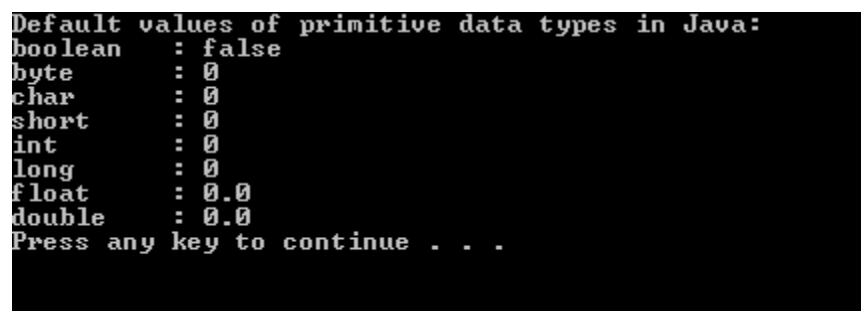


1. a). Write a JAVA program to display default value of all primitive data type of JAVA

```
public class DefaultValues {  
  
    static boolean aBoolean;  
    static byte aByte;  
    static char aChar;  
    static short aShort;  
    static int anInt;  
    static long aLong;  
    static float aFloat;  
    static double aDouble;  
  
    public static void main(String[] args) {  
        System.out.println("Default values of primitive data types in Java:");  
        System.out.println("boolean   : " + aBoolean);  
        System.out.println("byte     : " + aByte);  
        System.out.println("char     : " + (int) aChar); // Cast to int to display as number  
        System.out.println("short    : " + aShort);  
        System.out.println("int      : " + anInt);  
        System.out.println("long     : " + aLong);  
        System.out.println("float    : " + aFloat);  
        System.out.println("double   : " + aDouble);  
    }  
}
```

OUTPUT:

```
Default values of primitive data types in Java:  
boolean   : false  
byte      : 0  
char      : 0  
short     : 0  
int       : 0  
long      : 0  
float     : 0.0  
double    : 0.0  
Press any key to continue . . .
```

b). Write a java program that display the roots of a quadratic equation $ax^2+bx=0$. Calculate the discriminate D and basing on value of D, describe the nature of root.

```
import java.util.Scanner;

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

        System.out.println("Enter the coefficients of the quadratic equation  $ax^2 + bx = 0$ :");
        System.out.print("Enter a: ");
        double a = scanner.nextDouble();
        System.out.print("Enter b: ");
        double b = scanner.nextDouble();

        // Calculate discriminant
        double discriminant = b * b - 4 * a * 0;

        // Determine nature of roots based on discriminant
        if (discriminant > 0) {
            // Two distinct real roots
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("The equation has two distinct real roots:");
            System.out.println("Root 1 = " + root1);
            System.out.println("Root 2 = " + root2);
        } else if (discriminant == 0) {
            // One real root (repeated)
            double root = -b / (2 * a);
            System.out.println("The equation has one real root (repeated):");
            System.out.println("Root = " + root);
        } else {
            // No real roots (complex roots)
            System.out.println("The equation has no real roots (complex roots).");
        }

        scanner.close();
    }
}
```

Output

```
Enter the coefficients of the quadratic equation ax^2 + bx = 0:
Enter a: 4
Enter b: 3
The equation has two distinct real roots:
Root 1 = 0.0
Root 2 = -0.75
Press any key to continue . . .
```

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 QualifyingRacers {

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

        // Input speeds of all 5 racers
        System.out.println("Enter the speeds of the 5 racers:");
        double[] speeds = new double[5];
        for (int i = 0; i < 5; i++) {
            System.out.print("Racer " + (i + 1) + ": ");
            speeds[i] = scanner.nextDouble();
        }

        // Calculate average speed
        double sum = 0;
        for (double speed : speeds) {
            sum += speed;
        }
        double averageSpeed = sum / 5;

        // Display speeds of qualifying racers
        System.out.println("Qualifying racers (speed > average speed):");
        for (int i = 0; i < 5; i++) {
            if (speeds[i] > averageSpeed) {
                System.out.println("Racer " + (i + 1) + ": " + speeds[i]);
            }
        }

        scanner.close();
    }
}
```

Output

```
Enter the speeds of the 5 racers:
Racer 1: 56
Racer 2: 55
Racer 3: 78
Racer 4: 67
Racer 5: 45
Qualifying racers <speed > average speed>:
Racer 3: 78.0
Racer 4: 67.0
Press any key to continue . . .
```

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

In Java, the public static void main method is the entry point of a Java program. It is the starting point where the program begins execution. In this case study, we will explore the significance of public static void main and its role in Java programming.

What is public static void main?

public static void main is a special method in Java that is used to define the entry point of a Java program. It is declared as public to make it accessible from outside the class, static to allow it to be called without creating an instance of the class, and void to indicate that it does not return any value.

Why is public static void main necessary?

public static void main is necessary because it provides a single entry point for the Java Virtual Machine (JVM) to start executing the program. Without it, the JVM would not know where to begin executing the program.

How does public static void main work?

When a Java program is executed, the JVM searches for the public static void main method in the main class. Once found, the JVM calls this method, passing in an array of strings as arguments. The main method then executes the code inside it, which typically includes creating objects, calling methods, and performing other operations.

Best Practices

When writing a public static void main method, it is essential to follow best practices to ensure that the program is executed correctly. These include:

- Declaring the method as public and static to make it accessible from outside the class.
- Using the void return type to indicate that the method does not return any value.
- Providing a clear and concise name for the method, such as main.
- Keeping the method short and focused on initializing the program.

Conclusion

In conclusion, public static void main is a critical component of Java programming, providing a single entry point for the JVM to start executing the program. By following best practices and understanding the role of public static void main, developers can write efficient and effective Java programs.

Week 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.

