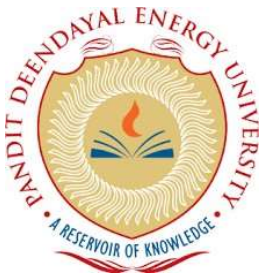


SCHOOL OF TECHNOLOGY
PANDIT DEENDAYAL ENERGY UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA

Computer Science & Engineering

LAB FILE (2022-23)

Object Oriented Programming with Java Lab

(20CP204P)

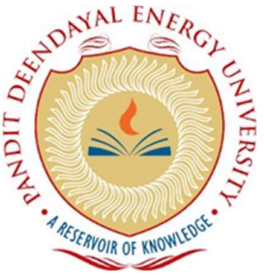
Student Name: Harsh Shah

Enrollment No.: 21BCP359 **Course with Semester: B.Tech (3rd) CSE**

Division: 6

Group: G11

Instructor: Dr. Shivangi Surati



SCHOOL OF TECHNOLOGY
PANDIT DEENDAYAL ENERGY UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA

CERTIFICATE

This is to certify that Mr. /Miss **Harsh Shah** Enrolment no**. 21BCP359** of **3rd Semester** degree course in **Computer Science and Engineering** has satisfactorily prepared and presented his Term Work in **Object Oriented Programming with Java Lab (20CP204P)** within four walls of the laboratory of this Institute during **July-2022** to **November-2022**.

Date of Submission:

Head of Department **Dr. Shivangi Surati**

Index
OBJCT ORIENTED PROGRAMMING with JAVA (20CP204P)

Sr. No.	First module	Pg. No.	Date	Sign
1.	Install JDK, setup Java environment and write a program to print -"CODING IS FUN, ENJOY IT!".			
2.	Write a program in Java to generate first *n* prime numbers.			
3.	Write a program to enter two numbers and perform all arithmetic, comparison, logical and bitwise operations on them.			
4.	Write a program that scans marks and credits of 2 subjects of the student and calculate the following: Grade of each subject (using else if ladder), Gradepoint of each subject from grade (using switch case), SPI using gradepoints and credits of 2 subjects.			
5.	Write a program in Java to find maximum of three numbers using nested if-else and conditional operator.			
6.	Write a program to accept a line and check how many consonants and vowels are there in line.			
7.	Write a program to count the number of words that start with capital letters.			
8.	Create a class which ask the user to enter a sentence, and it should display count of each vowel type in the sentence. The program should continue till user enters a word "quit". Display the total count of each vowel for all sentences.			
9.	Write an interactive program to print a string entered in a pyramid form. For instance, the string "stream" has to be displayed as follows: S S t S t r S t r e S t r e a S t r e a m			

10.	Write an interactive program to print a diamond shape. For example, if user enters the number 3, the diamond will be as follows: * * * * * * * * *			
11.	Develop minimum 4 program based on variation in methods i.e., passing by value, passing by reference, returning values and returning objects from methods.			
12.	Write a Java Program to find area of Geometric figures using method Overloading.			
13.	Write a program in Java to create a simple scientific calculator using Math Class.			
14.	Write a program in Java to sort the elements of list so that they are in ascending order (Take dynamic array).			
15.	Write a program in Java to multiply two matrixes (Take dynamic arrays).			
Sr. No.	**Second module**	**Pg. No.**	**Date**	**Sign**
1.	Write a program to create a “distance” class with methods where distance is computed in terms of feet and inches, how to create objects of a class.			
2.	Modify the “distance” class by creating constructor for assigning values (feet and inches) to the distance object. Create another object and assign second object as reference variable to another object reference variable. Further create a third object which is a clone of the first object.			
3.	Write a program in Java in which a subclass constructor invokes the constructor of the super class and instantiate the values.			
4.	Write a program in Java to develop overloaded constructor. Also develop the copy constructor to create a new object with the state of the existing object. (class Point)			

5.	Write a program to show the difference between public and private access specifiers. The program should also show that primitive data types are passed by value and objects are passed by reference and to learn use of final keyword.			
6.	Write a program to show the use of static functions and to pass variable length arguments in a function.			
7.	Write programs in Java to use Wrapper class of each primitive data types.			
8.	Write a program that implements two constructors in the class. We call the other constructor using 'this' pointer, from the default constructor of the class.			
9.	Write a program in Java to demonstrate single inheritance, multilevel inheritance and hierarchical inheritance.			
10.	Java Program to demonstrate the real scenario (e.g., bank) of Java Method Overriding where three classes are overriding the method of a parent class. Creating a parent class.			
11.	Write a program that implements simple example of Runtime Polymorphism with multilevel inheritance. (e.g., Animal or Shape).			
12.	Write a program to compute if one string is a rotation of another. For example, pit is rotation of tip as pit has same character as tip.			
13.	Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e. area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.			
14.	Write a program in Java to demonstrate multiple inheritance.			
15.	a) Write an application that illustrates method overriding in the same package and different packages. b) Also demonstrate accessibility rules in inside and outside packages.			
Sr. No.	**Third module**	**Pg. No.**	**Date**	**Sign**
1.	Read a content from file: calculate number of sentences, words and characters from the file.			

2.	Read from a file convert it to uppercase and save it into another file.			
3.	Remove duplicate lines from a File.			
4.	Create a class called Student. Write a student manager program to manipulate the student information from files by using FileInputStream and FileOutputStream.			
5.	Refine the student manager program to manipulate the student information from files by using the BufferedReader and BufferedWriter.			
6.	Write a program to manipulate the information from files by using the Reader and Writer class. Assume suitable data.			
7.	Write a program "DivideByZero" that takes two numbers a and b as input, computes a/b, and invokes Arithmetic Exception to generate a message when the denominator is zero.			
8.	Write a program to show the use of nested try statements that emphasizes the sequence of checking for catch handler statements.			
9.	Write a program to create your own exception types to handle situation specific to your application (Hint: Define a subclass of Exception which itself is a subclass of Throwable).			
10.	Write a small application in Java to develop Banking Application in which user deposits the amount Rs 1000.00 and then start withdrawing of Rs 400.00, Rs 300.00 and it throws exception "Not Sufficient Fund" when user withdraws Rs. 500 thereafter.			
11.	Write a program to handle ArrayIndexOutOfBounds exception for binary search.			
12.	Write a Java Program that demonstrates thread class and few methods.			
13.	Write a program to demonstrate thread example by implementing runnable interface.			
14.	Write a program to demonstrate priorities among multiple threads.			
15.	Write a program to demonstrate multithread communication by implementing synchronization among threads (Hint: you can implement a simple producer and consumer problem).			

Sr. No.	Fourth module	Pg. No.	Date	Sign
1.	Write a program to demonstrate different Window handling events.			
2.	Write a program to demonstrate different mouse handling events like mouseClicked(), mouseEntered(), mouseExited(), mousePressed, mouseReleased() and mouseDragged().			
3.	Write a program to demonstrate different keyboard handling events.			
4.	Write a program to generate a window without an applet window using main() function.			
5.	Write a program to demonstrate the use of push buttons.			
6.	WAP to create a Menu using the frame.			
7.	WAP to create a Frame that display the student information.			
8.	WAP to create a Dialogbox.			
9.	WAP to implement the FlowLayout and BorderLayout.			
10.	WAP to implement the GridLayout and CardLayout.			
11.	WAP to implement the GroupLayout and BoxLayout.			
12.	Write a program that demonstrates the life cycle of an applet.			
13.	WAP to demonstrate System clock.			
14.	WAP to demonstrate Painting in applet.			
15.	WAP to demonstrate Graphics in applet.			

MODULE: 1

- **<u>Practical-1</u>: Install JDK, setup Java environment and write a program to print- "CODING IS FUN, ENJOY IT!".**

```
// Install JDK
public class one {
    public static void main(String[] args) {
        System.out.println("\nCODING IS FUN, ENJOY IT!\n");
    }
}
```

> Output:

```
CODING IS FUN, ENJOY IT!
```

♦ Practical- 2: Write a program in Java to generate first n prime numbers.

```
// Generate first n Prime Number
import java.util.Scanner;
public class two{
  public static void main (String[]args){
    Scanner sc = new Scanner(System.in);
    int count = 0, n = 0, i = 1, j = 1;
    System.out.print("Enter number of prime numbers to be printed: ");
    int a = sc.nextInt();
    while (n < a)
      {
    j = 1;
    count = 0;
    while (j <= i)
      {
       if (i % j == 0)
         count++;
       j++;
      }
    if (count == 2)
      {
       System.out.printf ("%d ", i);
       n++;
      }
    i++;
      }
  }
}
```

> Output:

```
Enter number of prime numbers to be printed: 10
2 3 5 7 11 13 17 19 23 29
```

