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

Chapala Bhargav Krishna Srivastav
(1BM23MC026)

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 **Chapala Bhargav Krishna Srivastav** (1BM23MC026) 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	Signature of HoD
Prof. R. V. Raghavendra Rao	Dr. Ch. Ram Mohan Reddy

Examiner:

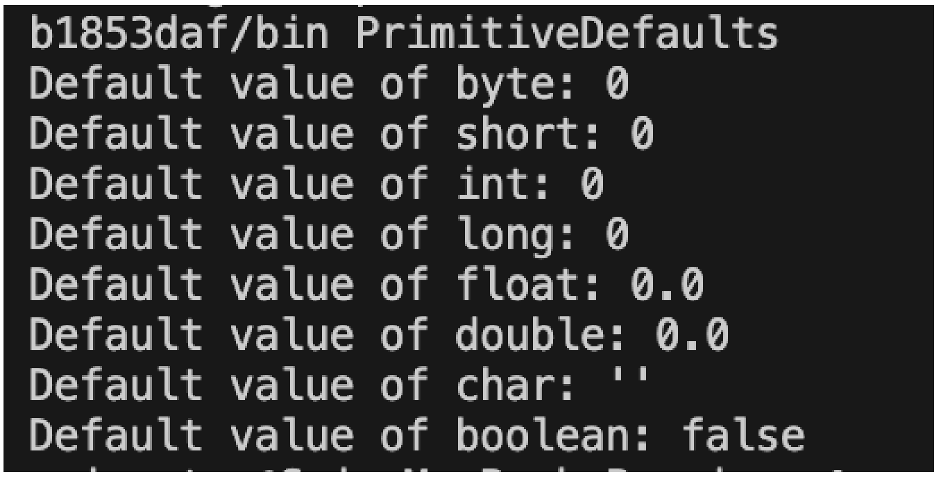
CONTENTS

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. b). Write a Java program that implements producer consumer problem using the concept of inter thread communication.	19-22
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(). 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.) c) Write a program to create a directory and check whether the directory is created.	23-25
8.	a). Write a program that can read a host name and convert it to an IP address 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.	26-28
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. 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 compute the 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.	29-35
10.	a).Write a JAVA program to paint like paint brush in applet. b) Write a JAVA program to display analog clock using Applet. c). Write a JAVA program to create different shapes and fill colors using Applet.	36-40
11.	a).Write a JAVA program to build a Calculator in Swings b). Write a JAVA program to display the digital watch using swings.	41-44

1)

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

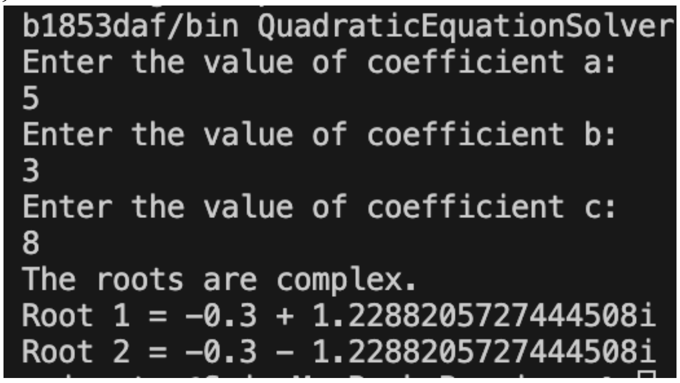
```
public class PrimitiveDefaults {
    byte byteValue;
    short shortValue;
    int intValue;
    long longValue;
    float floatValue;
    double doubleValue;
    char charValue;
    boolean booleanValue;

    public static void main(String[] args) {
        PrimitiveDefaults defaults = new PrimitiveDefaults();

        System.out.println("Default value of byte: " + defaults.byteValue);
        System.out.println("Default value of short: " + defaults.shortValue);
        System.out.println("Default value of int: " + defaults.intValue);
        System.out.println("Default value of long: " + defaults.longValue);
        System.out.println("Default value of float: " + defaults.floatValue);
        System.out.println("Default value of double: " + defaults.doubleValue);
        System.out.println("Default value of char: '" + defaults.charValue + "'");
        System.out.println("Default value of boolean: " + defaults.booleanValue);
    }
}
```



```
b1853daf/bin PrimitiveDefaults
Default value of byte: 0
Default value of short: 0
Default value of int: 0
Default value of long: 0
Default value of float: 0.0
Default value of double: 0.0
Default value of char: ''
Default value of boolean: false
```

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

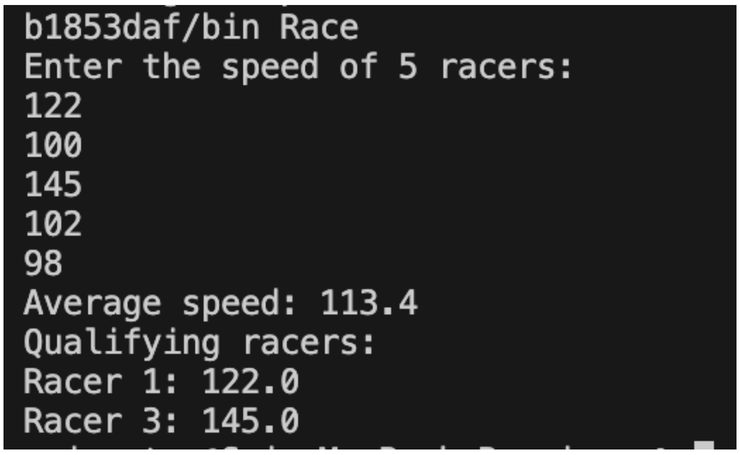
```java
import java.util.Scanner;

class QuadraticEquationSolver {
    public static void solveEquation(double coefficientA, double coefficientB, double coefficientC) {
        if (coefficientA == 0) {
            System.out.println("This is not a quadratic equation (a cannot be zero).");
            return;
        }
        double discriminant = coefficientB * coefficientB - 4 * coefficientA * coefficientC;

        if (discriminant > 0) {
            double root1 = (-coefficientB + Math.sqrt(discriminant)) / (2 * coefficientA);
            double root2 = (-coefficientB - Math.sqrt(discriminant)) / (2 * coefficientA);
            System.out.println("The roots are real and distinct.");
            System.out.println("Root 1 = " + root1);
            System.out.println("Root 2 = " + root2);
        } else if (discriminant == 0) {
            double root = -coefficientB / (2 * coefficientA);
            System.out.println("The equation has one real root.");
            System.out.println("Root = " + root);
        } else {
            System.out.println("The roots are complex.");
            double realPart = -coefficientB / (2 * coefficientA);
            double imaginaryPart = Math.sqrt(-discriminant) / (2 * coefficientA);
            System.out.println("Root 1 = " + realPart + " + " + imaginaryPart + "i");
            System.out.println("Root 2 = " + realPart + " - " + imaginaryPart + "i");
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the value of coefficient a:");
        double a = scanner.nextDouble();
        System.out.println("Enter the value of coefficient b:");
        double b = scanner.nextDouble();
        System.out.println("Enter the value of coefficient c:");
        double c = scanner.nextDouble();
        solveEquation(a, b, c);
    }
}
```



```
b1853daf/bin QuadraticEquationSolver
Enter the value of coefficient a:
5
Enter the value of coefficient b:
3
Enter the value of coefficient c:
8
The roots are complex.
Root 1 = -0.3 + 1.2288205727444508i
Root 2 = -0.3 - 1.2288205727444508i
```

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 Race {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
double[] speeds = new double[5];
double totalSpeed = 0;
System.out.println("Enter the speed of 5 racers:");
for (int i = 0; i < 5; i++) {
speeds[i] = scanner.nextDouble();
totalSpeed += speeds[i];
}
double averageSpeed = totalSpeed / 5;
System.out.println("Average speed: " + averageSpeed);
System.out.println("Qualifying racers:");
for (int i = 0; i < 5; i++) {
if (speeds[i] > averageSpeed) {
System.out.println("Racer " + (i + 1) + ": " + speeds[i]);
}
}
}
}
```