```
abstract class Shape {
    abstract void draw();
    abstract void erase();
}

class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Circle: draw");
    }

    @Override
    void erase() {
        System.out.println("Circle: erase");
    }
}

class Triangle extends Shape {
    @Override
    void draw() {
        System.out.println("Triangle: draw");
    }

    @Override
    void erase() {
        System.out.println("Triangle: erase");
    }
}

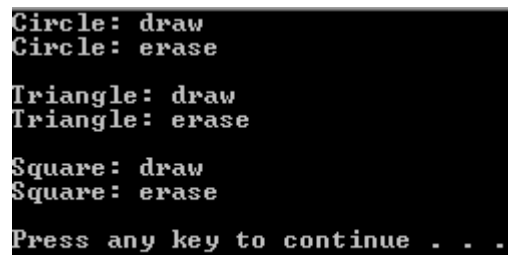
class Square extends Shape {
    @Override
    void draw() {
        System.out.println("Square: draw");
    }
}
```

```
@Override
void erase() {
    System.out.println("Square: erase");
}

public class Main {
    public static void main(String[] args) {
        Shape[] shapes = { new Circle(), new Triangle(), new Square() };

        for (Shape shape : shapes) {
            shape.draw();
            shape.erase();
            System.out.println();
        }
    }
}
```

Output

A screenshot of a terminal window showing the output of a Java program. The text is as follows:

```
Circle: draw
Circle: erase

Triangle: draw
Triangle: erase

Square: draw
Square: erase

Press any key to continue . . .
```

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.

```
import java.util.Scanner;
class Room{
    private static int roomNo;
    private static String roomType;
    private static double roomArea;
    static boolean hasAC;

    public static void setData(Scanner sc){
        System.out.print("Enter Room No:");
        roomNo=sc.nextInt();
        System.out.print("Enter Room Type:");
        roomType=sc.next();
        System.out.print("Enter Room Area(sqft):");
        roomArea=sc.nextDouble();
        System.out.print("is AC is Available? :");
        hasAC=sc.nextBoolean();
    }

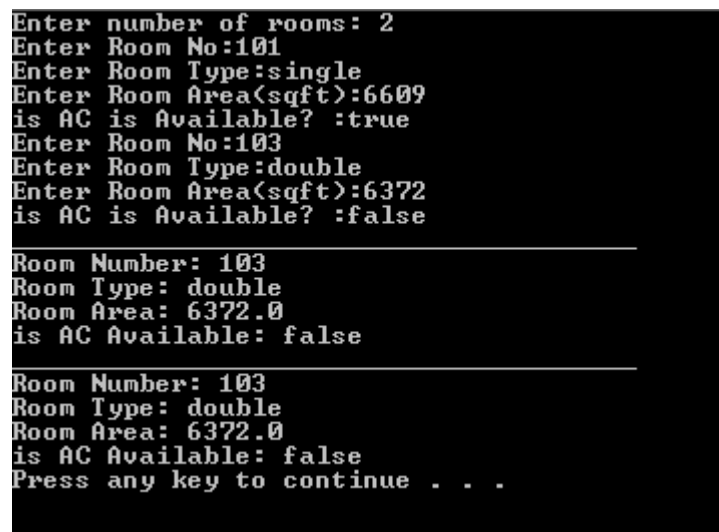
    public static void displayData(){
        System.out.println("Room Number: "+roomNo+"\nRoom Type: "+roomType+"\nRoom
Area: "
        +roomArea+"\nis AC Available: "+hasAC);
    }

    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter number of rooms: ");
        int n=sc.nextInt();

        Room[] room=new Room[n];
        for(int i=0;i<n;i++){
            room[i]=new Room();
            room[i].setData(sc);
        }
    }
}
```

```
        for(int i=0;i<n;i++){  
            System.out.println("_____");  
            room[i].displayData();  
        }  
  
        sc.close();  
    }  
}
```

Output



```
Enter number of rooms: 2  
Enter Room No:101  
Enter Room Type:single  
Enter Room Area(sqft):6609  
is AC is Available? :true  
Enter Room No:103  
Enter Room Type:double  
Enter Room Area(sqft):6372  
is AC is Available? :false  
  
Room Number: 103  
Room Type: double  
Room Area: 6372.0  
is AC Available: false  
  
Room Number: 103  
Room Type: double  
Room Area: 6372.0  
is AC Available: false  
Press any key to continue . . .
```


Week 3

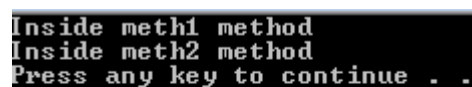
3.a). Write a program to create interface A in this interface we have two method meth1 and meth2. Implements this interface in another class named MyClass.

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

class MyClass implements A {
    @Override
    public void meth1() {
        System.out.println("Inside meth1 method");
    }

    @Override
    public void meth2() {
        System.out.println("Inside meth2 method");
    }
}

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

OutputA screenshot of a terminal window showing the output of the Java program. The text displayed is: "Inside meth1 method", "Inside meth2 method", and "Press any key to continue . .".

```
Inside meth1 method
Inside meth2 method
Press any key to continue . .
```

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.