♦ **Practical- 3: Write a program to enter two numbers and perform all arithmetic, comparison, logical and bitwise operations on them.**

```
// Arithmetic, Comparison, Bitwise Operations
import java.util.Scanner;
public class three {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter first number: ");
      int a = sc.nextInt();
      System.out.print("Enter second number: ");
      int b = sc.nextInt();
      //Arithmetic Operations
      System.out.println("\nAddition: " + (a + b));
      System.out.println("Multiplication: " + (a * b));
      System.out.println("Subtraction: " + (a - b));
      System.out.println("Division: " + (a / b));
      System.out.println("Modulus: " + (a % b));
      System.out.println("Exponation: " + (a ^ b));
      //Comparision Operations
      System.out.println("\nGreater than: " + (a > b));
      System.out.println("Greater than or equals to: " + (a >= b));
      System.out.println("Less than: " + (a < b));
      System.out.println("Less than or equals to: " + (a <= b));
      System.out.println("Equals to: " + (a == b));
      System.out.println("Not equals to: " + (a != b));
      //Bitwise Operations
      System.out.println("\na&b = " + (a & b));
      System.out.println("a|b = " + (a | b));
      System.out.println("a^b = " + (a ^ b));
      System.out.println("~a = " + ~a);
   }
}
```

```
Enter first number: 6
Enter second number: 3

Addition: 9
Multiplication: 18
Subtraction: 3
Division: 2
Modulus: 0
Exponation: 5

Greater than: true
Greater than or equals to: true
Less than: false
Less than or equals to: false
Equals to: false
Not equals to: true

a&b = 2
a|b = 7
a^b = 5
~a = -7
```

- **<u>Practical-</u> 4: Write a program that scans marks and credits of 2 subjects of the student and Calculate: Grade of each subject (using else if ladder), Gradepoint of each subject from grade (using switch case), SPI using gradepoints and credits of 2 subjects.**

```
// Calculating Grade & SPI
import java.util.Scanner;
public class four {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      //Calculating grade, gradepoints and SPI
      String grade1, grade2;
      int gradepoints1, gradepoints2;
      //Inputing marks and credits of first subject
      System.out.print("\nEnter marks of first subject: ");
      int m1 = sc.nextInt();
      System.out.print("Enter credits of first subject: ");
      int c1 = sc.nextInt();
      //Inputing marks and credits of second subject
      System.out.print("\nEnter marks of second subject: ");
      int m2 = sc.nextInt();
      System.out.print("Enter credits of second subject: ");
      int c2 = sc.nextInt();
      //Calculating grades of first subject
      if(m1 >= 80){
         grade1 = "O";
      }
      else if(m1 >= 70){
         grade1 = "A+";
      }
      else if(m1 >= 60){
         grade1 = "A";
      }
      else if(m1 >= 55){
         grade1 = "B+";
      }
      else if(m1 >= 50){
```

```
    grade1 = "B";
}
else if(m1 >= 45){
    grade1 = "C";
}
else if(m1 >= 40){
    grade1 = "P";
}
else {
    grade1 = "F";
}
//Calculating grades of second subject
if(m2 >= 80){
    grade2 = "O";
}
else if(m2 >= 70){
    grade2 = "A+";
}
else if(m2 >= 60){
    grade2 = "A";
}
else if(m2 >= 55){
    grade2 = "B+";
}
else if(m2 >= 50){
    grade2 = "B";
}
else if(m2 >= 45){
    grade2 = "C";
}
else if(m2 >= 40){
    grade2 = "P";
}
else {
    grade2 = "F";
}
```

```
//Calculating gradepoints of first subject
switch(grade1){
    case "O":
        gradepoints1 = 10;
        break;
    case "A+":
        gradepoints1 = 9;
        break;
    case "A":
        gradepoints1 = 8;
        break;
    case "B+":
        gradepoints1 = 7;
        break;
    case "B":
        gradepoints1 = 6;
        break;
    case "C":
        gradepoints1 = 5;
        break;
    case "P":
        gradepoints1 = 4;
        break;
    default:
        gradepoints1 = 0;
        break;
}
switch(grade2){
    case "O":
        gradepoints2 = 10;
        break;
    case "A+":
        gradepoints2 = 9;
        break;
    case "A":
        gradepoints2 = 8;
```

```
            break;
        case "B+":
            gradepoints2 = 7;
            break;
        case "B":
            gradepoints2 = 6;
            break;
        case "C":
            gradepoints2 = 5;
            break;
        case "P":
            gradepoints2 = 4;
            break;
        default:
            gradepoints2 = 0;
            break;
    }
    float spi = (float)((c1 * gradepoints1) + (c2 * gradepoints2)) / (c1 + c2);
    System.out.println("\nGrade of first subject is " + grade1);
    System.out.println("Gradepoint of first subject is " + gradepoints1);
    System.out.println("Grade of second subject is " + grade2);
    System.out.println("Gradepoint of second subject is " + gradepoints2);
    System.out.println("SPI is " + spi);
  }
}
```

> Output:

```
Enter marks of first subject: 98
Enter credits of first subject: 2

Enter marks of second subject: 95
Enter credits of second subject: 1

Grade of first subject is O
Gradepoint of first subject is 10
Grade of second subject is O
Gradepoint of second subject is 10
SPI is 10.0
```

♦ **<u>Practical-</u> 5: Write a program in Java to find maximum of three numbers using nested if-else and conditional operator.**

```
// Maximum of Three
import java.util.Scanner;
public class five {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      //Finding maximum of three numbers
      System.out.print("Enter first number: ");
      int a = sc.nextInt();
      System.out.print("Enter second number: ");
      int b = sc.nextInt();
      System.out.print("Enter third number: ");
      int c = sc.nextInt();
      if(a > b){
         if(a > c){
            System.out.println(a + " is the greatest number");
         }
         else{
            System.out.println(c + " is the greatest number");
         }
      }
      else{
         if(c > b){
            System.out.println(c + " is the greatest number");
         }
         else{
            System.out.println(b + " is the greatest number");
         }
      }
   }
}
```

```
Enter first number: 6
Enter second number: 3
Enter third number: 2
6 is the greatest number
```

♦ **<u>Practical-</u> 6: Write a program to accept a line and check how many consonants and vowels are there in line.**

```
// Checking number of Consonants & Vowels
import java.util.Scanner;
public class six {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      //Counting vowels and consonants in a line
      System.out.print("Enter a string: ");
      String s = sc.nextLine();
      int vowel = 0;
      int space = 0;
      for(int i = 0; i < s.length(); i ++){
         char letter = s.charAt(i);
         if(letter == 'a' || letter == 'e' || letter == 'i' || letter == 'o' || letter  == 'u'){
            vowel = vowel + 1;
         }
         else if(letter == ' '){
            space ++;
         }
      }
      int conso = s.length() - vowel - space;
      System.out.println("Number of vowels are " + vowel);
      System.out.println("Number of consonants are " + conso);
   }
}
```

> Output:

```
Enter a string: Objected oriented programming with java
Number of vowels are 12
Number of consonants are 23
```

- **Practical- 7: Write a program to count the number of words that start with capital letters.**

```
// Number of words that start with capital letter
import java.util.Scanner;
public class seven {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      //Counting number of words that start with capital letters
      System.out.print("Enter a sentence: ");
      String line = sc.nextLine();
      int word = 0;
      char firstletter = line.charAt(0);
         if(firstletter >= 65 && firstletter <= 90){
            word ++;
         }
      for(int i = 0; i < line.length(); i ++){
         char letter = line.charAt(i);
         if(letter == ' '){
            char first = line.charAt(i + 1);
            if(first >= 65 && first <= 90){
               word ++;
            }
         }
      }
      System.out.println("Total words starting with capital letters are " + word);
   }
}
```

> Output:

```
Enter a sentence: Objected Oriented Programming With Java
Total words starting with capital letters are 5
```

- **<u>Practical-</u> 8: Create a class which ask the user to enter a sentence, and it should display count of each vowel type in the sentence. The program should continue till user enters a word "quit". Display the total count of each vowel for all sentences.**

```
import java.util.Scanner;
public class CountVowels {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String x = "yes";
      while (x.equals("yes")){
         System.out.print("Enter a sentence: ");
         String str = sc.nextLine();
         String s = str.toLowerCase();
         int n = s.length();
         int a=0, e=0, i=0, o=0, u=0;
         for (int j=0;j<n;j++){
            char letter = s.charAt(j);
            if(letter == 'a' ){
               a++;
            }
            if(letter == 'e' ){
               e++;
            }
            if(letter == 'i' ){
               i++;
            }
            if(letter == 'o' ){
               o++;
            }
            if(letter == 'u' ){
               u++;
            }
         }
         System.out.println("Number of 'a': " + a);
         System.out.println("Number of 'e': " + e);
         System.out.println("Number of 'i': " + i);
         System.out.println("Number of 'o': " + o);
```

```
            System.out.println("Number of 'u': " + u);
            System.out.println("Do you want to continue?");
            x = sc.nextLine();
        }
        System.out.println("Thankyou! for your Time");
        sc.close();
    }
}
```

> Output:

```
Enter a sentence: Objected oriented Programming with java
Number of 'a': 3
Number of 'e': 4
Number of 'i': 3
Number of 'o': 3
Number of 'u': 0
Do you want to continue?
quit
Thankyou! for your Time
```

