COLLEGE OF ENGINEERING

THIRUVANANTHAPURAM

OBJECT ORIENTED PROGRAMMING LAB RECORD

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MARCH 2022

College of Engineering

Thiruvananthapuram

Object Oriented Programming Lab Record

University Reg. No: TVE20CS021

Name: ANIRUDH A V

Roll No: 11 Batch: 2

Semester **S3** from page No: **1** to page No: **90.**

CERTIFIED BONAFIDE RECORD OF WORK DONE BY

ANIRUDH A V

External Examiner Staff Member in Charge

Contents

Date: 01/01/2022

CYCLE 1

1 Java program to check whether a given number is prime or not

1.1 Aim

To write a java program to check whether a number is prime or not.

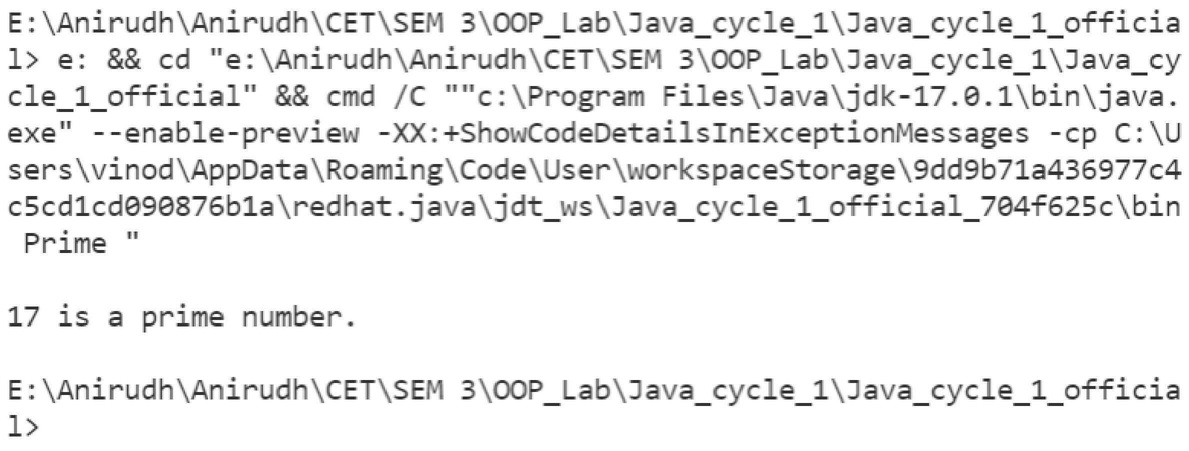
1.2 Algorithm

```
Step - 1   Start
Step - 2   Declare and initialise variables num = 17, flag = 0, i=2.
Step - 3   If num <= 1, Display “number is not a prime number”. Go to step 6.
Step - 4   Repeat the following steps until i < [(num/2) +1].
      Step - 4.1. If (num % i) = 0, increment flag, and break the loop, go to step 5.
      Step - 4.2. Increment i.
Step - 5   If flag not equal to zero, Display “number is not a prime number”.
      Step - 5.1. Else Display “number is a prime number”.
Step - 6   Stop
```

1.3 Code

```java
/** This program is intended to check whether a number is prime or not.
    This is written by Anirudh A V on 22 Dec 2021*/

public class Prime{
    public static void main(String[] args){
        int number = 17, flag =0;
        if (number<=1) {
            System.out.println("The number is neither prime nor composite");
            return;
        }
        for (int i = 2;i < number/2 ;i++ ) {
            if (number % i == 0) {
                flag++;
                break;
            }
        }
        if (flag == 1){
            System.out.println(number+" is not a prime number.");
        }
        else{
            System.out.println(number + " is a prime number.");
        }
    }
}
```

1.4 Sample Output



```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cy
cle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.
exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\U
sers\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4
c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin
 Prime "

17 is a prime number.

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l>
```

2 Java program that checks whether a given string is palindrome or not

2.1 Aim

To Implement a Java program to check whether the given string is a palindrome or not.

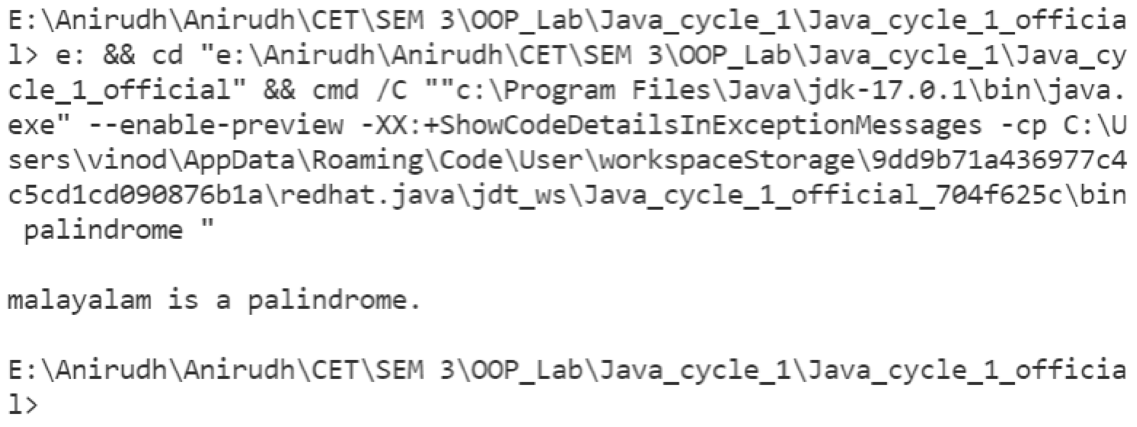
2.2 Algorithm

```
Step - 1   Start
Step - 2   Declare and initialize string [] = “malayalam”, flag = 0, i = 0,
 and len as the length of string [].
Step - 3   Repeat the following steps until i < len/2.
      Step - 3.1. If string [i] != string [len -  i], increment flag and go to
     step 6.
      Step - 3.2. Increment i.
Step - 4   If flag != 0, then Display “The entered word is not a
 palindrome”. Else, Display “The entered word is a palindrome”.
Step - 5   Stop
```

2.3 Code

```java
/** This program is intended to check whether a string is palindrome or not.
    This is written by Anirudh A V on 22 Dec 2021*/

public class palindrome{
  public static void main(String[] args){
    String str = "malayalam";
    int flag = 0, len = str.length();
    for (int i = 0;i < len ;i++ ) {
      if (str.charAt(i) != str.charAt(len-i-1)) {
        flag++;
        break;
      }
    }
    if (flag == 0) {
      System.out.println(str+" is a palindrome.");
    }
    else{
      System.out.println(str+" is not a palindrome.");
    }
  }
}
```

2.4 Sample Output



```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cy
cle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.
exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\U
sers\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4
c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin
 palindrome "

malayalam is a palindrome.

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l>
```

3 Java Program to find the frequency of a character in a string

3.1 Aim

To Implement a Java program to determine the frequency or occurrence of characters in a given sentence.

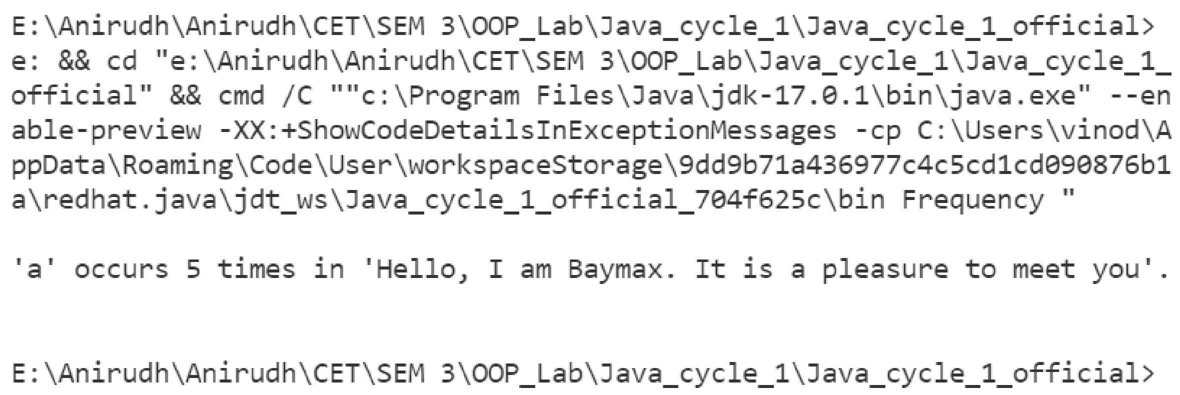
3.2 Algorithm

```
Step - 1   Start
Step - 2   Declare and initialize a String str, and variables i = 0, count = 0,
 key = ‘a’ and len as the length of str.
Step - 3   Repeat the following steps until i = len.
      Step - 3.1. If str[i] = key, increment count.
      Step - 3.2. Increment i.
Step - 4   Display count as the number of occurrences of key in str.
Step - 5   Stop
```

3.3 Code

```java
/**
 * This program is intended to determine the occurrence or frequency of a
 * character in a given sentence.
 * This is written by Anirudh A V on 22 Dec 2021
 */

public class Frequency {
  public static void main(String[] args){
    String str = "Hello, I am Baymax. It is a pleasure to meet you";
    char key = 'a';
    int len = str.length(), count = 0;
    for (int i = 0;i < len ;i++ ) {
      if (str.charAt(i) == key) {
        count++;
      }
    }
    System.out.println(key + " occurs " + count + " times in '" + str + "'.");
  }
}
```

3.4 Sample Output



```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_official>
e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_
official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.exe" --en
able-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\vinod\A
ppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4c5cd1cd090876b1
a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin Frequency "

'a' occurs 5 times in 'Hello, I am Baymax. It is a pleasure to meet you'.


E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_official>
```

4 Java program to reverse any given string

4.1 Aim

To implement a Java program to reverse a string.

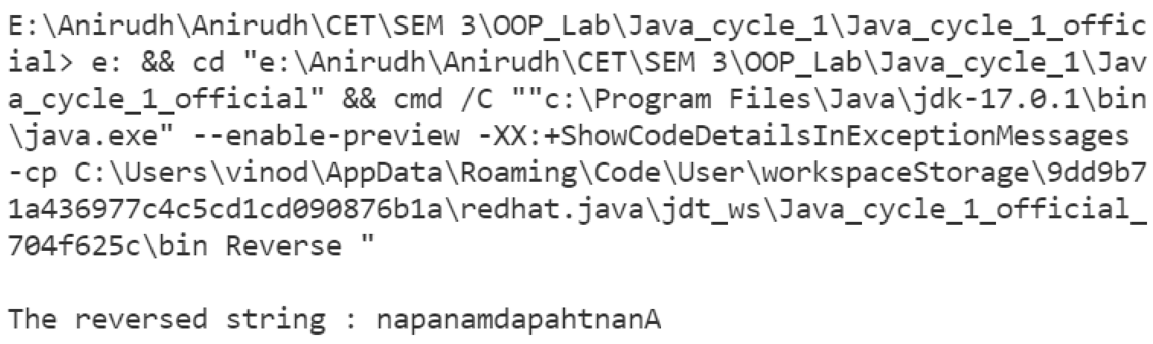
4.2 Algorithm

```
Step - 1   Start
Step - 2   Declare and initialize String str, variables i = 0 and len as the
 length of str.
Step - 3   Declare a character array rev [] of size len + 1.
Step - 4   Repeat the following steps until i = len.
     Step - 4.1.   rev [i] = str [len - 1 – i]
     Step - 4.2.   Increment i.
Step - 5   Assign rev [len] = ‘\0’.
Step - 6   Print rev
Step - 7   Stop
```

4.3 Code

```java
/**
 * This program is intended to reverse a given string.
 * This is written by Anirudh A V on 22 Dec 2021
 */

public class Reverse{
  public static void main(String[] args){
    String str = "Ananthapadmanapan";
    int len = str.length(),i =0;
    char[] rev = new char[len+1];

    for (i = 0;i < len ;i++ ) {
      rev[i] = str.charAt(len-1-i);
    }
    rev[len] ='\0';
    System.out.printf("The reversed string : ");
    System.out.println(rev);
  }
}
```

4.4 Sample Output



```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_offic
ial> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Jav
a_cycle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin
\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages
-cp C:\Users\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b7
1a436977c4c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_
704f625c\bin Reverse "

The reversed string : napanamdapahtnanA
```

5 Java Program to find the second smallest element in an array

5.1 Aim

To implement a Java program to find the second smallest element in an array.

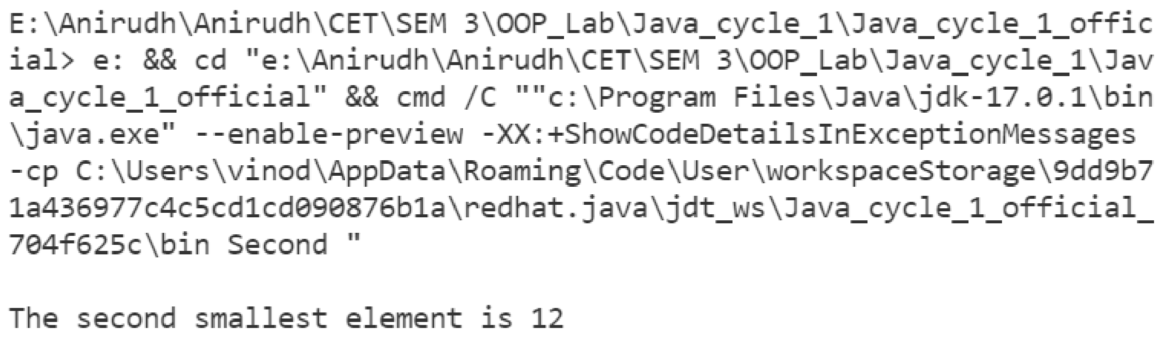
5.2 Algorithm

```
Step - 1   Start
Step - 2   Declare and initialize an array and, variables i = 0, and len as
 the length of the array.
Step - 3   Repeat the following steps until i = 0.
      Step - 3.1. Initialize j = i + 1, repeat the following steps until j = len.
           Step - 3.1.1. If array [i] > array [j], swap array [i] & array [j].
           Step - 3.1.2. Increment j.
      Step - 3.2. Increment i.
Step - 4   Display array [1] as the second smallest element.
Step - 5   Stop
```

5.3 Code

```java
/**
 * This program is intended to find the second smallest element
 * in an array.
 * This is written by Anirudh A V on 22 Dec 2021
 */
public class Second{
  public static void main(String[] args){
    int[] array = {12, 23, 1, 56, 789, 24};
    int len = 6, temp;
    for (int i = 0;i < len ;i++ ) {
      for (int j= i+1;j<len ;j++ ) {
        if (array[i]>array[j]) {
          temp = array[i];
          array[i] = array[j];
          array[j] = temp;
        }
      }
    }
    System.out.println("The second smallest element is "+ array[1]);
  }
}
```

5.4 Sample Output



```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_offic
ial> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Jav
a_cycle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin
\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages
-cp C:\Users\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b7
1a436977c4c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_
704f625c\bin Second "

The second smallest element is 12
```

6 Java Program to multiply two matrices

6.1 Aim

To implement a Java program to multiply two matrices and display their product.

6.2 Algorithm

```
Step - 1   Start
Step - 2   Declare and initialize 2D integer arrays MatrixA, MatrixB and
 mult. And i = 0, j = 0, k = 0.
Step - 3   Check if the matrix can be multiplied or not, if column of
 MatrixA is not equal to row of MatrixB, matrices can't be multiplied and
 an error message is generated.
Step - 4   Repeat until i < row of MatrixA
  Step - 4.1.     Repeat until j < column of MatrixB
      Step - 4.1.1.     Initialize mult[i][j] = 0.
      Step - 4.1.2.     Repeat until k < row of MatrixA
          Step - 4.1.2.1.     Set mult[i][j] = mult[i][j] + MatrixA[i][k] *
        MatrixB[k][j]
          Step - 4.1.2.2.     Increment k.
      Step - 4.1.3.       Increment j.
  Step - 4.2.     Increment i
Step - 5   Display Mult as the required matrix.
Step - 6   Stop
```

6.3 Code

```java
/**
 * This program is intended to multiply two matrices and
 * Display their product.
 * This is written by Anirudh A V on 22 Dec 2021
 */

public class Matrix{

  public static void display(int[][] matrix, int row, int col){
    for (int i = 0;i < row ;i++ ) {
      for (int j = 0;j < col ;j++ ) {
        System.out.printf(matrix[i][j] + " ");
      }
```

```
   System.out.println("");
  }
 }

 public static void main(String[] args){
  int[][] matrixA = {{1,2,3},{4,5,6},{7,8,9}};
  int[][] matrixB = {{1,0,0},{0,1,0},{0,0,1}};
  int[][] mult = new int[3][3];

  for (int i = 0;i < 3 ;i++ ) {
   for (int j = 0;j < 3 ;j++ ) {
    mult[i][j] = 0;
    for (int k = 0;k < 3 ;k++ ) {
     mult[i][j] = mult[i][j] + matrixA[i][k]*matrixB[k][j];
    }
   }
  }
  System.out.println("Matrix A : ");
  display(matrixA, 3, 3);
  System.out.println("Matrix B : ");
  display(matrixB, 3, 3);
  System.out.println("Product : ");
  display(mult, 3, 3);
 }
}
```

6.4 Sample Output

```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_offic
ial> cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.exe" --enabl
e-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\vinod
\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4c5cd1cd0
90876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin Matri
x "
Matrix A :
1 2 3
4 5 6
7 8 9
Matrix B :
1 0 0
0 1 0
0 0 1
Product :
1 2 3
4 5 6
7 8 9
```

7 Java Program to display transpose of a matrix

7.1 Aim

To implement a Java program to find the transpose of a given matrix.