```
import java.util.Scanner;
interface Test {
    int square(int number);
}
class Arithmetic implements Test {
    @Override
    public int square(int number) {
        return number * number;
    }
}
public class ToTestInt {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

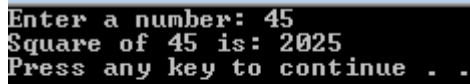
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();

        Arithmetic arithmeticObj = new Arithmetic();
        int result = arithmeticObj.square(number);

        System.out.println("Square of " + number + " is: " + result);

        scanner.close();
    }
}
```

Output



```
Enter a number: 45
Square of 45 is: 2025
Press any key to continue . .
```

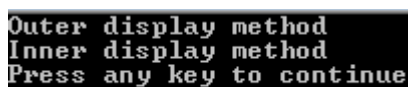
c). Create an outer class with a function display, again create another class inside the outer class named inner with a function called display and call the two functions in the main class.

```
class Outer {
    void display() {
        System.out.println("Outer display method");
    }

    class Inner {
        void display() {
            System.out.println("Inner display method");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Outer outerObj = new Outer();
        outerObj.display();
        Outer.Inner innerObj = outerObj.new Inner();
        innerObj.display();
    }
}
```

Output



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

Week 4

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

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

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

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

public class Main {
    public static void main(String[] args) {
        GoodMorningThread t1 = new GoodMorningThread();
        HelloThread t2 = new HelloThread();
        WelcomeThread t3 = new WelcomeThread();

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

Output

```
Good Morning
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
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
```

Runnable

// Using runnable interface

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

```
    }  
    }  
}  
  
class HelloRunnable implements Runnable {  
    public void run() {  
        try {  
            while (true) {  
                System.out.println("Hello");  
                Thread.sleep(2000); // Sleep for 2 seconds  
            }  
        } catch (InterruptedException e) {  
            System.out.println(e);  
        }  
    }  
}  
  
class WelcomeRunnable implements Runnable {  
    public void run() {  
        try {  
            while (true) {  
                System.out.println("Welcome");  
                Thread.sleep(3000); // Sleep for 3 seconds  
            }  
        } catch (InterruptedException e) {  
            System.out.println(e);  
        }  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Thread t1 = new Thread(new GoodMorningRunnable());  
        Thread t2 = new Thread(new HelloRunnable());  
        Thread t3 = new Thread(new WelcomeRunnable());  
  
        t1.start();  
        t2.start();  
        t3.start();  
    }  
}
```

Output

```
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Hello
Welcome
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Hello
Welcome
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
```

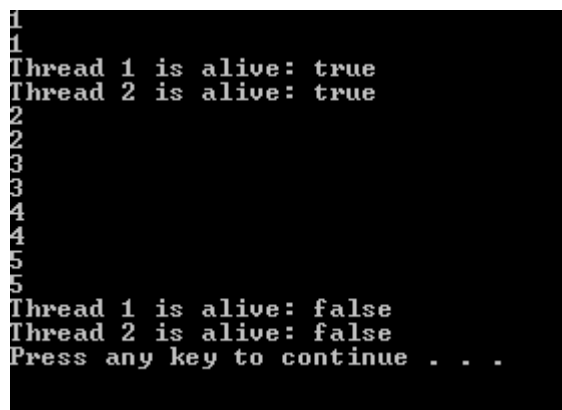
b). Write a program illustrating isAlive and join ()

```
class PrintNumbers extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(i);
        }
    }
}
```



```
        try {
            Thread.sleep(500); // Sleep for 0.5 seconds
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}
}
}
public class Main {
    public static void main(String[] args) {
        PrintNumbers t1 = new PrintNumbers();
        PrintNumbers t2 = new PrintNumbers();
        t1.start();
        t2.start();
        System.out.println("Thread 1 is alive: " + t1.isAlive());
        System.out.println("Thread 2 is alive: " + t2.isAlive());
        try {
            t1.join(); // Main thread waits for t1 to finish
            t2.join(); // Main thread waits for t2 to finish
        } catch (InterruptedException e) {
            System.out.println(e);
        }
        System.out.println("Thread 1 is alive: " + t1.isAlive());
        System.out.println("Thread 2 is alive: " + t2.isAlive());
    }
}
```

Output



The screenshot shows the output of the Java program in a black terminal window. The output consists of two lines: "Thread 1 is alive: true" and "Thread 2 is alive: true". Below these, there are several lines of numbers (1, 2, 3, 4, 5) which appear to be part of the program's execution or a sequence of characters. At the bottom, the text "Thread 1 is alive: false" and "Thread 2 is alive: false" is displayed, followed by "Press any key to continue . . .".

Week 5

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 bpay;
    private double hra;
    private double da;
    private double npay;
    private double pf;
    private double grosspay;
    private double incometax;
    private double allowance;

    public Emp(String name, int empid, String category, double bpay) {
        this.name = name;
        this.empid = empid;
        this.category = category;
        this.bpay = bpay;
        this.hra = 0;
        this.da = 0;
        this.pf = 0;
        this.grosspay = 0;
        this.incometax = 0;
        this.allowance = 0;
        this.npay = 0;
    }

    public void calculateHRA() {
        hra = 0.20 * bpay;
    }

    public void calculateDA() {
        da = 0.10 * bpay;
    }

    public void calculatePF() {
        pf = 0.12 * bpay;
    }

    public void calculateGrossPay() {
        grosspay = bpay + hra + da + allowance;
    }

    public void calculateIncomeTax() {
        incometax = 0.05 * (grosspay - pf);
    }
}
```

```
    }

    public void calculateNetPay() {
        npay = grosspay - pf - incometax;
    }

    public void setAllowance(double allowance) {
        this.allowance = allowance;
    }

    public void printDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee ID: " + empid);
        System.out.println("Category: " + category);
        System.out.println("Basic Pay: " + bpay);
        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: " + npay);
    }
}

import employee.Emp;

public class Emppay {
    public static void main(String[] args) {
        Emp e = new Emp("Kavan H M", 10001, "Manager", 800000);

        e.setAllowance(5000);

        e.calculateHRA();
        e.calculateDA();
        e.calculatePF();
        e.calculateGrossPay();
        e.calculateIncomeTax();
        e.calculateNetPay();

        e.printDetails();
    }
}
```

