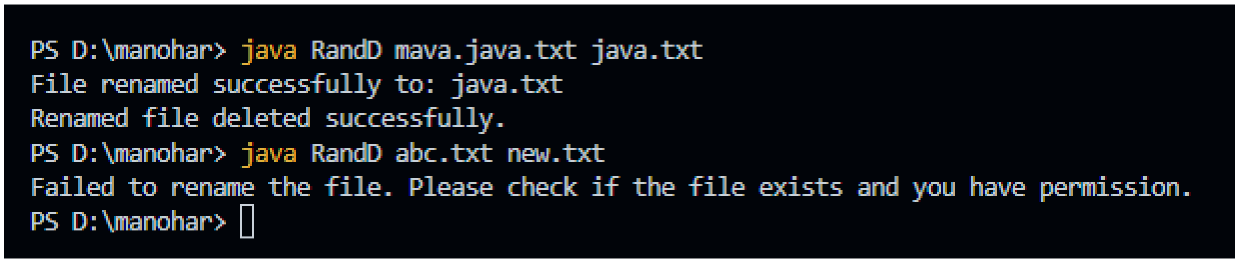


```
PS D:\manohar> java opps
File created successfully.
File exists.
File is not a directory.
File is a regular file.
File name: abc2.txt
Absolute path: c:\java\abc2.txt
```

b) Write a program to rename the given file, after renaming the file delete the renamed file. (Accept the file name using command line arguments.)

```
import java.io.File;
public class RandD {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java RenameAndDeleteFile <oldFileName> <newFileName>");
            return;
        }
        String oldFileName = args[0];
        String newFileName = args[1];
        File oldFile = new File(oldFileName);
        File newFile = new File(newFileName);
        boolean renamed = oldFile.renameTo(newFile);
        if (renamed) {
            System.out.println("File renamed successfully to: " + newFileName);
        } else {
            System.out.println("Failed to rename the file. Please check if the file exists and you have
permission.");
            return;
        }
        boolean deleted = newFile.delete();
        if (deleted) {
            System.out.println("Renamed file deleted successfully.");
        } else {
            System.out.println("Failed to delete the renamed file. Please check if you have
permission.");
        }
    }
}
```

OUTPUT:

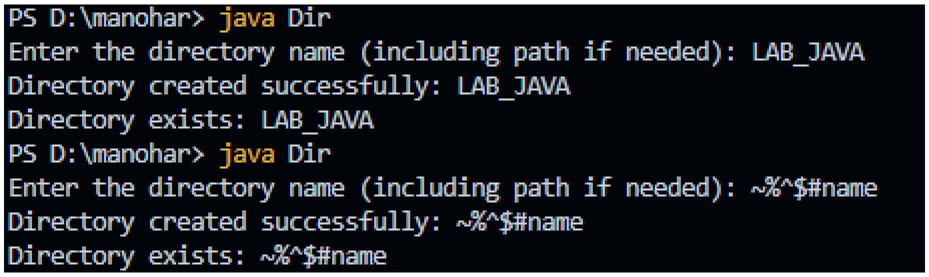


```
PS D:\manohar> java RandD mava.java.txt java.txt
File renamed successfully to: java.txt
Renamed file deleted successfully.
PS D:\manohar> java RandD abc.txt new.txt
Failed to rename the file. Please check if the file exists and you have permission.
PS D:\manohar> 
```

c) Write a program to create a directory and check whether the directory is created.

```
import java.io.File;
import java.util.Scanner;
public class Dir{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the directory name (including path if needed): ");
        String directoryPath = scanner.nextLine();
        File directory = new File(directoryPath);
        boolean created = directory.mkdir();
        if (created) {
            System.out.println("Directory created successfully: " + directoryPath);
        } else {
            if (directory.exists()) {
                System.out.println("Directory already exists: " + directoryPath);
            } else {
                System.out.println("Failed to create directory. Please check the path and permissions.");
            }
        }
        if (directory.exists() && directory.isDirectory()) {
            System.out.println("Directory exists: " + directoryPath);
        } else {
            System.out.println("Directory does not exist: " + directoryPath);
        }
    }
}
```

OUTPUT:



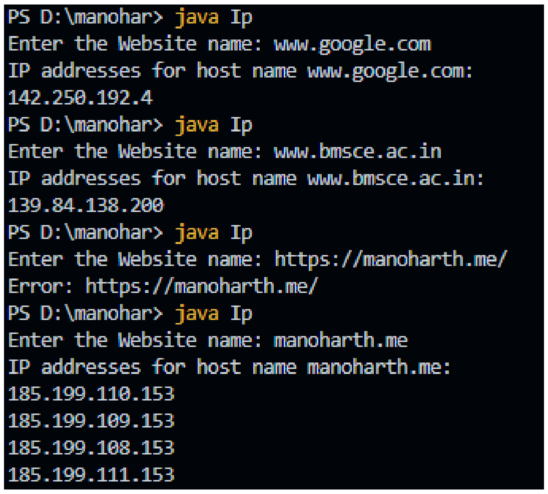
```
PS D:\manohar> java Dir
Enter the directory name (including path if needed): LAB_JAVA
Directory created successfully: LAB_JAVA
Directory exists: LAB_JAVA
PS D:\manohar> java Dir
Enter the directory name (including path if needed): ~%^$#name
Directory created successfully: ~%^$#name
Directory exists: ~%^$#name
```