```
b1853daf/bin Race
Enter the speed of 5 racers:
122
100
145
102
98
Average speed: 113.4
Qualifying racers:
Racer 1: 122.0
Racer 3: 145.0
```

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

Introduction

In the realm of Java programming, the public static void main method serves as the cornerstone of program execution. It's the initial point where the JVM (Java Virtual Machine) begins to execute a Java application. This case study delves into the significance of this method, its structure, and its role in the overall execution of Java programs.

Understanding public static void main

public: This keyword makes the method accessible from outside the class.

static: This modifier allows the method to be called without creating an instance of the class.

void: This indicates that the method does not return any value.

main: This is the specific name that the JVM recognizes as the entry point.

String[] args: This parameter represents an array of strings that can be passed as command-line arguments to the program.

Why is it necessary?

The main method acts as a bridge between the JVM and the application. It's the starting point where the JVM begins its execution process. Without this method, the JVM wouldn't know where to start.

Best Practices:

Clear and concise: Keep the main method focused on initializing the program and delegating tasks to other methods.

Command-line arguments: If your application needs to accept command-line arguments, use the args array to process them.

Error handling: Implement appropriate error handling mechanisms to gracefully handle unexpected situations.

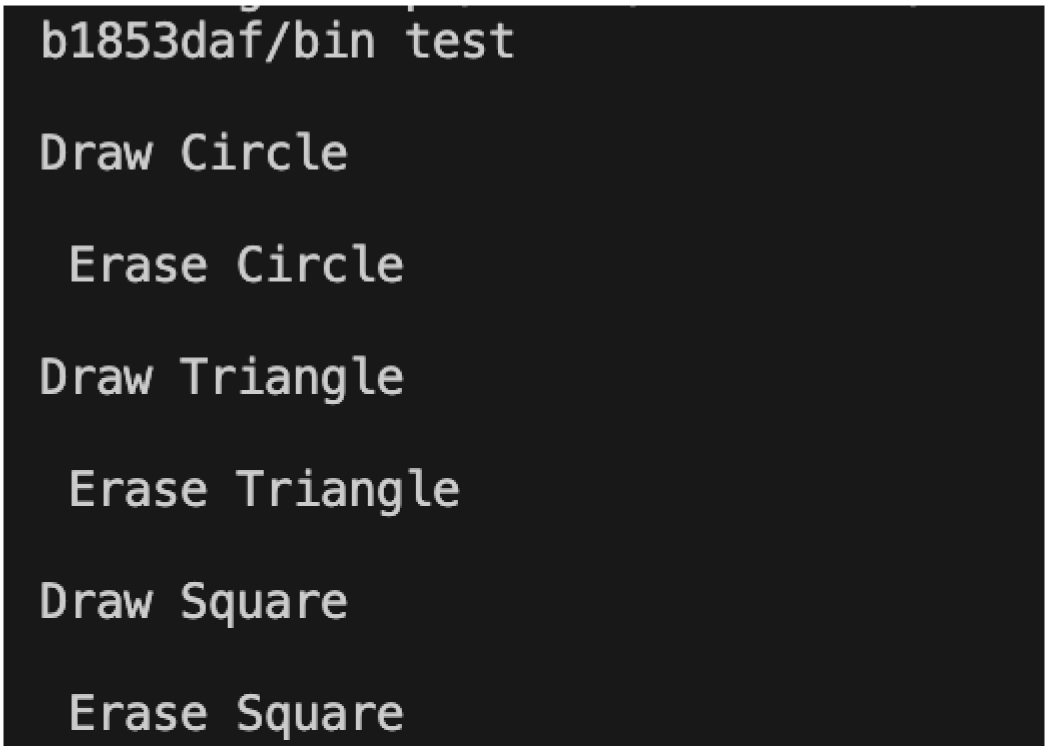
Maintainability: Write clean, well-structured code within the main method for easier maintenance.

Conclusion

The public static void main method plays a vital role in Java programming, serving as the entry point for program execution. By understanding its structure and purpose, developers can effectively create and manage Java applications.

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.

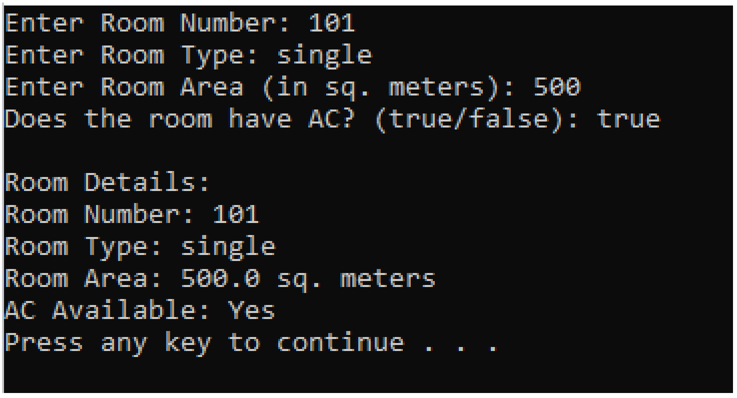
```
class Shape
{
  void draw()
  {
   System.out.println("\nDraw any shape");

  }

  void erase()
  {
  System.out.println("\nErase any shape ");
  }
}

class Circle extends Shape
{
 void draw()
 {
 System.out.println("\nDraw Circle");
 }
 void erase()
 {
 System.out.println("\n Erase Circle ");
 }

}

class Triangle extends Shape
{
 void draw()
 {
 System.out.println("\nDraw Triangle");
 }
 void erase()
 {
 System.out.println("\n Erase Triangle ");
 }

}
class Square extends Shape
{
 void draw()
 {
 System.out.println("\nDraw Square");
 }
 void erase()
 {
    System.out.println("\n Erase Square ");
}

}

public class test
{
public static void main(String args[])
{
Shape a=new Circle();
Shape b=new Triangle();
Shape c=new Square();

a.draw();
a.erase();

b.draw();
b.erase();

c.draw();
c.erase();
}
}
```



```
b1853daf/bin test

Draw Circle

 Erase Circle

Draw Triangle

 Erase Triangle

Draw Square

 Erase Square
```

b). Write a program to create a room class, the attributes of this class are room no, room type, room area and AC machine. In this class the member functions are setdata and displaydata.

```
import java.util.Scanner;

class Room {
    private int roomNo;
    private String roomType;
    private double roomArea;
    private boolean hasAC;

    public void setData(int roomNo, String roomType, double roomArea, boolean hasAC) {
        this.roomNo = roomNo;
        this.roomType = roomType;
        this.roomArea = roomArea;
        this.hasAC = hasAC;
    }
     public void displayData() {
        System.out.println("Room Number: " + roomNo);
        System.out.println("Room Type: " + roomType);
        System.out.println("Room Area: " + roomArea + " sq. meters");
        System.out.println("AC Available: " + (hasAC ? "Yes" : "No"));
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Room Number: ");
        int roomNo = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter Room Type: ");
        String roomType = scanner.nextLine();
        System.out.print("Enter Room Area (in sq. meters): ");
        double roomArea = scanner.nextDouble();
        System.out.print("Does the room have AC? (true/false): ");
        boolean hasAC = scanner.nextBoolean();
        Room room = new Room();
        room.setData(roomNo, roomType, roomArea, hasAC);
        System.out.println("\nRoom Details:");
        room.displayData();
    }
}
```

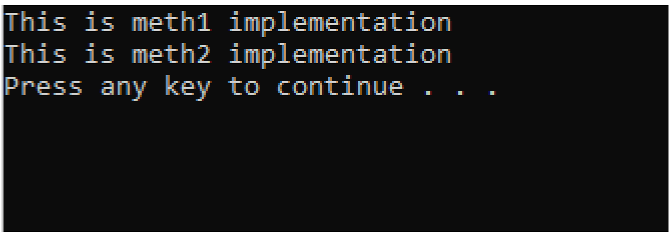


```
Enter Room Number: 101
Enter Room Type: single
Enter Room Area (in sq. meters): 500
Does the room have AC? (true/false): true

Room Details:
Room Number: 101
Room Type: single
Room Area: 500.0 sq. meters
AC Available: Yes
Press any key to continue . . .
```

3)

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

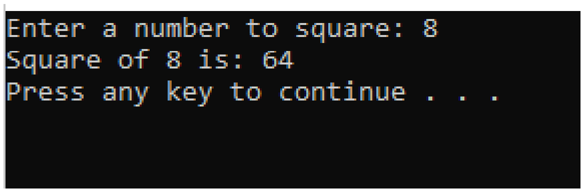
```
interface A {
   void meth1();
   void meth2();
}

class MyClass implements A {

   public void meth1() {
      System.out.println("This is meth1 implementation");
   }


   public void meth2() {
      System.out.println("This is meth2 implementation");
   }

   public static void main(String[] args) {
      MyClass obj = new MyClass();
      obj.meth1();
      obj.meth2();
   }
}
```



```
This is meth1 implementation
This is meth2 implementation
Press any key to continue . . .
```

b). Write a program to create an 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 using the object of the arithmetic class.

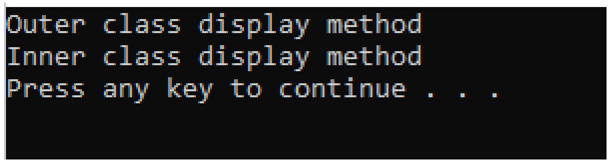
```
import java.util.Scanner;
interface Test {
    int square(int num);
}
class Arithmetic implements Test {
    public int square(int num) {
        return num * num;
    }
}
public class ToTestInt {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Arithmetic obj = new Arithmetic();
        System.out.print("Enter a number to square: ");
        int number = scanner.nextInt();
        int result = obj.square(number);
        System.out.println("Square of " + number + " is: " + result);
        scanner.close();
    }
}
```



```
Enter a number to square: 8
Square of 8 is: 64
Press any key to continue . . .
```

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.