Output

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 != null ? marks : new int[0]; // Avoid null marks array
        calculateTotalMarks();
        calculatePercentage();
    }

    private void calculateTotalMarks() {
        totalMarks = 0;
        for (int mark : marks) {
            totalMarks += mark;
        }
    }

    private void calculatePercentage() {
        if (marks.length > 0) {
            // Assuming each mark is out of 100
            percentage = (totalMarks / (double) (marks.length * 100)) * 100;
        } else {
            percentage = 0;
        }
    }

    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 + "%");
    }

    // Getter methods
```

```
    public String getName() {
        return name;
    }

    public int getRollNumber() {
        return rollNumber;
    }

    public int[] getMarks() {
        return marks;
    }

    public int getTotalMarks() {
        return totalMarks;
    }

    public double getPercentage() {
        return percentage;
    }
}

import MCA.Student;

public class Main {
    public static void main(String[] args) {
        int[] marks = {90, 94, 68, 82, 86};

        Student student = new Student("Kavan", 123, marks);
        student.display();
    }
}
```

Output

Week 6

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 LowPriorityThread extends Thread {
    public void run() {
        System.out.println("Low-priority thread started");
        for (int i = 0; i < 5; i++) {
            System.out.println("Low-priority thread: iteration " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("Low-priority thread finished");
    }
}

class HighPriorityThread extends Thread {
    public void run() {
        System.out.println("High-priority thread started");
        for (int i = 0; i < 5; i++) {
            System.out.println("High-priority thread: iteration " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("High-priority thread finished");
    }
}

public class ThreadPriorityDemo {
    public static void main(String[] args) {
        LowPriorityThread lowThread = new LowPriorityThread();
        lowThread.setPriority(Thread.NORM_PRIORITY - 2); // set priority to 3 (lower than
        default)
        HighPriorityThread highThread = new HighPriorityThread();
        highThread.setPriority(Thread.NORM_PRIORITY + 2); // set priority to 7 (higher than
        default)
        lowThread.start();
        highThread.start();
        try {
            lowThread.join();
            highThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

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

Output



```
Low-priority thread started
High-priority thread started
High-priority thread: iteration 0
Low-priority thread: iteration 0
Low-priority thread: iteration 1
High-priority thread: iteration 1
High-priority thread: iteration 2
Low-priority thread: iteration 2
Low-priority thread: iteration 3
High-priority thread: iteration 3
High-priority thread: iteration 4
Low-priority thread: iteration 4
Low-priority thread finished
High-priority thread finished
Main thread finished
Press any key to continue . . .
```


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 data) {
        while (available) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        this.data = data;
        available = true;
        notifyAll();
    }
    public synchronized int get() {
        while (!available) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        available = false;
        notifyAll();
        return data;
    }
}

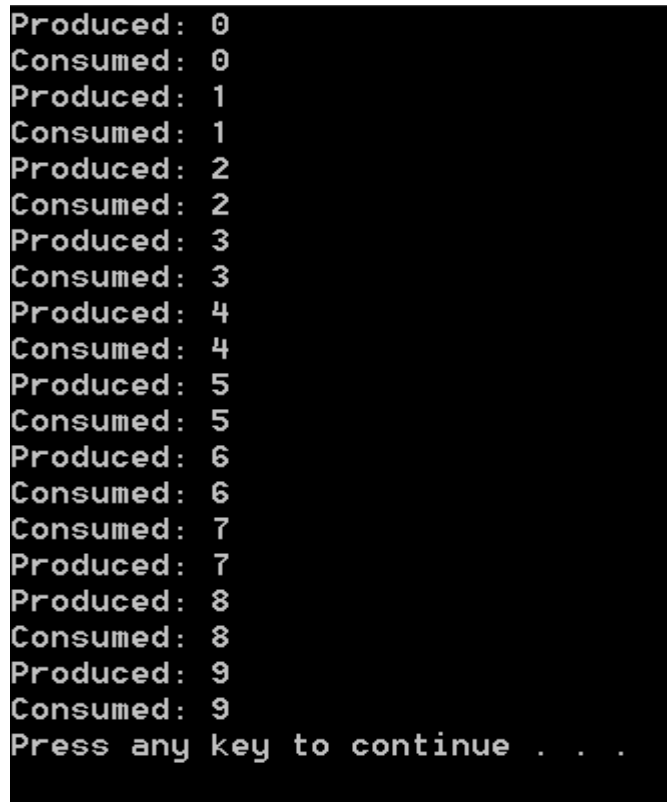
class Producer extends Thread {
    private Buffer buffer;
    public Producer(Buffer buffer) {
        this.buffer = buffer;
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            buffer.put(i);
            System.out.println("Produced: " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer extends Thread {
```

```
private Buffer buffer;
public Consumer(Buffer buffer) {
    this.buffer = buffer;
}
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) {
            e.printStackTrace();
        }
    }
}
}

public class ProducerConsumer {
    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



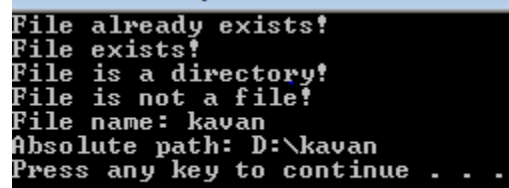
```
Produced: 0
Consumed: 0
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Consumed: 7
Produced: 7
Produced: 8
Consumed: 8
Produced: 9
Consumed: 9
Press any key to continue . . .
```

Week 7

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 command exists(), isDirectory(), isFile(), getName() and getAbsolutePath().