♦ **<u>Practical-</u> 9: Write an interactive program to print a string entered in a pyramid form. For instance, the string "stream" has to be displayed as follows:**

S

S t

S t r

S t r e

S t r e a

S t r e a m

```
// String in Pyramid form
import java.util.*;
class Pyramid
{
   public static void main(String[] m)
      {
       char c;
       int i,j;
       Scanner in= new Scanner(System.in);
       String s;
       System.out.println("Enter A string : ");
       s=in.next();
       int k,d;
       for(i=0;i<s.length();i++){
          for(k=0;k<s.length()-i;k++) {
             System.out.print(" ");
          }
          for(j=0;j<=i;j++){
             c=s.charAt(j);
             System.out.print(c+" ");
          }
       System.out.println(" ");
       }
   }
```

```
}
```

> Output:

```
Enter A string :
stream
      s
     s t
    s t r
   s t r e
  s t r e a
 s t r e a m
```

- **<u>Practical-</u> 10: Write an interactive program to print a diamond shape. For example, if user enters the number 3, the diamond will be as follows:**

```
  *
 * *
* * *
 * *
  *
```

```
// Diamond Pattern
import java.util.Scanner;
public class Diamond {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        for (int i=1; i<=n; i++){
            for (int j=n-i; j>0; j--){
                System.out.print(" ");
            }
            for (int k=1; k<=i; k++){
                System.out.print("* ");
            }
            System.out.println();
        }
        for (int i=1; i<=n; i++){
            for (int k=1; k<=i; k++){
                System.out.print(" ");
            }
```

```
            for (int j=n-i; j>0; j--){
                System.out.print("* ");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

> Output:

```
3
   *
  * *
* * *
  * *
   *
```

- **<u>Practical-</u> 11: Develop minimum 4 program based on variation in methods i.e., passing by value, passing by reference, returning values and returning objects from methods.**

```
// Passing by value

import java.util.Scanner;

class CallByValue {

   static void swap(int a, int b){

      int c = a;

      a=b;

      b=c;

   }

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      int a = sc.nextInt();

      int b = sc.nextInt();


      System.out.println("Values before: "+a+", "+b);

      swap(a,b);

      System.out.println("Values after: "+a+", "+b);

   }

}
```

> Output:

```
3
6
Values before: 3, 6
Values after: 3, 6
```

// Passing By Reference

```
import java.util.Scanner;
class Test{
    int a, b;
    void swap(Test t){
        int c = a;
        a = b;
        b = c;
    }
}
public class CallByReference {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        Test t = new Test();
        t.a = sc.nextInt();
        t.b = sc.nextInt();
        // t.a = 5;
        // t.b = 6;
        System.out.println("Values before: "+t.a+", "+t.b);
        t.swap(t);
        System.out.println("Values after: "+t.a+", "+t.b);
    }
}
```

```
3
6
Values before: 3, 6
Values after: 6, 3
```

// Returning value

```
import java.util.Scanner;

class CallByValue {
    static void swap(int a, int b){
        int c = a;
        a=b;
        b=c;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();


        System.out.println("Values before: "+a+", "+b);
        swap(a,b);
        System.out.println("Values after: "+a+", "+b);
    }
}
```

> Output:

```
3
6
Values before: 3, 6
Values after: 3, 6
```

// Returning Object

```
import java.util.Scanner;
class Test{
    int a;
    Test(int i){
        a = i;
    }
    Test incr(){
        Test t = new Test(a + 10);
        return t;
    }
}
class Main
{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        Test t1 = new Test(n);
        Test t2;
        t2 = t1.incr();
        System.out.println("Value before: "+t1.a);
        System.out.println("Value after: "+t2.a);
    }
}
```

```
3
Value before: 3
Value after: 13
```

♦ **Practical- 12: Write a Java Program to find area of Geometric figures using method Overloading.**

```
// Area of various Geometric Figures using Method Overloading

public class AreaMethOverload{

   public static void findArea(int l){

      System.out.println("Area of Square is :"+ (l*l));

   }

   public static void findArea(int l, int b){

      System.out.println("Area of Rectangle is :"+ (l*b));

   }

   public static void findArea(int l, int b, int h){

      System.out.println("Area of Trapezoid is :"+ (0.5*(l+b)*h));

   }

   public static void main(String[] args) {

      findArea(5);

      findArea(5,6);

      findArea(5,6,7);

   }

}
```

> Output:

```
Area of Square is :25
Area of Rectangle is :30
Area of Trapezoid is :38.5
```

- **<u>Practical-</u> 13: Write a program in Java to create a simple scientific calculator using Math Class.**

```
// Scientific Calculator using Math Function

import java.util.Scanner;

import java.lang.Math;

public class Calculator {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      System.out.println("(+) Addition\n(-) Subtraction\n(*) Multiplication\n(/) Division\n(%)
Remainder\n(^) Square\n(s) Square root");

      System.out.print("\nEnter your Choice: ");

      String n = sc.nextLine();

      System.out.print("Enter two numbers: ");

      int a = sc.nextInt();

      int b = sc.nextInt();

      if (n=="+"){

         System.out.println("Addition: "+ Math.addExact(a,b));

      }

      else if (n=="-"){

         System.out.println("Subtraction: "+ Math.subtractExact(a,b));

      }

      else if (n=="*"){

         System.out.println("Multiplication: "+ Math.multiplyExact(a,b));

      }

      else if (n=="/"){

         System.out.println("Division: "+ Math.floorDiv(a,b));

      }

      else if (n=="%"){
```

```
            System.out.println("Modulus: "+ (a%b));
        }
        else if (n=="s"){
            System.out.println("Square root: "+ (Math.sqrt(a)));
        }
        else if (n=="^"){
            System.out.println("Square root: "+ (Math.pow(a,2)));
        }
        sc.close();
    }
}
```

> Output:

```
(+) Addition
(-) Subtraction
(*) Multiplication
(/) Division
(%) Remainder
(^) Square
(s) Square root

Enter your Choice: +
Enter two numbers: 6 3

Addition: 9
```

- **<u>Practical-</u> 14: Write a program in Java to sort the elements of list so that they are in ascending order (Take dynamic array).**

```
// Sorting Array

import java.util.Scanner;

public class SortArray {

   public static void main(String[] args) {

      int n, temp;

      Scanner sc = new Scanner(System.in);

      // Taking input of array

      System.out.print("Enter no. of elements you want in array: ");

      n = sc.nextInt();

      int a[] = new int[n];

      System.out.println("Enter all the elements:");

      for (int i = 0; i < n; i++)

      {

         a[i] = sc.nextInt();

      }

      // Sorting loop

      for (int i = 0; i < n; i++)

      {

         for (int j = i + 1; j < n; j++)

         {

            if (a[i] > a[j])

            {

               temp = a[i];

               a[i] = a[j];

               a[j] = temp;

            }
```

```
        }
    }
    // Printing the array
    System.out.print("Ascending Order: ");
    for (int i = 0; i < n - 1; i++)
    {
        System.out.print(a[i] + ", ");
    }
    System.out.print(a[n - 1]);
  }
}
```

> Output:

```
Enter no. of elements you want in array: 6
Enter all the elements:
4
8
2
1
0
6
Ascending Order: 0, 1, 2, 4, 6, 8
```

- **Practical- 15: Write a program in Java to multiply two matrixes (Take dynamic arrays).**

```
// Multiplication of Matrices

import java.util.Arrays;

import java.util.Scanner;

public class MatrixMultiply {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      int i,j,k;

      // Taking dimensions of Arrays

      System.out.print("Enter no. of rows of 1st array: ");

      int row1 = sc.nextInt();

      System.out.print("Enter no. of columns of 1st array: ");

      int col1 = sc.nextInt();

      System.out.print("Enter no. of rows of 1st array: ");

      int row2 = sc.nextInt();

      System.out.print("Enter no. of columns of 1st array: ");

      int col2 = sc.nextInt();

      if (col1==row2){

         // First Array

         System.out.println("\nEnter elements of 1st array");

         int a[][] = new int[row1][col1];

         for (i=0; i<row1; i++){

            for (j=0; j<col1; j++){

               System.out.printf("Enter element[%d][%d] : ",i,j);

               a[i][j] = sc.nextInt();

            }

         }
```

```
        System.out.println("1st Array: "+Arrays.deepToString(a));
        // Second Array
        System.out.println("\nEnter elements of 2nd array");
        int b[][] = new int[row2][col2];
        for (i=0; i<row2; i++){
            for (j=0; j<col2; j++){
                System.out.printf("Enter element[%d][%d] : ",i,j);
                b[i][j] = sc.nextInt();
            }
        }
        System.out.println("2nd Array: "+Arrays.deepToString(a));
        // Multiplying Array
        int c[][] = new int[row1][col2];
        for (i = 0; i < row1; i++) {
            for (j = 0; j < col2; j++) {
                for (k = 0; k < row2; k++){
                    c[i][j] += a[i][k] * b[k][j];
                }
            }
        }
        System.out.print("\nThe Multiplied array is ");
        System.out.println(Arrays.deepToString(c));
      }
      else{
        System.out.println("\nArrays can't be multiplied!!");
      }
    }
}
```

> Output:

```
Enter no. of rows of 1st array: 2
Enter no. of columns of 1st array: 2
Enter no. of rows of 1st array: 2
Enter no. of columns of 1st array: 2

Enter elements of 1st array
Enter element[0][0] : 1
Enter element[0][1] : 2
Enter element[1][0] : 3
Enter element[1][1] : 4
1st Array: [[1, 2], [3, 4]]

Enter elements of 2nd array
Enter element[0][0] : 1
Enter element[0][1] : 2
Enter element[1][0] : 3
Enter element[1][1] : 4
2nd Array: [[1, 2], [3, 4]]

The Multiplied array is [[7, 10], [15, 22]]
```

MODULE: 2

- **<u>Practical-1</u>: Write a program to create a "distance" class with methods where distance is computed in terms of feet and inches, how to create objects of a class.**

```
import java.util.Scanner;
class Distance1{
   float feet;
   float inches;
   void distCalc(int length){
      feet = (length/(12f*2.54f));
      inches = (length*2.54f);
      System.out.println("Feet: " + feet + "\nInches: " + inches);
   }
}
public class OneDistance {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the length: ");
      int length = sc.nextInt();
      Distance1 d1 = new Distance1();
      d1.distCalc(length);
   }
}
```

Output:

```
Enter the length: 320
Feet: 10.498688
Inches: 812.8
```

- **<u>Practical-2</u>: Modify the "distance" class by creating constructor for assigning values (feet and inches) to the distance object. Create another object and assign second object as reference variable to another object reference variable. Further create a third object which is a clone of the first object.**

```
import java.util.Scanner;

class distance1 {
   double feet, inch;

   distance1() {
      double feet, inch;
      System.out.println("In default constructor");
   }

   void printData() {
      System.out.println("The distance is: " + feet + "ft.");
      System.out.println("The distance is: " + inch + "in.");
   }

   void setData(double f, double i) {
      feet = f;
      inch = i;
   }
}

class distance2 {
      public static void main(String[] args) {
         distance1 D1 = new distance1();
         distance1 D2 = new distance1();

         distance1 D3 = D1;
```

```
        //taking input of values...
        Scanner ob = new Scanner(System.in);

        System.out.print("Enter the distance in feet: ");
        double f = ob.nextDouble();

        System.out.print("Enter the distance in inch: ");
        double i = ob.nextDouble();

        //setting and printing data for D1
        D1.setData(f,i);
        D1.printData();

        D2.setData(f,i);
        D2.printData();

        //setting data of D3 as 50 feet..
        D3.feet = 50;

        System.out.println("The feet of D1: " + D1.feet);
    }
}
```

Output:

```
Constructor
Clone Constructor
Feet: 7
Inch: 5
Feet: 3
Inch: 5
Feet: 7
Inch: 5
```

- **<u>Practical-3</u>: Write a program in Java in which a subclass constructor invokes the constructor of the super class and instantiate the values.**

```
package src;
class Animal {
   // No-arg constructor
   Animal() {
      System.out.println("I am an animal");
   }
   // parameterized constructor
   Animal(String type) {
      System.out.println("Type: "+type);
   }
}
class Dog extends Animal {
   // default constructor
   Dog() {
      // calling parameterized constructor of the superclass
      super("Animal");
      System.out.println("I am a dog");
   }
}
public class ThreeSubSupConst {
   public static void main(String[] args) {
      Dog dog1 = new Dog();
   }
}
```

Output:

```
Type: Animal
I am a dog
```

- **Practical-4: Write a program in Java to develop overloaded constructor. Also develop the copy constructor to create a new object with the state of the existing object. (Class Point)**

```
class Point1{
   int x,y,z;
   Point1(){
      x=0;
      y=0;
      z=0;
      System.out.println("Invoked Default constructor");
      System.out.println(x + ", " + y + ", " + z);
   }

   Point1(int ix, int iy, int iz){
      x = ix;
      y = iy;
      z = iz;
      System.out.println("Invoked Parameterized constructor");
      System.out.println(x + ", " + y + ", " + z);
   }

   Point1(Point1 p){
      x = p.x;
      y = p.y;
      z = p.z;
      System.out.println("Invoked Copy Constructor");
      System.out.println(x + ", " + y + ", " + z);
   }
}

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

```
        Point1 p1 = new Point1();
        Point1 p2 = new Point1(5,10,15);
        Point1 p3 = new Point1(p2);
    }
}
```

Output:

```
Invoked Default constructor
0, 0, 0
Invoked Parameterized constru
5, 10, 15
Invoked Copy Constructor
```

- **<u>Practical-5</u>: Write a program to show the difference between public and private access specifiers. The program should also show that primitive data types are passed by value and objects are passed by reference and to learn use of final keyword.**

```
class Distance{
   public int feet, inch;
   private int meter, centimeter;
   final int kilometer = 1500;
   Distance(int f, int i){
      this.feet = f;
      this.inch = i;
      meter = 10, centimeter = 100;
   }
   void display(){
      System.out.println("Feet: " + feet);
      System.out.println("Inch: " + inch);
      System.out.println("Meter: " +meter);
      System.out.println("Centimeter: " + centimeter);
      kilometer = 2000; // kilometer can't be changed as it is final
   }
}
public class FiveAccessFinal {
   public static void main(String[] args) {
    Distance d1 = new Distance(23, 12);

      Distance d2 = new Distance(20, 30);
      d1.display(), d2.display();

      System.out.println(meter); // As it's Private, can't be called
      System.out.println(centimeter); // As it's Private, can't be called
   }
}
```

Output

```
Constructor
Constructor
Feet: 23
Inch: 12
Feet: 20
Inch: 30
```

- **Practical-6: Write a program to show the use of static functions and to pass variable length arguments in a function.**

```
class Demo {
   //non-static function
   void display() {
      System.out.println("A non-static function is called.");
   }
   //static function
   static void show() {
      System.out.println("The static function is called.");
   }
   //Varargs method
   static void displayVarargs(String... values){
      System.out.println("Displaying Varargs method ");
      for(String s:values){
         System.out.println(s);
      }
   }
}
public class SixStaticVarargs {
   public static void main(String args[]) {
      Demo obj = new Demo();
      obj.display();
      Demo.show();
      Demo.displayVarargs("This","is","Object","Oriented", "Programming");

   }
}
```

Output

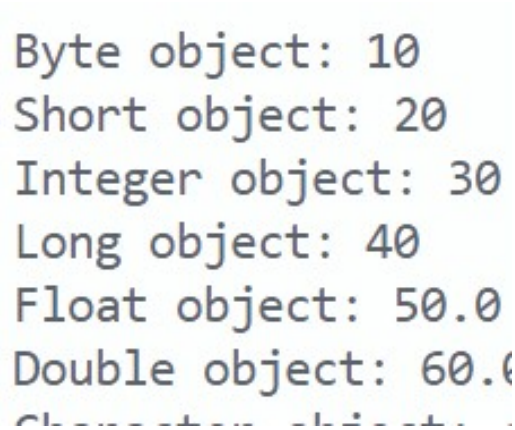
```
A non-static function is called.
The static function is called.

Displaying Varargs method
This
is
Object
Oriented
Programming
```

- **<u>Practical-7</u>: Write programs in Java to use Wrapper class of each primitive data types.**

```
public class SevenWrapperClass {

   public static void main(String[] args) {

      byte b = 10;

      short s=20;

      int i=30;

      long l=40;

      float f=50.0F;

      double d=60.0D;

      char c='a';

      boolean b2=true;


      Byte byteobj=b;

      Short shortobj=s;

      Integer intobj=i;

      Long longobj=l;

      Float floatobj=f;

      Double doubleobj=d;

      Character charobj=c;

      Boolean boolobj=b2;


      System.out.println("\nByte object: "+byteobj);

      System.out.println("Short object: "+shortobj);

      System.out.println("Integer object: "+intobj);

      System.out.println("Long object: "+longobj);

      System.out.println("Float object: "+floatobj);
```

```
        System.out.println("Double object: "+doubleobj);
        System.out.println("Character object: "+charobj);
        System.out.println("Boolean object: "+boolobj);
    }
}
```

Output:



- **<u>Practical-8</u>: Write a program that implements two constructors in the class. We call the other constructor using 'this' pointer, from the default constructor of the class**

```
class Point{
   int x,y,z;
   Point(){
      this.x=0;
      this.y=0;
      this.z=0;
      System.out.println("Invoked 1st constructor");
   Point(int x, int y, int z){
      this.x = x;
      this.y = y;
      this.z = z;
      System.out.println("Invoked 2nd constructor");
   }
}
class EightThisPointer {
   public static void main(String[] args) {
      Point p1 = new Point();
      Point p2 = new Point(5,10,15);
      System.out.println(p1.x+", "+p1.y+", "+p1.z);
      System.out.println(p2.x+", "+p2.y+", "+p2.z);
   }
}
```

Output:

```
Invoked 1st constructor
Invoked 2nd constructor
0, 0, 0
5, 10, 15
```