```
class Outer {
    void display() {
        System.out.println("Outer class display method");
    }
    class Inner {
        void display() {
            System.out.println("Inner class display method");
        }
    }
}
public class innerouter {
    public static void main(String[] args) {
        Outer outer = new Outer();
        outer.display();
        Outer.Inner inner = outer.new Inner();
        inner.display();
    }
}
```



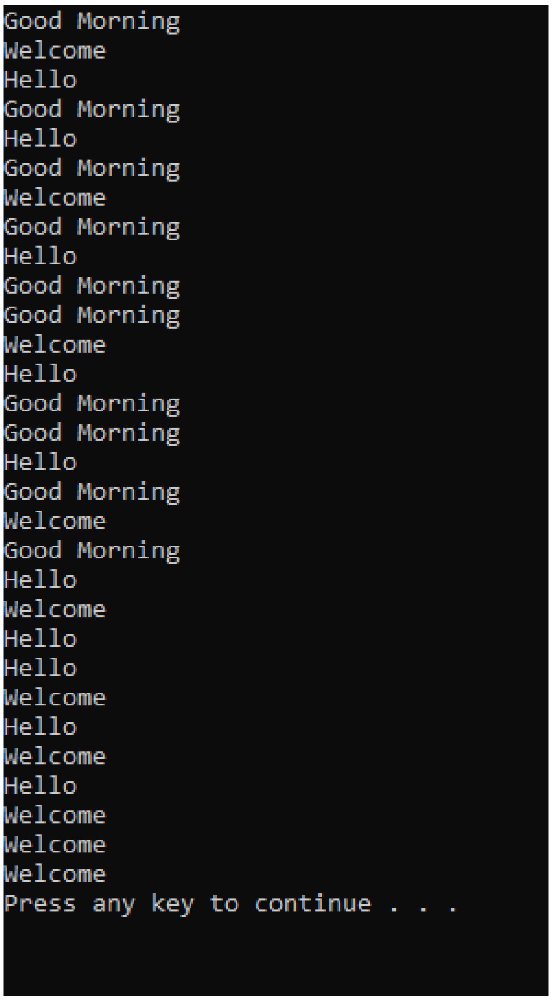
```
Outer class display method
Inner class display method
Press any key to continue . . .
```

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 Morning extends Thread {
   public void run() {
      try {
         for (int i = 0; i < 10; i++) {
            System.out.println("Good Morning");
            Thread.sleep(1000);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread interrupted");
      }
   }
}

class Hello extends Thread {
   public void run() {
      try {
         for (int i = 0; i < 10; i++) {
            System.out.println("Hello");
            Thread.sleep(2000);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread interrupted");
      }
   }
}

class Welcome extends Thread {
   public void run() {
      try {
         for (int i = 0; i < 10; i++) {
            System.out.println("Welcome");
            Thread.sleep(3000);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread interrupted");
      }
   }
}

public class Thread2 {
   public static void main(String[] args) {
      Morning m1 = new Morning();
      Hello h1 = new Hello();
      Welcome w1 = new Welcome();
      m1.start();
      h1.start();
```

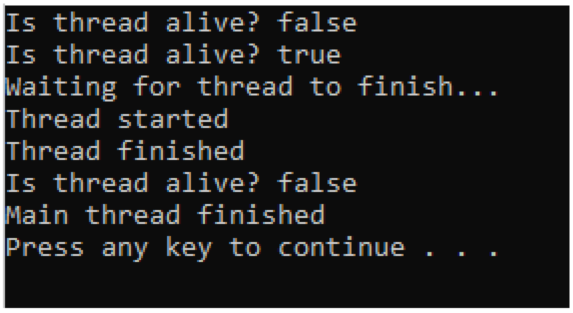
```
        w1.start();
    }
}
```



```
Good Morning
Welcome
Hello
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
Welcome
Hello
Hello
Welcome
Hello
Welcome
Hello
Welcome
Welcome
Welcome
Press any key to continue . . .
```

b) Write a program illustrating isAlive and join ()

```
class MyThread extends Thread {
   @Override
   public void run() {
      System.out.println("Thread started");
      try {
         Thread.sleep(2000); // sleep for 2 seconds
      } catch (InterruptedException e) {
         Thread.currentThread().interrupt();
      }
      System.out.println("Thread finished");
   }
}

public class ThreadDem {
   public static void main(String[] args) throws InterruptedException {
      MyThread thread = new MyThread();

      // Check if thread is alive before starting
      System.out.println("Is thread alive? " + thread.isAlive());

      // Start the thread
      thread.start();

      // Check if thread is alive after starting
      System.out.println("Is thread alive? " + thread.isAlive());

      // Wait for the thread to finish using join()
      System.out.println("Waiting for thread to finish...");
      thread.join();
       // Check if thread is alive after joining
              System.out.println("Is thread alive? " + thread.isAlive());

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



```
Is thread alive? false
Is thread alive? true
Waiting for thread to finish...
Thread started
Thread finished
Is thread alive? false
Main thread finished
Press any key to continue . . .
```

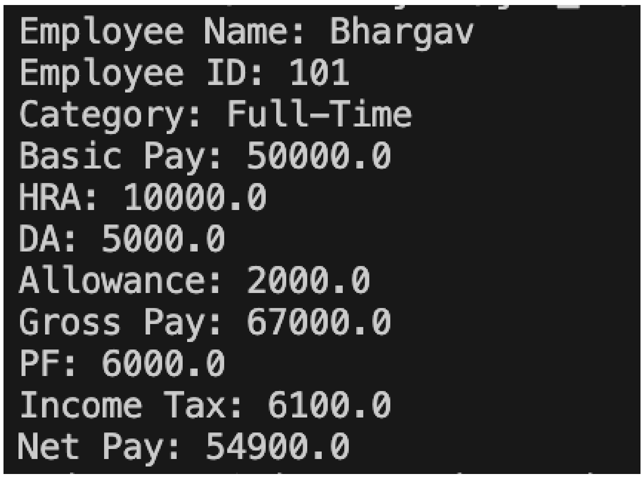
5.
a). Write a Java program to perform employee payroll processing using packages. In the java file, Emp.java creates a package employee and creates a class Emp. Declare the variables name,empid, category, bpay, hra, da, npay, pf, grosspay, incometax, and allowance. Calculate the values in methods. Create another java file Emppay.java. Create an object e to call the methods to perform and print values.

```
package employee;

public class Emp {

   private String name;

   private int empId;

   private String category;

   private double basicPay;

   private double hra;

   private double da;

   private double allowance;

   private double pf;

   private double grossPay;

   private double incomeTax;

   private double netPay;

public Emp(String name, int empId, String category, double basicPay, double hra, double da, double
allowance) {

      this.name = name;

      this.empId = empId;

      this.category = category;

      this.basicPay = basicPay;

      this.hra = hra;

      this.da = da;

      this.allowance = allowance;

      calculatePay();

   }
```

```
    private void calculatePay() {
        grossPay = basicPay + hra + da + allowance;
        pf = 0.12 * basicPay; // Assume PF is 12% of basic pay
        incomeTax = 0.1 * (grossPay - pf); // Assume income tax is 10% of (gross pay - PF)
        netPay = grossPay - pf - incomeTax;
    }
    public void displayInfo() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee ID: " + empId);
        System.out.println("Category: " + category);
        System.out.println("Basic Pay: " + basicPay);
        System.out.println("HRA: " + hra);
        System.out.println("DA: " + da);
        System.out.println("Allowance: " + allowance);
        System.out.println("Gross Pay: " + grossPay);
        System.out.println("PF: " + pf);
        System.out.println("Income Tax: " + incomeTax);
        System.out.println("Net Pay: " + netPay);
    }
}
```

Emppay.java

```
import employee.Emp; // Import the Emp class from the employee package
public class Emppay {
    public static void main(String[] args) {
        Emp e = new Emp("John Doe", 101, "Full-Time", 50000, 10000, 5000, 2000);
        e.displayInfo();
    }
}
```

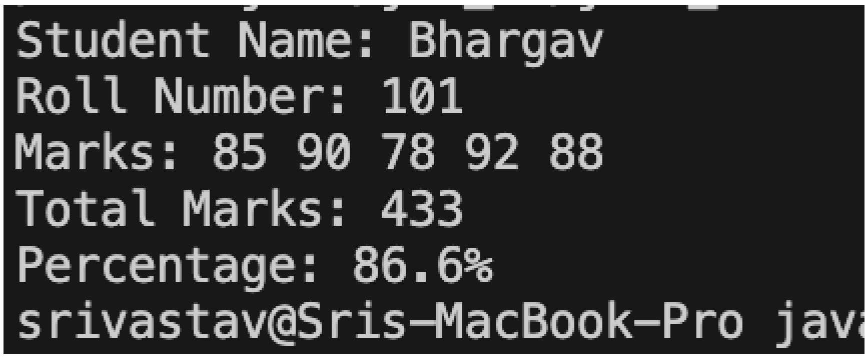


```
Employee Name: Bhargav
Employee ID: 101
Category: Full-Time
Basic Pay: 50000.0
HRA: 10000.0
DA: 5000.0
Allowance: 2000.0
Gross Pay: 67000.0
PF: 6000.0
Income Tax: 6100.0
Net Pay: 54900.0
```

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.