8

a). Write a program that can read a host name and convert it to an IP address

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Scanner;
public class Ip{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the Website name: ");
        String hostName = scanner.nextLine();
        try {
            InetAddress[] addresses = InetAddress.getAllByName(hostName);
            System.out.println("IP addresses for host name " + hostName + ":");
            for (InetAddress address : addresses) {
                System.out.println(address.getHostAddress());
            }
        } catch (UnknownHostException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

OUTPUT:



```
PS D:\manohar> java Ip
Enter the Website name: www.google.com
IP addresses for host name www.google.com:
142.250.192.4
PS D:\manohar> java Ip
Enter the Website name: www.bmsce.ac.in
IP addresses for host name www.bmsce.ac.in:
139.84.138.200
PS D:\manohar> java Ip
Enter the Website name: https://manoharth.me/
Error: https://manoharth.me/
PS D:\manohar> java Ip
Enter the Website name: manoharth.me
IP addresses for host name manoharth.me:
185.199.110.153
185.199.109.153
185.199.108.153
185.199.111.153
```

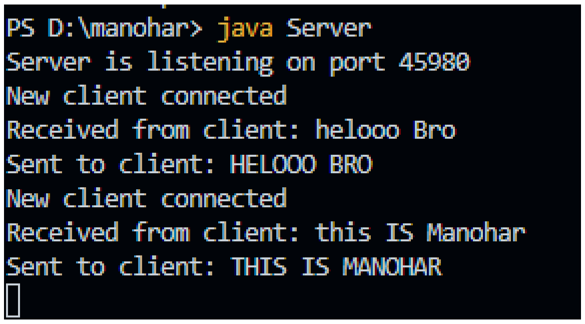
b). Write a Socket base java server program that responds to client messages as follows: When it receives a message from client, it simply converts the message into all uppercase letters and sends back to the client. Write both client and server programs demonstrating this.

Server.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) {
        int port = 45980;
        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Server is listening on port " + port);
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected");
                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                String message = in.readLine();
                System.out.println("Received from client: " + message);
                String response = message.toUpperCase();
                out.println(response);
                System.out.println("Sent to client: " + response);
                clientSocket.close();
            }
        }
```

```
 catch (IOException e) {
            System.out.println("Server exception: " + e.getMessage());
        }
    }
}
```

OUTPUT:

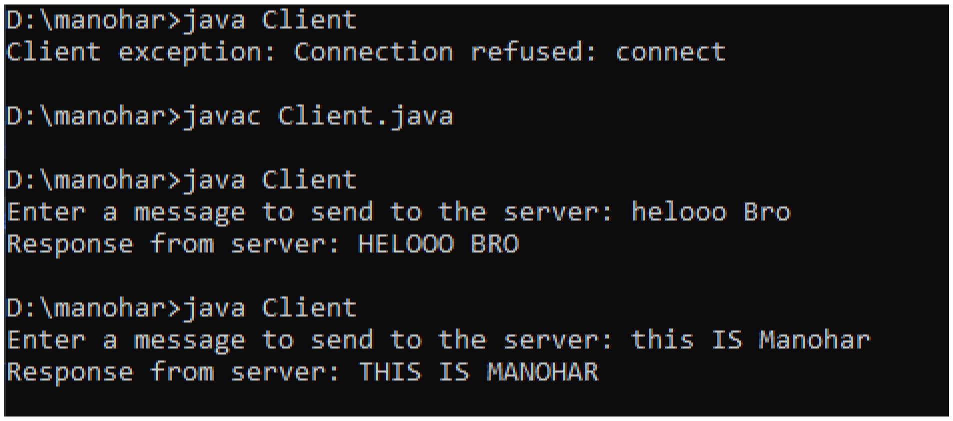


```
PS D:\manohar> java Server
Server is listening on port 45980
New client connected
Received from client: helooo Bro
Sent to client: HELOOO BRO
New client connected
Received from client: this IS Manohar
Sent to client: THIS IS MANOHAR
```

Client.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 45980; // Port number for the server

        try (Socket socket = new Socket(hostname, port)) {
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new
BufferedReader(new    InputStreamReader(socket.getInputStream()));
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a message to send to the server: ");
            String message = scanner.nextLine();
            out.println(message);
            String response = in.readLine();
            System.out.println("Response from server: " + response);
        } catch (IOException e) {
            System.out.println("Client exception: " + e.getMessage());
        }
    }
}
```

OUTPUT:



```
D:\manohar>java Client
Client exception: Connection refused: connect

D:\manohar>javac Client.java

D:\manohar>java Client
Enter a message to send to the server: helooo Bro
Response from server: HELOOO BRO

D:\manohar>java Client
Enter a message to send to the server: this IS Manohar
Response from server: THIS IS MANOHAR
```