```
import java.io.File;
import java.io.IOException;
public class FileExample {
    public static void main(String[] args) {
        String filePath = "D:\\kavan";
        File file = new File(filePath);
        try {
            // Create the file if it doesn't exist
            if (!file.exists()) {
                file.createNewFile();
                System.out.println("File created successfully!");
            } else {
                System.out.println("File already exists!");
            }
            // Check if the file exists
            if (file.exists()) {
                System.out.println("File exists!");
            } else {
                System.out.println("File does not exist!");
            }
            // Check if the file is a directory
            if (file.isDirectory()) {
                System.out.println("File is a directory!");
            } else {
                System.out.println("File is not a directory!");
            }
            // Check if the file is a file
            if (file.isFile()) {
                System.out.println("File is a file!");
            } else {
                System.out.println("File is not a file!");
            }
            // Get the file name
            System.out.println("File name: " + file.getName());
            // Get the absolute path of the file
            System.out.println("Absolute path: " + file.getAbsolutePath());
        } catch (IOException e) {
            System.out.println("Error creating file: " + e.getMessage());
        }
    }
}
```

Output



```
File already exists!  
File exists!  
File is a directory!  
File is not a file!  
File name: kavan  
Absolute path: D:\kavan  
Press any key to continue . . .
```

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 FileRenameAndDelete {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage:  java  FileRenameAndDelete  <oldFileName>
<newFileName>");
            return;
        }

        String oldFileName = args[0];
        String newFileName = args[1];

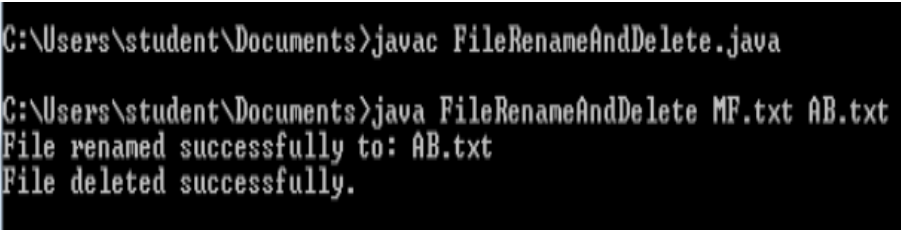
        File oldFile = new File(oldFileName);
        File newFile = new File(newFileName);

        if (!oldFile.exists()) {
            System.out.println("The file " + oldFileName + " does not exist.");
            return;
        }

        if (oldFile.renameTo(newFile)) {
            System.out.println("File renamed successfully to: " + newFileName);

            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.");
        }
    }
}
```

Output



```
C:\Users\student\Documents>javac FileRenameAndDelete.java

C:\Users\student\Documents>java FileRenameAndDelete MF.txt AB.txt
File renamed successfully to: AB.txt
File deleted successfully.
```

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

        System.out.print("Enter the path for the new directory: ");
        String directoryPath = scanner.nextLine();

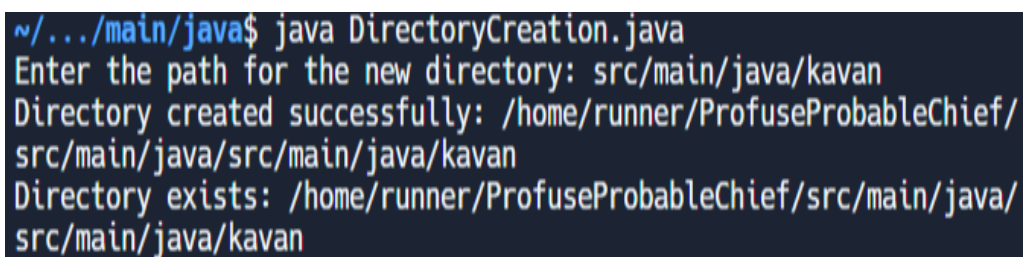
        File directory = new File(directoryPath);

        if (directory.mkdirs()) {
            System.out.println("Directory created successfully: " + directory.getAbsolutePath());
        } else {
            System.out.println("Directory already exists or failed to create.");
        }

        if (directory.exists() && directory.isDirectory()) {
            System.out.println("Directory exists: " + directory.getAbsolutePath());
        } else {
            System.out.println("Directory does not exist.");
        }

        scanner.close();
    }
}
```

Output



```
~/.../main/java$ java DirectoryCreation.java
Enter the path for the new directory: src/main/java/kavan
Directory created successfully: /home/runner/ProfuseProbableChief/
src/main/java/src/main/java/kavan
Directory exists: /home/runner/ProfuseProbableChief/src/main/java/
src/main/java/kavan
```

Week 8.

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

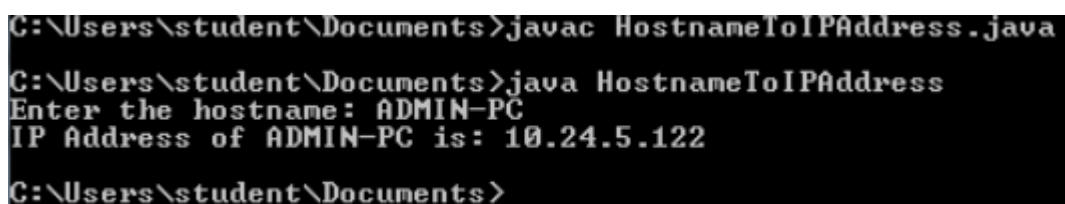
```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Scanner;

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

        System.out.print("Enter the hostname: ");
        String hostname = scanner.nextLine();