- **<u>Practical-9</u>: Write a program in Java to demonstrate single inheritance, multilevel inheritance and hierarchical inheritance.**

```
import java.util.Scanner;
public class singleInheritance {
    public static void main(String[] args) {
        programmer p = new programmer();
        p.setData();
        p.printData();
        System.out.println("Bonus of programmer is " + p.bonus);
    }
}
class employee {
    Scanner input = new Scanner(System.in);
    int salary;
    int empId;
    void setData(){
        System.out.print("Enter employee salary: ");
        salary = input.nextInt();
        System.out.print("Enter Employee Id: ");
        empId = input.nextInt();
    }
    void printData(){
        System.out.println("Salary of employee is " + salary);
        System.out.println("Employee ID of employee is " + empId);
    }
}
class programmer extends employee {
```

```
Enter employee salary: 65000
Enter Employee Id: 23
Salary of employee is 65000
Employee ID of employee is 23
Bonus of programmer is 100
```

```
    int bonus = 100;
}
    // Multilevel inheritance
    import java.util.Scanner;
    public class multilevelInheritance {
      public static void main(String[] arguments) {
        Cube cube = new Cube();
        cube.display();
        cube.area();
        cube.volume();
      }
    }
    class Shape {
      int areaofshape;
      void display() {
        System.out.println("Inside display");
      }
    }
    class Rectangle extends Shape {
      int l , d;
      Scanner input = new Scanner(System.in);
      void area() {
        System.out.print("Enter length of Rectangle: ");
        l = input.nextInt();
        System.out.print("Enter width of Rectangle: ");
        d = input.nextInt();
        areaofshape = l * d;
```

```
    System.out.println("Area of rectangle is " + areaofshape);
  }
}
class Cube extends Rectangle {
  int h;
  void volume() {
    System.out.print("Enter height of cube: ");
    h = input.nextInt();
    int vol = l * d * h;
    System.out.println("Volume of cube is " + vol);
  }
}
```

Output:

```
Inside display
Enter length of Rectangle: 65
Enter width of Rectangle: 32
Area of rectangle is 2080
Enter height of cube: 6
Volume of cube is 12480
```

// Hierarchical Inheritance

```
import java.util.Scanner;
public class hierachicalInheritance {
   public static void main(String[] args) {
      Twowheeler b = new Twowheeler();
      b.finalPrice();
      Fourwheeler c = new Fourwheeler();
      c.finalPricefour();
   }
}
class Vehicle {
   Scanner input = new Scanner(System.in);
   double baseprice = input.nextDouble();
}
class Twowheeler extends Vehicle {
```

```
    double increasePriceby = 0.2;
    void finalPrice() {
        baseprice = baseprice + (baseprice * increasePriceby);
        System.out.println("Final price of Two Wheeler is " + baseprice);
    }
}
class Fourwheeler extends Vehicle {
    double increasePriceby = 0.5;
    void finalPricefour() {
        baseprice = baseprice + (baseprice * increasePriceby);
        System.out.println("Final price of Four Wheeler is " + baseprice);
    }
}
```

Output:

```
30000
Final price of Two Wheeler is 36000.0
```

- **<u>Practical-10</u>: Java Program to demonstrate the real scenario (e.g., bank) of Java Method Overriding where three classes are overriding the method of a parent class.**

```
// SuperClass
class Bank {
  void loanInterest() {
     System.out.println("Interest Rate is 8%");
  }
}

// SubClass-1
class SBI extends Bank {
  void loanInterest() {
     System.out.println("SBI Bank Rate of Interest: 8.2%");
  }
}

// SubClass-2
class Axis extends Bank {
void loanInterest() {
     System.out.println("Axis Bank Rate of Interest: 9.8%");
  }
}

// SubClass-3
class ICICI extends Bank {
```

```
  void loanInterest() {
        System.out.println("ICICI Bank Rate of Interest: 9%");
    }
  }

 // Main Class
public class TenMethOverride
{
  public static void main(String[] args) {
     SBI s = new SBI();
     ICICI i = new ICICI();
     Axis a = new Axis();
     s.loanInterest();
     i.loanInterest();
     a.loanInterest();
   }
}
```

Output:

```
SBI Bank Rate of Interest:
ICICI Bank Rate of Interest
Axis Bank Rate of Interest:
```

- **<u>Practical-11</u>: Write a program that implements simple example of Runtime Polymorphism with multilevel inheritance. (e.g., Animal or Shape).**

```
class Animal{
   void sound(){System.out.println("Animals hae different sounds");}
}
class Cat extends Animal{
   void sound(){System.out.println("Meow");}
}
class Kitten extends Cat{
   void sound(){System.out.println("Meowww");}
}
public class ElevenMultiLevelInheritance {
   public static void main(String args[]) {
      // Creating Reference of Animal
      Animal a1;
      Cat c1 = new Cat();
      Kitten k1 = new Kitten();

      a1 = c1;
      a1.sound();
      a1 = k1;
      a1.sound();
   }
}
```

Output:

```
Meow
Meowww
```

- **<u>Practical-12</u>: Write a program to compute if one string is a rotation of another. For example, pit is rotation of tip as pit has same character as tip.**

```
import java.util.Scanner;


public class Twelve {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter word 1: ");
      String a = sc.next();
      System.out.print("Enter word 2: ");
      String b = sc.next();
      int count = 0;
      for(int i = 0; i < a.length(); i ++) {
         if(a.charAt(i) == (b.charAt(b.length() - i - 1))) {
            count ++;
         }
      }
      if(count == a.length()) {
         System.out.println(a + " is rotation of " + b);
      }
      else {
         System.out.println(a + " is not rotation of " + b);
      }
   }
}
```

Output

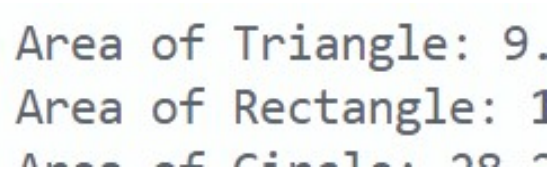
```
Enter word 1: JAVA
Enter word 2: AVAJ
JAVA is rotation of AVAJ
```

- **<u>Practical-13</u>: Describe abstract class called Shape which has three subclasses say Triangle, Rectangle, Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e. area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.**

```
abstract class Shape {
   abstract void area();
}

class Triangle{
   void area(float b, float h) {
      System.out.println("Area of Triangle: " + (0.5f*b*h));
   }
}

class Rectangle{
   void area(int l, int b) {
      System.out.println("Area of Rectangle: "+(l*b));
   }
}

class Circle{
   void area(float r) {
      System.out.println("Area of Circle: " + ((float)(3.14*r*r)));
   }
}

public class ThirteenAreaMethOverride {
```

```
    public static void main(String[] args) {
        Triangle t = new Triangle();
        Rectangle r = new Rectangle();
        Circle c = new Circle();
        t.area(3.3f,5.6f);
        r.area(3,6);
        c.area(3);
    }
}
```

Output:



- **<u>Practical-14</u>: Write a program in Java to demonstrate multiple inheritance.**

```
Interface BankA{
   float rateOfInterest();
}
interface BankB{
   float securities(long amount);
}
class SBI implements BankA, BankB{
   public float rateOfInterest() {
      return 9.75f;
   }
   public float securities(long amount){
      return (float)0.1f*amount;
   }
}
class ICICI implements BankA, BankB{
   public float rateOfInterest() {
      return 8f;
   }
   public float securities(long amount){
      return (float)0.2f*amount;
   }
}
class Axis implements BankA, BankB{
   public float rateOfInterest() {
      return 9f;
```

```
    }
    public float securities(long amount){
        return (float)0.3f*amount;
    }
}
public class multipleInheritanceInterfacing {
    public static void main(String[] args) {
        SBI s = new SBI();
        BankA I = new ICICI();
        BankB a = new Axis();

        System.out.println(s.rateOfInterest());
        System.out.println(s.securities(5000863));
        System.out.println(i.rateOfInterest());
        System.out.println(a.securities(8645167));
    }
}
```

Output:

```
9.75
500086.3
8.0
```

- **Practical-15:**

a) Write an application that illustrates method overriding in the same package and different packages.

```
class A {
   void show() {
      System.out.println("Inside Class-A");
   }
}
class B extends A {
   void show() {
      System.out.println("Inside Class-B");
   }
}
public class FifteenA {
   public static void main(String arg[]) {
      B obj=new B();
      obj.show(); //show() method of class-B will called.
   }
}
```

Output:

```
Inside Class-B
```

b) Also demonstrate accessibility rules in inside and outside packages.

package package1;

```
public class accessModifiers {
   public static void printData() {
      System.out.println("This is public access specifier");
   }
   protected static void print() {
      System.out.println("This is protected access specifier");
   }
   public static int a = 4;
   protected static int b = 3;
   int c = 2;
}
class A extends accessModifiers {
   void printdata() {
      System.out.println(c);
   }
}
```

```
package package2;
import package1.accessModifiers;
public class accessModifiers1 extends accessModifiers {
    public static void main(String[] args) {
        printData();
        print(); // Will give error
        System.out.println(a);
        System.out.println(b); // Will give error
    }
}
```