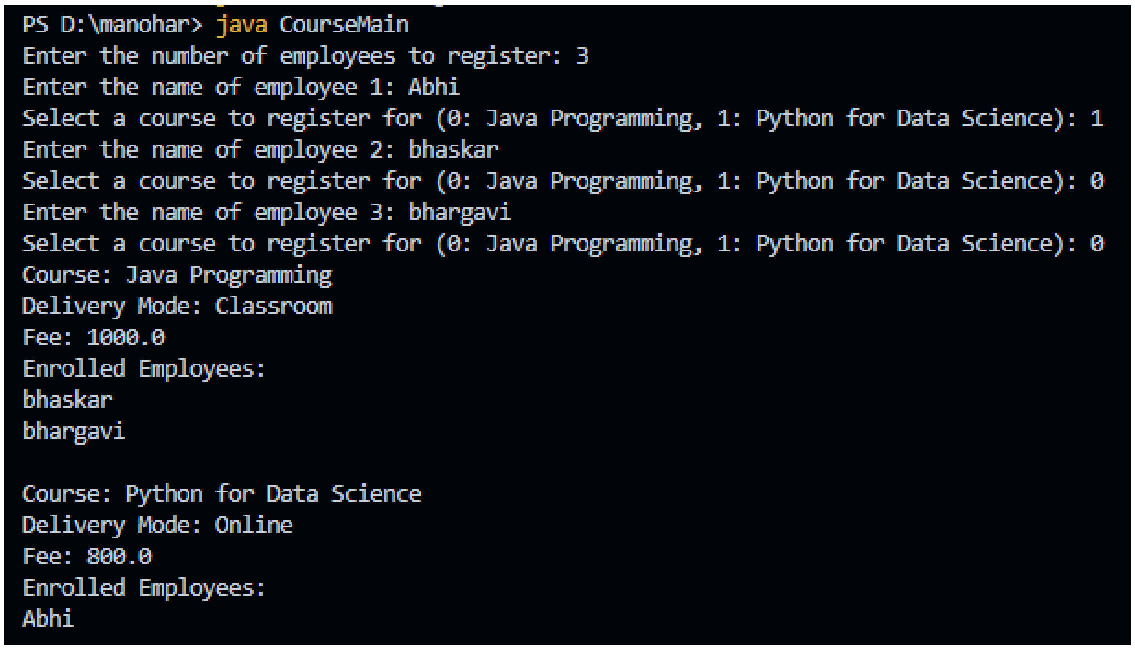
9

a). Create the classes required to store data regarding different types of courses that employees in a company can enroll for. All courses have name and course fee. Courses are also either classroom delivered or delivered online. Courses could also be full time or part time. The program must enable the course coordinator to register employees for courses and list out employees registered for specific courses.

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class Course {
    private String name;
    private double fee;
    private String deliveryMode; // "Classroom" or "Online"
    public Course(String name, double fee, String deliveryMode) {
        this.name = name;
        this.fee = fee;
        this.deliveryMode = deliveryMode;
    }
    public String getName() {
        return name;
    }
    public double getFee() {
        return fee;
    }
    public String getDeliveryMode() {
        return deliveryMode;
    }
}
```

```
class Employee {
    private String name;

    public Employee(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}

class CourseCoordinator {
    private List<Course> courses;
    private List<List<Employee>> enrolledEmployees; // List of lists for enrolled employees
    public CourseCoordinator() {
        courses = new ArrayList<>();
        enrolledEmployees = new ArrayList<>();
    }
    public void addCourse(Course course) {
        courses.add(course);
        enrolledEmployees.add(new ArrayList<Employee>()); // Initialize the list for enrolled
employees
    }