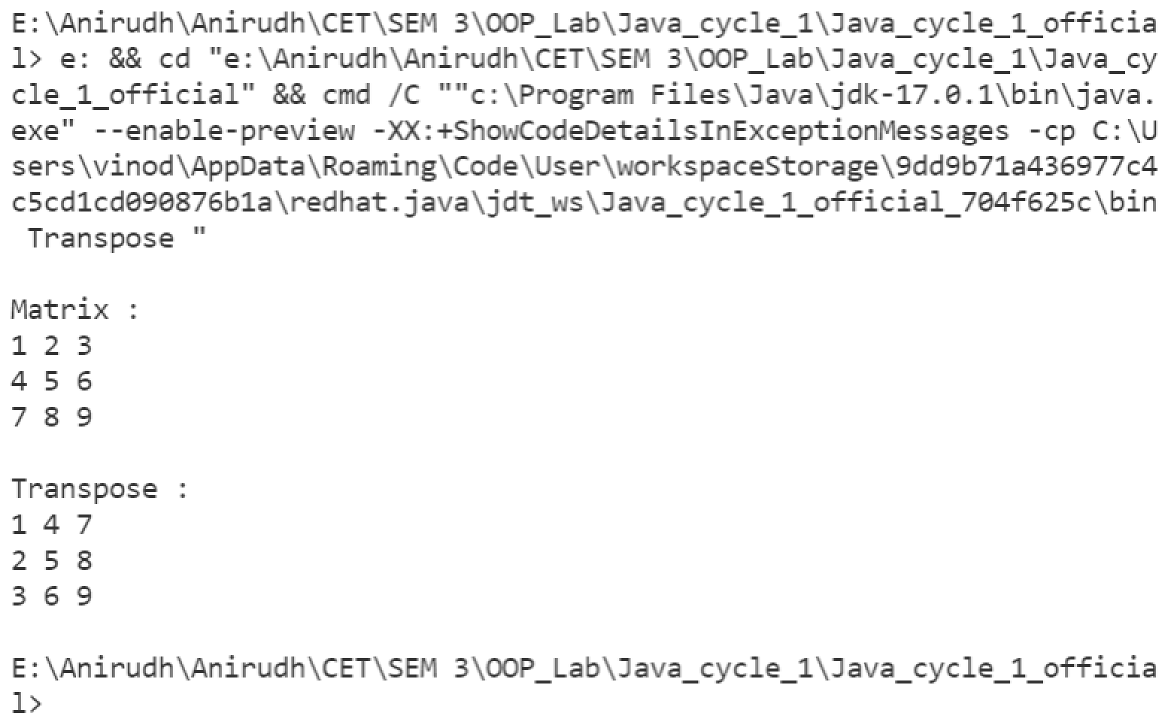
7.2 Algorithm

```
Step - 1   Start
Step - 2   Declare and initialize a two-dimensional array matrix. Initialize
 variables i = 0, j = 0, row and col as the number of rows and columns of
 the matrix.
Step - 3   Print the elements of the matrix.
Step - 4   Repeat until i = row
   Step - 4.1.     Repeat until j = col
        Step - 4.1.1.        Print matrix[j][i]
        Step - 4.1.2.        Increment j.
   Step - 4.2.     Print "\n"
   Step - 4.3.     Increment i.
Step - 5   Stop
```

7.3 Code

```java
/**
 * This program is intended to find the transpose of a given matrix.
 * This is written by Anirudh A V on 22 Dec 2021
 */

public class Transpose{

  public static void display(int[][] matrix, int row, int col){
    for (int i = 0;i < row ;i++ ) {
      for (int j = 0;j < col ;j++ ) {
        System.out.printf(matrix[i][j] + " ");
      }
      System.out.println("");
    }
  }

  public static void main(String[] args){
    int[][] matrix = {{1,2,3},{4,5,6},{7,8,9}};
    System.out.println("\nMatrix : ");
    display(matrix, 3, 3);
    System.out.println("\nTranspose : ");
    for (int i = 0;i < 3 ;i++ ) {
      for (int j = 0;j < 3 ;j++ ) {
        System.out.printf(matrix[j][i] + " ");
      }
      System.out.println("");
    }
  }
}
```

7.4 Sample Output



```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cy
cle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.
exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\U
sers\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4
c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin
 Transpose "

Matrix :
1 2 3
4 5 6
7 8 9

Transpose :
1 4 7
2 5 8
3 6 9

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l>
```

Date: 28/01/2022

CYCLE 2

8 Java Program to calculate area of shapes

8.1 Aim

To write a Java program to calculate the area of different shapes namely circle, rectangle, and triangle using the concept of method overloading.

8.2 Algorithm

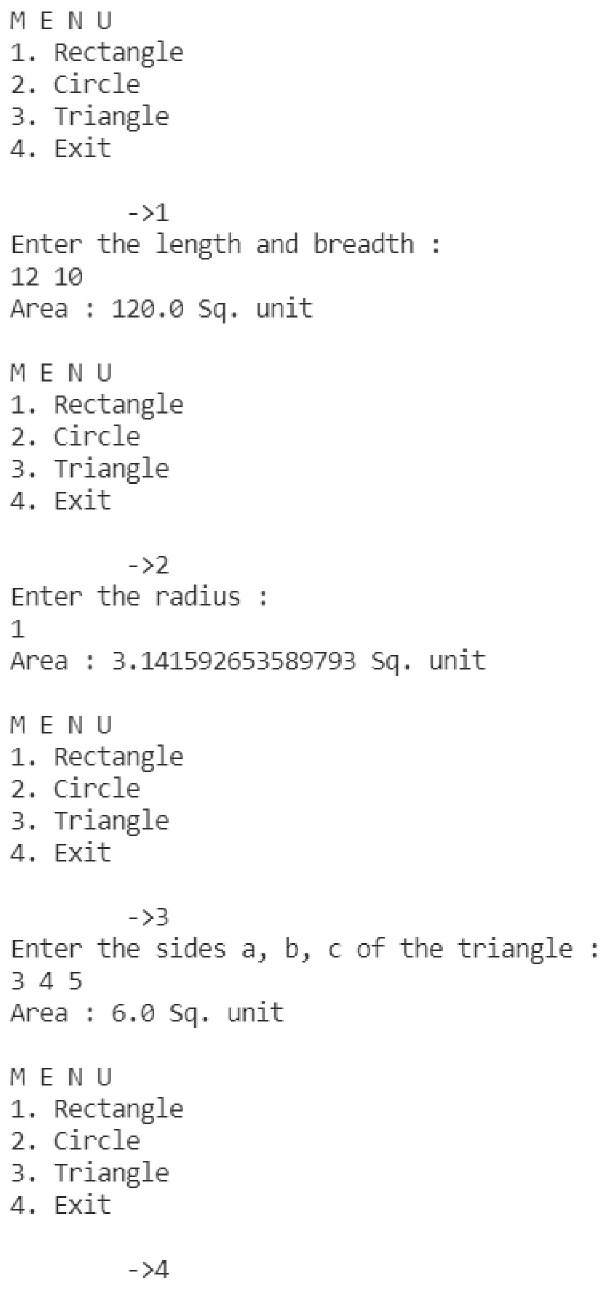
```
Step - 1   Start
Step - 2   Create a class Area and initialize all the variables.
Step - 3   Declare the overloaded function area and its various forms.
Step - 4   Declare the main function and create an object for Area.
Step - 5   Declare a Scanner object to accept user input.
Step - 6   Display the Menu to the user.
        Step - 6.1.     For option 1 go to STEP 7.
        Step - 6.2.     For option 1 go to STEP 11.
        Step - 6.3.     For option 1 go to STEP 15.
        Step - 6.4.     For option 1 go to STEP 19.
Step - 7   Accept the length and breadth.
Step - 8   Call the area method of the class and pass the length and breadth
 as the parameter.
Step - 9   Print the area.
Step - 10 Go to STEP 6.
Step - 11 Accept the radius from the user.
Step - 12 Call the area method of the class and pass the radius as the
 parameter.
Step - 13 Print the area.
Step - 14 Go to STEP 6.
Step - 15 Accept the length of the three sides from the user.
Step - 16 Call the area method of the class and pass the three lengths as
 the parameters.
Step - 17 Print the area.
Step - 18 Go to STEP 6.
Step - 19 Stop
```

8.3 Code

```
/**
 * This program is intended to calculate the are of different shapes
 * using method overloading.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

import java.util.Scanner;

public class Area {

    static double area(double radius) {
        return Math.PI * Math.pow(radius, 2);
    }

    static double area(double length, double breadth) {
        return length * breadth;
    }

    static double area(double a, double b, double c) {
        double s = (a + b + c) / 2;
        double x = s * (s - a) * (s - b) * (s - c);
        return Math.sqrt(x);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int response = 1, choice;
        double radius, a, b, c, length, breadth;
        do {
            System.out.println("\t\nM E N U");
            System.out.println("1. Rectangle\n2. Circle\n3. Triangle\n4. 
Exit");
            System.out.printf("\n\t->");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Enter the length and breadth : ");
                    length = sc.nextDouble();
                    breadth = sc.nextDouble();
                    System.out.println("Area : " + area(length, breadth)+" Sq. 
unit");
                    break;

                case 2:
                    System.out.println("Enter the radius : ");
                    radius = sc.nextDouble();
```

```java
                    System.out.println("Area : " + area(radius)+" Sq. unit");
                    break;

                case 3:
                    System.out.println("Enter the sides a, b, c of the 
triangle : ");
                    a = sc.nextDouble();
                    b = sc.nextDouble();
                    c = sc.nextDouble();
                    System.out.println("Area : " + area(a, b, c)+" Sq. unit");
                    break;

                case 4:
                    System.exit(0);
                    break;

                default:
                    break;
            }
        } while (response == 1);
        sc.close();
    }
}
```

8.4 Sample Output



```
M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->1
Enter the length and breadth :
12 10
Area : 120.0 Sq. unit

M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->2
Enter the radius :
1
Area : 3.141592653589793 Sq. unit

M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->3
Enter the sides a, b, c of the triangle :
3 4 5
Area : 6.0 Sq. unit

M E N U
1. Rectangle
2. Circle
3. Triangle
4. Exit

        ->4
```

9 Java program to demonstrate Inheritance

9.1 Aim

To write a Java program that creates a class named 'Employee' having the following members: Name, Age, Phone number, Address, Salary. It also has a method named 'printSalary()' which prints the salary of the Employee. Two classes 'Officer' and 'Manager' inherit the employee class. The 'Officer' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign a name, age, phone number, address, and salary to an officer and a manager by making an object of both of these classes and printing the same.

9.2 Algorithm

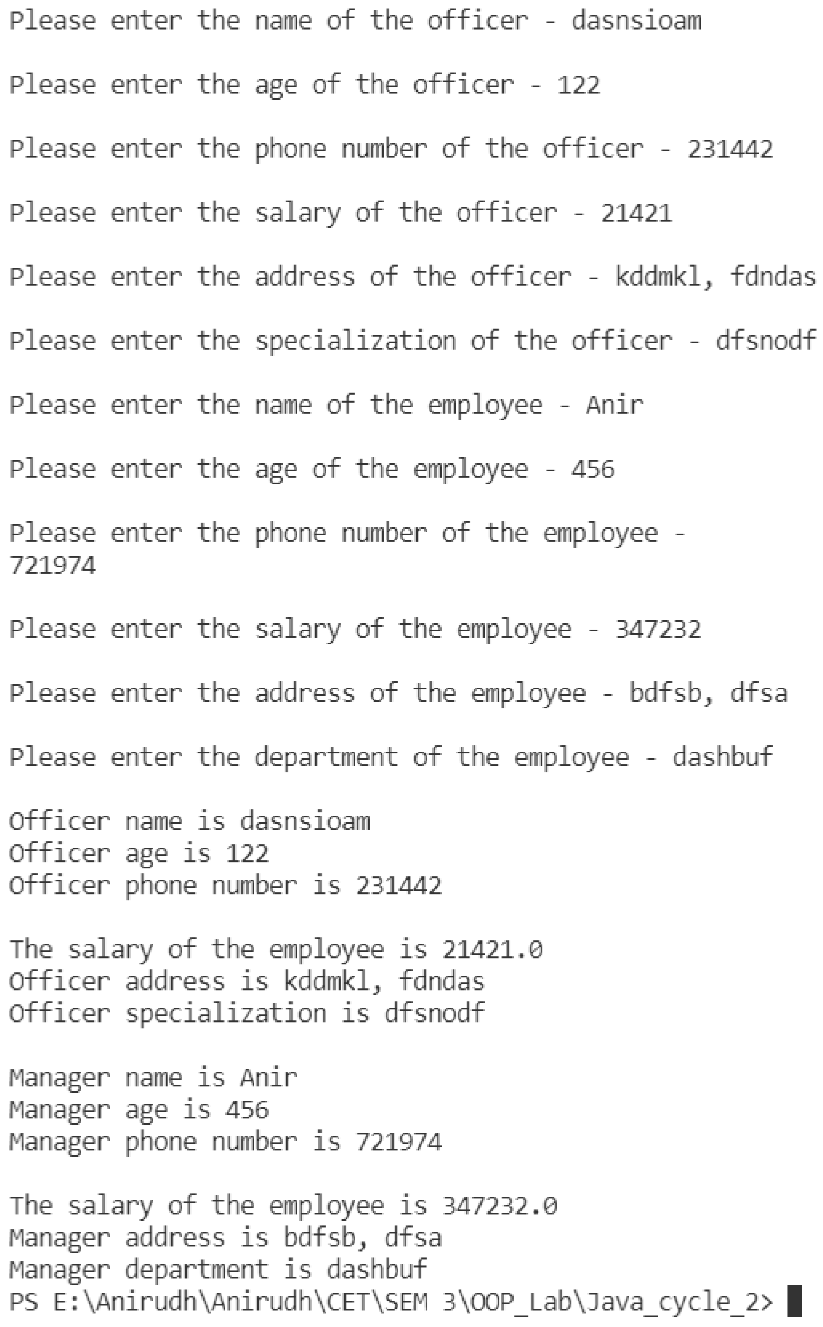
```
Step - 1   Start
Step - 2   Create the class Employee and declare all the member variables
           and member functions.
Step - 3   Create the class Officer that inherits the class Employee.
Step - 4   Declare the member variable and all the member functions.
Step - 5   Create the class Manager that inherits the class Employee.
Step - 6   Declare the member variable and all the member functions.
Step - 7   Declare all the main functions.
Step - 8   Declare objects of the classes.
Step - 9   Call the functions that demonstrate the properties of inheritance
Step - 10  Stop
```

9.3 Code

```java
/**
 * This program is intended to understand the concept of Inheritance.
 * It takes input for classes Officer and Employee, which are the child
 * of the class Employee. Then prints the entered data.
 * This program is written by Anirudh A V on Jan 25 2022.
 */
import java.util.Scanner;

class Employee {
    String name;
    int age;
    int ph;
    double salary;
    String adr;

    void printSalary() {
        System.out.println("\nThe salary of the employee is " + salary);
    }
}

class Officer extends Employee {
    String specialization;

    void newOfficer() {
        Scanner sc = new Scanner(System.in);
        System.out.printf("\nPlease enter the name of the officer - ");
        this.name = sc.nextLine();
        System.out.printf("\nPlease enter the age of the officer - ");
        this.age = sc.nextInt();
        System.out.printf("\nPlease enter the phone number of the officer -
");
        this.ph = sc.nextInt();
        System.out.printf("\nPlease enter the salary of the officer - ");
        this.salary = sc.nextDouble();
        sc.nextLine();
        System.out.printf("\nPlease enter the address of the officer - ");
        this.adr = sc.nextLine();
        System.out.printf("\nPlease enter the specialization of the officer -
");
        this.specialization = sc.nextLine();
        sc.close();
    }

    void printData() {
        System.out.println("\nOfficer name is " + name);
        System.out.println("Officer age is " + age);
        System.out.println("Officer phone number is " + ph);
        printSalary();
        System.out.println("Officer address is " + adr);
        System.out.println("Officer specialization is " + specialization);
    }
}

class Manager extends Employee {
    String department;

    void newManager() {
        Scanner sc = new Scanner(System.in);
        System.out.printf("\nPlease enter the name of the employee - ");
        this.name = sc.nextLine();
```

```java
        System.out.printf("\nPlease enter the age of the employee - ");
        this.age = sc.nextInt();
        System.out.println("\nPlease enter the phone number of the employee - 
");
        this.ph = sc.nextInt();
        System.out.printf("\nPlease enter the salary of the employee - ");
        this.salary = sc.nextDouble();
        sc.nextLine();
        System.out.printf("\nPlease enter the address of the employee - ");
        this.adr = sc.nextLine();
        System.out.printf("\nPlease enter the department of the employee - ");
        this.department = sc.nextLine();
        sc.close();
    }

    void printData() {
        System.out.println("\nManager name is " + name);
        System.out.println("Manager age is " + age);
        System.out.println("Manager phone number is " + ph);
        printSalary();
        System.out.println("Manager address is " + adr);
        System.out.println("Manager department is " + department);
    }
}

public class Office {
    public static void main(String args[]) {
        Manager m1 = new Manager();
        Officer off = new Officer();
        off.newOfficer();
        m1.newManager();
        off.printData();
        m1.printData();
    }
}
```

9.4 Sample Output



```
Please enter the name of the officer - dasnsioam

Please enter the age of the officer - 122

Please enter the phone number of the officer - 231442

Please enter the salary of the officer - 21421

Please enter the address of the officer - kddmkl, fdndas

Please enter the specialization of the officer - dfsnodf

Please enter the name of the employee - Anir

Please enter the age of the employee - 456

Please enter the phone number of the employee -
721974

Please enter the salary of the employee - 347232

Please enter the address of the employee - bdfsb, dfsa

Please enter the department of the employee - dashbuf

Officer name is dasnsioam
Officer age is 122
Officer phone number is 231442

The salary of the employee is 21421.0
Officer address is kddmkl, fdndas
Officer specialization is dfsnodf

Manager name is Anir
Manager age is 456
Manager phone number is 721974

The salary of the employee is 347232.0
Manager address is bdfsb, dfsa
Manager department is dashbuf
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2>
```

10 Java Program to demonstrate Method Overriding

10.1 Aim

To Write two Java classes, Employee and Engineer. The engineer should inherit from the Employee class. Employee class to have two methods display() and calcSalary(). Write a program to display the engineer's salary and to display from Employee class using a single object instantiation (i.e., only one object creation is allowed).

- display () only prints the name of the class and does not return any value. Ex. "Name of class is Employee."
- calcSalary() in Employee displays "Salary of employee is 10000" and calcSalary() in Engineer displays "Salary of employee is 20000."