```
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;
        calculateTotalAndPercentage();
    }

    private void calculateTotalAndPercentage() {
        totalMarks = 0;
        for (int mark : marks) {
            totalMarks += mark;
        }
        percentage = (totalMarks / (double)(marks.length * 100)) * 100;
    }

    public void display() {
        System.out.println("Student Name: " + name);
        System.out.println("Roll Number: " + rollNumber);
        System.out.print("Marks: ");
        for (int mark : marks) {
            System.out.print(mark + " ");
        }
        System.out.println();
        System.out.println("Total Marks: " + totalMarks);
        System.out.println("Percentage: " + percentage + "%");
    }
}

import mca.Student; // Import the Student class from the mca package

public class Main {
    public static void main(String[] args) {
        int[] marks = {85, 90, 78, 92, 88};

        Student student = new Student("Bhargav", 101, marks);

        student.display();
    }
```

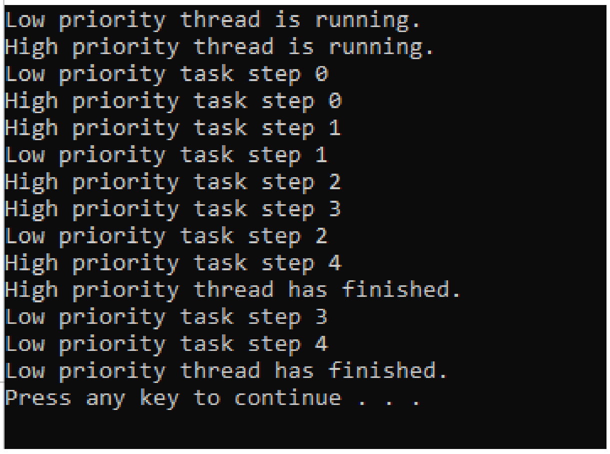
}



```
Student Name: Bhargav
Roll Number: 101
Marks: 85 90 78 92 88
Total Marks: 433
Percentage: 86.6%
srivastav@Sris-MacBook-Pro jav
```

6.
a). Write a complex program to illustrate 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.

```
public class ThreadPriority1 {
   public static void main(String[] args) {
      Thread highPriorityThread = new Thread(new HighPriorityTask());
      highPriorityThread.setPriority(Thread.MAX_PRIORITY);

      Thread lowPriorityThread = new Thread(new LowPriorityTask());
      lowPriorityThread.setPriority(Thread.MIN_PRIORITY);


      lowPriorityThread.start();
      highPriorityThread.start();
   }

   static class HighPriorityTask implements Runnable {
      @Override
      public void run() {
         try {

            System.out.println("High priority thread is running.");
            for (int i = 0; i < 5; i++) {
               System.out.println("High priority task step " + i);
               Thread.sleep(100);
            }
            System.out.println("High priority thread has finished.");
         } catch (InterruptedException e) {
            System.out.println("High priority thread was interrupted.");
         }
      }
   }
   static class LowPriorityTask implements Runnable {
         @Override
         public void run() {
            try {

               System.out.println("Low priority thread is running.");
               for (int i = 0; i < 5; i++) {
                  System.out.println("Low priority task step " + i);
                  Thread.sleep(200);
               }
               System.out.println("Low priority thread has finished.");
            } catch (InterruptedException e) {
               System.out.println("Low priority thread was interrupted.");
            }
         }
      }
}
```

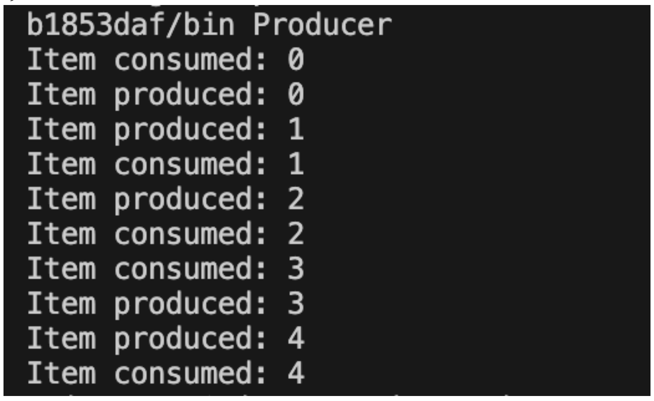


```
Low priority thread is running.
High priority thread is running.
Low priority task step 0
High priority task step 0
High priority task step 1
Low priority task step 1
High priority task step 2
High priority task step 3
Low priority task step 2
High priority task step 4
High priority thread has finished.
Low priority task step 3
Low priority task step 4
Low priority thread has finished.
Press any key to continue . . .
```

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

```
class Storage {
   private int item;
   private boolean hasItem = false;

   public synchronized void addItem(int item) {
      while (hasItem) {
         try {
            wait();
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
      this.item = item;
      hasItem = true;
      notifyAll();
   }

   public synchronized int retrieveItem() {
      while (!hasItem) {
         try {
            wait();
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
      hasItem = false;
      notifyAll();
      return item;
   }
}

 class ItemProducer extends Thread {
   private Storage storage;

   public ItemProducer(Storage storage) {
      this.storage = storage;
   }

   public void run() {
      for (int i = 0; i < 5; i++) {
         storage.addItem(i);
         System.out.println("Item produced: " + i);
         try {
            Thread.sleep(2000);
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
```

```
  }
}

class ItemConsumer extends Thread {
  private Storage storage;

  public ItemConsumer(Storage storage) {
    this.storage = storage;
  }

  public void run() {
    for (int i = 0; i < 5; i++) {
      int item = storage.retrieveItem();
      System.out.println("Item consumed: " + item);
      try {
        Thread.sleep(1000);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}

public class Producer {
  public static void main(String[] args) {
    Storage storage = new Storage();
    ItemProducer producer = new ItemProducer(storage);
    ItemConsumer consumer = new ItemConsumer(storage);



    producer.start();
    consumer.start();
  }
}
```

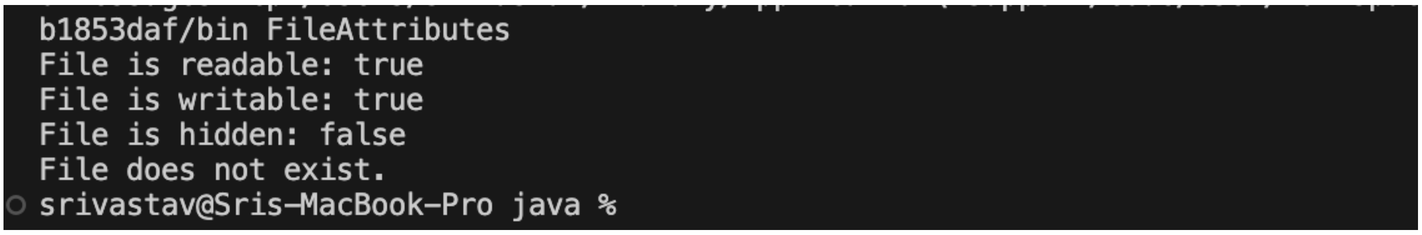


```
b1853daf/bin Producer
Item consumed: 0
Item produced: 0
Item produced: 1
Item consumed: 1
Item produced: 2
Item consumed: 2
Item consumed: 3
Item produced: 3
Item produced: 4
Item consumed: 4
```

7.

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

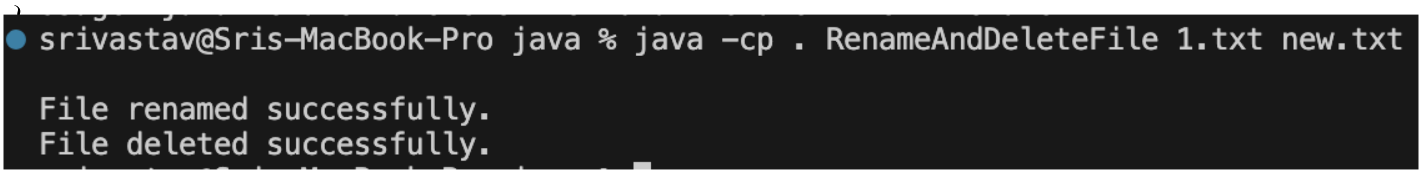
```
import java.io.File;
public class FileAttributes {
public static void main(String[] args) {
String Path = "/Users/srivastav/Downloads/chapter5-ML.pptx"; File file = new File(Path);
if (file.exists()) {
        boolean isReadable = file.canRead(); System.out.println("File is readable: " + isReadable);
        boolean isWritable = file.canWrite(); System.out.println("File is writable: " + isWritable);
        boolean isHidden = file.isHidden();
        System.out.println("File is hidden: " + isHidden);
}
else {
}
System.out.println("File does not exist.");
}
}
```