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

        scanner.close();
    }
}
```

Output

```
C:\Users\student\Documents>javac HostnameToIPAddress.java
C:\Users\student\Documents>java HostnameToIPAddress
Enter the hostname: ADMIN-PC
IP Address of ADMIN-PC is: 10.24.5.122
C:\Users\student\Documents>
```

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.

Client Server

```
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 UppercaseClient {
    public static void main(String[] args) {
        final String SERVER_ADDRESS = "localhost";
        final int SERVER_PORT = 12345;

        try (Socket socket = new Socket(SERVER_ADDRESS, SERVER_PORT)) {
            System.out.println("Connected to server");

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

            while (true) {
                System.out.print("Enter message to send to server (type 'exit' to quit): ");
                String message = scanner.nextLine();

                if ("exit".equalsIgnoreCase(message)) {
                    break;
                }
                output.println(message);
                String response = input.readLine();
                System.out.println("Response from server: " + response);
            }
        } catch (IOException e) {
            System.out.println("Error communicating with server: " + e.getMessage());
        }
    }
}
```

Server Side

```
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 UppercaseServer {
    public static void main(String[] args) {
        final int PORT = 12345;

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

            while (true) {
                try (Socket socket = serverSocket.accept()) {
                    System.out.println("Client connected");

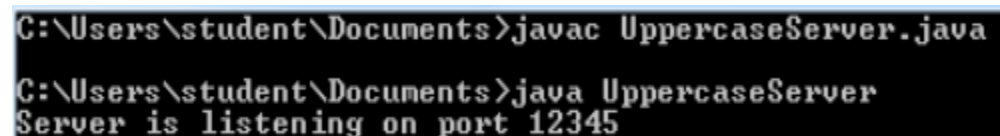
                    BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                    PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

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

                    String response = message.toUpperCase();
                    output.println(response);
                    System.out.println("Sent to client: " + response);
                } catch (IOException e) {
                    System.out.println("Error in communication with client: " + e.getMessage());
                }
            }
        } catch (IOException e) {
            System.out.println("Error starting server: " + e.getMessage());
        }
    }
}
```

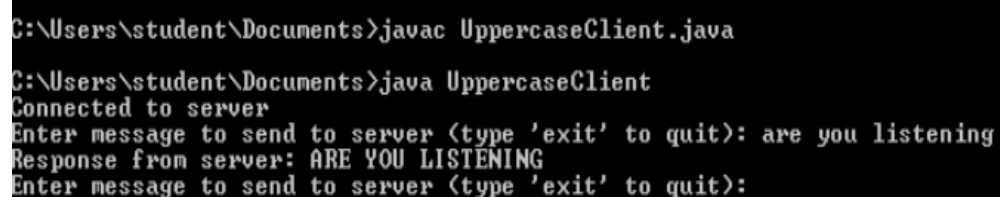
Output

Connect Server



```
C:\Users\student\Documents>javac UppercaseServer.java
C:\Users\student\Documents>java UppercaseServer
Server is listening on port 12345
```

Connect Client and send message



```
C:\Users\student\Documents>javac UppercaseClient.java
C:\Users\student\Documents>java UppercaseClient
Connected to server
Enter message to send to server <type 'exit' to quit>: are you listening
Response from server: ARE YOU LISTENING
Enter message to send to server <type 'exit' to quit>:
```

Server response

```
C:\Users\student\Documents>javac UppercaseServer.java  
C:\Users\student\Documents>java UppercaseServer  
Server is listening on port 12345  
Client connected  
Received from client: are you listening  
Sent to client: ARE YOU LISTENING
```

Week 9

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.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 Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        CourseCoordinator coordinator = new CourseCoordinator(10);  
        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: ");  
        while (true) {  
            try {  
                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());
}
}
```

Output

```
1. Add Course
2. Register Employee
3. List Employees for Course
4. Exit
Enter your choice: Enter your choice: 1
Enter course name: Java
Enter course fee: 4000
Enter maximum number of employees: 2
Is the course online? (true/false): true
Is the course full-time? (true/false): full-time
Course added successfully!
Enter your choice: 2
Enter course name to register: Java
Enter employee name: Kavan
Employee registered successfully!
Enter your choice: 2
Enter course name to register: Java
Enter employee name: Suman
Employee registered successfully!
Enter your choice: 2
Enter course name to register: Java
Enter employee name: Sagar
Cannot register more employees. Course is full.
Employee registered successfully!
```

```
Enter your choice: 3
Enter course name to list employees: Java
Employees registered for Java:
Kavan
Suman
Enter your choice: 4
Exiting...
```

```
=== Code Execution Successful ===|
```

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.

```
public class WorkerExample {

    public static abstract class Worker {
        private String name;
        private double salaryRate;

        public Worker(String name, double salaryRate) {
            this.name = name;
            this.salaryRate = salaryRate;
        }
        public String getName() {
            return name;
        }
        public double getSalaryRate() {
            return salaryRate;
        }
        public abstract double pay(int hours);
        @Override
        public String toString() {
            return name + " (Salary Rate: ₹" + salaryRate + ")";
        }
    }

    public static class DailyWorker extends Worker {
        public DailyWorker(String name, double dailyRate) {
            super(name, dailyRate);
        }
        @Override
        public double pay(int days) {
            return getSalaryRate() * days;
        }
        @Override
        public String toString() {
            return super.toString() + " - Daily Worker";
        }
    }

    public static class SalariedWorker extends Worker {
        private static final int WEEKLY_HOURS = 40;
        public SalariedWorker(String name, double weeklySalary) {
            super(name, weeklySalary / WEEKLY_HOURS);
        }

        @Override
        public double pay(int hours) {
```