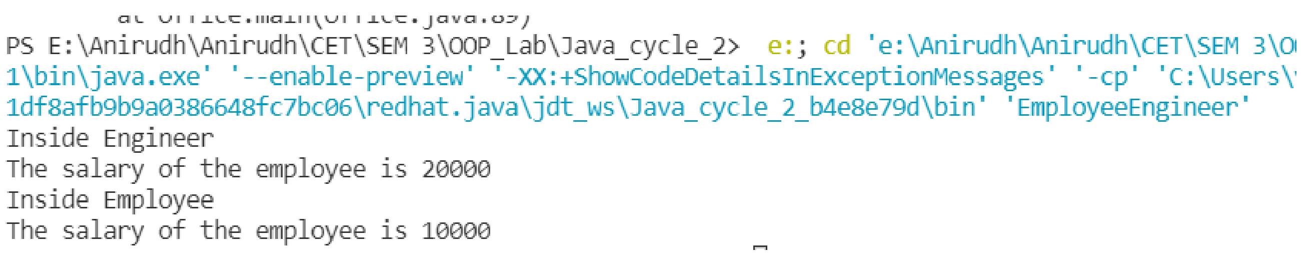
10.2 Algorithm

```
Step - 1   Start
Step - 2   Create the Employee class and declare all the variables and the
 member functions.
Step - 3   Create the Engineer class that inherits from Employee.
Step - 4   Declare the member variables and the member functions
 (Redeclare calcSalary()).
Step - 5   Declare the main function and create an object of Engineer.
Step - 6   The function overriding takes over and the function in Engineer
 is called.
Step - 7   Stop.
```

10.3 Code

```java
/**
 * This program is intended to demonstrate the concept of method overriding.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

class Employee {
    void display() {
        System.out.println("Inside Employee");
        return;
    }

    void calcSalary() {
        System.out.println("The salary of the employee is 10000");
        return;
    }
}

class Engineer extends Employee {
    void display() {
        System.out.println("Inside Engineer");
        return;
    }

    void calcSalary() {
        System.out.println("The salary of the employee is 20000");
        super.calcSalary();
        return;
    }
}

public class EmployeeEngineer {
    public static void main(String args[]) {
        Engineer eng = new Engineer();
        eng.display();
        eng.calcSalary();
    }
}
```

10.4 Sample Output



```
        at Office.main(Office.java:89)
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2>  e:; cd 'e:\Anirudh\Anirudh\CET\SEM 3\O
1\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
1df8afb9b9a0386648fc7bc06\redhat.java\jdt_ws\Java_cycle_2_b4e8e79d\bin' 'EmployeeEngineer'
Inside Engineer
The salary of the employee is 20000
Inside Employee
The salary of the employee is 10000
```

11 Java Program to create an Abstract class

11.1 Aim

To write a java program to create an abstract class named Shape that contains an empty method named numberOfSides(). Provide three classes named Rectangle, Triangle, and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides() that shows the number of sides in the given geometrical structures.

11.2 Algorithm

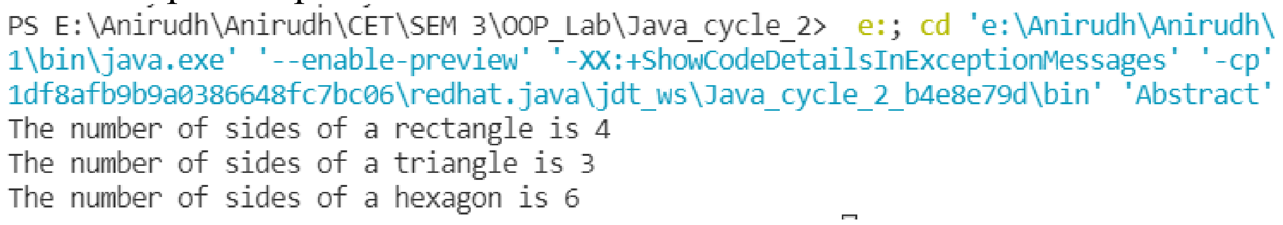
```
Step - 1   Start
Step - 2   Declare an abstract class and create all the variables.
Step - 3   Declare the class Rectangle and make it inherit from the
           abstract class ‘Shape’.
Step - 4   Declare the methods in Rectangle.
Step - 5   Declare the class Triangle and make it inherit from the abstract
           class ‘Shape’.
Step - 6   Declare the methods in Triangle.
Step - 7   Declare the class Hexagon and make it inherit from the
           abstract class ‘Shape’.
Step - 8   Declare the methods in Hexagon.
Step - 9   Stop
```

11.3 Code

```java
/**
 * This program is intended to understand the concept of polymorphism in java.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

abstract class Shape {
    abstract void numberOfSides();
}

class Rectangles extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides of a rectangle is 4");
    }
}
```

```java
class Triangles extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides of a triangle is 3");
    }
}

class Hexagons extends Shape {
    void numberOfSides() {
        System.out.println("The number of sides of a hexagon is 6");
    }
}

public class Abstract {
    public static void main(String args[]) {
        Rectangles rt = new Rectangles();
        Triangles tr = new Triangles();
        Hexagons hx = new Hexagons();
        rt.numberOfSides();
        tr.numberOfSides();
        hx.numberOfSides();
    }
}
```

11.4 Sample Output



```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2>  e:; cd 'e:\Anirudh\Anirudh\
1\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
1df8afb9b9a0386648fc7bc06\redhat.java\jdt_ws\Java_cycle_2_b4e8e79d\bin' 'Abstract'
The number of sides of a rectangle is 4
The number of sides of a triangle is 3
The number of sides of a hexagon is 6
```

12 Java Program to Illustrate Interface Inheritance

12.1 Aim

To write a Java program to illustrate Interface inheritance.

12.2 Algorithm

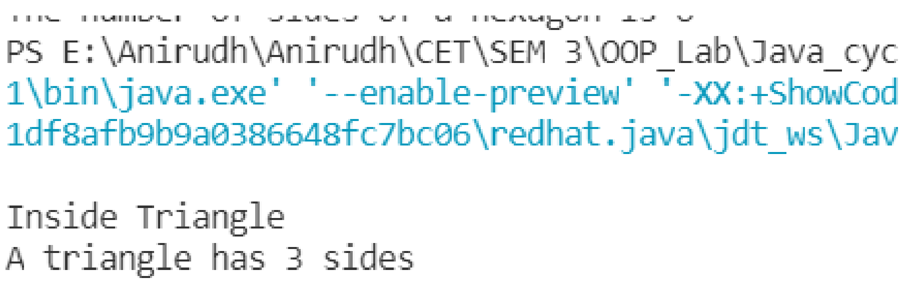
```
Step - 1  Start
Step - 2  Initialize the first interface ShapeName.
Step - 3  Layout definitions for all the methods in ShapeName.
Step - 4  Create another interface in ShapeInfo that inherits
          ShapeName.
Step - 5  Layout definitions for all the methods in ShapeName.
Step - 6  Create a class that inherits from ShapeInfo and declare
          functions inside it.
Step - 7  Layout the definitions for the functions in the interfaces in
          the class definition.
Step - 8  Declare the main function and create an object of the class.
Step - 9  Call the member function of the class and interface
          inheritance is observed.
Step - 10 Stop
```

12.3 Code

```
/**
 * This program is intended to demonstrate the concept of interface
inheritance.
 * This program is written by Anirudh A V on Jan 25, 2022.
 */

interface ShapeName {
    public void name();
}

interface ShapeInfo extends ShapeName {
    public void info();
}
```

```
class Triangle implements ShapeInfo{
    public void name(){
    System.out.println("\nInside Triangle");
    }
    public void info(){
    System.out.println("A triangle has 3 sides\n");
    }
    }

public class InterfaceInheritance {
    public static void main(String args[]) {
        Triangle tri = new Triangle();
        tri.name();
        tri.info();
    }
}
```

12.4 Sample Output



```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cyc
1\bin\java.exe' '--enable-preview' '-XX:+ShowCod
1df8afb9b9a0386648fc7bc06\redhat.java\jdt_ws\Jav

Inside Triangle
A triangle has 3 sides
```

13 Java Program to demonstrate the use of Garbage Collectors

13.1 Aim

To write a Java program to demonstrate the use of garbage collectors.

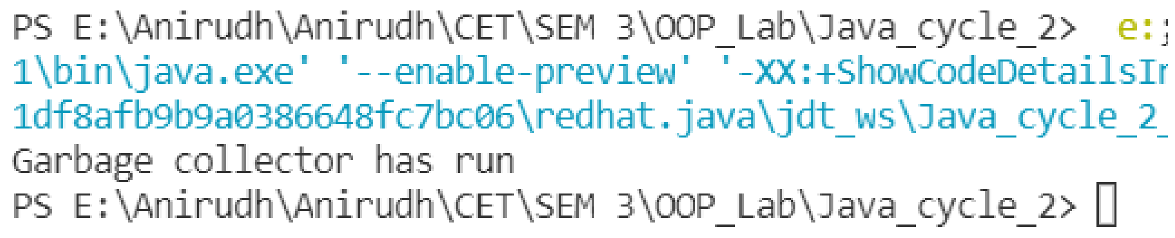
13.2 Algorithm

```
Step - 1   Start
Step - 2   Define the garbage_collector class and declare the finalize()
 function inside it.
Step - 3   Print the message “Garbage collector has run”.
Step - 4   Declare the main function and create an object of the class in
 it.
Step - 5   Call the display() method in the class.
Step - 6   Assign the object to null.
Step - 7   The garbage collector then activates and the finalizer() method
 in the class is called.
Step - 8   Stop
```

13.3 Code

```java
/**
 * This program is intended to demonstrate the concept of garbage collection.
 * This program is written by Anirudh A V on Jan 25 2022.
 */

public class garbage_collector {
    protected void finalize() throws Throwable {
        System.out.println("Garbage collector has run");
    }

    public static void main(String args[]) {
        garbage_collector gb = new garbage_collector();
        gb = null;
        // Calling garbage collector
        System.gc();
    }
}
```

13.4 Sample Output



```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2>  e:
1\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsI
1df8afb9b9a0386648fc7bc06\redhat.java\jdt_ws\Java_cycle_2_
Garbage collector has run
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_2> 
```

Date: 12/02/2022

CYCLE 3

14 File Handling in Java using Reader/Writer

14.1 Aim

To Write a file handling program in Java with reader/writer.

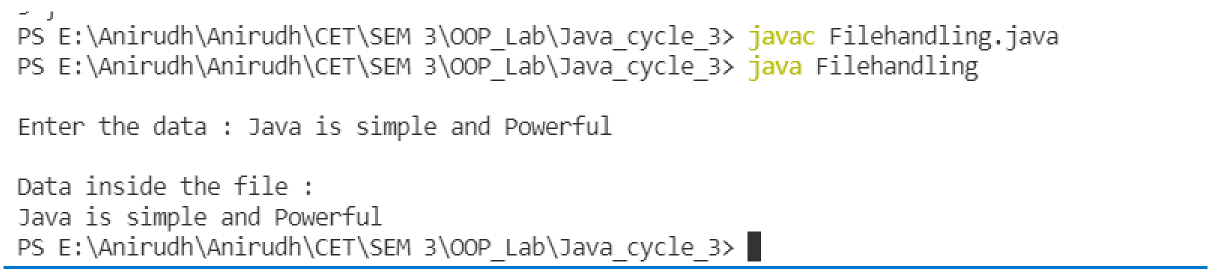
14.2 Algorithm

```
Step - 1   Start
Step - 2   Use the try statement to create the FileWriter object.
Step - 3   Create the file through the path if it doesn't exist or open it with
           path if it does exist.
Step - 4   Write a sample string "data" into the file using the file object.
Step - 5   Accept the user input and write the string into the file using file
           object.
Step - 6   Close the file object/stream.
Step - 7   Declare catch statements to catch any IO exceptions if it occurs.
Step - 8   Open the file just created by creating a FileReader object in a
           try statement.
Step - 9   Read the characters from the file and display them to the user.
Step - 10  Close the file object/stream.
Step - 11  Catch statement is specified to catch IO exceptions.
Step - 12  Stop
```

14.3 Code

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

public class Filehandling {
    public static void main(String [] args){
        try {
            Writer w = new FileWriter("file.txt");
            Scanner sc = new Scanner(System.in);
            System.out.printf("\nEnter the data : ");
            String content = sc.nextLine();
            w.write(content);
            w.close();
            sc.close();
            Reader r = new FileReader("file.txt");
            if (r.ready()) {
                System.out.println("\nData inside the file : ");
                int character = r.read();
                while (character != -1) {
                    System.out.print((char)character);
                    character = r.read();
                }
                r.close();
            } else {
                System.err.println("\nFile not Found.\n");
            }
        } catch (Exception e) {
            System.err.println(e+" occurred.\n");
        }
    }

}
```

14.4 Sample Output



```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3> javac Filehandling.java
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3> java Filehandling

Enter the data : Java is simple and Powerful

Data inside the file :
Java is simple and Powerful
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3>
```

15 Java program to reads a file and displays on the screen with a line number before each line

15.1 Aim

To write a Java program that reads a file and displays the file on the screen, with a line number before each line.

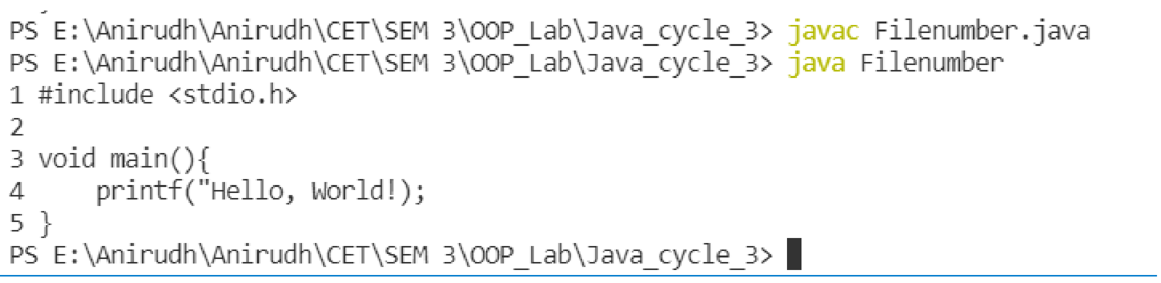
15.2 Algorithm

```
Step - 1   Start
Step - 2   Use the try statement to create the FileWriter object.
Step - 3   Create the file through the path if it doesn’t exist or open it with
           path if it does exist.
Step - 4   Accept the number of lines the user wants to write into the file.
Step - 5   Accept the lines from the user and write them to the file using
           the FileWriter object.
Step - 6   Close the file object/stream.
Step - 7   Declare catch statements to catch any IO exeptions if it occurs.
Step - 8   Open the file just created by creating a FileReader object in a
           try statement.
Step - 9   Keep count of every line using an incrementer.
Step - 10  Print the line number followed by the line in the file (read using
           the FilReader object).
Step - 11  Close the file object/stream.
Step - 12  Catch statement is specified to catch an IO exceptions.
Step - 13  Stop
```

15.3 Code

```java
import java.io.*;

public class Filenumber {
    public static void main(String[] args) {
        try {
            FileReader f = new FileReader("code.txt");
            String str = "";
            if (f.ready()) {
                int data = f.read();
                while (data != -1) {
                    str = str + (char) data;
                    data = f.read();
                }
            } else {
                System.err.println("\nFile not Found\n");
            }
            int n = 2;
            System.out.print("1 ");
            // int len = str.length();
            for (int i = 0; i < str.length(); i++) {
                if (str.charAt(i) == '\n') {
                    System.out.print(str.charAt(i));
                    System.out.print(n + " ");
                    n++;
                } else {
                    System.out.print(str.charAt(i));
                }
                f.close();
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

15.4 Sample Output



```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3> javac Filenumber.java
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3> java Filenumber
1 #include <stdio.h>
2
3 void main(){
4     printf("Hello, World!);
5 }
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3>
```

16 Java program that displays the number of characters, lines, and words in a text file

16.1 Aim

To write a Java program that displays the number of characters, lines, and words in a text file.

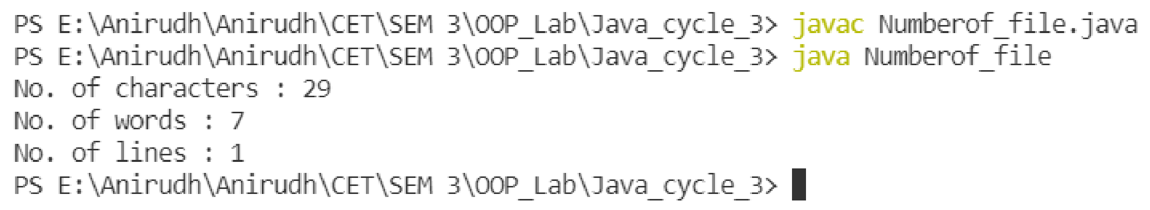
16.2 Algorithm

```
Step - 1   Start
Step - 2   Create a class Counter.
Step - 3   A counter() method is created to read from the file and display all the counts.
    Step - 3.1.   Use try statement to create the FileWriter object.
    Step - 3.2.   Open the file by creating a FileReader object in a try statement.
    Step - 3.3.   As characters are read, increment n_chars variable to keep count.
    Step - 3.4.   As spaces are detected increment n_wrds to keep count of all the words.
    Step - 3.5.   As the new line character is read, increment n_lines to keep count of all the lines.
    Step - 3.6.   Print the number of lines, number of words, and the number of characters to the user.
    Step - 3.7.   Close the file object/stream.
    Step - 3.8.   Catch statement is specified to catch IO exceptions.
Step - 4   Declare the main function and create an object of the class Counter.
Step - 5   Call the counter() method of the object.
Step - 6   Stop
```

16.3 Code

```java
import java.io.*;

public class Numberof_file {
    public static void main(String [] args){
        try {
            FileReader f = new FileReader("file.txt");
            String str = "";
            if (f.ready()) {
                int data = f.read();
                while (data !=-1) {
                    str = str + (char)data;
                    data = f.read ();
                }
            } else {
                System.err.println("\nFile not Found\n");
            }
            f.close();
            int new_line_count = 0, len = str.length(), no_of_words = 0,
no_of_lines = 0;;
            for (int i = 0; i < len; i++) {
                if (str.charAt(i) == '\n') {
                    new_line_count++;
                    no_of_lines++;
                }
                if ((str.charAt(i) == '\n' || str.charAt(i)==' ') &&
str.charAt(i-1) != ' ' && str.charAt(i-1) != '\n') {
                    no_of_words++;
                }
            }
            System.out.printf("No. of characters : %d\nNo. of words : %d\nNo.
of lines : %d", len-new_line_count, no_of_words+1, no_of_lines+1);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

16.4 Sample Output