Output:

This is public access specifier

MODULE: 3

- **Practical-1: Read a content from file: calculate number of sentences, words and characters from the file.**

```
import java.io.FileReader;
import java.io.IOException;

public class OneFileReader {
   public static void main(String[] args) throws IOException {
      int i, nword=1, nline=1, nsentence=1, nchar=0;
      try{
         FileReader fr = new FileReader("D:\\Files\\one.txt");
         while((i= fr.read())!=-1){
            System.out.print((char)i);
            nchar++;
            switch((char)i){
               case ' ':
                  nword++;
                  break;
               case '.':
                  nsentence++;
                  break;
               case '\n':
                  nline++;
                  nword++;
                  break;
            }
         }

         System.out.print("\nNumber of Sentences: "+nsentence);
         System.out.print("\nNumber of Words: "+nword);
         System.out.print("\nNumber of Characters: "+(nchar-nword));
         System.out.print("\nNumber of Lines: "+nline);
```

```
            fr.close();
        }
        catch (Exception e){
            System.out.println(e);
        }
    }
}
```

Output:

```
Object Oriented Programming
in Java
Number of Sentences: 1
Number of Words: 5
Number of Characters: 31
Number of Lines: 2
```

one.txt

```
1  Object Oriented Programming
2  in Java
```

- **<u>Practical-2:</u> Read from a file convert it to uppercase and save it into another file.**

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class TwoFileWriter {
   public static void main(String[] args) throws IOException {
      int i;
      try {
         FileReader fr = new FileReader("D:\\Files\\one.txt");
         FileWriter wr = new FileWriter("D:\\Files\\two.txt");
         System.out.println("\nReading file - one.txt : ");
         while((i= fr.read())!=-1) {
            System.out.print((char)i);
            char j =Character.toUpperCase((char)i);
            wr.write(j);
         }
         wr.close();
         System.out.println("\n\nWritten file in UpperCase - two.txt : ");

                                    FileReader fwr = new FileReader("D:\\Files\\two.txt");
         while((i= fwr.read())!=-1) {
            System.out.print((char)i);
         }
         fr.close();
         fwr.close();
      }

      catch (Exception e) {
         System.out.println(e);
      }
```

```
    }
}
```

Output:

```
Reading file - one.txt :
Object Oriented Programming
in Java

Written file in UpperCase - two.txt :
OBJECT ORIENTED PROGRAMMING
IN JAVA
```

one.txt

```
1  Object Oriented Programming
2  in Java
```

two.txt

```
1  OBJECT ORIENTED PROGRAMMING
2  IN JAVA
```

- **<u>Practical-3</u>: Remove duplicate lines from a File.**

```
import java.io.*;
public class FileOperation{
    public static void main(String[] args) throws IOException {
        PrintWriter pw = new PrintWriter("output.txt");
        BufferedReader br1 = new BufferedReader(new FileReader("input.txt"));

        String line1 = br1.readLine();
        while(line1 != null){
            boolean flag = false;
            BufferedReader br2 = new BufferedReader(new FileReader("output.txt"));

            String line2 = br2.readLine();
            while(line2 != null){
                if(line1.equals(line2)){
                    flag = true;
                    break;
                }
                line2 = br2.readLine();
            }
            if(!flag){
                pw.println(line1);

                // flushing is important here
                pw.flush();
            }
            line1 = br1.readLine();
```

```
            }
            br1.close();
            pw.close();
            System.out.println("File operation performed successfully");
    }
}
```

Output:

```
File operation performed successfully
```

- **<u>Practical-4:</u> Create a class called Student. Write a student manager program to manipulate the student information from files by using FileInputStream and FileOutputStream.**

```
import java.io.Serializable;
import java.io.*;
class Student implements Serializable{
   int rollNo;
   String name;
   public Student(int rollNo, String name){
      this.rollNo = rollNo;
      this.name = name;
   }
}

public class FourStudent {
    public static void main(String[] args) {
       try{
          Student s1 = new Student(359, "Harsh");
          Student s2 = new Student(377, "Karan");
          Student s3 = new Student(387, "Kushagara");

          FileOutputStream fout = new FileOutputStream("f1.txt");
          ObjectOutputStream oot = new ObjectOutputStream(fout);

          oot.writeObject(s1);
          oot.writeObject(s2);
          oot.writeObject(s3);
          oot.flush();
          oot.close();
          System.out.println("Done Succesfully!");

                    // Reading the file
```

```
                FileReader fr = new FileReader("D:\\Files\\f1.txt");
                int i;
                while((i= fr.read())!=-1){
            System.out.print((char)i);
                }
                fr.close();


        }
        catch(Exception e){
            System.out.println(e);
        }
    }
}
```

Output:

```
Done Succesfully!
??♣srStudent??Le?↨??@@I♠rollNoL♦namet↕Ljava/lang/String;xp@gt♣Harshsq~@yt♣Karansq~@?t   Kushagara
```

f1.txt

```
1   ��NULENGsrNULBELStudent��Le�ETB��STXNULSTXINULACrollNoLNULEOTnametNULDCLjava/lang/String;xpNULNULSOHgtNULEMHarshsqNULANULNULNULNULSOHytNULENDKaransqNULANULNULNULNULSOH?tNUL Kushagara
```

- **<u>Practical-5</u>: Refine the student manager program to manipulate the student information from files by using the BufferedReader and BufferedWriter.**

```
import java.io.*;
import java.io.BufferedReader;


public class FiveBufferedStudent {
   public static void main(String[] args) throws IOException{


      // Buffered Write
      FileWriter writer = new FileWriter("D:\\Files\\f1.txt");
      BufferedWriter buffer = new BufferedWriter(writer);
      buffer.write("This is Java");
      buffer.close();


      System.out.println("Success");


      // Buffered Read
      FileReader fr = new FileReader("D:\\Files\\f1.txt");
      BufferedReader br = new BufferedReader(fr);
      int i;
      while((i=br.read())!=-1) {
         System.out.print((char)i);
      }
      br.close();
      fr.close();
   }
}
```

Output:

```
Success
This is Java
```

f1.txt

```
1  This is Java
```

- **<u>Practical-6</u>: Write a program to manipulate the information from files by using the Reader and Writer class. Assume suitable data.**

```
import java.io.*;
public class SixReadWrite {
   public static void main(String[] args) throws IOException {
      int data;
      try {
         Writer wr = new FileWriter("D:\\Files\\six.txt");
         Reader fr = new FileReader("D:\\Files\\six.txt");
         // Writing in file - six.txt
         wr.write("This is some text in six.txt");
         wr.close();

         while((data= fr.read())!=-1) {
            System.out.print((char)data);
         }
         fr.close();
      }
      catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

Output:

```
This is some text in six.txt
```

six.txt

```
1  This is some text in six.txt
```

- **<u>Practical-7:</u> Write a program "DivideByZero" that takes two numbers a and b as input, computes a/b, and invokes Arithmetic Exception to generate a message when the denominator is zero.**

```
import java.util.Scanner;


public class SevenDivideByZero {
   public static void main(String[] args) {
      int div=0;
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter two numbers: ");
      int n1 = sc.nextInt();
      int n2 = sc.nextInt();
      try{
         div = n1/n2;
      }

      catch(ArithmeticException e){
         System.out.println("Divide By Zero Error");
         System.out.println("n1 / n2 = "+(div));
      }
   }
}
```

Output:

```
Enter two numbers: 6
0
Divide By Zero Error
n1 / n2 = 0
```

- **<u>Practical-8</u>: Write a program to show the use of nested try statements that emphasizes the sequence of checking for catch handler statements.**

```
public class EIghtNestedTryBlock {
   public static void main(String[] args) {
      try{
         try{
            int n = 62/0;
         }
         catch(ArithmeticException e){
            System.out.println(e);
         }

         try{
            int a[] = new int[5];
            a[5]=62;
         }
         catch (ArrayIndexOutOfBoundsException e){
            System.out.println(e);
         }
      }
      catch (Exception e){
         System.out.println("Handled the Exception");
      }
   }
}
```

Output:

```
java.lang.ArithmeticException: / by zero
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
```

- **<u>Practical-9:</u> Write a program to create your own exception types to handle situation specific to your application (Hint: Define a subclass of Exception which itself is a subclass of Throwable).**

```
// Use of Throw
import java.util.Scanner;
class LongLength extends Exception{
   LongLength(String str){
      super(str);
   }
}

public class NineUserDefinedException1 {
   public static void main(String[] args) throws Exception {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter 10 digit Mobile Number: ");
      try {
      long mobileNo = sc.nextLong();
      int len = (int) (Math.log10(mobileNo) + 1);
         if (!(len == 10)) {
            throw new Exception("Please enter 10 digit numbers");
         }
      }
      catch (Exception e) {
         System.out.println("Caught Exception");
         System.out.println(e);
      }
   }
}
```

Output:

```
Enter 10 digit Mobile Number: 3695924d
Caught Exception
java.util.InputMismatchException
```