    public void registerEmployee(int courseIndex, Employee employee) {
        if (courseIndex >= 0 && courseIndex < courses.size()) {
            enrolledEmployees.get(courseIndex).add(employee);
        } else {
            System.out.println("Invalid course index.");
        }
    }
    public void listEnrolledEmployees(int courseIndex) {
        if (courseIndex >= 0 && courseIndex < courses.size()) {
            Course course = courses.get(courseIndex);
            System.out.println("Course: " + course.getName());
            System.out.println("Delivery Mode: " + course.getDeliveryMode());
            System.out.println("Fee: " + course.getFee());
            System.out.println("Enrolled Employees:");
            List<Employee> employees = enrolledEmployees.get(courseIndex);
            if (employees.isEmpty()) {
                System.out.println("No employees enrolled.");
            } else {
                for (Employee employee : employees) {
                    System.out.println(employee.getName());
                }
            }
            System.out.println();
        } else {
            System.out.println("Invalid course index.");
        }
    }
    public List<Course> getCourses() {
        return courses;
    }
}
public class CourseMain {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        CourseCoordinator coordinator = new CourseCoordinator();
        coordinator.addCourse(new Course("Java Programming", 1000.0, "Classroom"));
        coordinator.addCourse(new Course("Python for Data Science", 800.0, "Online"));
        System.out.print("Enter the number of employees to register: ");
        int numEmployees = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character
        for (int i = 0; i < numEmployees; i++) {
            System.out.print("Enter the name of employee " + (i + 1) + ": ");
            String employeeName = scanner.nextLine();
            Employee employee = new Employee(employeeName);
            System.out.print("Select a course to register for (0: Java Programming, 1: Python for Data
Science): ");
            int courseIndex = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character
            coordinator.registerEmployee(courseIndex, employee);
        }
        for (int i = 0; i < coordinator.getCourses().size(); i++) { // Loop through the number of courses
            coordinator.listEnrolledEmployees(i);
        }
    }
}
```

Output:



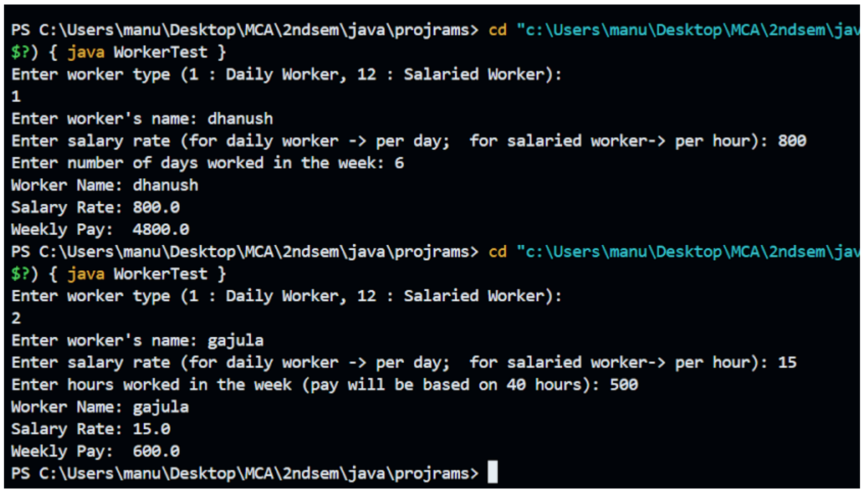
```
PS D:\manohar> java CourseMain
Enter the number of employees to register: 3
Enter the name of employee 1: Abhi
Select a course to register for (0: Java Programming, 1: Python for Data Science): 1
Enter the name of employee 2: bhaskar
Select a course to register for (0: Java Programming, 1: Python for Data Science): 0
Enter the name of employee 3: bhargavi
Select a course to register for (0: Java Programming, 1: Python for Data Science): 0
Course: Java Programming
Delivery Mode: Classroom
Fee: 1000.0
Enrolled Employees:
bhaskar
bhargavi

Course: Python for Data Science
Delivery Mode: Online
Fee: 800.0
Enrolled Employees:
Abhi
```

b). Write a java program in to create a class Worker. Write classes DailyWorker and SalariedWorker that inherit from Worker. Every worker has a name and a salaryrate. Write method Pay (int hours) to computethe week pay of every worker. A Daily worker ia paid on the basis of the number of days she/he works. The salaried worker gets paid the wage for 40 hours a week no matter what the actual hours are. Test this program to calculate the pay of workers.

```
import java.util.Scanner;
class Worker {
    protected String name;
    protected double salaryRate;
    public Worker(String name, double salaryRate) {
        this.name = name;
        this.salaryRate = salaryRate;
    }
    public double pay(int hours) {
        return 0.0;
    }
    public void displayDetails() {
        System.out.println("Worker Name: " + name);
        System.out.println("Salary Rate: " + salaryRate);
    }
}

class DailyWorker extends Worker {
    public DailyWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }
    @Override
    public double pay(int daysWorked) {
        return salaryRate * daysWorked;
    }
}
```