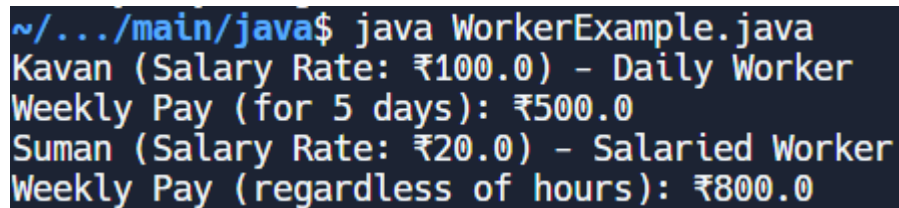
```
        return getSalaryRate() * WEEKLY_HOURS;
    }

    @Override
    public String toString() {
        return super.toString() + " - Salaried Worker";
    }
}

public static void main(String[] args) {
    Worker dailyWorker = new DailyWorker("Alice", 100.0);
    Worker salariedWorker = new SalariedWorker("Bob", 800.0);

    int daysWorked = 5;
    int hoursWorked = 45;
    System.out.println(dailyWorker);
    System.out.println("Weekly Pay (for " + daysWorked + " days): ₹" +
dailyWorker.pay(daysWorked));
    System.out.println(salariedWorker);
    System.out.println("Weekly Pay (regardless of hours): ₹" +
salariedWorker.pay(hoursWorked));
}
}
```

Output

A screenshot of a terminal window showing the execution of a Java program. The prompt is '~/.../main/java\$'. The output consists of four lines: 'Kavan (Salary Rate: ₹100.0) - Daily Worker', 'Weekly Pay (for 5 days): ₹500.0', 'Suman (Salary Rate: ₹20.0) - Salaried Worker', and 'Weekly Pay (regardless of hours): ₹800.0'.

```
~/.../main/java$ java WorkerExample.java
Kavan (Salary Rate: ₹100.0) - Daily Worker
Weekly Pay (for 5 days): ₹500.0
Suman (Salary Rate: ₹20.0) - Salaried Worker
Weekly Pay (regardless of hours): ₹800.0
```

Week 10**10.a).Write a JAVA program to paint like paint brush in applet.**

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

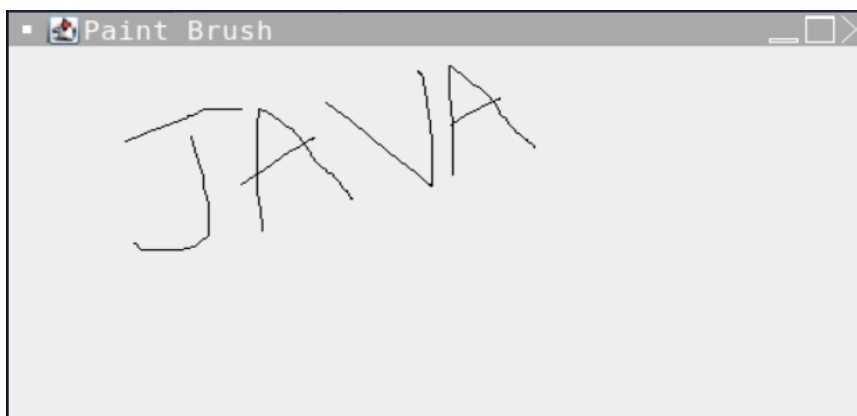
public class PaintBrushApp extends JFrame implements MouseMotionListener {
    int lastX, lastY;

    public PaintBrushApp() {
        setTitle("Paint Brush");
        setSize(500, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        addMouseMotionListener(this);
        setVisible(true);
    }

    public void mouseDragged(MouseEvent me) {
        Graphics g = getGraphics();
        g.setColor(Color.black);
        g.drawLine(lastX, lastY, me.getX(), me.getY());
        lastX = me.getX();
        lastY = me.getY();
    }

    public void mouseMoved(MouseEvent me) {
        lastX = me.getX();
        lastY = me.getY();
    }