```
b1853daf/bin FileAttributes
File is readable: true
File is writable: true
File is hidden: false
File does not exist.
srivastav@Sris-MacBook-Pro java %
```

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.)

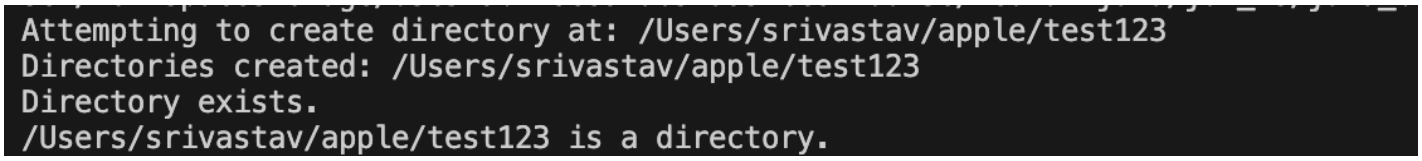
```java
import java.io.File;

public class RenameAndDeleteFile {
    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);

        if (oldFile.renameTo(newFile)) {
            System.out.println("File renamed successfully.");
            if (newFile.delete()) {
                System.out.println("File deleted successfully.");
            } else {
                System.out.println("Failed to delete the file.");
            }
        } else {
            System.out.println("Failed to rename the file.");
        }
    }
```



```
srivastav@Sris-MacBook-Pro java % java -cp . RenameAndDeleteFile 1.txt new.txt

File renamed successfully.
File deleted successfully.
```

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

```
import java.io.File;

public class DirectoryCheck {
    public static void main(String[] args) {
        String directoryPath = "/Users/srivastav/apple/test123"; // Ensure this path is correct

        File directory = new File(directoryPath);
        System.out.println("Attempting to create directory at: " + directoryPath);
        if (directory.mkdirs()) {
            System.out.println("Directories created: " + directory.getAbsolutePath());
        } else {
            System.out.println("Directory already exists or could not be created: " + directory.getAbsolutePath());
        }
        if (directory.exists()) {
            System.out.println("Directory exists.");

            if (directory.isDirectory()) {
                System.out.println(directory.getAbsolutePath() + " is a directory.");
            } else {
                System.out.println(directory.getAbsolutePath() + " is not a directory.");
            }
        } else {
            System.out.println("Directory does not exist.");
        }
    }
}
```

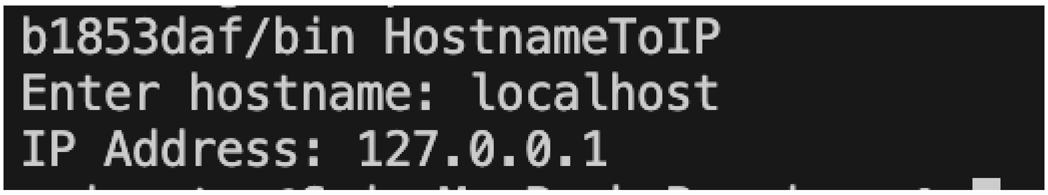


```
Attempting to create directory at: /Users/srivastav/apple/test123
Directories created: /Users/srivastav/apple/test123
Directory exists.
/Users/srivastav/apple/test123 is a directory.
```

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 HostnameToIP {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter hostname: ");
        String hostname = scanner.nextLine();

        try {
            InetAddress inetAddress = InetAddress.getByName(hostname);
            String ipAddress = inetAddress.getHostAddress();
            System.out.println("IP Address: " + ipAddress);
        } catch (UnknownHostException e) {
            System.out.println("Unable to resolve hostname: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```



```
b1853daf/bin HostnameToIP
Enter hostname: localhost
IP Address: 127.0.0.1
```

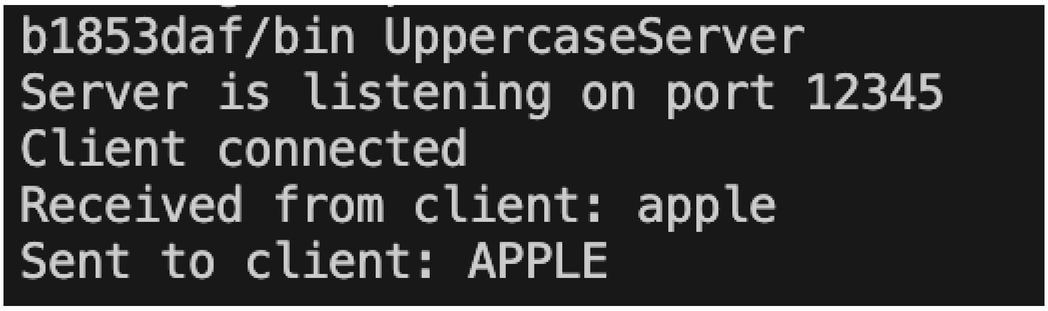
b). Write a Socket base java server program that responds to client messages as follows: When it receives a message from client, it simply converts the message into all uppercase letters and sends back to the client. Write both client and server programs demonstrating this.

UppercaseServer.java

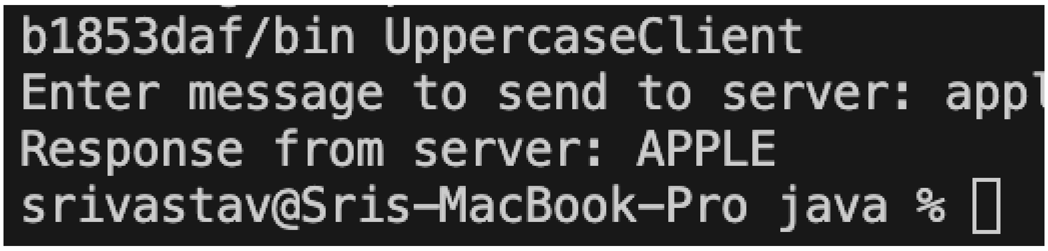
```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class UppercaseServer {
    public static void main(String[] args) {
        int port = 12345; // Port number to listen on

        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Server is listening on port " + port);

            while (true) {
                // Accept a client connection
                Socket socket = serverSocket.accept();
                System.out.println("Client connected");

                // Create input and output streams
                BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

                // Read the message from the client
                String message = input.readLine();
                System.out.println("Received from client: " + message);

                // Convert the message to uppercase
                String response = message.toUpperCase();

                // Send the response back to the client
                output.println(response);
                System.out.println("Sent to client: " + response);

                // Close the connection
                socket.close();
            }
        } catch (IOException e) {
            System.out.println("Server error: " + e.getMessage());
        }
    }
}
```

UppercaseClient.java

```
import java.io.*;
import java.net.Socket;
import java.util.Scanner;

public class UppercaseClient {
    public static void main(String[] args) {
        String hostname = "localhost"; // Server hostname
        int port = 12345; // Port number to connect to

        try (Socket socket = new Socket(hostname, port);
             PrintWriter output = new PrintWriter(socket.getOutputStream(), true);
             BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
             Scanner scanner = new Scanner(System.in)) {

            // Create a Scanner for user input
            System.out.print("Enter message to send to server: ");
            String message = scanner.nextLine();

            // Send the message to the server
            output.println(message);

            // Read and display the response from the server
            String response = input.readLine();
            System.out.println("Response from server: " + response);
        } catch (IOException e) {
            System.out.println("Client error: " + e.getMessage());
        }
    }
}
```



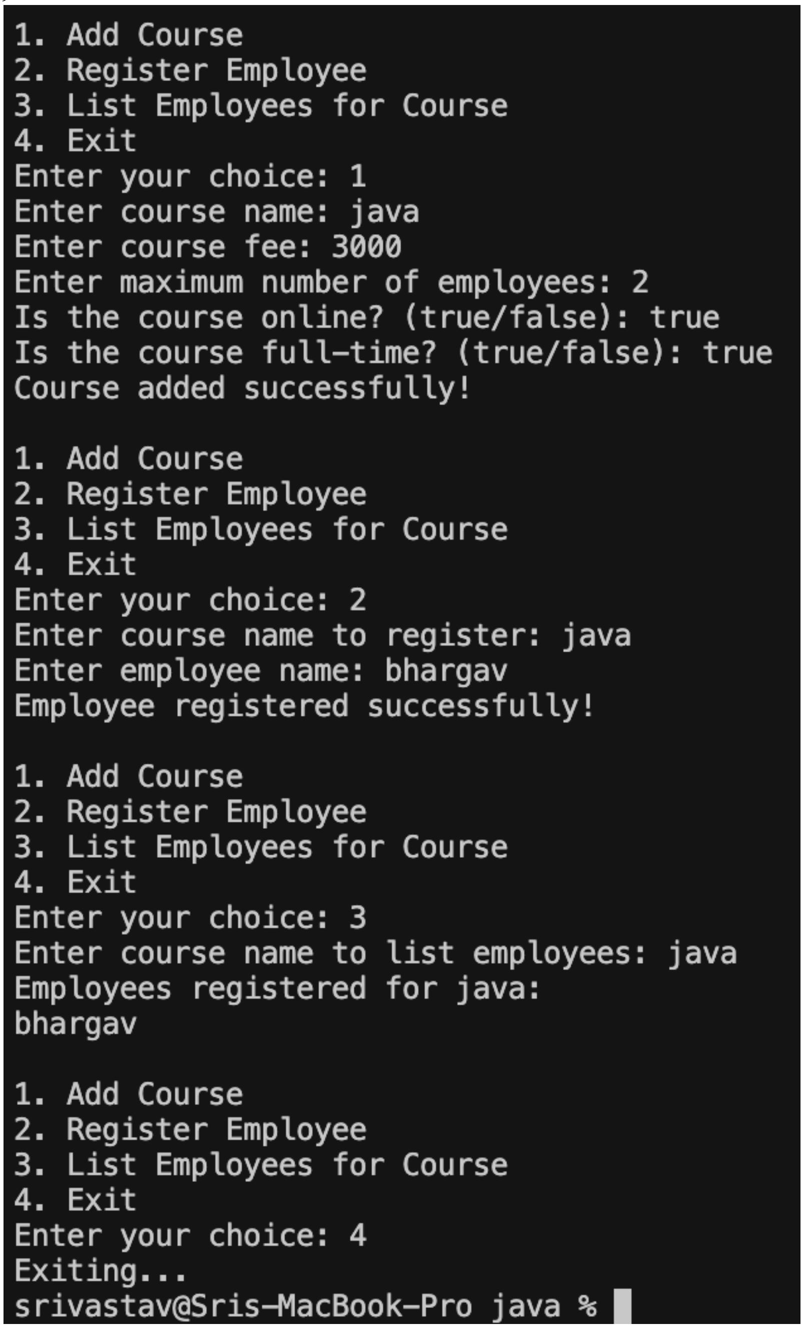
```
b1853daf/bin UppercaseServer
Server is listening on port 12345
Client connected
Received from client: apple
Sent to client: APPLE
```