```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3> javac Numberof_file.java
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3> java Numberof_file
No. of characters : 29
No. of words : 7
No. of lines : 1
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3>
```

17 Java program that reads from a file and writes to a file by handling all file-related exceptions

17.1 Aim

To write a Java program that reads from a file and writes to a file by handling all file-related exceptions.

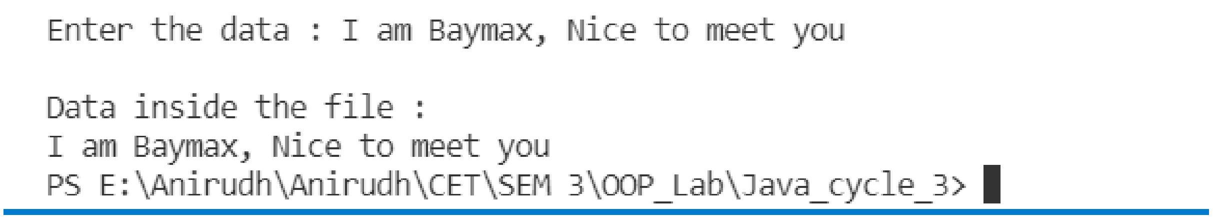
17.2 Algorithm

Step - 1 Start
Step - 2 Use try statement to create the FileWriter object.
Step - 3 Create the file through the path if it doesn’t exist or open it with path if it does exist.
Step - 4 Wrtite sample lines from into the file using the FileWriter object.
Step - 5 Close the file object/stream.
Step - 6 Declare catch statements to catch all IO exeptions if it occurs.
Step - 7 Open the file just created by creating a FileReader object in a try statement.
Step - 8 Loop till the end of the file and Print all the lines from the file
Step - 9 Close the file object/stream.
Step - 10 Catch statement is specified to catch all IO exceptions if any.
Step - 11 Stop

17.3 Code

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

public class readnwrite {

    public static void main(String[] args) {
        try {
            Writer w = new FileWriter("file.txt");
            Scanner sc = new Scanner(System.in);
            System.out.printf("\nEnter the data : ");
            String content = sc.nextLine();
            w.write(content);
            w.close();
            sc.close();

            Reader r = new FileReader("file.txt");
            if (r.ready()) {
                System.out.println("\nData inside the file : ");
                int character = r.read();
                while (character != -1) {
                    System.out.print((char)character);
                    character = r.read();
                }
                r.close();
            } else {
                System.err.println("\nFile not Found.\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

17.4 Sample Output



```
Enter the data : I am Baymax, Nice to meet you

Data inside the file :
I am Baymax, Nice to meet you
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3>
```

18 Java Program that uses String Tokenizers

18.1 Aim

To write a Java program that reads a line of integers, and then displays each integer, and the sum of all the integers.

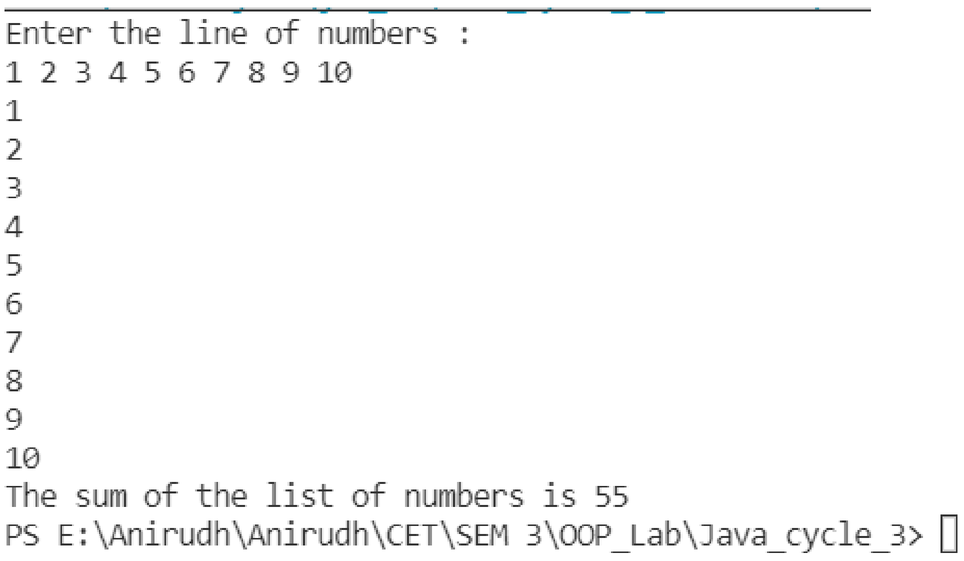
18.2 Algorithm

```
Step - 1  Start
Step - 2  Use the try statement to create the FileWriter object.
Step - 3  Create the file through the path if it doesn't exist or open it
          with the path if it does exist.
Step - 4  Accept the list of integers separated by spaces.
Step - 5  Write the list into the file using the FileWriter object.
Step - 6  Close the file object/stream.
Step - 7  Declare catch statements to catch an IO exception if it
          occurs.
Step - 8  Open the file just created by creating a FileReader object in
          a try statement.
Step - 9  Check if the file is empty. If yes, exit out of the program
          else continues:
    Step - 9.1.   Create the StringTokenizer object
    Step - 9.2.   Loop through all the tokens
        Step - 9.2.1.   Read the tokens to a variable
        Step - 9.2.2.   Convert the token characters to numbers of
                        type int.
        Step - 9.2.3.   Display the integer to the user.
        Step - 9.2.4.   Add the value of the integers to a sum
                        variable.
    Step - 9.3.   Display the value of the sum to the user.
Step - 10   Close the file object/stream.
Step - 11   Catch statement is specified to catch IO exceptions if
            any.
Step - 12   Stop
```

18.3 Code

```java
import java.util.Scanner;
import java.util.StringTokenizer;

public class Line_of_Integers {
    public static void main(String[] args) {
        int sum = 0;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the line of numbers : ");
        String str = sc.nextLine();

        StringTokenizer num = new StringTokenizer(str, " ");
        while (num.hasMoreTokens()) {
            String s = num.nextToken();
            System.out.println(s);
            sum = sum + Integer.parseInt(s);
        }
        System.out.println("The sum of the list of numbers is "+sum);
        sc.close();
    }
}
```

18.4 Sample Output



```
Enter the line of numbers :
1 2 3 4 5 6 7 8 9 10
1
2
3
4
5
6
7
8
9
10
The sum of the list of numbers is 55
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_3> 
```

Date: 19/02/2022

CYCLE 4

19 Java program that shows the usage of try, catch, throws, and finally

19.1 Aim

To Write Java program that shows the usage of try, catch, throws, and finally.

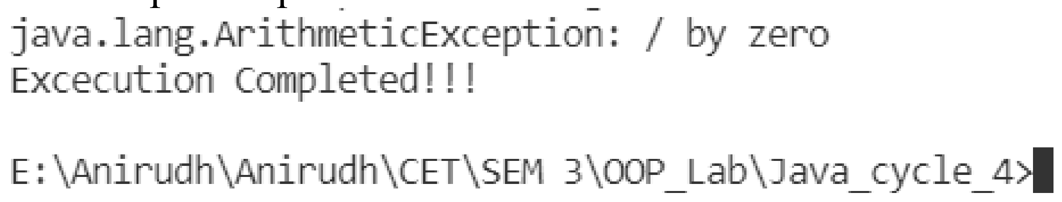
19.2 Algorithm

```
Step - 1   Start
Step - 2   Create the first try block.
    Step - 2.1.   Store the first command-line arguments in a string.
    Step - 2.2.   Obtain the length of the string and store it in an
    integer.
    Step - 2.3.   Store the result of the operation in which 100 is
    divided by the length of the string (first command-line
    argument).
    Step - 2.4.   Print the result of the operation to the user.
Step - 3   Write the first catch block which catches
 ArithemeticException if any and displays the exception to the user.
Step - 4   Write the second catch block which catches
 ArrayIndexOutOfBoundsException if any and displays the exception to
 the user.
Step - 5   Declare the finally block which displays a message no matter
 the exception
Step - 6   Stop
```

19.3 Code

```java
public class ExceptionHandling{
    public static void main(String[] args) throws ArithmeticException{
        try {
            int a = 0;
            int b = 6;
            System.out.println(b/a);
        } catch (Exception e) {
            System.out.println(e);
        }
        finally{

            System.out.println("Execution Completed!!!");
        }
    }
}
```

19.4 Sample Output



```
java.lang.ArithmeticException: / by zero
Excecution Completed!!!

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_4>
```

20 Java program that shows how to create a user-defined exception

20.1 Aim

To Write a Java program that shows how to create a user-defined exception.

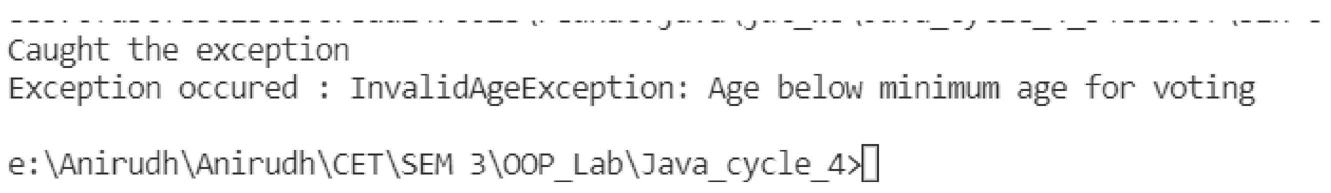
20.2 Algorithm

```
Step - 1  Start
Step - 2  Create a CustomException class that inherits from the
          Exception main class.
Step - 3  The constructor for the class accepts an argument that is
          stored in a class variable.
Step - 4  Override the toString() function inherited from the
          Throwable class.
    Step - 4.1.  Return a string that describes an error to the user
Step - 5  Create the CustomException public class.
    Step - 5.1.  Define a static method that throws a
                 CustomException error.
    Step - 5.2.  It checks for the value of the variable and
                 displays a message or throws an error as per the given
                 condition
    Step - 5.3.  Declare the public static main class.
    Step - 5.4.  It accepts a number less than 10 from the user.
    Step - 5.5.  Within a try block, call the static method defined
                 earlier that can throw the custom exception.
    Step - 5.6.  Define a catch statement that can catch that
                 custom exception and display an error message to the user.
Step - 6  Stop
```

20.3 Code

```java
class InvalidAgeException extends Exception {

  InvalidAgeException(String str) {
    super(str);
  }
}

public class UserException {

  public static void validate(int age) throws InvalidAgeException {
    if (age < 18) {
      throw new InvalidAgeException("Age below minimum age for voting");
    } else {
      System.out.println("Please cast your vote");
    }
  }

  public static void main(String[] args) {
    try {
      validate(13);
    } catch (Exception e) {
      System.out.println("Caught the exception");
      System.out.println("Exception occured : " + e);
    }
  }
}
```

20.4 Sample Output



```
Caught the exception
Exception occured : InvalidAgeException: Age below minimum age for voting

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_4>
```

21 Java Program that implements a multi-threaded program

21.1 Aim

To Write a Java program that implements a multi-threaded program which has three threads. First thread generates a random integer every 1 second. If the value is even, second thread computes the square of the number and prints. If the value is odd the third thread will print the value of cube of the number.

21.2 Algorithm

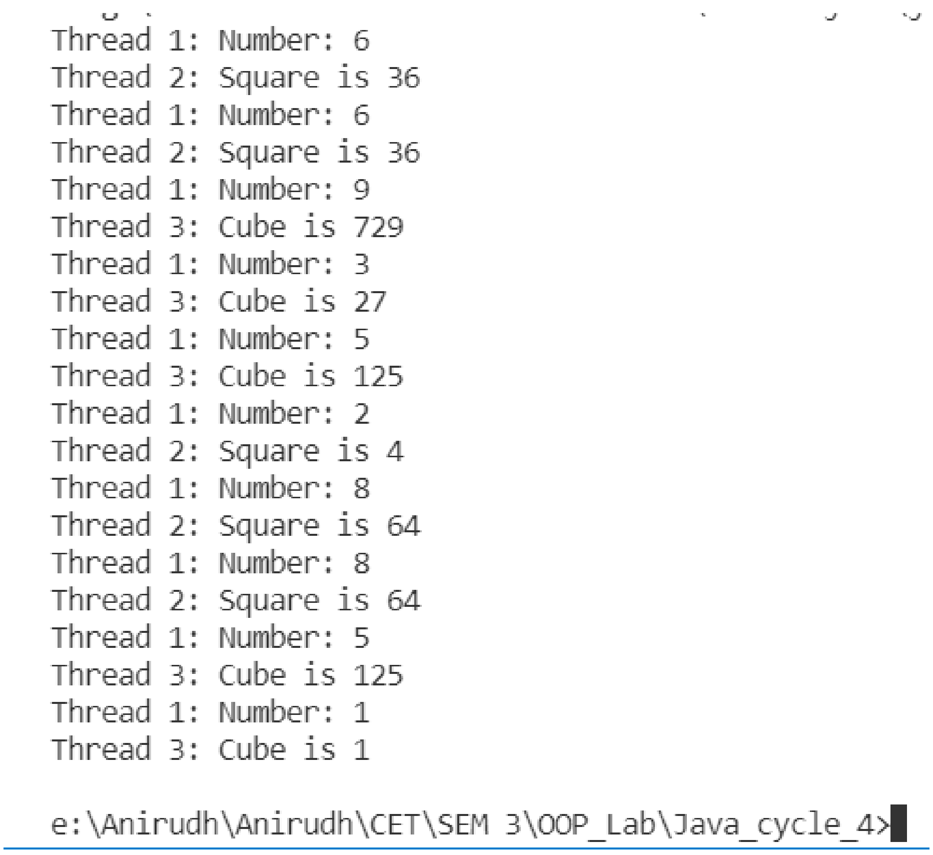
```
Step - 1  Start
Step - 2  Create a class Square that implements Runnable.
    Step - 2.1.   Define a thread, string and a integer variable.
    Step - 2.2.   Accept the integer, and the thread name in the
                  constructor.
    Step - 2.3.   Store the integer and thread name in the class
                  variables and initialise the thread.
    Step - 2.4.   Declare the run() function.
        Step - 2.4.1.   Print the name of the thread and the square
                        of the integer.
        Step - 2.4.2.   Sleep the thread for 1 second.
        Step - 2.4.3.   Catch any exception if it occurs.
Step - 3  Create a class Cube that also implements Runnable.
    Step - 3.1.   Define a thread, string and a integer variable.
    Step - 3.2.   Accept the integer, and the thread name in the
                  constructor.
    Step - 3.3.   Store the integer and thread name in the class
                  variables and initialise the thread.
    Step - 3.4.   Declare the run() function.
        Step - 3.4.1.   Print the name of the thread and the cube
                        of the integer.
        Step - 3.4.2.   ii. Sleep the thread for 1 second.
        Step - 3.4.3.   iii. Catch any exception if it occurs.
Step - 4  Create a class Generates that also implements Runnable.
```

```
        Step - 4.1.   Define a thread and a string.
        Step - 4.2.   Accept the thread name as an argument in the
        constructor.
        Step - 4.3.   Store the thread name in the class variables and
        initialise the thread.
        Step - 4.4.   Declare the run() function.
            Step - 4.4.1.   In the run() function declare an object of
            Random().
            Step - 4.4.2.   Create a loop that loops ten times
                Step - 4.4.2.1.   In each iteration, generate a random
                integer in the range of 0 to 10.
                Step - 4.4.2.2.   If the generated number is divisible
                by 2, create an object of the Square class and thus create
                a new thread.
                Step - 4.4.2.3.   Else, create an object of the Cube
                class and thus create a new thread.
                Step - 4.4.2.4.   Sleep the thread for 1 second.
            Step - 4.4.3.   Catch any exception if it occurs.
    Step - 5  Declare the main function and create an object of the
    Generate class.
    Step - 6  Stop
```

21.3 Code

```java
import java.util.Random;

class Square implements Runnable {
    Thread t;
    String thread;
    int num;

    Square(int num, String thr) {
        thread = thr;
        this.num = num;
        t = new Thread(this, thread);
        t.start();
    }

    public void run() {
        try {
            System.out.println(thread + ": Square is " + num * num);
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Exception Occurred!! " + e);
        }
    }
}

class Cube implements Runnable {
    Thread t;
    String thread;
    int num;

    Cube(int num, String thr) {
        this.num = num;
        thread = thr;
        t = new Thread(this, thread);
        t.start();
    }

    public void run() {
        try {
            System.out.println(thread + ": Cube is " + num * num * num);
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Exception Occurred!! " + e);
        }
    }
}

class Generate implements Runnable {
    Thread t;
    String thread;

    Generate(String thr) {
        thread = thr;
        t = new Thread(this, thread);
        t.start();
    }

    public void run() {
        Random rd = new Random();
        for (int i = 0; i < 10; i++) {
            try {
                int num = rd.nextInt(10);
                System.out.println(thread + ": Number: " + num);
                if (num % 2 == 0) {
                    Square sq = new Square(num, "Thread 2");
                } else {
                    Cube cb = new Cube(num, "Thread 3");
                }
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                System.out.println("Exception Occurred!! " + e);
            }
        }
    }
}

public class Even_oddThread {
    public static void main(String args[]) {
        Generate g1 = new Generate("Thread 1");
    }
}
```

21.4 Sample Output



```
Thread 1: Number: 6
Thread 2: Square is 36
Thread 1: Number: 6
Thread 2: Square is 36
Thread 1: Number: 9
Thread 3: Cube is 729
Thread 1: Number: 3
Thread 3: Cube is 27
Thread 1: Number: 5
Thread 3: Cube is 125
Thread 1: Number: 2
Thread 2: Square is 4
Thread 1: Number: 8
Thread 2: Square is 64
Thread 1: Number: 8
Thread 2: Square is 64
Thread 1: Number: 5
Thread 3: Cube is 125
Thread 1: Number: 1
Thread 3: Cube is 1

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_4>
```

22 Java program that shows thread Synchronization

22.1 Aim

To write a Java program that shows thread synchronization.

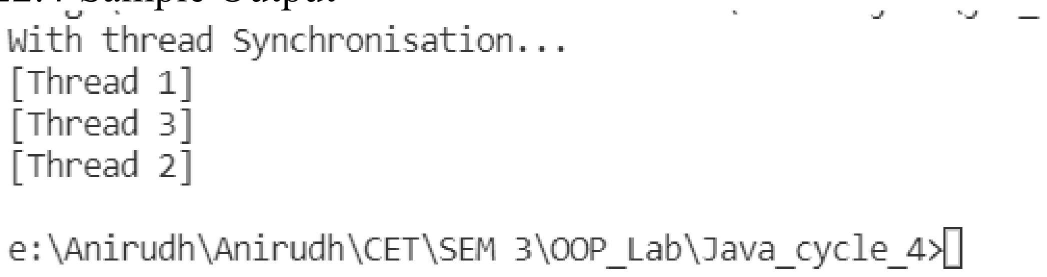
22.2 Algorithm