```
class SalariedWorker extends Worker {
    public SalariedWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }
    @Override
    public double pay(int hoursWorked) {
        return salaryRate * 40;
    }
}
public class WorkerTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter worker type (1 : Daily Worker, 12 : Salaried Worker): ");
        int workerType = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter worker's name: ");
        String name = scanner.nextLine();
        System.out.print("Enter salary rate (for daily worker -> per day;  for salaried worker-> per
hour): ");
        double salaryRate = scanner.nextDouble();
        Worker worker;
        if (workerType == 1) {
            worker = new DailyWorker(name, salaryRate);
            System.out.print("Enter number of days worked in the week: ");
            int daysWorked = scanner.nextInt();
            worker.displayDetails();
            System.out.println("Weekly Pay:  " + worker.pay(daysWorked));
        }
 else if (workerType == 2) {
            worker = new SalariedWorker(name, salaryRate);
            System.out.print("Enter hours worked in the week (pay will be based on 40 hours): ");
            int hoursWorked = scanner.nextInt();
            worker.displayDetails();
            System.out.println("Weekly Pay:  " + worker.pay(hoursWorked));
        } else {
            System.out.println("Invalid worker type entered.");
        }
        scanner.close();
    }
}
```

OUTPUT:



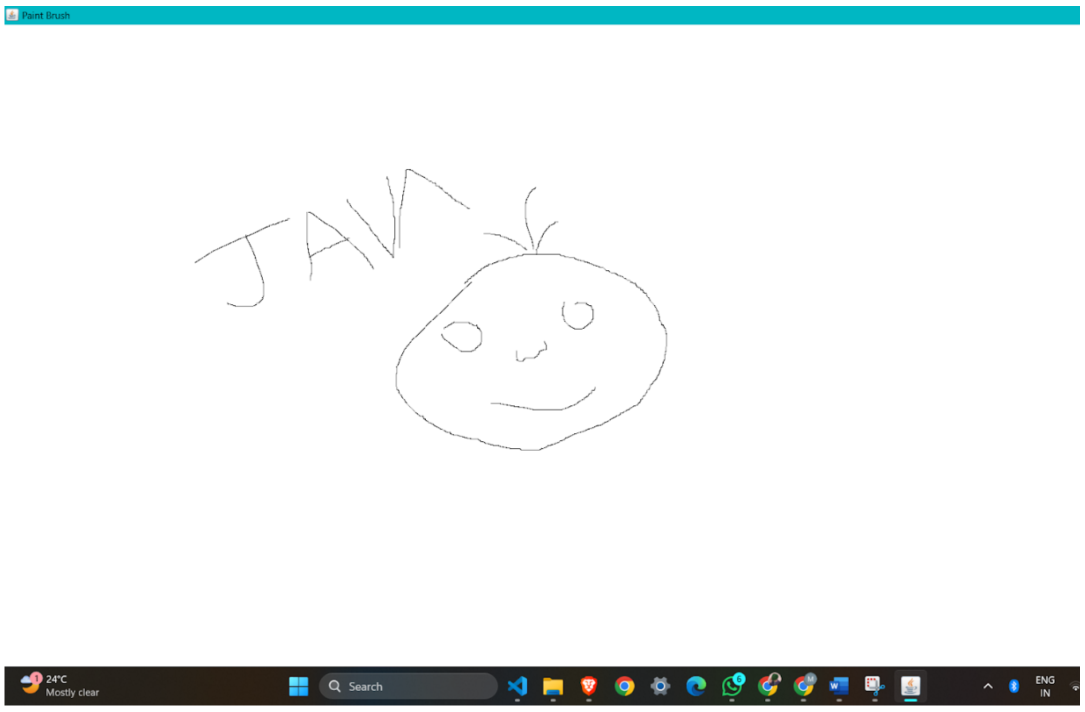
```
PS C:\Users\manu\Desktop\MCA\2ndsem\java\projrams> cd "c:\Users\manu\Desktop\MCA\2ndsem\jav
$?) { java WorkerTest }
Enter worker type (1 : Daily Worker, 12 : Salaried Worker):
1
Enter worker's name: dhanush
Enter salary rate (for daily worker -> per day;  for salaried worker-> per hour): 800
Enter number of days worked in the week: 6
Worker Name: dhanush
Salary Rate: 800.0
Weekly Pay:  4800.0
PS C:\Users\manu\Desktop\MCA\2ndsem\java\projrams> cd "c:\Users\manu\Desktop\MCA\2ndsem\jav
$?) { java WorkerTest }
Enter worker type (1 : Daily Worker, 12 : Salaried Worker):
2
Enter worker's name: gajula
Enter salary rate (for daily worker -> per day;  for salaried worker-> per hour): 15
Enter hours worked in the week (pay will be based on 40 hours): 500
Worker Name: gajula
Salary Rate: 15.0
Weekly Pay:  600.0
PS C:\Users\manu\Desktop\MCA\2ndsem\java\projrams> 
```

10.

a).Write a JAVA program to paint like paint brush in applet.

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.awt.Graphics2D;
import java.awt.BasicStroke;
public class PaintApplet extends Applet {
    private int lastX, lastY;
    private boolean dragging = false;
    @Override
    public void init() {
        setSize(800, 600);
        setBackground(Color.WHITE);
        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                lastX = e.getX();
                lastY = e.getY();
                dragging = true;
            }
            @Override
            public void mouseReleased(MouseEvent e) {
                dragging = false;
            }
        });
    addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                if (dragging) {
                    Graphics g = getGraphics();
                    Graphics2D g2d = (Graphics2D) g;
                    g2d.setColor(Color.BLACK);
                    g2d.setStroke(new BasicStroke(2));
                    g2d.drawLine(lastX, lastY, e.getX(), e.getY());
                    lastX = e.getX();
                    lastY = e.getY();
                }
            }
        });
    }
}
```

OUTPUT:



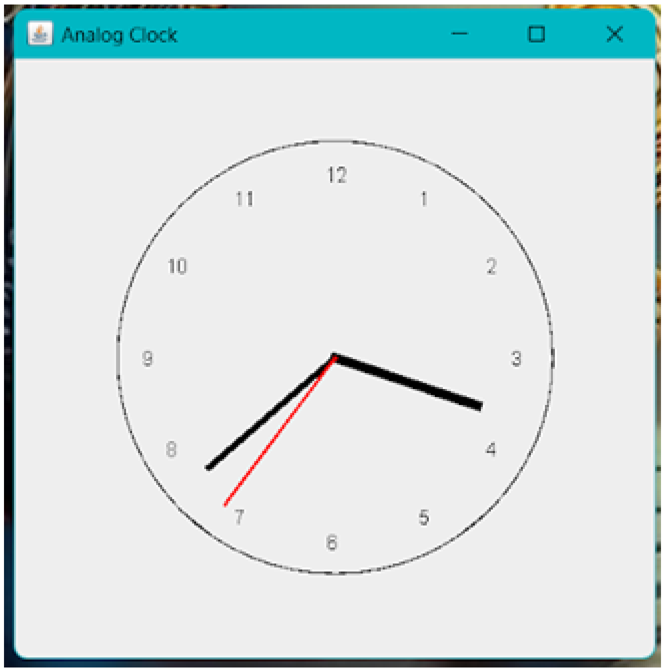
10) b) Write a JAVA program to display analog clock using Applet.

```
import java.applet.*;
import java.awt.*;
import java.util.Calendar;
public class AnalogClock extends Applet implements Runnable {
        Thread t = null;
        int width, height;
        int hours = 0, minutes = 0, seconds = 0;
        String timeString = "";
        public void init() {
        width = getSize().width;
     height = getSize().height;
     setBackground(Color.white);
     setForeground(Color.black);
        }
        public void start() {
        t = new Thread(this);
     t.start();
        }
        public void run() {
        while (true) {
        repaint();
        try {
            Thread.sleep(1000); // Redraw every second
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        }
        }


        public void paint(Graphics g) {
```

```
    width = getSize().width;
height = getSize().height;

Calendar cal = Calendar.getInstance();
    hours = cal.get(Calendar.HOUR);
minutes = cal.get(Calendar.MINUTE);
seconds = cal.get(Calendar.SECOND);
g.setColor(Color.black);
g.drawOval(50, 50, 200, 200);
    int xCenter = 150;
    int yCenter = 150;

    // Calculate angles for each hand
double secondAngle = (seconds * 6) * (Math.PI / 180); // 6 degrees per second
double minuteAngle = (minutes * 6 + seconds * 0.1) * (Math.PI / 180); // 6 degrees per minute
double hourAngle = (hours * 30 + minutes * 0.5) * (Math.PI / 180); // 30 degrees per hour

    // Second hand
    int xSecond = (int) (xCenter + 90 * Math.sin(secondAngle));
    int ySecond = (int) (yCenter - 90 * Math.cos(secondAngle));
g.setColor(Color.red);
g.drawLine(xCenter, yCenter, xSecond, ySecond);

    // Minute hand
    int xMinute = (int) (xCenter + 70 * Math.sin(minuteAngle));
    int yMinute = (int) (yCenter - 70 * Math.cos(minuteAngle));
g.setColor(Color.blue);
g.drawLine(xCenter, yCenter, xMinute, yMinute);

    // Hour hand
    int xHour = (int) (xCenter + 50 * Math.sin(hourAngle));
    int yHour = (int) (yCenter - 50 * Math.cos(hourAngle));
```

```
        g.setColor(Color.green);
           g.drawLine(xCenter, yCenter, xHour, yHour);


           // Draw the numbers around the clock
        g.setColor(Color.black);
        g.drawString("12", 140, 70);
        g.drawString("3", 240, 150);
        g.drawString("6", 145, 240);
           g.drawString("9", 60, 150);
           }
  }
```

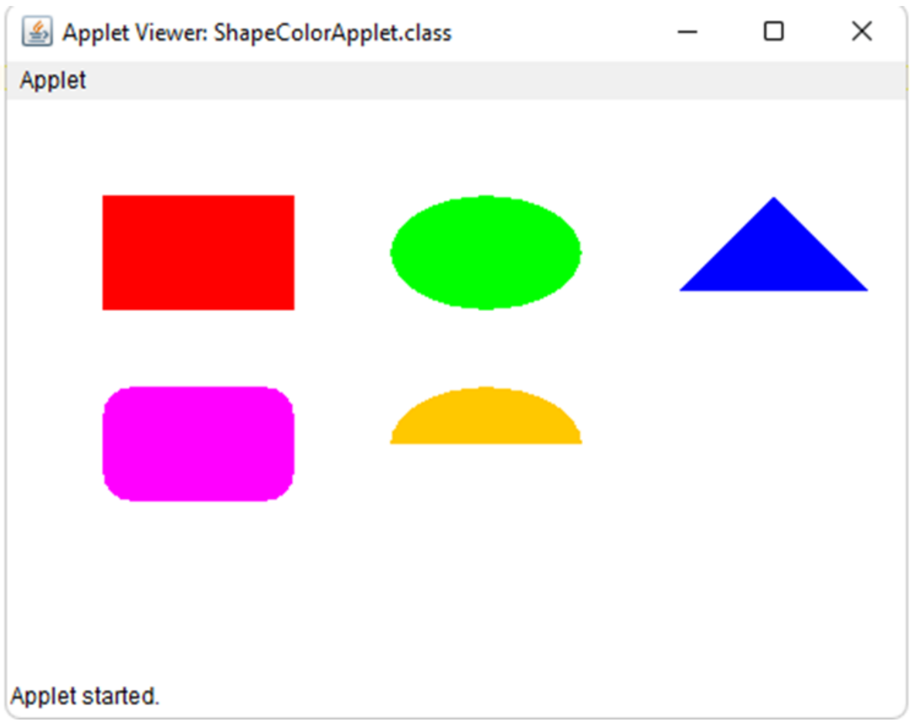
OUTPUT:



c) Write a JAVA program to create different shapes and fill colors using Applet.

```
import java.applet.Applet;
import java.awt.*;
public class ShapeApplet extends Applet {

        // Method to paint the shapes
        public void paint(Graphics g) {
        // Cast the Graphics object to Graphics2D for more control
     Graphics2D g2d = (Graphics2D) g;

     g2d.setColor(Color.RED);
     g2d.fillRect(50, 50, 100, 60); // (x, y, width, height)

        // Set color and draw a filled oval
     g2d.setColor(Color.GREEN);
     g2d.fillOval(200, 50, 100, 60); // (x, y, width, height)

        // Set color and draw a filled polygon (triangle)
     g2d.setColor(Color.BLUE);
        int[] xPoints = {350, 400, 450}; // X coordinates of vertices
        int[] yPoints = {100, 50, 100};  // Y coordinates of vertices
     g2d.fillPolygon(xPoints, yPoints, 3); // 3 is the number of vertices

        // Set color and draw a filled round rectangle
     g2d.setColor(Color.MAGENTA);
     g2d.fillRoundRect(50, 150, 100, 60, 30, 30); // (x, y, width, height, arcWidth, arcHeight)
      g2d.setColor(Color.ORANGE);
     g2d.fillArc(200, 150, 100, 60, 0, 180); // (x, y, width, height, startAngle, arcAngle)
        }
}
```

OUTPUT:



```
Applet started.
```

11)

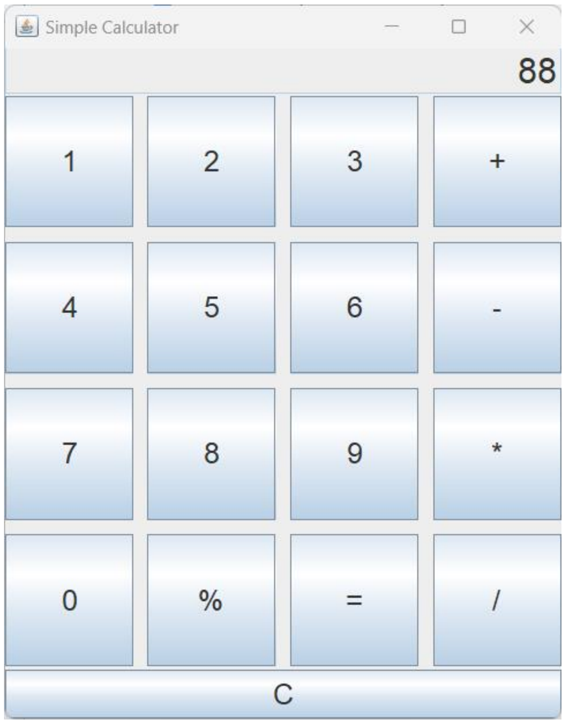
a). Write a JAVA program to build a Calculator in Swings.

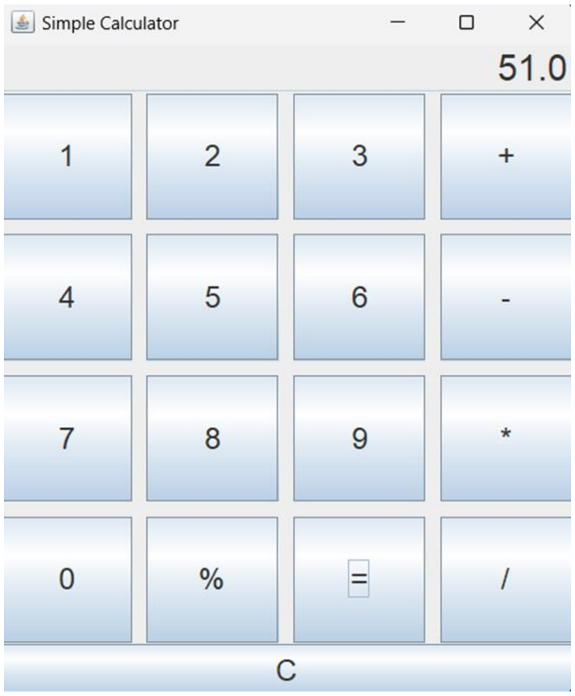
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Calculator extends JFrame implements ActionListener {
    // Components of the calculator
    private JTextField display;
    private JButton[] numberButtons;
    private JButton addButton, subButton, mulButton, divButton, equalButton, clearButton;
    private JPanel panel;

    private double num1, num2, result;
    private char operator;