```
b1853daf/bin UppercaseClient
Enter message to send to server: appl
Response from server: APPLE
srivastav@Sris-MacBook-Pro java % 
```

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 a 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.

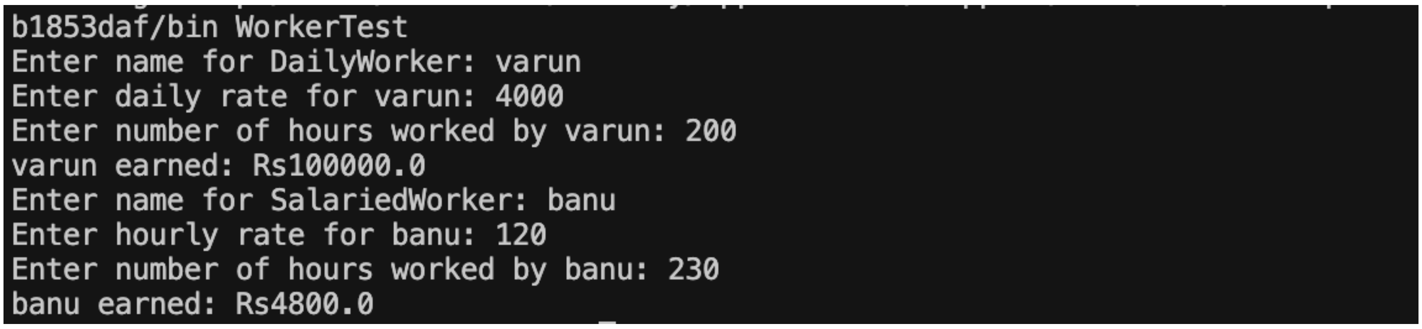
```java
import java.util.Scanner;

abstract class Course {
    private String name;
    private double fee;
    private Employee[] employeesRegistered;
    private int numEmployees;
    private boolean isOnline;
    private boolean isFullTime;

    public Course(String name, double fee, int maxEmployees, boolean isOnline, boolean isFullTime) {
        this.name = name;
        this.fee = fee;
        this.employeesRegistered = new Employee[maxEmployees];
        this.numEmployees = 0;
        this.isOnline = isOnline;
        this.isFullTime = isFullTime;
    }

    public String getName() {
        return name;
    }

    public double getFee() {
        return fee;
    }

    public boolean isOnline() {
        return isOnline;
    }

    public boolean isFullTime() {
        return isFullTime;
    }

    public void registerEmployee(Employee employee) {
        if (numEmployees < employeesRegistered.length) {
            employeesRegistered[numEmployees++] = employee;
        } else {
            System.out.println("Cannot register more employees. Course is full.");
        }
    }

    public Employee[] getEmployeesRegistered() {
        Employee[] registeredEmployees = new Employee[numEmployees];
        System.arraycopy(employeesRegistered, 0, registeredEmployees, 0, numEmployees);
        return registeredEmployees;
    }

    public String[] listEmployees() {
        String[] employeeNames = new String[numEmployees];
        for (int i = 0; i < numEmployees; i++) {
            employeeNames[i] = employeesRegistered[i].getName();
        }
        return employeeNames;
    }

    @Override
    public String toString() {
        return "Course{name='" + name + "', fee=" + fee + ", online=" + isOnline + ", fullTime=" +
isFullTime + "}";
    }
}

class ConcreteCourse extends Course {
    public ConcreteCourse(String name, double fee, int maxEmployees, boolean isOnline, boolean
isFullTime) {
        super(name, fee, maxEmployees, isOnline, isFullTime);
    }
}

class Employee {
    private String name;

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

    public String getName() {
        return name;
    }
}

class CourseCoordinator {
    private Course[] courses;
    private int numCourses;

    public CourseCoordinator(int maxCourses) {
        this.courses = new Course[maxCourses];
        this.numCourses = 0;
    }

    public void addCourse(Course course) {
        if (numCourses < courses.length) {
            courses[numCourses++] = course;
        } else {
            System.out.println("Cannot add more courses. Coordinator is full.");
        }
    }

    public void registerEmployee(String courseName, Employee employee) {
        Course course = findCourseByName(courseName);
        if (course != null) {
            course.registerEmployee(employee);
        } else {
            System.out.println("Course " + courseName + " not found.");
        }
    }

    public String[] listEmployeesForCourse(String courseName) {
        Course course = findCourseByName(courseName);
        if (course != null) {
            return course.listEmployees();
        } else {
            System.out.println("Course " + courseName + " not found.");
            return new String[0];
        }
    }

    private Course findCourseByName(String courseName) {
        for (int i = 0; i < numCourses; i++) {
            if (courses[i].getName().equalsIgnoreCase(courseName)) {
                return courses[i];
            }
        }
        return null;
    }
}

public class coursemanageMain {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        CourseCoordinator coordinator = new CourseCoordinator(10);

        while (true) {
            try {
                System.out.println("\n1. Add Course");
                System.out.println("2. Register Employee");
                System.out.println("3. List Employees for Course");
                System.out.println("4. Exit");
                System.out.print("Enter your choice: ");

                int choice = Integer.parseInt(scanner.nextLine().trim());

                switch (choice) {
                    case 1:
                        System.out.print("Enter course name: ");
                        String courseName = scanner.nextLine();
                        System.out.print("Enter course fee: ");
                        double fee = Double.parseDouble(scanner.nextLine().trim());
                        System.out.print("Enter maximum number of employees: ");
                        int maxEmployees = Integer.parseInt(scanner.nextLine().trim());
                        System.out.print("Is the course online? (true/false): ");
                        boolean isOnline = Boolean.parseBoolean(scanner.nextLine().trim());
                        System.out.print("Is the course full-time? (true/false): ");
                        boolean isFullTime = Boolean.parseBoolean(scanner.nextLine().trim());

                        Course course = new ConcreteCourse(courseName, fee, maxEmployees, isOnline,
isFullTime);
                        coordinator.addCourse(course);
                        System.out.println("Course added successfully!");
                        break;

                    case 2:
                        System.out.print("Enter course name to register: ");
                        String regCourseName = scanner.nextLine();
                        System.out.print("Enter employee name: ");
                        String employeeName = scanner.nextLine();

                        Employee employee = new Employee(employeeName);
                        coordinator.registerEmployee(regCourseName, employee);
                        System.out.println("Employee registered successfully!");
                        break;

                    case 3:
                        System.out.print("Enter course name to list employees: ");
                        String listCourseName = scanner.nextLine();

                        String[] employees = coordinator.listEmployeesForCourse(listCourseName);
                        System.out.println("Employees registered for " + listCourseName + ":");
                        if (employees.length == 0) {
                            System.out.println("No employees registered.");
                        } else {
                            for (String emp : employees) {
                                System.out.println(emp);
                            }
                        }
                        break;

                    case 4:
                        System.out.println("Exiting...");
                        scanner.close();
                        return;

                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter numeric values where required.");
            } catch (Exception e) {
                System.out.println("An error occurred: " + e.getMessage());
            }
        }
    }
}
```



```
1. Add Course
2. Register Employee
3. List Employees for Course
4. Exit
Enter your choice: 1
Enter course name: java
Enter course fee: 3000
Enter maximum number of employees: 2
Is the course online? (true/false): true
Is the course full-time? (true/false): true
Course added successfully!