```
Step - 1  Start
Step - 2  Create a class main.
        Step - 2.1.  Declare a method inside it that prints the first
        square bracket and sleeps the thread.
        Step - 2.2.  Catch Exceptions if any.
        Step - 2.3.  Print the thread name and the final square
        bracket.
Step - 3  Create the class Synchronisation which implements
 Runnable
Step - 4  Declare a thread, a string, an Object of the Main class, and
 an integer flag.
Step - 5  Accept the thread name, integer value, and the object of
 Main as the constructor arguments.
        Step - 5.1.  Assign the integer value to the class integer
        variable and the object to the class object variable.
        Step - 5.2.  Initialise the thread
Step - 6  Declare the run() function which class the method of the
 Main class in the synchronized block.
Step - 7  In the main function, Create 3 class variables. This should
 start the threads and the synchronized messages are printed
 thereafter.
Step - 8  Stop
```

22.3 Code

```
class Main {
    void call(String thread) {
        System.out.printf("[");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Exception Occurred!! " + e);
        }
        System.out.printf("%s", thread);
        System.out.println("]");
    }
}

class Synchronisation implements Runnable {
    String thread;
    Thread t;
    Main m1;
    int flag;
    Synchronisation(String thr, int flag, Main m) {
        thread = thr;
        m1 = m;
        this.flag = flag;
        t = new Thread(this, thread);
        t.start();
    }

    public void run() {
        // Main m1 = new Main();
        if (flag == 0) {
            m1.call(thread);
        } else {
            synchronized (m1) {
                m1.call(thread);
            }
        }
    }
}

public class Sync {
    public static void main(String args[]) {
        Main m = new Main();
        System.out.println("With thread Synchronisation...");
        Synchronisation s4 = new Synchronisation("Thread 1", 1, m);
        Synchronisation s5 = new Synchronisation("Thread 2", 1, m);
        Synchronisation s6 = new Synchronisation("Thread 3", 1, m);
    }
}
```

22.4 Sample Output



```
With thread Synchronisation...
[Thread 1]
[Thread 3]
[Thread 2]

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_4>
```

23 Java Program to Create Two Threads

23.1 Aim

To write a Java program to create two threads: One for displaying all odd numbers between 1 and 100 and the second thread for displaying all even numbers between 1 and 100.

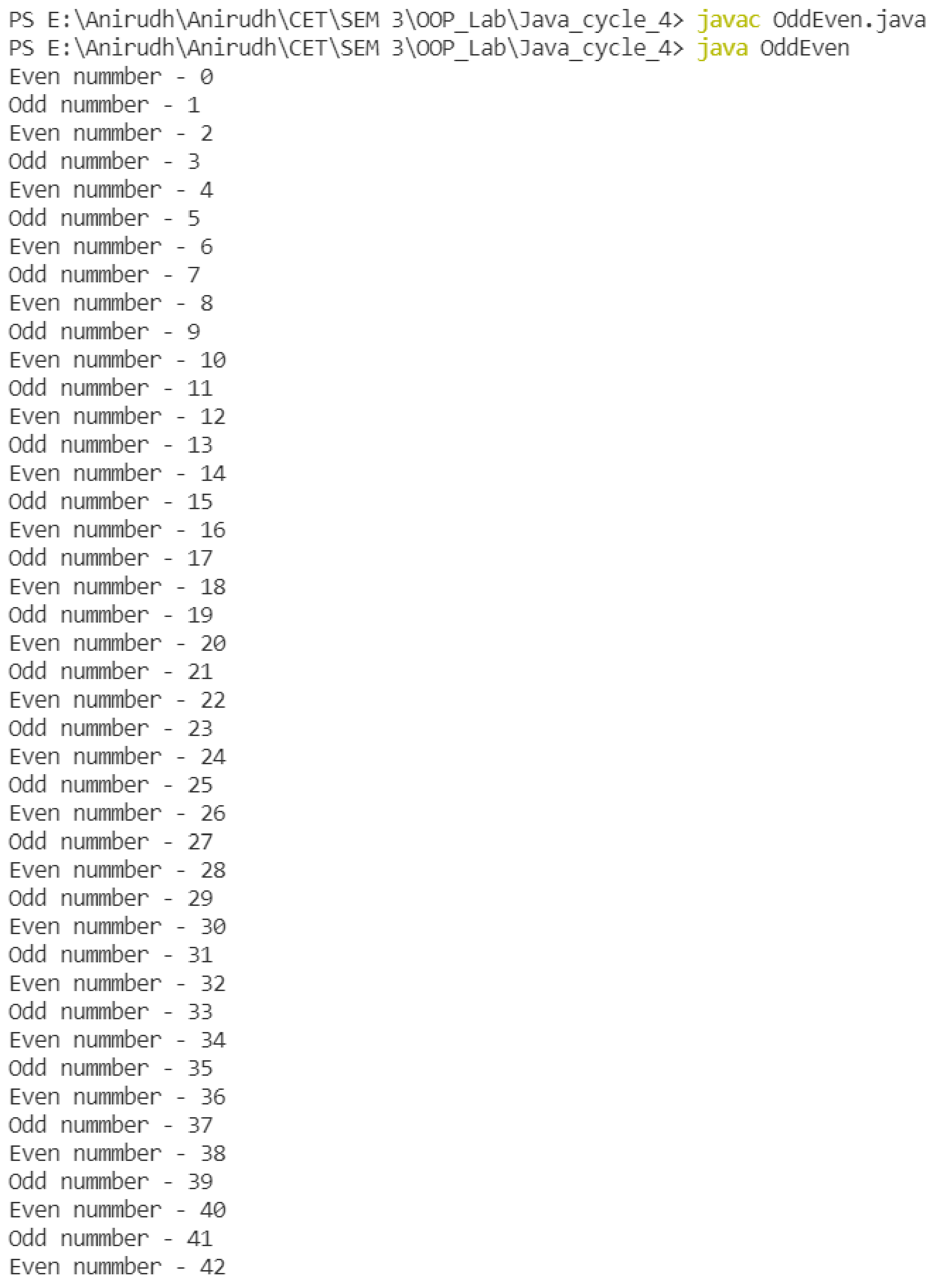
23.2 Algorithm

```
Step - 1  Start
Step - 2  Create a class main.
    Step - 2.1.  Declare a method call inside it that prints the thread name and an integer as arguments
    Step - 2.2.  Display the integer and the thread
Step - 3  Create the class Synchronisation which implements Runnable
Step - 4  Declare a thread, a string, an Object of the Main class, and an integer n.
Step - 5  Accept the thread name, integer value, and the object of Main as the constructor arguments.
    Step - 5.1.  Assign the integer value to the class integer variable and the object to the class object variable.
    Step - 5.2.  Initialise the thread
Step - 6  Declare the run() function which class the method of the Main class in the synchronized block.
Step - 7  In the main function, Create an object of the Main class.
Step - 8  Loop through numbers from 1 to 100.
Step - 9  If the number is divisible by 2, Create an object of the Synchronisation class and pass a thread name, an object of Main, and an integer as arguments. Else, Create an object of the Synchronization class and pass another thread name, the object of Main, and an integer as arguments.
Step - 10  Wait till the thread execution completes with the thread.join() method.
Step - 11  Catch exceptions if any.
Step - 12  Stop
```

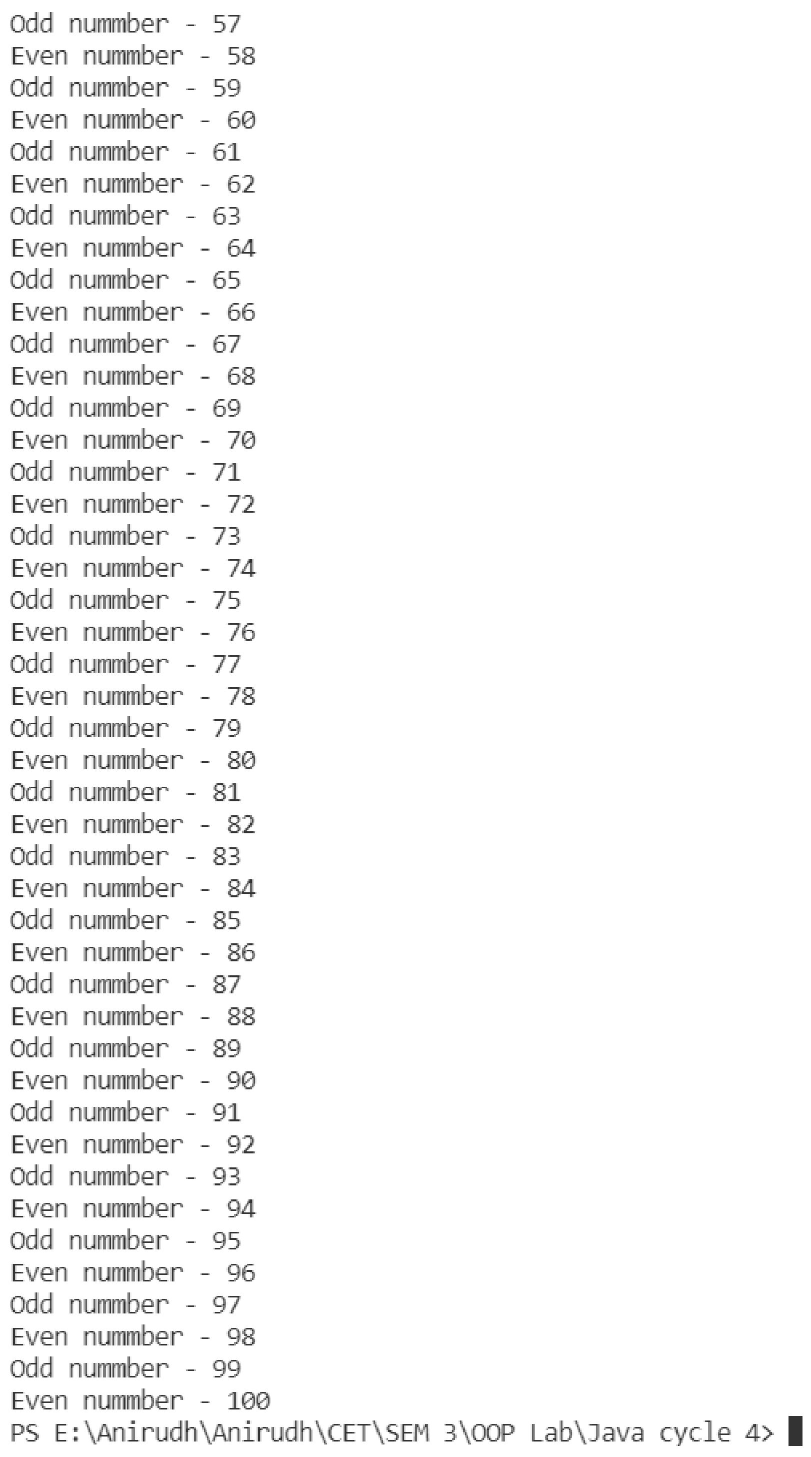
23.3 Code

```java
class Thread01 implements Runnable  {

    @Override
    public void run() {
        try {
            for (int i = 0; i <= 100; i++) {
                if (i%2==0){
                    System.out.println("Even nummber - "+i);
                }
                Thread.sleep(100);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
class Thread02 implements Runnable {
    @Override
    public void run() {
        try {

            for (int i = 0; i <= 100; i++) {
                if (i%2!=0){
                    System.out.println("Odd nummber - "+i);
                }
                Thread.sleep(100);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}

public class OddEven {
    public static void main(String[] args) {
        Thread01 obj1 = new Thread01();
        Thread t1 = new Thread(obj1);

        Thread02 obj2 = new Thread02();
        Thread t2 = new Thread(obj2);

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

23.4 Sample Output



```
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_4> javac OddEven.java
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_4> java OddEven
Even nummber - 0
Odd nummber - 1
Even nummber - 2
Odd nummber - 3
Even nummber - 4
Odd nummber - 5
Even nummber - 6
Odd nummber - 7
Even nummber - 8
Odd nummber - 9
Even nummber - 10
Odd nummber - 11
Even nummber - 12
Odd nummber - 13
Even nummber - 14
Odd nummber - 15
Even nummber - 16
Odd nummber - 17
Even nummber - 18
Odd nummber - 19
Even nummber - 20
Odd nummber - 21
Even nummber - 22
Odd nummber - 23
Even nummber - 24
Odd nummber - 25
Even nummber - 26
Odd nummber - 27
Even nummber - 28
Odd nummber - 29
Even nummber - 30
Odd nummber - 31
Even nummber - 32
Odd nummber - 33
Even nummber - 34
Odd nummber - 35
Even nummber - 36
Odd nummber - 37
Even nummber - 38
Odd nummber - 39
Even nummber - 40
Odd nummber - 41
Even nummber - 42
```



```
Odd nummber - 57
Even nummber - 58
Odd nummber - 59
Even nummber - 60
Odd nummber - 61
Even nummber - 62
Odd nummber - 63
Even nummber - 64
Odd nummber - 65
Even nummber - 66
Odd nummber - 67
Even nummber - 68
Odd nummber - 69
Even nummber - 70
Odd nummber - 71
Even nummber - 72
Odd nummber - 73
Even nummber - 74
Odd nummber - 75
Even nummber - 76
Odd nummber - 77
Even nummber - 78
Odd nummber - 79
Even nummber - 80
Odd nummber - 81
Even nummber - 82
Odd nummber - 83
Even nummber - 84
Odd nummber - 85
Even nummber - 86
Odd nummber - 87
Even nummber - 88
Odd nummber - 89
Even nummber - 90
Odd nummber - 91
Even nummber - 92
Odd nummber - 93
Even nummber - 94
Odd nummber - 95
Even nummber - 96
Odd nummber - 97
Even nummber - 98
Odd nummber - 99
Even nummber - 100
PS E:\Anirudh\Anirudh\CET\SEM 3\OOP Lab\Java cycle 4>
```

24 Java program that shows thread priorities

24.1 Aim

To Write a Java program that shows thread priorities.

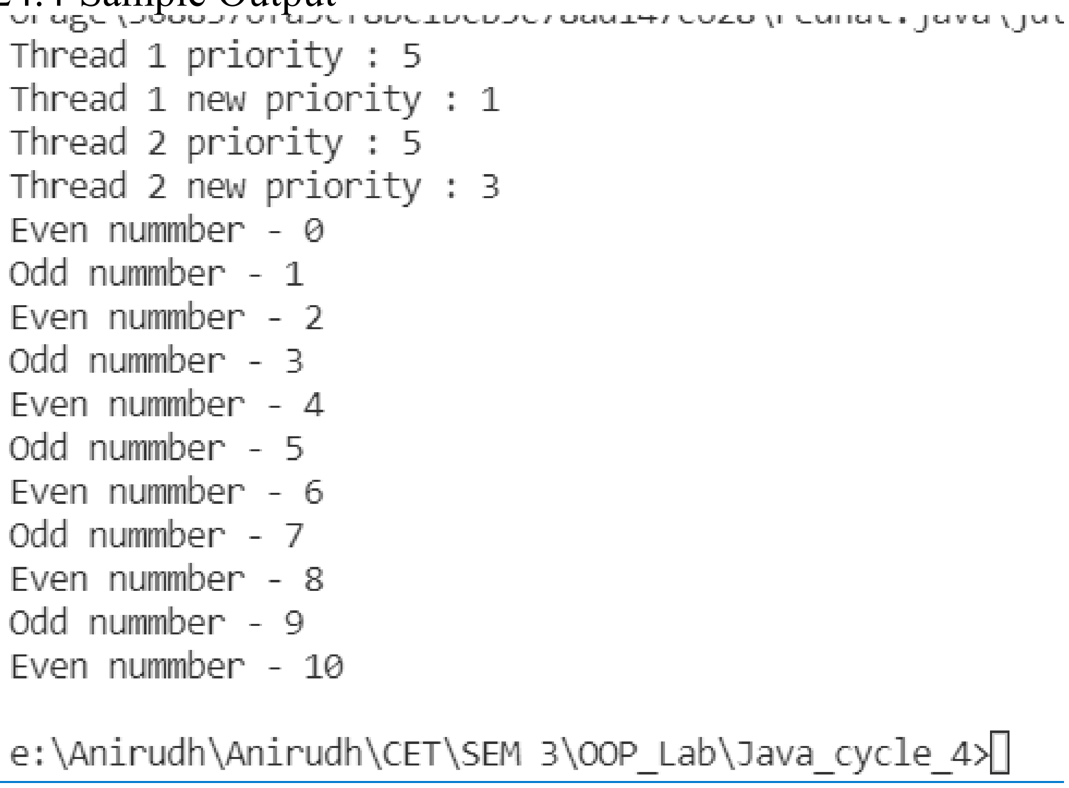
24.2 Algorithm

```
Step - 1  Start
Step - 2  Create a class main.
Step - 3  Declare a method call inside it that accepts a thread name, a message, and an integer as arguments
Step - 4  Display the thread, message, and priority.
Step - 5  Create the class Synchronisation which implements Runnable
Step - 6  Declare a thread, a string, an Object of the Main class, and an integer n.
Step - 7  Accept the thread name, integer value, and the object of Main as the constructor arguments.
Step - 8  Assign the integer value to the class integer variable and the object to the class object variable and set the priority of the thread as the value of the integer argument.
Step - 9  Initialise the thread
Step - 10  Declare the run() function.
    Step - 10.1.  Call the call method of the Main class object twice. Pass the thread name, a string message, and the priority value as the arguments each time.
    Step - 10.2.  Sleep the thread for one second.
    Step - 10.3.  Catch Exceptions if any.
    Step - 10.4.  Call the call method of the Main class object once more. Pass the thread name, a string message, and the priority value as the arguments each time.
Step - 11  In the main function, Create 3 class variables. This should start the threads and the synchronized messages are printed thereafter.
Step - 12  Stop
```

24.3 Code

```java
class T1 implements Runnable {

    @Override
    public void run() {
        try {
            for (int i = 0; i <= 10; i++) {
                Thread.sleep(1000);
                if (i%2==0){
                    System.out.println("Even number - "+i);
                }

            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

class T2 implements Runnable {

    @Override
    public void run() {
        try {

            for (int i = 0; i <= 10; i++) {
                Thread.sleep(1000);
                if (i % 2 != 0) {
                    System.out.println("Odd number - " + i);
                }

            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}

public class Priority {
    public static void main(String[] args) {
        // T1 obj1 = new T1();
        Thread t1 = new Thread(new T1());

        // T2 obj2 = new T2();
        Thread t2 = new Thread(new T2());
        t1.start();

        System.out.println("Thread 1 priority : " + t1.getPriority());
        t1.setPriority(1);
        System.out.println("Thread 1 new priority : " + t1.getPriority());
        t2.start();
        System.out.println("Thread 2 priority : " + t2.getPriority());
        t2.setPriority(3);
        System.out.println("Thread 2 new priority : " + t2.getPriority());
    }
}
```

24.4 Sample Output



```
Thread 1 priority : 5
Thread 1 new priority : 1
Thread 2 priority : 5
Thread 2 new priority : 3
Even nummber - 0
Odd nummber - 1
Even nummber - 2
Odd nummber - 3
Even nummber - 4
Odd nummber - 5
Even nummber - 6
Odd nummber - 7
Even nummber - 8
Odd nummber - 9
Even nummber - 10

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_4>
```

Date: 10/03/2022

CYCLE 5

25 Java program that allows the user to draw lines, rectangles, and ovals.

25.1 Aim

To Write a Java program that allows the user to draw lines, rectangles, and ovals.