    // Constructor to set up the calculator
    public Calculator() {
        // Set up the frame
        setTitle("Simple Calculator");
        setSize(400, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Create display field
        display = new JTextField();
        display.setEditable(false);
        display.setFont(new Font("Arial", Font.PLAIN, 24));
        add(display, BorderLayout.NORTH);
```

```
// Create number buttons
numberButtons = new JButton[10];
for (int i = 0; i < 10; i++) {
    numberButtons[i] = new JButton(String.valueOf(i));
    numberButtons[i].addActionListener(this);
}

// Create operator buttons
addButton = new JButton("+");
subButton = new JButton("-");
mulButton = new JButton("*");
divButton = new JButton("/");
equalButton = new JButton("=");
clearButton = new JButton("C");

// Add action listeners for operator buttons
addButton.addActionListener(this);
subButton.addActionListener(this);
mulButton.addActionListener(this);
divButton.addActionListener(this);
equalButton.addActionListener(this);
clearButton.addActionListener(this);
panel = new JPanel();
panel.setLayout(new GridLayout(4, 4, 10, 10));
panel.add(numberButtons[1]);
panel.add(numberButtons[2]);
panel.add(numberButtons[3]);
panel.add(addButton);
panel.add(numberButtons[4]);
panel.add(numberButtons[5]);
panel.add(numberButtons[6]);
panel.add(subButton);
```

```
        panel.add(numberButtons[7]);
        panel.add(numberButtons[8]);
        panel.add(numberButtons[9]);
        panel.add(mulButton);
        panel.add(clearButton);
        panel.add(numberButtons[0]);
        panel.add(equalButton);
        panel.add(divButton);
        add(panel, BorderLayout.CENTER);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        for (int i = 0; i < 10; i++) {
            if (e.getSource() == numberButtons[i]) {
                display.setText(display.getText() + i);
            }
        }
        if (e.getSource() == addButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '+';
            display.setText("");
        } else if (e.getSource() == subButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '-';
            display.setText("");
        } else if (e.getSource() == mulButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '*';
            display.setText("");
        } else if (e.getSource() == divButton) {
            num1 = Double.parseDouble(display.getText());
            operator = '/';
            display.setText("");
        } else if (e.getSource() == equalButton) {
            num2 = Double.parseDouble(display.getText());
            switch (operator) {
                case '+':
                    result = num1 + num2;
                    break;
                case '-':
                    result = num1 - num2;
                    break;
                case '*':
                    result = num1 * num2;
                    break;
                case '/':
                    result = num1 / num2;
                    break;
            }
            display.setText(String.valueOf(result));
        } else if (e.getSource() == clearButton) {
            display.setText("");
        }
    }
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        calculator.setVisible(true);
    }
}
```

OUTPUT:

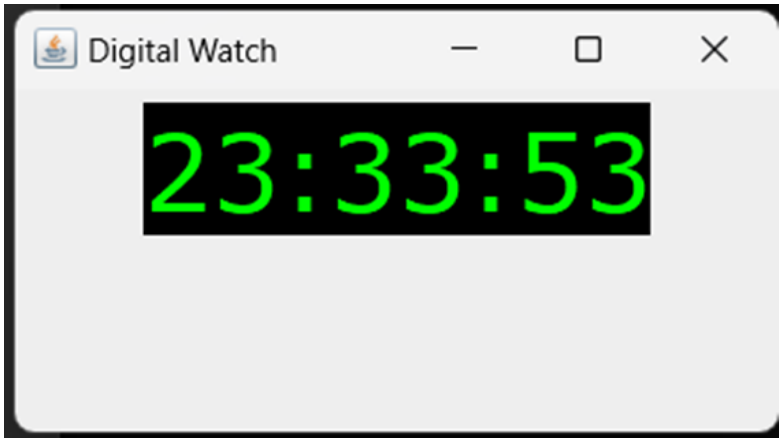

Simple Calculator
88
1
2
3
+
4
5
6
-
7
8
9
*
0
%
=
/
C


Simple Calculator
51.0
1
2
3
+
4
5
6
-
7
8
9
*
0
%
=
/
C

b) Write a JAVA program to display the digital watch using swings.

```
import javax.swing.*;
import java.awt.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class DigitalWatch implements ActionListener {
    JFrame frame;
    JLabel timeLabel;
    Timer timer;

    public DigitalWatch() {
        frame = new JFrame("Digital Watch");
        frame.setLayout(new FlowLayout());
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 150);

        timeLabel = new JLabel();
        timeLabel.setFont(new Font("Verdana", Font.PLAIN, 40));
        timeLabel.setOpaque(true);
        timeLabel.setBackground(Color.BLACK);
        timeLabel.setForeground(Color.GREEN);

        frame.add(timeLabel);
        frame.setVisible(true);

        timer = new Timer(1000, this);
        timer.start();
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        String currentTime = new SimpleDateFormat("HH:mm:ss").format(new Date());
        timeLabel.setText(currentTime);
    }

    public static void main(String[] args) {
        new DigitalWatch();
    }
}
```

OUTPUT:



```
23:33:53
```