1. Add Course
2. Register Employee
3. List Employees for Course
4. Exit
Enter your choice: 2
Enter course name to register: java
Enter employee name: bhargav
Employee registered successfully!

1. Add Course
2. Register Employee
3. List Employees for Course
4. Exit
Enter your choice: 3
Enter course name to list employees: java
Employees registered for java:
bhargav

1. Add Course
2. Register Employee
3. List Employees for Course
4. Exit
Enter your choice: 4
Exiting...
srivastav@Sris-MacBook-Pro java %
```

9.
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 compute the 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;

abstract class Worker {
    protected String name;
    protected double salaryRate;

    public Worker(String name, double salaryRate) {
        this.name = name;
        this.salaryRate = salaryRate;
    }

    public abstract double pay(int hours);

    public String getName() {
        return name;
    }

    public double getSalaryRate() {
        return salaryRate;
    }
}

class DailyWorker extends Worker {
    public DailyWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }

    @Override
    public double pay(int hours) {
        return (hours / 8) * salaryRate;
    }
}

class SalariedWorker extends Worker {
    public SalariedWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }

    @Override
    public double pay(int hours) {
        return 40 * salaryRate;
    }
}

public class WorkerTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter name for DailyWorker: ");
        String dailyWorkerName = scanner.nextLine();
        System.out.print("Enter daily rate for " + dailyWorkerName + ": ");
        double dailyRate = scanner.nextDouble();
        System.out.print("Enter number of hours worked by " + dailyWorkerName + ": ");
        int dailyHoursWorked = scanner.nextInt();
        scanner.nextLine();

        DailyWorker dailyWorker = new DailyWorker(dailyWorkerName, dailyRate);
        double dailyWorkerPay = dailyWorker.pay(dailyHoursWorked);
        System.out.println(dailyWorker.getName() + " earned: Rs" + dailyWorkerPay);

        System.out.print("Enter name for SalariedWorker: ");
        String salariedWorkerName = scanner.nextLine();
        System.out.print("Enter hourly rate for " + salariedWorkerName + ": ");
        double hourlyRate = scanner.nextDouble();
        System.out.print("Enter number of hours worked by " + salariedWorkerName + ": ");
        int salariedHoursWorked = scanner.nextInt();
        scanner.nextLine();
        SalariedWorker salariedWorker = new SalariedWorker(salariedWorkerName, hourlyRate);
        double salariedWorkerPay = salariedWorker.pay(salariedHoursWorked);
        System.out.println(salariedWorker.getName() + " earned: Rs" + salariedWorkerPay);

        scanner.close();
    }
}
```

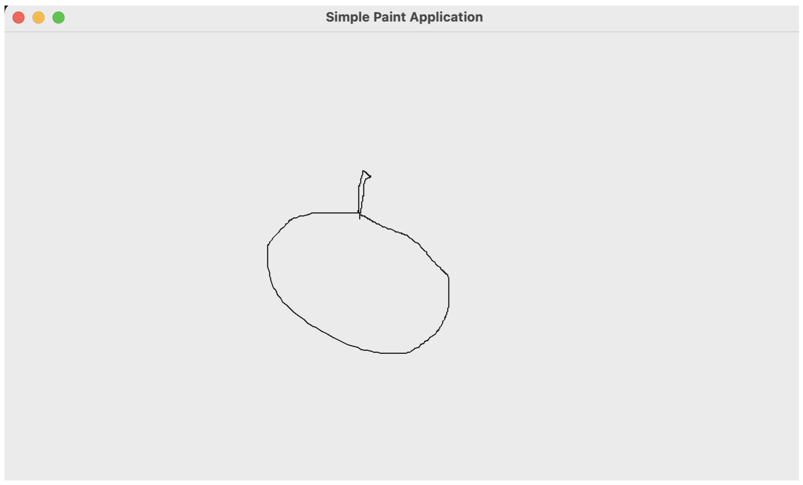


```
b1853daf/bin WorkerTest
Enter name for DailyWorker: varun
Enter daily rate for varun: 4000
Enter number of hours worked by varun: 200
varun earned: Rs100000.0
Enter name for SalariedWorker: banu
Enter hourly rate for banu: 120
Enter number of hours worked by banu: 230
banu earned: Rs4800.0
```

10.

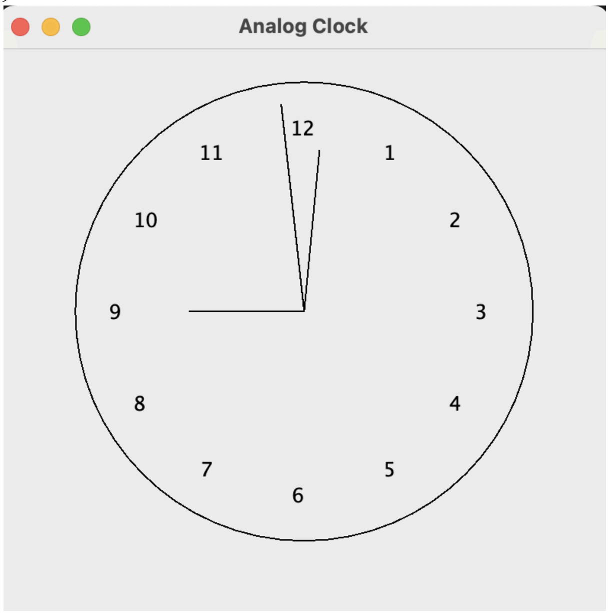
a).Write a JAVA program to paint like paint brush in applet.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.util.ArrayList;
import java.util.List;

public class SimplePaintApp extends JPanel {
    private List<Point> points = new ArrayList<>();
    private Color brushColor = Color.BLACK;

    public SimplePaintApp() {
        // Set up mouse listeners for drawing
        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                // Start a new line
                points.add(new Point(e.getX(), e.getY()));
            }
        });

        addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                // Add a point for the current drag
                points.add(new Point(e.getX(), e.getY()));
                repaint(); // Request a redraw
            }
        });
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(brushColor);

        // Draw all points stored
        for (int i = 1; i < points.size(); i++) {
            Point p1 = points.get(i - 1);
            Point p2 = points.get(i);
            g.drawLine(p1.x, p1.y, p2.x, p2.y);
        }
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Simple Paint Application");
        SimplePaintApp paintApp = new SimplePaintApp();
        frame.add(paintApp);
        frame.setSize(800, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



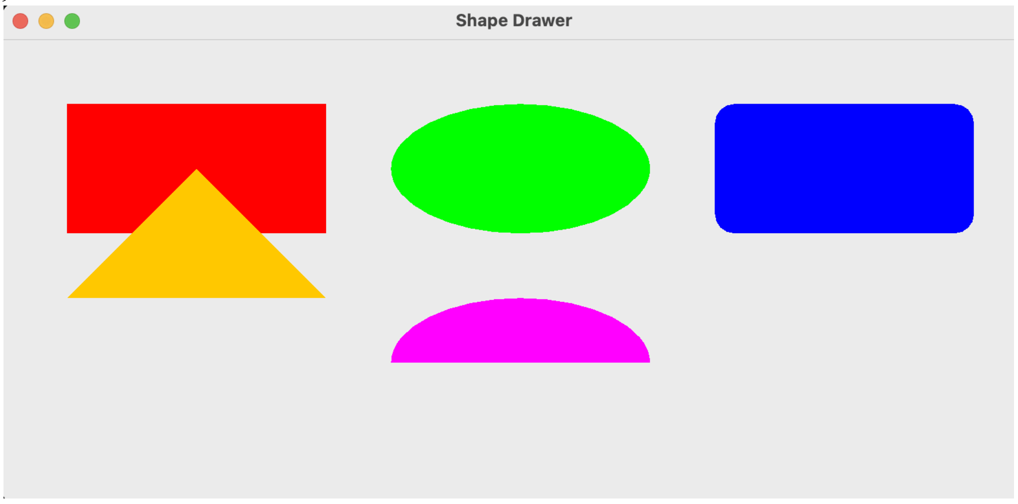
b) Write a JAVA program to display analog clock using Applet.

```
import javax.swing.*;
import java.awt.*;
import java.util.Calendar;

public class AnalogClockFrame extends JFrame implements Runnable {

    private Thread clockThread = null;

    public AnalogClockFrame() {
        setTitle("Analog Clock");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBackground(Color.white);
        setVisible(true);

        // Start the clock thread
        clockThread = new Thread(this);
        clockThread.start();
    }

    @Override
    public void run() {
        while (clockThread != null) {
            repaint(); // Request a redraw
            try {
                Thread.sleep(1000); // Pause for 1 second
            } catch (InterruptedException e) {
                // Handle the exception
                Thread.currentThread().interrupt(); // Restore interrupt status
            }
        }
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g); // Ensure the background is cleared

        Calendar now = Calendar.getInstance();
        int hours = now.get(Calendar.HOUR_OF_DAY);
        int minutes = now.get(Calendar.MINUTE);
        int seconds = now.get(Calendar.SECOND);

        // Convert 24-hour time to 12-hour format for display
        if (hours > 12) {
            hours -= 12;
        }

        int centerX = getWidth() / 2;
        int centerY = getHeight() / 2;
        int radius = Math.min(centerX, centerY) - 50;
        // Draw the circle for the clock
        g.setColor(Color.black);
        g.drawOval(centerX - radius, centerY - radius, 2 * radius, 2 * radius);

        // Draw the numbers on the clock
        for (int i = 1; i <= 12; i++) {
            int x = (int) (centerX + radius * 0.8 * Math.sin(i * Math.PI / 6));
            int y = (int) (centerY - radius * 0.8 * Math.cos(i * Math.PI / 6));
            g.drawString(String.valueOf(i), x - 8, y + 5); // Adjust position to center the text
        }
        double secondAngle = seconds * Math.PI / 30;
        double minuteAngle = (minutes + seconds / 60.0) * Math.PI / 30;
        double hourAngle = (hours + minutes / 60.0) * Math.PI / 6;
        drawHand(g, centerX, centerY, hourAngle, radius * 0.5, 8);
        drawHand(g, centerX, centerY, minuteAngle, radius * 0.7, 6);
        g.setColor(Color.red);
        drawHand(g, centerX, centerY, secondAngle, radius * 0.9, 2);
    }

    private void drawHand(Graphics g, int x, int y, double angle, double length, int width) {
        int xEnd = (int) (x + length * Math.sin(angle));
        int yEnd = (int) (y - length * Math.cos(angle));
        g.setColor(Color.black); // Set color for hands
        g.drawLine(x, y, xEnd, yEnd); // Draw hand
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new AnalogClockFrame());
    }
}
```