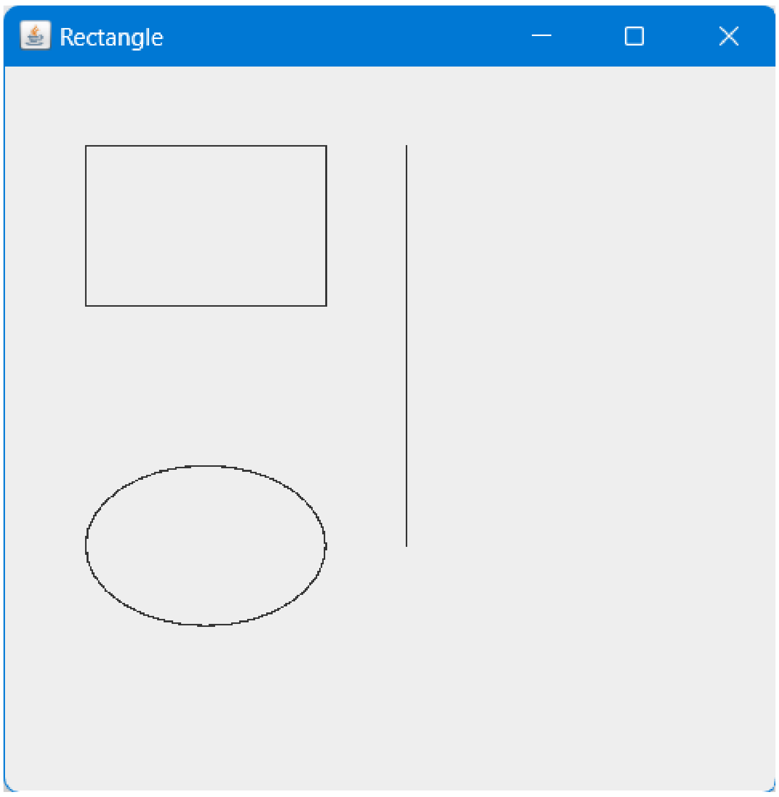
25.2 Algorithm

```
Step - 1   Start
Step - 2   Create a ‘mycanvas’ class extending JComponent, in which
           define a paint method with a Graphics as argument.
Step - 3   Inside the paint method, initialise Rectangles, Lines, and Ovals
           of required size.
Step - 4   Create a ‘Rectangle’ class as the main class that extends JFrame.
Step - 5   Create an instance of JFrame and set appropriate size and
           properties for it.
Step - 6   Add an anonymous object of ‘mycanvas’ class to the Frame.
Step - 7   Stop
```

25.3 Code

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

class mycanvas extends JComponent {
    public void paint(Graphics g){
        g.drawRect(40, 40, 120, 80);
        g.drawOval(40, 200, 120, 80);
        g.drawLine(200, 40, 200, 240);
    }

}

public class Rectangle extends JFrame {
    public static void main(String[] args) {
        JFrame F = new JFrame("Rectangle");
        F.setDefaultCloseOperation(EXIT_ON_CLOSE);
        F.setSize(400, 400);
        F.add(new mycanvas());
        F.setVisible(true);
    }
}
```

25.4 Sample Output



26 **Java program for handling mouse events**

26.1 Aim

To Write a Java program for handling mouse events.

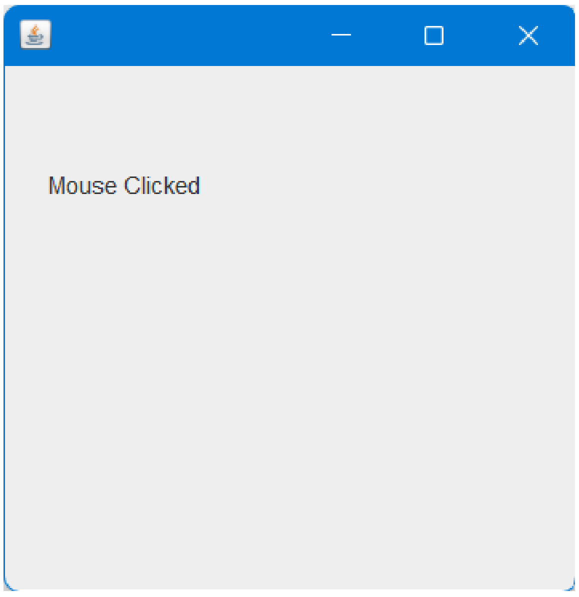
26.2 Algorithm

Step - 1 Start
Step - 2 Create a class 'Mouseevent' that inherits JFrame and implements MouseListener.
Step - 3 Create an instance of JFrame and set appropriate size and properties for it.
Step - 4 Add a JLabel 'l' to the frame. Add mouse listener to the frame.
Step - 5 If mouse enters the frame print 'Mouse entered' in the label.
Step - 6 If mouse button is pressed, print 'Mouse pressed' in the label.
Step - 7 If mouse button is released, print 'Mouse released' in the label
Step - 8 If mouse leaves the frame print 'Mouse exited' in the label
Step - 9 Stop

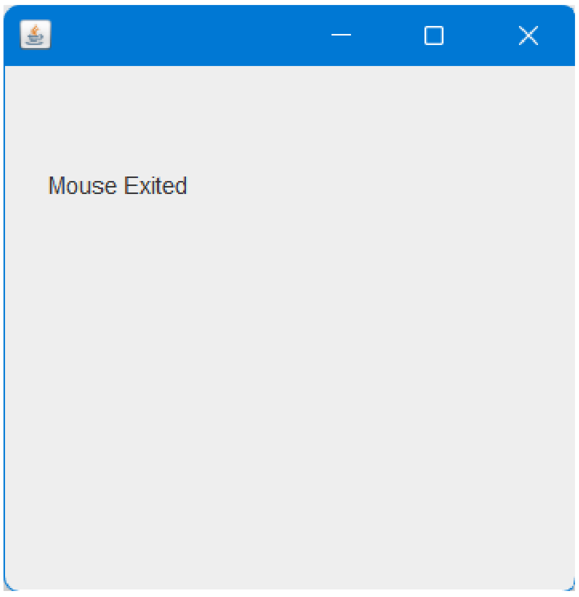
26.3 Code

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

public class Mouseevent extends JFrame implements MouseListener {
    Label l;

    Mouseevent() {
        addMouseListener(this);

        l = new Label();
        l.setBounds(20, 50, 100, 20);
        add(l);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 300);
        setLayout(null);
        setVisible(true);
    }

    public void mouseClicked(MouseEvent e) {
        l.setText("Mouse Clicked");
    }

    public void mouseEntered(MouseEvent e) {
        l.setText("Mouse Entered");
    }

    public void mouseExited(MouseEvent e) {
        l.setText("Mouse Exited");
    }

    public void mousePressed(MouseEvent e) {
        l.setText("Mouse Pressed");
    }

    public void mouseReleased(MouseEvent e) {
        l.setText("Mouse Released");
    }

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

26.4 Sample Output



```
Mouse Clicked
```



```
Mouse Exited
```

27 **Java program for handling key events using Adapter classes.**

27.1 Aim

To Write a Java program for handling key events using Adapter classes.

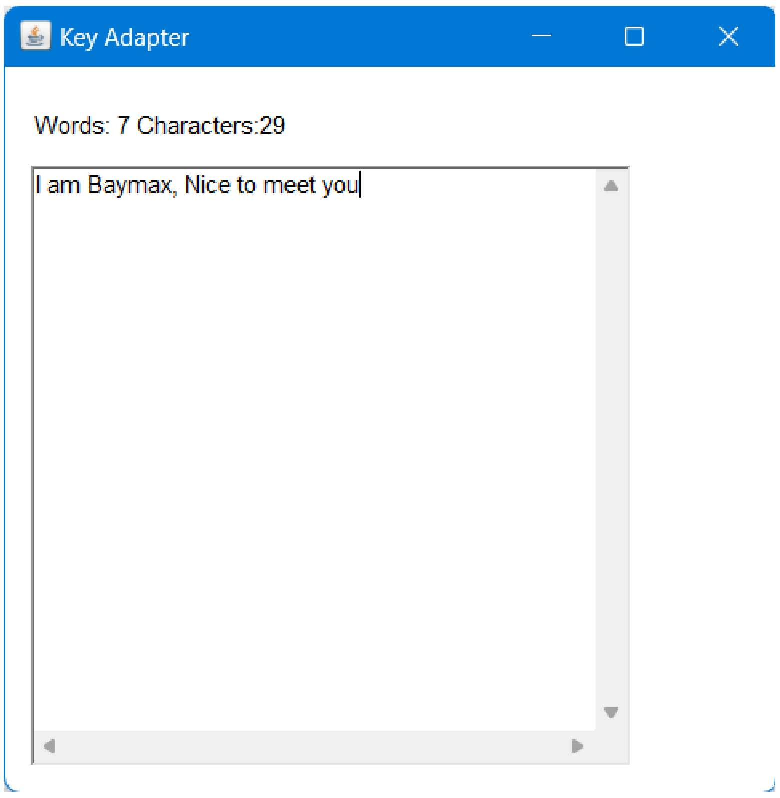
27.2 Algorithm

```
Step - 1   Start
Step - 2   Create a main class ‘Adapter’ that extends KeyAdapter
Step - 3   Create an instance of a Frame and set appropriate size and
 properties
Step - 4   Add a text are to the frame object
Step - 5   In the KeyReleased method, define the function for counting no.
 of words and no. of characters and set them to the Label.
Step - 6   Add Label to the Frame object.
Step - 7   Stop
```

27.3 Code

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

public class Adapter extends KeyAdapter {
    Label l;
    TextArea area;
    Frame f;

    Adapter() {
        f = new Frame("Key Adapter");
        l = new Label();
        l.setBounds(20, 50, 200, 20);
        area = new TextArea();
        area.setBounds(20, 80, 300, 300);
        area.addKeyListener(this);

        f.add(l);
        f.add(area);
        f.setSize(400, 400);

        f.addWindowListener (new WindowAdapter() {
            public void windowClosing (WindowEvent e) {
                f.dispose();
            }
        });

        f.setLayout(null);
        f.setVisible(true);
    }

    public void keyReleased(KeyEvent e) {
        String text = area.getText();
        String words[] = text.split(" ");
        l.setText("Words: " + words.length + " Characters:" + text.length());
    }

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

27.4 Sample Output



```
Words: 7 Characters:29
I am Baymax, Nice to meet you
```

28 Java Swing program to print a wave form on the output screen

28.1 Aim

To Write a Java Swing program to print a wave form on the output screen

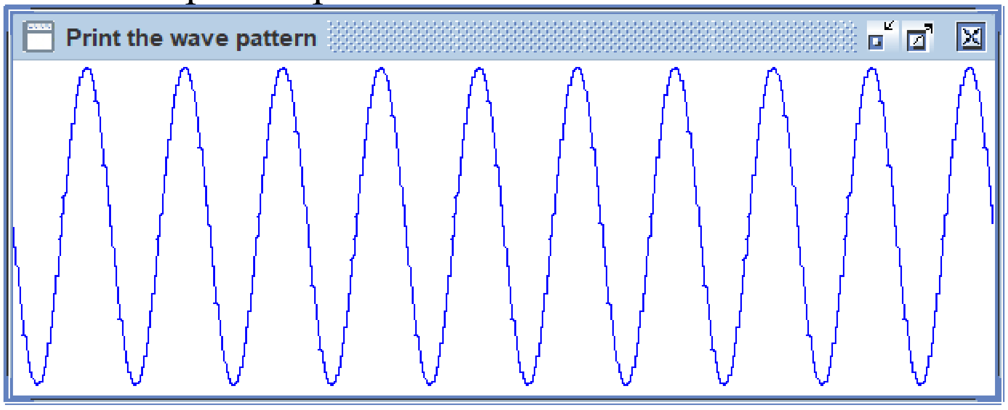
28.2 Algorithm

```
Step - 1  Start
Step - 2  Create a main class ‘WavePrint’ that extends JPanel
Step - 3  Declare necessary data members for waveform building.
Step - 4  Define a method ‘setCycles’ which assigns values for the
          data members of the class and sets the no. of cycles.
Step - 5  In the public method paintComponent, define the function
          for forming the waveform by passing a Graphics argument.
Step - 6  Create a JFrame instance and add the JPanel to it.
Step - 7  Set appropriate size and properties for the frame instance.
Step - 8  Stop
```

28.3 Code

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

public class WavePrint extends JPanel
{
    int SCALEFACTOR = 200;
    int cycles;
    int points;
    double[] sines;
    int[] pts;

    public  void setCycles(int cycles)
    {
        this.cycles = cycles;
        this.points = SCALEFACTOR * cycles * 2;
        this.sines = new double[points];
        for (int i = 0; i < points; i++)
        {
            double radians = (Math.PI / SCALEFACTOR) * i;
            this.sines[i] = Math.sin(radians);
        }
    }
    public void paintComponent(Graphics g)
    {
        int maxWidth = getWidth();
        double hstep = (double) maxWidth / (double) points;
        int maxHeight = getHeight();
        pts = new int[points];
        for (int i = 0; i < points; i++)
        {
            pts[i] = (int) (sines[i] * maxHeight / 2 * .95 + maxHeight / 2);
        }
        g.setColor(Color.BLUE);
        for (int i = 1; i < points; i++)
        {
            int x1 = (int) ((i - 1) * hstep);
            int x2 = (int) (i * hstep);
            int y1 = pts[i - 1];
            int y2 = pts[i];
            g.drawLine(x1, y1, x2, y2);
        }
    }

    public static void main(String[] args)
    {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame("Print the wave pattern");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setBackground(Color.white);
        frame.setSize(500, 200);

        WavePrint sw = new WavePrint();
        sw.setCycles(10);
        frame.add(sw);
        frame.setVisible(true);
    }
}
```

28.4 Sample Output



29 Java Program that works as a Simple Calculator

29.1 Aim

To Write a Java program that works as a simple calculator. Arrange Buttons for digits and the + - * % operations properly. Add a text field to display the result. Handle any possible exceptions like divide by zero. Use Java Swing

29.2 Algorithm

```
Step - 1  START
Step - 2  Create a java frame inside the constructor by extending the
  frame class and create button b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, bp,
  bs, bm, bd, be, bc as buttons in the calculator and a textfield txtf for
  display
Step - 3  Let str1 and str2 be two empty strings and let f be a Boolean
  value initialized to true
Step - 4  Register the listeners of each button with the event source
Step - 5  In the event handling method, if buttons corresponding to any
  of the numbers is presses and if f=true add the number corresponding to
  the button to str1 else add it to str2
Step - 6  If button corresponding to any operator is pressed, set f=false
  and let the operator be stored in op as a character
Step - 7  If button be is pressed, convert str1 and str2 to integers n1 and
  n2 and do the suitable operation using the operator stored in op and print
  it to the textfield and set str1 as the current answer, set f=true and set str2
  to empty string
Step - 8  Enclose the operation in a try block and catch the exceptions
  and handle it suitably
Step - 9  If button bc is pressed, empty the textfield and set str1 and str2
  to empty string and f to true
Step - 10 Call the constructor of the class in the main method
Step - 11 STOP
```