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

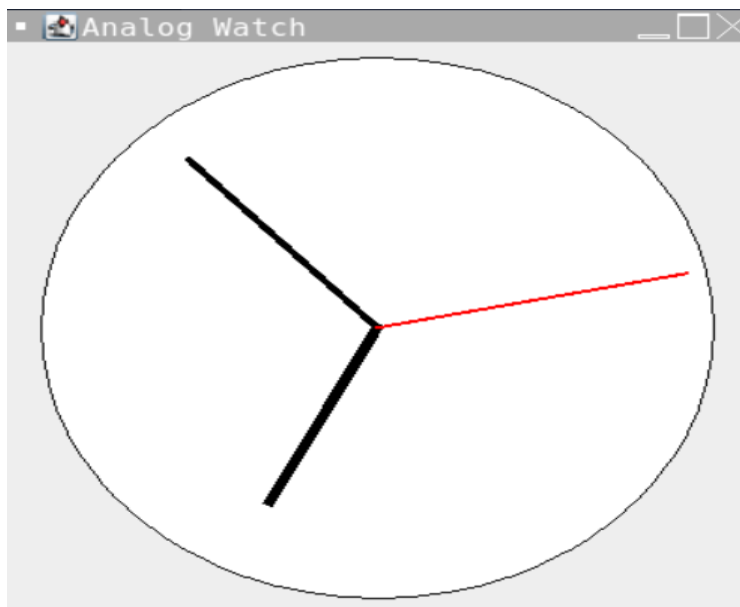
Output

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

```
import javax.swing.*;
import java.awt.*;
import java.util.Calendar;
public class AnalogClockApp extends JPanel implements Runnable {
    Thread clockThread;
    public AnalogClockApp() {
        clockThread = new Thread(this);
        clockThread.start();
    }
    public void run() {
        while (true) {
            repaint();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Calendar cal = Calendar.getInstance();
        int hours = cal.get(Calendar.HOUR);
        int minutes = cal.get(Calendar.MINUTE);
        int seconds = cal.get(Calendar.SECOND);

        g.drawOval(50, 50, 200, 200);
        drawHand(g, hours * 30 - 90, 60, 100, Color.black);
        drawHand(g, minutes * 6 - 90, 80, 100, Color.blue);
        drawHand(g, seconds * 6 - 90, 90, 100, Color.red);
    }
    private void drawHand(Graphics g, int angle, int length, int radius, Color color) {
        double radian = Math.toRadians(angle);
        int x = (int) (150 + length * Math.cos(radian));
        int y = (int) (150 + length * Math.sin(radian));
        g.setColor(color);
        g.drawLine(150, 150, x, y);
    }
    public static void main(String[] args) {
        JFrame frame = new JFrame("Analog Clock");
        AnalogClockApp clockApp = new AnalogClockApp();
        frame.add(clockApp);
        frame.setSize(300, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Output



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

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

public class ShapesApp extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);

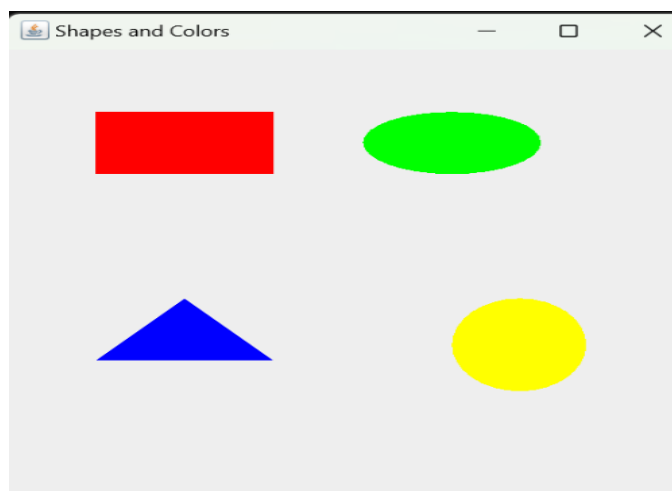
        g.setColor(Color.red);
        g.fillRect(50, 50, 100, 50);

        g.setColor(Color.green);
        g.fillOval(200, 50, 100, 50);

        g.setColor(Color.blue);
        int[] xPoints = { 100, 50, 150 };
        int[] yPoints = { 200, 250, 250 };
        g.fillPolygon(xPoints, yPoints, 3);

        g.setColor(Color.yellow);
        g.fillOval(250, 200, 75, 75);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Shapes and Colors");
        ShapesApp shapesApp = new ShapesApp();
        frame.add(shapesApp);
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Output

Week 11**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 {
    private JTextField textField;
    private String operator = "";
    private double num1 = 0;
    private double num2 = 0;
    private boolean start = true;

    public Calculator() {
        setTitle("Calculator");
        setSize(250, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        textField = new JTextField("0");
        textField.setFont(new Font("Arial", Font.BOLD, 20));
        textField.setHorizontalAlignment(JTextField.RIGHT);
        add(textField, BorderLayout.NORTH);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(4, 4, 2, 2));

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

        for (String text : buttons) {
            JButton button = new JButton(text);
            button.setFont(new Font("Arial", Font.BOLD, 14));
            button.setPreferredSize(new Dimension(50, 50));
            button.addActionListener(new ButtonClickListener());
            panel.add(button);
        }

        add(panel, BorderLayout.CENTER);
    }

    private class ButtonClickListener implements ActionListener {
```

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

    if (command.charAt(0) >= '0' && command.charAt(0) <= '9') {
        if (start) {
            textField.setText("");
            start = false;
        }
        textField.setText(textField.getText() + command);
    } else if (command.charAt(0) == 'C') {
        textField.setText("0");
        operator = "";
        start = true;
    } else if (command.charAt(0) == '=') {
        num2 = Double.parseDouble(textField.getText());
        switch (operator) {
            case "+":
                textField.setText(String.valueOf(num1 + num2));
                break;
            case "-":
                textField.setText(String.valueOf(num1 - num2));
                break;
            case "*":
                textField.setText(String.valueOf(num1 * num2));
                break;
            case "/":
                if (num2 != 0) {
                    textField.setText(String.valueOf(num1 / num2));
                } else {
                    textField.setText("Error");
                }
                break;
        }
        operator = "";
        start = true;
    } else {
        if (!operator.isEmpty()) {
            return;
        }
        operator = command;
        num1 = Double.parseDouble(textField.getText());
        start = true;
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
```

```
    public void run() {  
        Calculator calculator = new Calculator();  
        calculator.setVisible(true);  
    }  
});  
}  
}
```

Output

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

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

public class DigitalWatch {
    private JFrame window;
    private JLabel displayLabel;
    private Timer timeCounter;

    public DigitalWatch() {
        window = new JFrame("Digital Watch");
        displayLabel = new JLabel("00:00:00", SwingConstants.CENTER);
        window.setSize(300, 150);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setLayout(new BorderLayout());
        displayLabel.setFont(new Font("Serif", Font.BOLD, 48));
        window.add(displayLabel, BorderLayout.CENTER);
        timeCounter = new Timer(1000, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                updateDisplayLabel();
            }
        });
        timeCounter.start();
        window.setVisible(true);
    }

    private void updateDisplayLabel() {
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
        String currentTime = sdf.format(new Date());
        displayLabel.setText(currentTime);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new DigitalWatch();
            }
        });
    }
}
```

OUTPUT