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

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

public class ShapeDrawer extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);
        g.fillRect(50, 50, 200, 100);
        g.setColor(Color.GREEN);
        g.fillOval(300, 50, 200, 100);
        g.setColor(Color.BLUE);
        g.fillRoundRect(550, 50, 200, 100, 30, 30);
        g.setColor(Color.ORANGE);
        int[] xPoints = {50, 150, 250};
        int[] yPoints = {200, 100, 200};
        g.fillPolygon(xPoints, yPoints, 3);
        g.setColor(Color.MAGENTA);
        g.fillArc(300, 200, 200, 100, 0, 180);  // Draw an arc from 0 to 180 degrees
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Shape Drawer");
        ShapeDrawer panel = new ShapeDrawer();
        frame.add(panel);
        frame.setSize(800, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

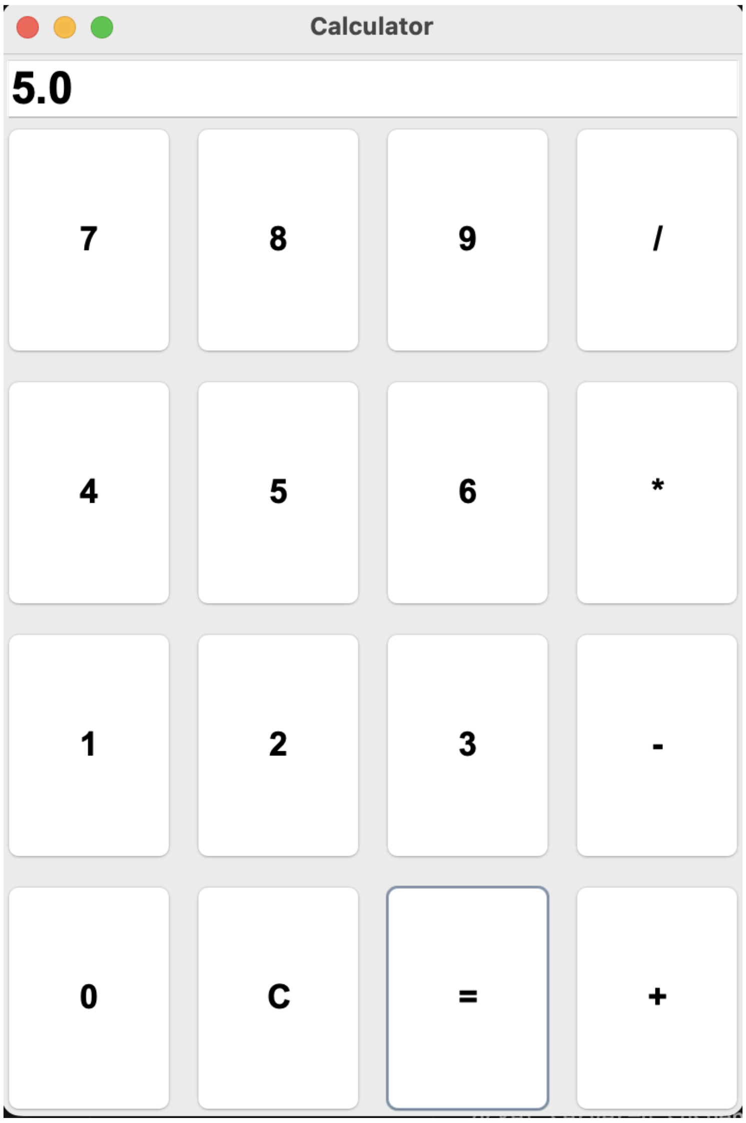


11.

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

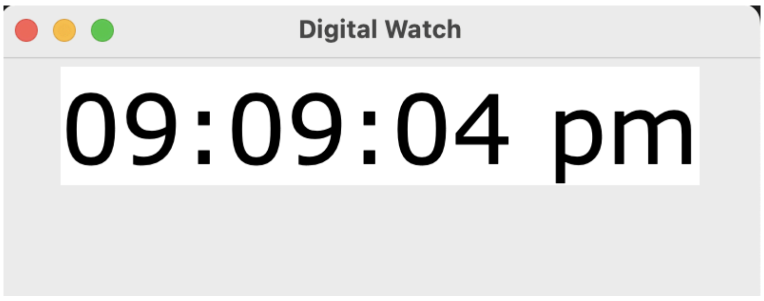
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Calculator extends JFrame implements ActionListener {
   private JTextField textField;
   private String operator = "";
   private double num1, num2, result;

   public Calculator() {
      // Create and set up the window
      setTitle("Calculator");
      setSize(400, 600);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLayout(new BorderLayout());

      // Create text field
      textField = new JTextField();
      textField.setFont(new Font("Arial", Font.BOLD, 24));
      add(textField, BorderLayout.NORTH);

      // Create panel for buttons
      JPanel panel = new JPanel();
      panel.setLayout(new GridLayout(4, 4, 10, 10));

      // Button labels
      String[] buttonLabels = {
         "7", "8", "9", "/",
         "4", "5", "6", "*",
         "1", "2", "3", "-",
         "0", "C", "=", "+"
      };

      // Create and add buttons to panel
      for (String label : buttonLabels) {
         JButton button = new JButton(label);
         button.setFont(new Font("Arial", Font.BOLD, 18));
         button.addActionListener(this);
         panel.add(button);
      }

      add(panel, BorderLayout.CENTER);
   }

   @Override
   public void actionPerformed(ActionEvent e) {
      String command = e.getActionCommand();
```

```
        switch (command) {
            case "C":
                textField.setText("");
                operator = "";
                break;
            case "=":
                num2 = Double.parseDouble(textField.getText());
                switch (operator) {
                    case "+":
                        result = num1 + num2;
                        break;
                    case "-":
                        result = num1 - num2;
                        break;
                    case "*":
                        result = num1 * num2;
                        break;
                    case "/":
                        result = num1 / num2;
                        break;
                }
                textField.setText(String.valueOf(result));
                operator = "";
                break;
            case "+":
            case "-":
            case "*":
            case "/":
                operator = command;
                num1 = Double.parseDouble(textField.getText());
                textField.setText("");
                break;
            default:
                textField.setText(textField.getText() + command);
                break;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            Calculator calculator = new Calculator();
            calculator.setVisible(true);
        });
    }
}
```


Calculator
5.0
7
8
9
/
4
5
6
*
1
2
3
-
0
C
=
+

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

```
import java.awt.*;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import javax.swing.*;


public class DigitalWatch extends JFrame {
    JLabel timeLabel;
    SimpleDateFormat timeFormat;
    String time;
    public DigitalWatch() {
        setTitle("Digital Watch");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());
        timeFormat = new SimpleDateFormat("hh:mm:ss a");
        timeLabel = new JLabel();
        timeLabel.setFont(new Font("Verdana", Font.PLAIN, 50));
        timeLabel.setForeground(Color.BLACK);
        timeLabel.setBackground(Color.WHITE);
        timeLabel.setOpaque(true);
        add(timeLabel);
        setVisible(true);
        updateTime();
    }
    public void updateTime() {
        while (true) {
            time = timeFormat.format(Calendar.getInstance().getTime());
            timeLabel.setText(time);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public static void main(String[] args) {
        new DigitalWatch();
    }
}
```



```
09:09:04 pm
```