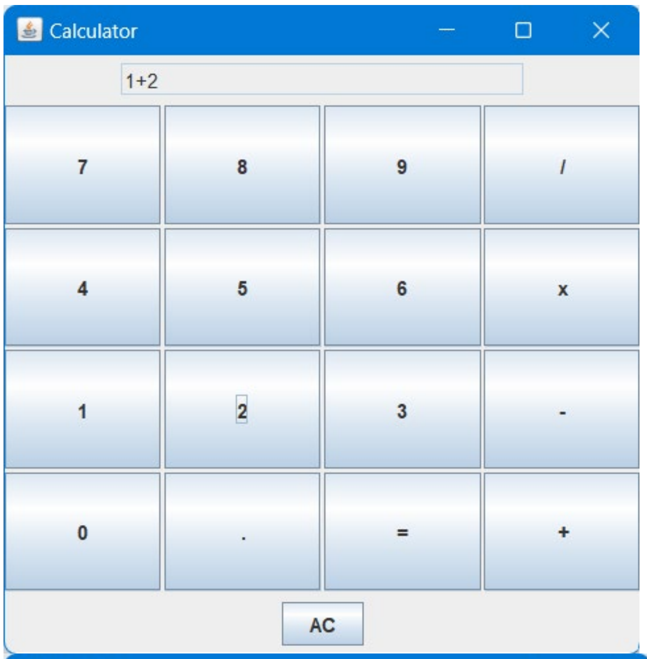
29.3 Code

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

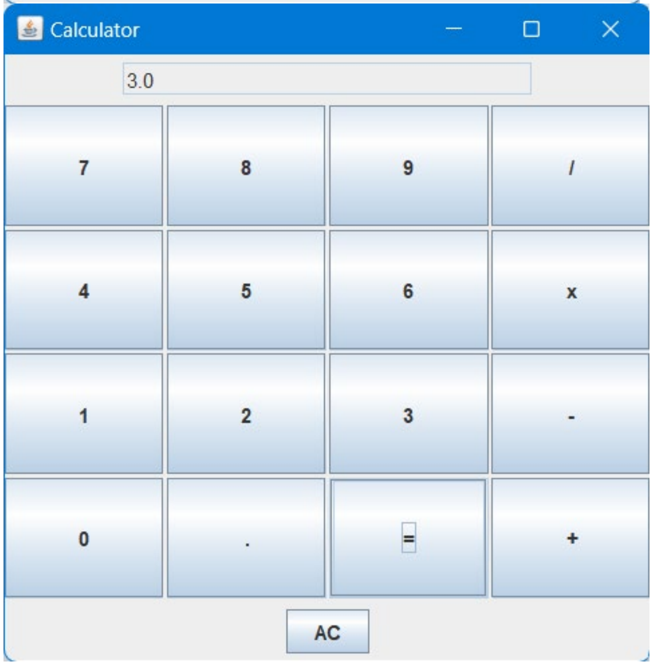
```java
public class Calculator extends JFrame implements ActionListener {
    static JFrame f;
    static JButton b1, b2, b3, b4, b5, b6, b7, b8, b9, b0, dot;
    static JButton plus, minus, mult, divide, clear, equals;
    static JTextField t1;
    static String s0, s1, s2;

    Calculator() {
        s1 = s2 = s0 = "";
    }

    public static void main(String[] args) {
        Calculator c = new Calculator();

        JPanel p1 = new JPanel();
        JPanel p2 = new JPanel();
        JPanel p3 = new JPanel();

        f = new JFrame("Calculator");

        b1 = new JButton("1");
        b2 = new JButton("2");
        b3 = new JButton("3");
        b4 = new JButton("4");
        b5 = new JButton("5");
        b6 = new JButton("6");
        b7 = new JButton("7");
        b8 = new JButton("8");
        b9 = new JButton("9");
        b0 = new JButton("0");
        dot = new JButton(".");
        plus = new JButton("+");
        minus = new JButton("-");
        mult = new JButton("x");
        divide = new JButton("/");
        equals = new JButton("=");
        clear = new JButton("AC");

        t1 = new JTextField(24);
        t1.setEditable(false);

        b1.addActionListener(c);
        b2.addActionListener(c);
        b3.addActionListener(c);
        b4.addActionListener(c);
        b5.addActionListener(c);
        b6.addActionListener(c);
        b7.addActionListener(c);
        b8.addActionListener(c);
        b9.addActionListener(c);
        b0.addActionListener(c);
        dot.addActionListener(c);
        plus.addActionListener(c);
        minus.addActionListener(c);
        mult.addActionListener(c);
        divide.addActionListener(c);
        equals.addActionListener(c);
        clear.addActionListener(c);

        p1.add(t1);

        p2.add(b7);
        p2.add(b8);
        p2.add(b9);
        p2.add(divide);

        p2.add(b4);
        p2.add(b5);
        p2.add(b6);
        p2.add(mult);

        p2.add(b1);
        p2.add(b2);
        p2.add(b3);
        p2.add(minus);

        p2.add(b0);
        p2.add(dot);
        p2.add(equals);
        p2.add(plus);

        p3.add(clear);

        p2.setLayout(new GridLayout(4, 4, 2, 2));
        // p1.setBounds(20, 10, 350, 30);
        f.add(p1, BorderLayout.NORTH);
        f.add(p2);
        f.add(p3, BorderLayout.SOUTH);

        f.setDefaultCloseOperation(EXIT_ON_CLOSE);
        f.setSize(400, 400);
        f.setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        String s = e.getActionCommand();
        if ((s.charAt(0) >= '0' && s.charAt(0) <= '9') || s.charAt(0) == '.')
    {

            if (!s1.equals(""))
                s2 = s2 + s;
            else
                s0 = s0 + s;

            t1.setText(s0 + s1 + s2);

        } else if (s.equals("AC")) {

            s0 = s1 = s2 = "";
            t1.setText(s0 + s1 + s2);

        } else if (s.charAt(0) == '=') {

            double result;

            if (s1.equals("+"))
                result = Double.parseDouble(s0) + Double.parseDouble(s2);
            else if (s1.equals("-"))
                result = Double.parseDouble(s0) - Double.parseDouble(s2);
            else if (s1.equals("x"))
                result = Double.parseDouble(s0) * Double.parseDouble(s2);
            else
                result = Double.parseDouble(s0) / Double.parseDouble(s2);

            s0 = Double.toString(result);
            t1.setText(s0);
            s1 = s2 = "";
        }
        else {
            if (s1.equals("") || s2.equals("")) {
                s1 = s;
            } else {
                double result;

            if (s1.equals("+"))
                result = Double.parseDouble(s0) + Double.parseDouble(s2);
            else if (s1.equals("-"))
                result = Double.parseDouble(s0) - Double.parseDouble(s2);
            else if (s1.equals("x"))
                result = Double.parseDouble(s0) * Double.parseDouble(s2);
            else
                result = Double.parseDouble(s0) / Double.parseDouble(s2);

            s0 = Double.toString(result);
```

```
                s1 = s;
                s2 = "";
                }
                t1.setText(s0+s1+s2);
            }
        }
    }
```

29.4 Sample Output



1+2



3.0

30 Java Program that simulates Traffic Lights

30.1 Aim

To Write a Java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time. No light is on when the program starts.

30.2 Algorithm

```
Step - 1   START
Step - 2   Create a java frame inside the constructor and create three radio
 buttons jr1, jr2, jr3 with labels Red, Yellow, Green and let x=0, y=0, z=0
Step - 3   Register each listener with the source and add the components
 to the frame
Step - 4   In the event handling method of itemstatechange, if the jr1 is
 pressed let x=1 and call the paint method. If the jr2 is pressed let y=1 and
 call the paint method. If the jr3 is pressed let z=1 and call the paint
 method.
Step - 5   Inside the paint method, draw three circles of radius 60
Step - 6   If x=1, then set the colour of first circle to red and the other to
 white and set x=0
Step - 7   If y=1, then set the colour of second circle to yellow and the
 other to white and set y=0
Step - 8   If z=1, then set the colour of third circle to green and the other
 to white and set z=0
Step - 9   STOP
```

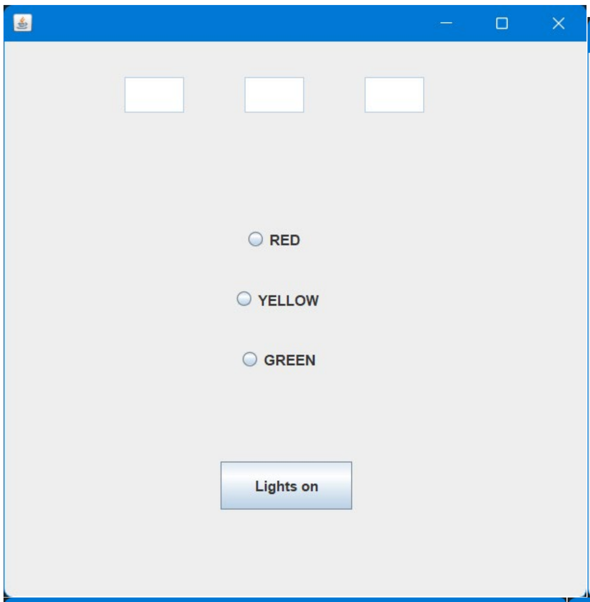
30.3 Code

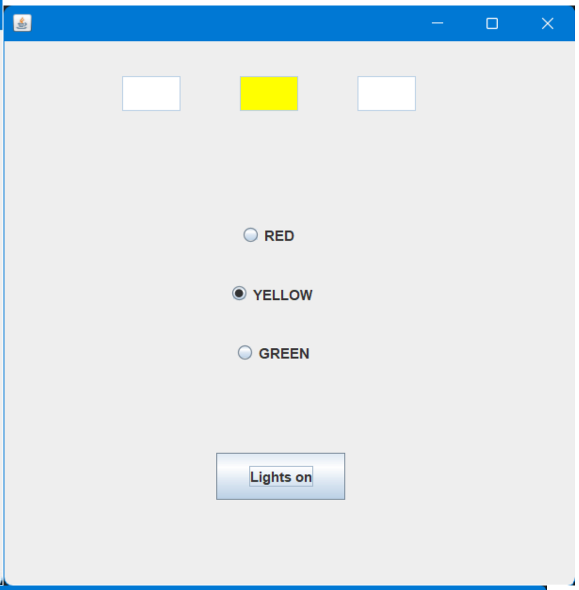
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Traffic extends JFrame implements ActionListener {
    static JRadioButton red, yellow, green;
    static JTextField R, Y, G;

    Traffic() {
    }
```

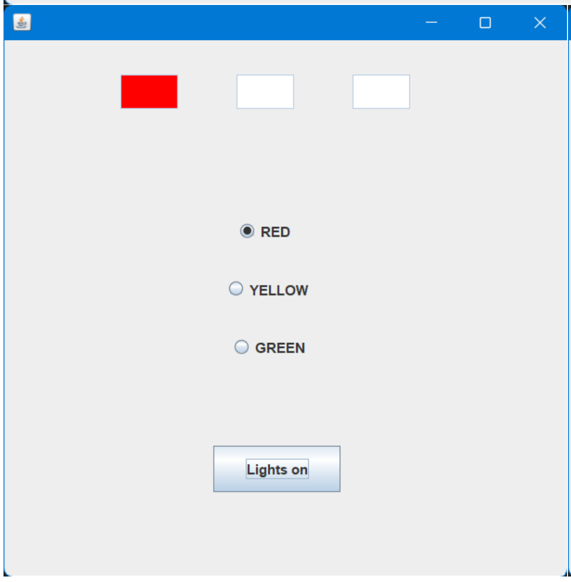
```
public void actionPerformed(ActionEvent e) {
    // String s = e.getActionCommand();
    if (red.isSelected()) {
        R.setBackground(Color.RED);
        Y.setBackground(Color.WHITE);
        G.setBackground(Color.WHITE);
    }
    if (yellow.isSelected()) {
        R.setBackground(Color.WHITE);
        Y.setBackground(Color.YELLOW);
        G.setBackground(Color.WHITE);
    }
    if (green.isSelected()) {
        R.setBackground(Color.WHITE);
        Y.setBackground(Color.WHITE);
        G.setBackground(Color.GREEN);
    }
}



public static void main(String[] args) {
    Traffic trf = new Traffic();

    JFrame f = new JFrame();
    f.setDefaultCloseOperation(EXIT_ON_CLOSE);
    f.setLayout(null);
    f.setSize(500, 500);
    f.setBackground(Color.WHITE);
    f.setVisible(true);

    R = new JTextField();
    Y = new JTextField();
    G = new JTextField();

    R.setEditable(false);
    Y.setEditable(false);
    G.setEditable(false);

    R.setBounds(100, 30, 50, 30);
    Y.setBounds(200, 30, 50, 30);
    G.setBounds(300, 30, 50, 30);

    R.setBackground(Color.WHITE);
    Y.setBackground(Color.WHITE);
    G.setBackground(Color.WHITE);
```

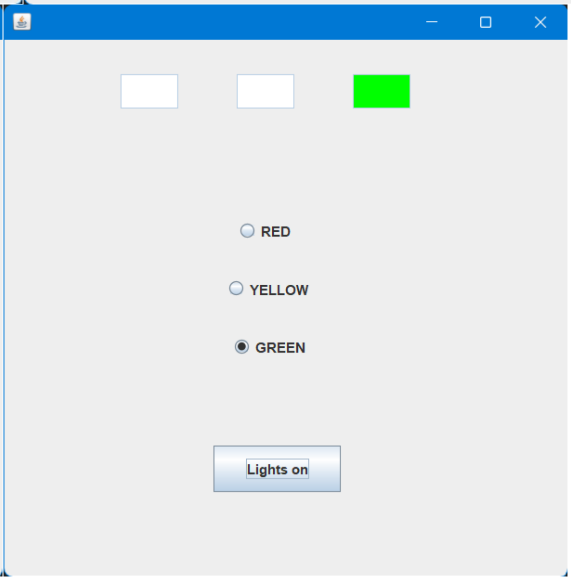
```
        red = new JRadioButton("RED");
        yellow = new JRadioButton("YELLOW");
        green = new JRadioButton("GREEN");

        red.setBounds(200, 150, 80, 30);
        yellow.setBounds(190, 200, 80, 30);
        green.setBounds(195, 250, 80, 30);

        ButtonGroup bg = new ButtonGroup();

        bg.add(red);
        bg.add(yellow);
        bg.add(green);

        red.addActionListener(trf);
        yellow.addActionListener(trf);
        green.addActionListener(trf);

        f.add(R);
        f.add(Y);
        f.add(G);
        f.add(red);
        f.add(yellow);
        f.add(green);

    }
}
```

30.4 Sample Output









Date: 24/02/2022

CYCLE 6

31 **Java Program that implements Doubly Linked List**

31.1 Aim

To Write a Java program for the following

A) Create a doubly linked list of elements.
B) Delete a given element from the above list.
C) Display the contents of the list after deletion.

31.2 Algorithm

```
Step - 1   Start
Step - 2   Create a class Node for representing the nodes withing the
           class LinkedList.
    Step - 2.1.   Create data members (object of the same class) for
                  pointing to the next and previous node.
Step - 3   Create a head node and assign it null.
Step - 4   Create the addNode(data) method which accepts the data to be
           added as a parameter and then creates a new node with a constructor of
           class Node.
Step - 5   Create a DeleteNode method which accepts the position of the
           node to be deleted.
Step - 6   It assigns the node as null and points the previous node to the
           node after it.
Step - 7   Create the display method to display the entire linked list after
           all addition operation and each deletion operation.
Step - 8   Stop
```

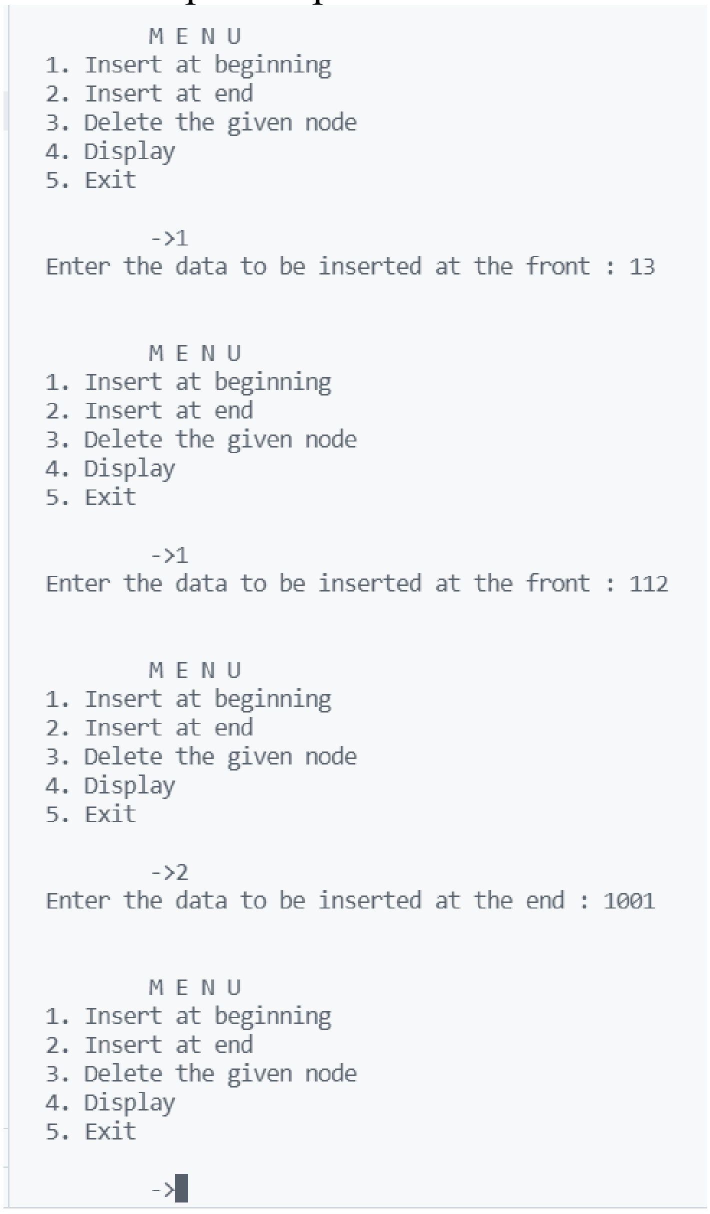
31.3 Code

```java
import java.util.Scanner;

public class LinkedList {

    static class node {
        int value;
        node next;

        node(int d) {
            value = d;
            next = null;
        }
    }

    node start = null;

    // public void finalize() {
    // System.out.println("object is garbage collected");
    // }

    public static LinkedList insert_beginning(LinkedList list, int data) {

        node Newnode = new node(data);
        if (list.start == null) {
            list.start = Newnode;
        } else {
            Newnode.next = list.start.next;
            list.start = Newnode;
        }

        return list;
    }

    public static LinkedList insert_end(LinkedList list, int data) {

        if (list.start == null) {
            System.out.println("The list is empty!!!");
        } else {
            node Newnode = new node(data);
            Newnode.next = null;

            node last = list.start;

            while (last.next != null) {
                last = last.next;
            }

            last.next = Newnode;
        }

        return list;
    }
```

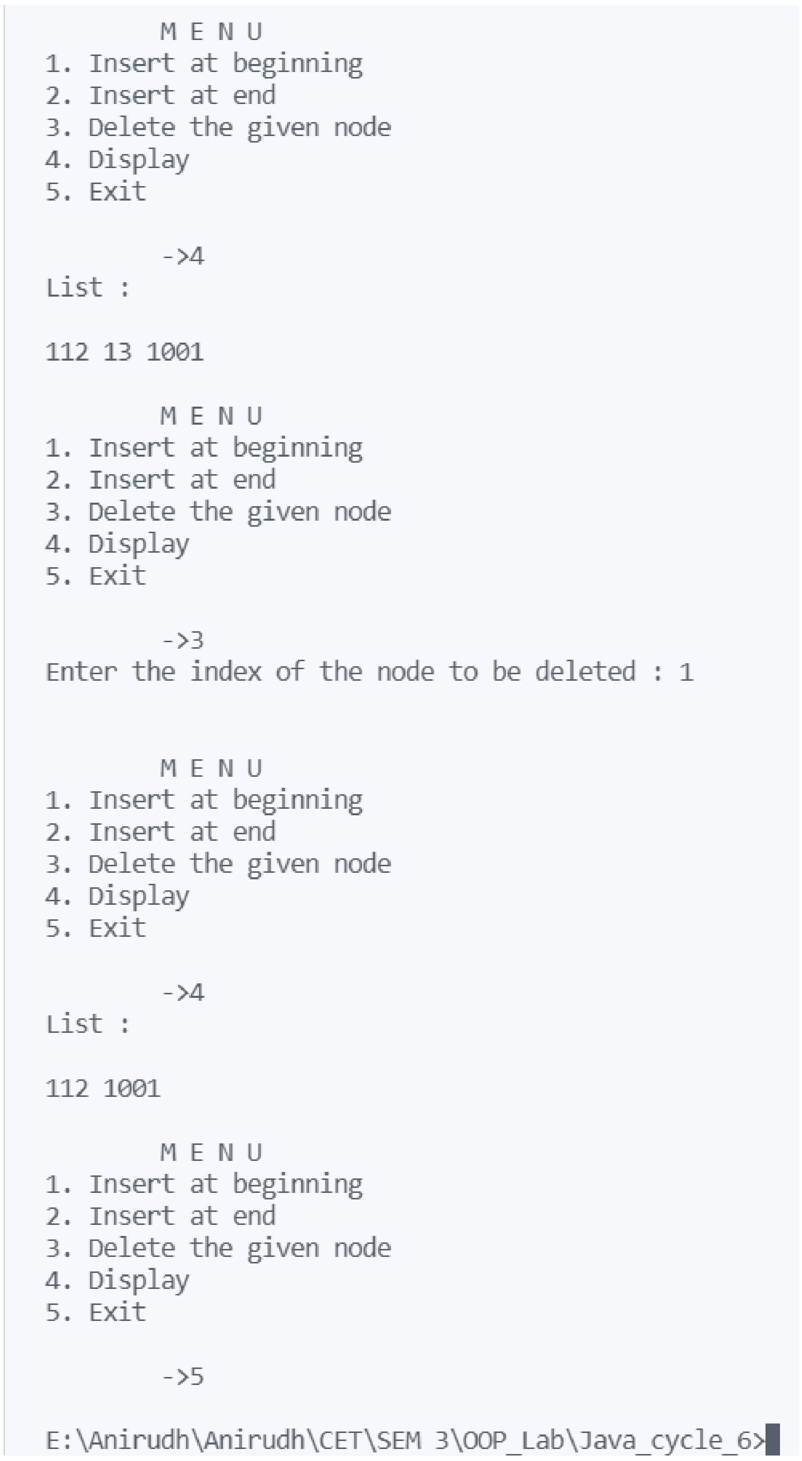
```
public static void traversal(LinkedList list) {

    int flag = 0;
    node temp = list.start;
    System.out.println("Linked list : ");

    while (temp.next != null) {
        System.out.printf("%d ", temp.value);
        temp = temp.next;
        flag = 1;
    }
    if (flag != 1) {
        System.out.println("List is empty!!");
    }


}

public static LinkedList delete(LinkedList list, int data) {

    if (list.start == null) {
        System.out.println("The list is empty!!!");
    } else {
        // node Newnode = new node(data);
        // Newnode.next = null;

        node temp1 = null, temp = list.start;
        // node temp1 = new node();
        while (temp.next != null) {
            temp1 = temp;
            temp = temp.next;
            if (temp.value == data) {
                break;
            }
        }

        temp1.next = temp.next;
        temp = null;

        System.gc();
    }

    return list;
}

public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);
```

```
        LinkedList list = new LinkedList();

        int choice, response = 1;
        do {
            System.out.println("\t\nM E N U");
            System.out
                    .println("1. Insert at beginning\n2. Insert at end\n3. 
Delete the given node\n4. Display\n5. Exit");
            System.out.printf("\n\t->");
            choice = sc.nextInt();
            int data;
            switch (choice) {
                case 1:
                    System.out.printf("Enter the data to be inserted at the 
front : ");
                    data = sc.nextInt();
                    insert_beginning(list, data);
                    break;

                case 2:
                    System.out.printf("Enter the data to be inserted at the 
end : ");
                    data = sc.nextInt();
                    insert_end(list, data);
                    break;

                case 3:
                    System.out.printf("Enter the data of the node to be 
deleted : ");
                    data = sc.nextInt();
                    delete(list, data);
                    break;

                case 4:
                    traversal(list);
                    break;

                case 5:
                    System.exit(0);
                    break;

                default:
                    break;
            }
        } while (response == 1);
        sc.close();
    }
}
```

31.4 Sample Output



```
        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->1
Enter the data to be inserted at the front : 13


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->1
Enter the data to be inserted at the front : 112


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->2
Enter the data to be inserted at the end : 1001


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->
```



```
        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->4
List :

112 13 1001

        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->3
Enter the index of the node to be deleted : 1


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->4
List :

112 1001

        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->5

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```

32 Java program that implements the binary search algorithm

32.1 Aim

To Write a Java program that implements the binary search algorithm

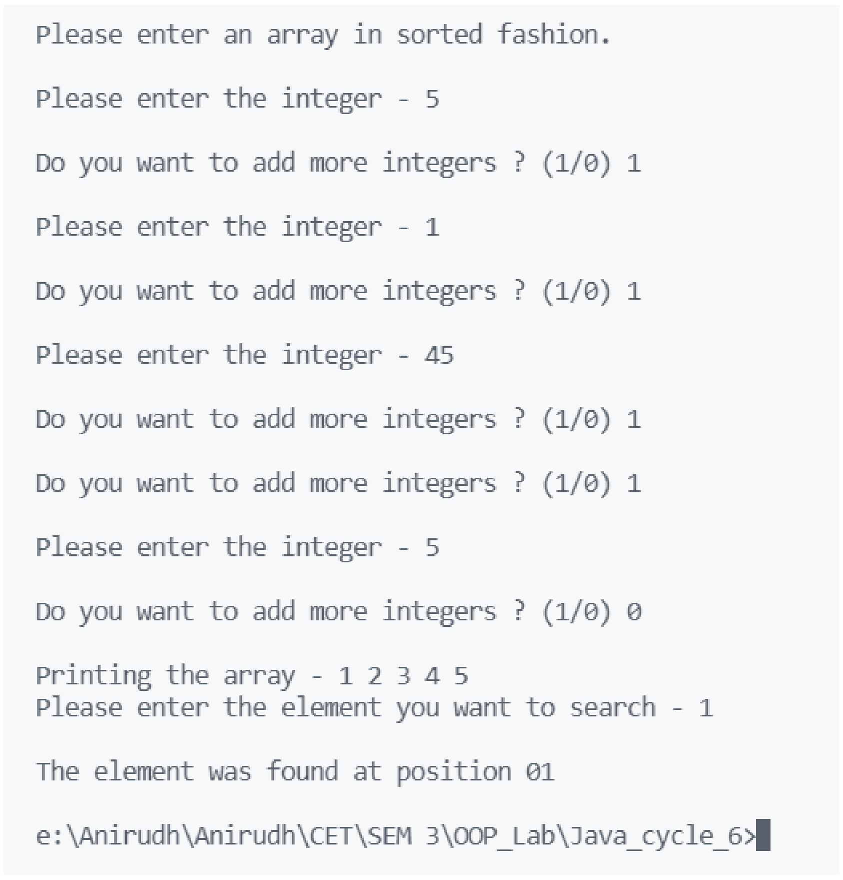
32.2 Algorithm

Step - 1 Start
Step - 2 Create a function binarySearch() where array is passed along with the element to be searched and the low and high value of the array
Step - 3 Now check the condition whether the required element is the mid one where mid=low+(high-low)/2
Step - 4 If element <array(mid) then high = low-1
Step - 5 If element>array(mid) then low=mid+1
Step - 6 Print the result whether the element is found and their respective position
Step - 7 Stop

32.3 Code

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

class Binarysearch{
    public static void main(String args[]){
        ArrayList<Integer> arr = new ArrayList<Integer>();
        Scanner sc = new Scanner(System.in);
        int x;
        System.out.print("\nPlease enter an array in sorted fashion.\n");
        while(true)
        {
            System.out.print("\nPlease enter the integer - ");

            x= sc.nextInt();
            arr.add(x);
            System.out.printf("\nDo you want to add more integers ? (1/0) ");
            x = sc.nextInt();
            if(x == 0)
            break;
        }
        Iterator<Integer> itr = arr.iterator();
        System.out.printf("\nPrinting the array - ");
        while(itr.hasNext())
        {
            x = itr.next();
            System.out.printf(x + " ");

        }

        System.out.printf("\nPlease enter the element you want to search - ");
        x = sc.nextInt();
        sc.close();

       // Performing binary search
        int start = 0, end = arr.size()-1, mid  = (start + end)/2, flag = 0;
        while(end >= start)
        {
            if(x > arr.get(mid))
            {
                start = mid +1;
                mid = (start + end)/2;
            }else if(x < arr.get(mid))
            {
                end = mid - 1;
                mid = (start + end)/2;

            }else{
                flag =  1;
                break;
            }
        }

        if(flag == 1)
        {
            System.out.println("\nThe element was found at position " +
mid+1);
        }
        else
        System.out.println("\nThe element was not found in the array");

    }
}
```

32.4 Sample Output



```
Please enter an array in sorted fashion.

Please enter the integer - 5

Do you want to add more integers ? (1/0) 1

Please enter the integer - 1

Do you want to add more integers ? (1/0) 1

Please enter the integer - 45

Do you want to add more integers ? (1/0) 1

Do you want to add more integers ? (1/0) 1

Please enter the integer - 5

Do you want to add more integers ? (1/0) 0

Printing the array - 1 2 3 4 5
Please enter the element you want to search - 1

The element was found at position 01

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```

33 Java program that implements Quick Sort algorithm

33.1 Aim

To Write a Java program that implements Quick Sort algorithm for sorting a list of names in ascending order.

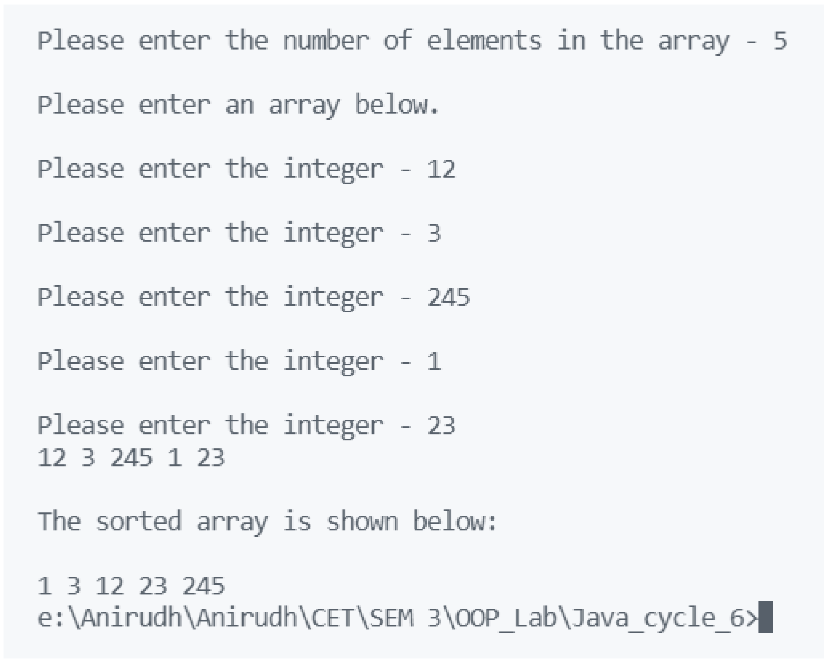
33.2 Algorithm

Step - 1 Start
Step - 2 Create a class Sort with a constructor that saves the array and a sort_now method for sorting the array.
Step - 3 Set the pivot as the first element and move the element till all the elements in its right and left are greater and smaller respectively. Recursively call this method by splitting the array again and again.
Step - 4 Accept the array in the main function and create an object of the class.
Step - 5 Call the sort method to sort the array. Then display it after sorting.
Step - 6 Stop

33.3 Code

```
import java.util.Scanner;

class Sort{
    int arr[], n;
    Sort(int arr[], int n){

        this.arr = arr;
        this.n = n;
    }

    void sort_now(int arr[], int st, int end)
    {
        int pivot = st, i=st+1, j=end, temp;
        if(st >= end)
        return;
        try{
            do{
                if(arr[i]>arr[pivot]){
                    // j=end;
                    do{
                        if(arr[pivot]>arr[j]){
                            temp = arr[j];
                            arr[j] = arr[i];
                            arr[i] = temp;
                            break;
                        }
                        j-=1;
                    }while(i<=j);

                }
                i+=1;
            }while(i<=j);
        }catch(Exception e){
            return;
        }
        temp = arr[j];
        arr[j] = arr[pivot];
        arr[pivot] = temp;

        sort_now(arr, j+1, end);
        sort_now(arr, st, j-1);
    }
}

public class QuickSort {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("\nPlease enter the number of elements in the array -
");
        int n = sc.nextInt(), x, arr[] = new int[n];

        System.out.print("\nPlease enter an array below.\n");

        for(int i=0; i<n; i++)
        {
            System.out.print("\nPlease enter the integer - ");
            x= sc.nextInt();
            arr[i] = x;
        }

        sc.close();
        Sort st = new Sort(arr, n);
        for(int p=0; p<n; p++)
        {
            System.out.printf("%d ", arr[p]);
        }
        System.out.println();
        st.sort_now(arr, 0, n-1);
        System.out.print("\nThe sorted array is shown below:\n\n");
        for(int i=0; i<n; i++)
        {
            System.out.printf("%d ", arr[i]);
        }

    }
}
```

33.4 Sample Output



```
Please enter the number of elements in the array - 5

Please enter an array below.

Please enter the integer - 12

Please enter the integer - 3

Please enter the integer - 245

Please enter the integer - 1

Please enter the integer - 23
12 3 245 1 23

The sorted array is shown below:

1 3 12 23 245
e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```

34 Java program to implement Heap Sort algorithm

34.1 Aim

To Write a Java program to implement Heap Sort algorithm using array.

34.2 Algorithm

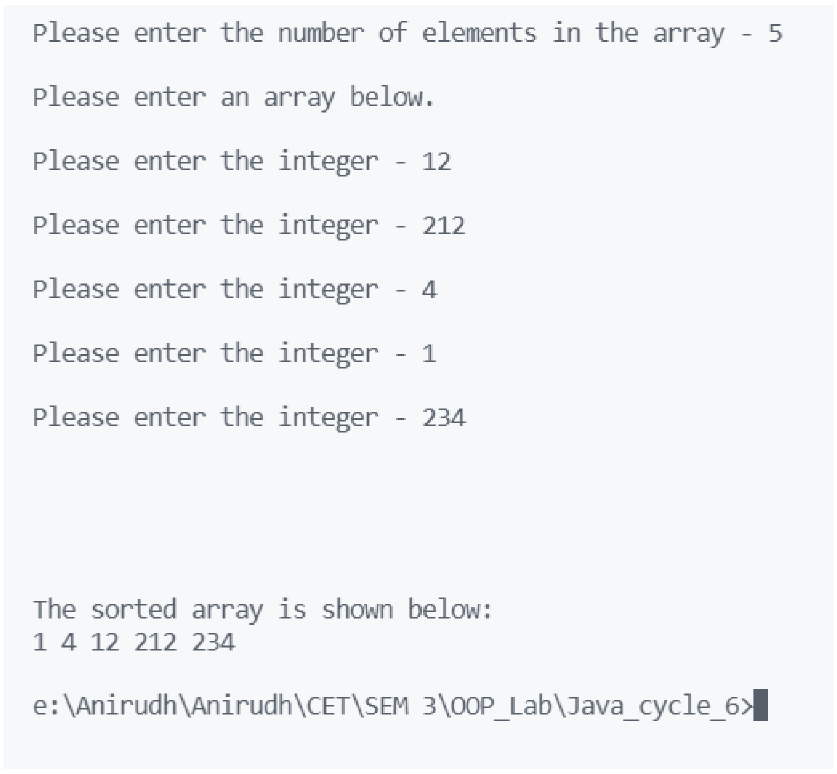
```
Step - 1  Start
Step - 2  Create the static method called Heapify which makes a heap
 from the given array.
Step - 3  Create the HeapSort method which accepts the array and the
 number of elements as parameters.
   Step - 3.1.  Move from the bottom most branch upwards and
   called the heapify function recursively to create a heap from the
   given array.
   Step - 3.2.  Now find the first element (which will also be the
   largest) and swap it with the end of the array. Call the heapify
   method again with n-1 elments of the array to recreate the heap.
   Step - 3.3.  Proceed like this till n=1.
Step - 4  Accept the array in the main method and call the methods
 to obtain the sorted array.
Step - 5  Display the array to the user.
Step - 6  Stop
```

34.3 Code

```java
import java.util.*;

public class HeapSort {


    static void heapify(int arr[], int n, int i){

        int largest = i, lc = 2*i+1, rc = 2*i+2;

        if(lc<n && arr[lc] > arr[largest]){
            largest = lc;
        }
        if(rc<n && arr[rc] > arr[largest]){
            largest = rc;
        }

        // Moving across the largest path if it exists
        if(i != largest){
            int temp = arr[largest];
            arr[largest] = arr[i];
            arr[i] = temp;

            heapify(arr, n, largest);
        }
        return;
    }

    static void Heapsort(int arr[], int n){

        // Initially heapify the tree. For this we start from the bottom most
parent (n/2 - 1)
        for(int i= n/2 - 1; i>=0; i--){
            heapify(arr, n, i);
        }


        // Due to heapify the largest element is already at the top. So, we're
going to push it out.
        int temp;

        // Now we extract the largest elements from the heap tree in an
iterative way.
        for(int i=n-1; i>0; i--)
        {

            temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;
            System.out.println();
            heapify(arr, i, 0);
        }

    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("\nPlease enter the number of elements in the array -
");
        int n = sc.nextInt(), x, arr[] = new int[n];
```

```java
        System.out.print("\nPlease enter an array below.\n");

        for(int i=0; i<n; i++)
        {
            System.out.print("\nPlease enter the integer - ");
            x= sc.nextInt();
            arr[i] = x;
        }

        sc.close();
        Heapsort(arr, n);

        System.out.println("\nThe sorted array is shown below:");

        // Printing the Sorted array
        for(int i=0; i<n; i++){
            System.out.printf("%d ", arr[i]);
        }
        System.out.println();
    }
}
```

34.4 Sample Output



```
Please enter the number of elements in the array - 5

Please enter an array below.

Please enter the integer - 12

Please enter the integer - 212

Please enter the integer - 4

Please enter the integer - 1

Please enter the integer - 234



The sorted array is shown below:
1 4 12 212 234

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```