// Use of Finally

```
public class NineUserDefinedException2 {
   public static void main(String[] args) {
      try {
         System.out.println("Try Block");
         int data = 65/0;
         System.out.println(data);
      }
      catch (ArithmeticException e){
         System.out.println(e);
      }
      finally {
         System.out.println("Finally gets executed");
      }
   }
}
```

Output:

```
Try Block
java.lang.ArithmeticException: / by zero
Finally gets executed
```

// Use of Throws

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class NineUserDefinedException3 {
   static void input() throws InputMismatchException {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter any number: ");
      int num = sc.nextInt();
      throw new InputMismatchException("Demo");
   }
   public static void main(String[] args) {
      try{
         input();
      }
      catch (InputMismatchException e){
         System.out.print("Caught ");
         System.out.println(e);
      }
   }
}
```

Output:

```
Enter any number: 6516v
Caught java.util.InputMismatchException
```

- **<u>Practical-10:</u> Write a small application in Java to develop Banking Application in which user deposits the amount Rs 1000.00 and then start withdrawing of Rs 400.00, Rs 300.00 and it throws exception "Not Sufficient Fund" when user withdraws Rs. 500 thereafter.**

```
import java.util.Scanner;


public class TenBank {
   static int Balance;
   static int exceptionCached = 0;


   // Deposit
   static void deposit(int amountDeposited){
      Balance += amountDeposited;
   }


   // Withdraw
   static void withdraw(int amountWithdrawed){
      try{
         if (amountWithdrawed > Balance){
            exceptionCached = 1;
            throw new Exception("NotSufficientFundException");
         }
         else{
            Balance -= amountWithdrawed;
         }
      }
      catch (Exception e){
         System.out.println(e);
      }
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
```

```
        int answer = 1;
        int amountWithdraw;
        int amountDeposited;

        try{
            System.out.print("Enter amount to be deposited: ");
            amountDeposited = sc.nextInt();
            deposit(amountDeposited);

            while (answer==1 && exceptionCached==0){
                System.out.print("Enter amount to withdraw: ");
                amountWithdraw = sc.nextInt();
                withdraw(amountWithdraw);
                if (exceptionCached==0){
                    System.out.print("Do you want to withdraw more money???\n1-Yes\n0-
No\nEnter: ");
                    answer = sc.nextInt();
                }
            }
            System.out.println("Balance: " + Balance);

        }
        catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output

```
Enter amount to be deposited: 1000
Enter amount to withdraw: 400
Do you want to withdraw more money???
1-Yes
0-No
Enter: 1
Enter amount to withdraw: 300
Do you want to withdraw more money???
1-Yes
0-No
Enter: 1
Enter amount to withdraw: 500
java.lang.Exception: NotSufficientFundException
Balance: 300
```

- **<u>Practical-11</u>: Write a program to handle ArrayIndexOutOfBounds exception for binary search.**public class ElevenArrayIndexOutOfBound {

```
   public static void main(String[] args) {
      int arr[] = {1,2,3,4,5};
      try{
         System.out.println(arr[5]);
      }
      catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Array Index Out Of Bound Exception");
      }
   }
}
```

Output:

```
Array Index Out Of Bound Exception
```

]

- **Practical-12: Write a Java Program that demonstrates thread class and few methods.**

```
public class Main extends Thread{
   public static void main(String[] args) {
      Main t1 = new Main();
      t1.start();
      Thread t2 = currentThread();

      System.out.println("Id: "+currentThread().getId());
      System.out.println("Name: "+currentThread().getName());
      System.out.println("Priority: "+currentThread().getPriority());
      t2.setName("Current Thread");
      System.out.println("Name: "+currentThread().getName());
   }
}
```

Output:

```
Id: 1
Name: main
Priority: 5
Name: Current Thread
```

- **Practical-13: Write a program to demonstrate thread example by implementing runnable interface.**

```
// Taking in TestThread
class testThread implements Runnable{
   public void run() {
      System.out.println("Running the Thread from different class");
   }
}

public class Main{
   public static void main(String[] args) {
      testThread t1 = new testThread();
      Thread t2 = new Thread(t1);
      t2.start();
   }
}
```

Output:

```
Running the Thread from different class
```

- **Practical-14: Write a program to demonstrate priorities among multiple threads.**

```
import java.lang.*;
public class FourteenPriorityThread extends Thread {
   public void run(){
      System.out.println("Inside the run method");
   }
   public static void main(String[] args) {
      FourteenPriorityThread t1 = new FourteenPriorityThread();
      FourteenPriorityThread t2 = new FourteenPriorityThread();
      FourteenPriorityThread t3 = new FourteenPriorityThread();

      System.out.println("Priority of Thread t1: " + t1.getPriority());
      System.out.println("Priority of Thread t2: " + t2.getPriority());
      System.out.println("Priority of Thread t2: " + t3.getPriority());

      t1.setPriority(8);
      t2.setPriority(5);
      t3.setPriority(2);

      System.out.println("Priority of Thread t1: " + t1.getPriority());
      System.out.println("Priority of Thread t2: " + t2.getPriority());
      System.out.println("Priority of Thread t2: " + t3.getPriority());

      Thread.currentThread().setPriority(10);

      System.out.println("\Priority of the main thread: " +
Thread.currentThread().getPriority());
   }
}
```

Output:

```
Priority of Thread t1: 5
Priority of Thread t2: 5
Priority of Thread t2: 5
Priority of Thread t1: 8
Priority of Thread t2: 5
Priority of Thread t2: 2

Priority of the main thread: 10
```

- **Practical-15: Write a program to demonstrate multithread communication by implementing synchronization among threads (Hint: you can implement a simple producer and consumer problem).**

```
class BookTheatreSeat{
   int totalSeats = 10;
   void bookSeat(int seats){
      synchronized(this){
         if (seats <= totalSeats) {
            System.out.println("Seats booked succesfully");
            totalSeats -= seats;
            System.out.println("Seats available: " + totalSeats);
         }
         else {
            System.out.println("Sorry! " + seats + " seats cannot be booked");
            System.out.println("Seats available: " + totalSeats);
         }
      }
   }
}



class BookMovie extends Thread{
   static BookTheatreSeat bts;
   int seats;
   public void run(){
      bts.bookSeat(seats);
   }

   public static void main(String[] args) {
      bts = new BookTheatreSeat();

      BookMovie m1 = new BookMovie();
```

```
        m1.seats = 7;
        m1.start();

        BookMovie m2 = new BookMovie();
        m2.seats = 6;
        m2.start();
    }
}
```

Output:

```
Seats booked succesfully
Seats available: 3
Sorry! 6 seats cannot be booked
Seats available: 3
```

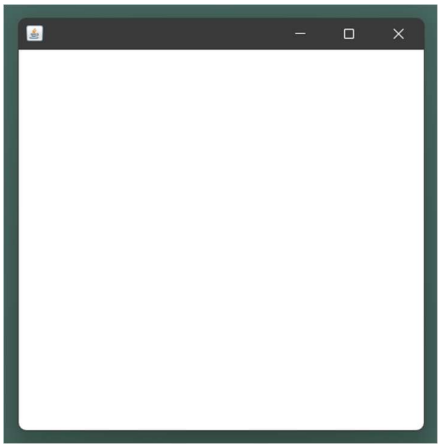
MODULE: 4

Practical-1: Write a program to demonstrate different Window handling events.

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

public class OneWindowListener extends Frame implements WindowListener {
   OneWindowListener() {
      addWindowListener(this);
      setSize (400, 400);
      setLayout (null);
      setVisible (true);
   }
   public static void main(String[] args) {
      new OneWindowListener();
   }
   public void windowActivated (WindowEvent arg0) {
      System.out.println("activated");
   }
   public void windowClosed (WindowEvent arg0) {
      System.out.println("closed");
   }
   public void windowClosing (WindowEvent arg0) {
      System.out.println("closing");
      dispose();
   }
   public void windowDeactivated (WindowEvent arg0) {
      System.out.println("deactivated");
   }
   public void windowDeiconified (WindowEvent arg0) {
      System.out.println("deiconified");
```

```
    }
    public void windowIconified(WindowEvent arg0) {
        System.out.println("iconified");
    }
    public void windowOpened(WindowEvent arg0) {
        System.out.println("opened");
    }
}
```

Output:



```
activated
opened
deactivated
activated
closing
deactivated
closed
```

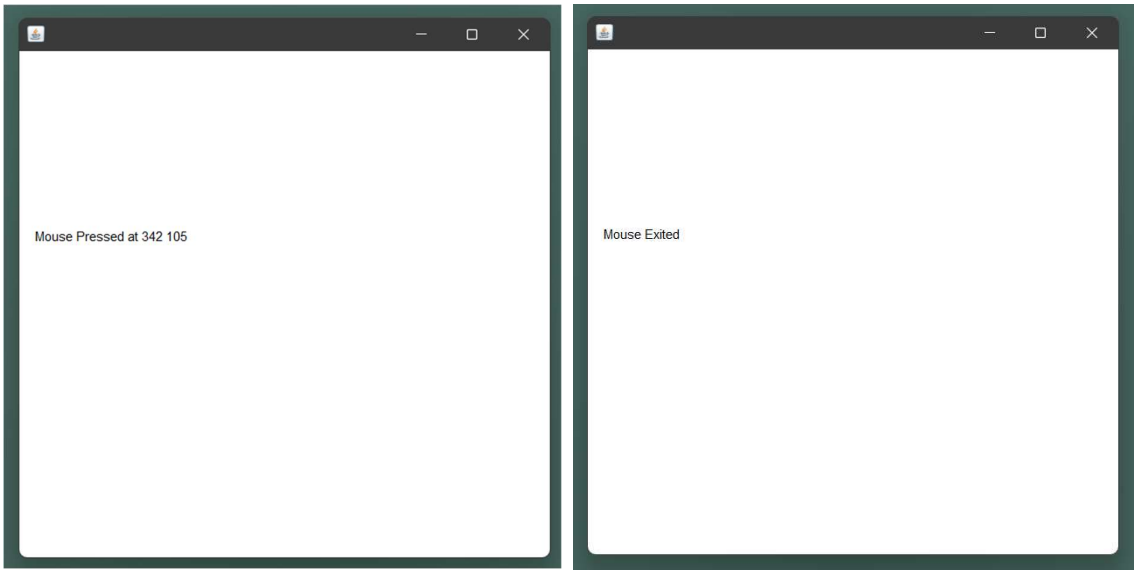
Practical-2: Write a program to demonstrate different mouse handling events like mouseClicked(), mouseEntered(), mouseExited(), mousePressed, mouseReleased() and mouseDragged().

// Mouse Listener

```
import java.awt.*;
import java.awt.event.*;
public class TwoMouseListener extends Frame implements MouseListener{
   Label l;
   TwoMouseListener(){
      addMouseListener(this);
      l=new Label();
      l.setBounds(20,50,300,300);
      add(l);
      setSize(500,500);
      setLayout(null);
      setVisible(true);
   }
   public static void main(String[] args) {
      new TwoMouseListener();
   }
   public void mouseClicked(MouseEvent e) {
      l.setText("Mouse Clicked at " + e.getX() + " " + e.getY());
      System.out.println(getAlignmentX() + " " + getAlignmentY());
   }
   public void mousePressed(MouseEvent e) {
      l.setText("Mouse Pressed at " + getX() + " " + getY());
      System.out.println(getAlignmentX() + " " + getAlignmentY());
   }
   public void mouseEntered(MouseEvent e) {
      l.setText("Mouse Entered");
   }
   public void mouseExited(MouseEvent e) {
```

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

Output:



// **MouseMotionListener**

```
import java.awt.*;
import java.awt.event.*;
public class TwoMouseMotionListener extends Frame implements MouseMotionListener{
    Label l;
    TwoMouseMotionListener(){
        addMouseMotionListener(this);
        l=new Label();
        l.setBounds(20,50,100,20);
        add(l);
        setSize(500,500);
```

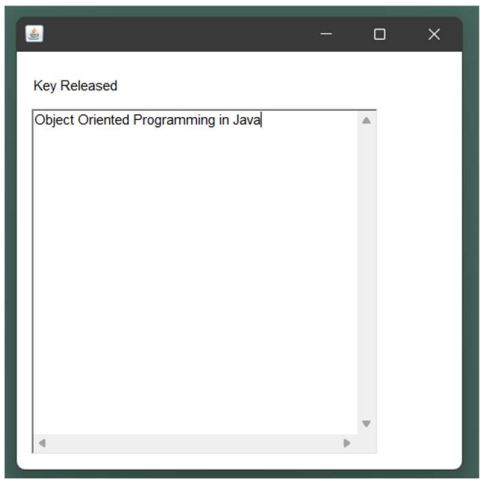
```
        setLayout(null);
        setVisible(true);
    }
    public static void main(String[] args) {
        new TwoMouseMotionListener();
    }
    public void mouseDragged(MouseEvent e) {
        Graphics g=getGraphics();
        g.setColor(Color.RED);
        g.fillOval(e.getX(),e.getY(),20,20);
    }
    public void mouseMoved(MouseEvent e) {
        l.setText("mouse is moved to point " + e.getX() + " " + e.getY());
    }
}
```

Output:

Practical-3: Write a program to demonstrate different keyboard handling events.

```
import java.awt.*;
import java.awt.event.*;
public class ThreeKeyListener extends Frame implements KeyListener {
   Label l;
   TextArea area;
   ThreeKeyListener() {
      l = new Label();
      l.setBounds(20, 50, 100, 20);
      area = new TextArea();
      area.setBounds(20, 80, 300, 300);
      area.addKeyListener(this);
      add(l);
      add(area);
      setSize(400, 400);
      setLayout(null);
      setVisible(true);
   }
   public void keyPressed(KeyEvent e) {
      l.setText("Key Pressed");
   }
   public void keyReleased(KeyEvent e) {
      l.setText("Key Released");
   }
   public void keyTyped(KeyEvent e) {
      l.setText("Key Typed");
   }
   public static void main(String[] args) {
      new ThreeKeyListener();
   }
}
```

Output



Practical-4: Write a program to generate a window without an applet window using main() function.

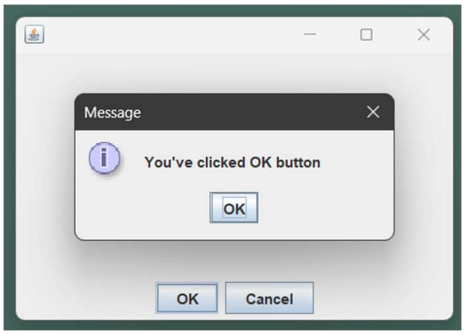
```
import java.awt.Frame;
public class FourApplet extends Frame{
   FourApplet(String title){
      super();
      this.setTitle(title);
      this.setVisible(true);
   }

   public static void main(String args[]){
      FourApplet window = new FourApplet("Create Window Example");
   }
}
```

Output:

Practical-5: Write a program to demonstrate the use of push buttons.

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

public class FivePush {
   public static void main(String[] args) {
      final JFrame frame = new JFrame();
      JButton btnOK = new JButton("OK");
      btnOK.addActionListener(
            new ActionListener(){
               public void actionPerformed(ActionEvent e) {
                  JOptionPane.showMessageDialog(frame,"You've clicked OK button");
               }
            });
      JButton btnCancel = new JButton("Cancel");
      btnCancel.addActionListener(
            new ActionListener(){
               public void actionPerformed(ActionEvent e) {
                  JOptionPane.showMessageDialog(frame,"You've clicked Cancel button"
                  );
               }
            });
      JPanel buttonPanel = new JPanel( );
      buttonPanel.add(btnOK);
      buttonPanel.add(btnCancel);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(300, 200);
      frame.getContentPane( ).add(buttonPanel,BorderLayout.SOUTH);
      frame.setVisible(true);
   }
}
```

Output:

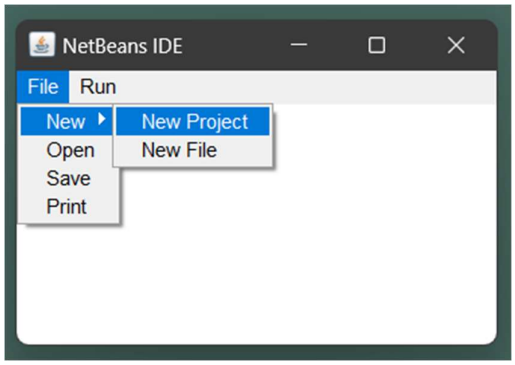


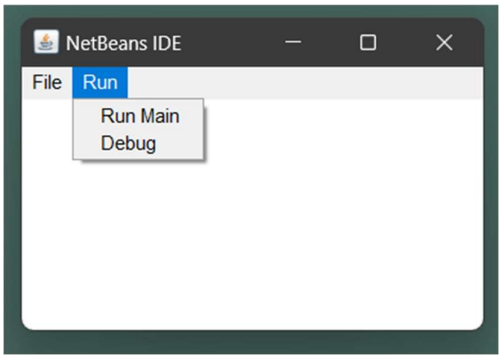
Practical-6: WAP to create a Menu using the frame.

```
import java.awt.*;
class MenuExample {
   MenuExample(){
      Frame f= new Frame("NetBeans IDE");
      MenuBar mb=new MenuBar();

      // File Menu
      Menu menu=new Menu("File");
      Menu submenu=new Menu("New");
      MenuItem i1=new MenuItem("New Project");
      MenuItem i2=new MenuItem("New File");
      MenuItem i3=new MenuItem("Open");
      MenuItem i4=new MenuItem("Save");
      MenuItem i5=new MenuItem("Print");

      submenu.add(i1);
      submenu.add(i2);
      menu.add(submenu);
      menu.add(i3);
      menu.add(i4);
      menu.add(i5);
      mb.add(menu);

      // Run Menu
      Menu Run=new Menu("Run");
      MenuItem run=new MenuItem("Run Main");
      MenuItem debug=new MenuItem("Debug");

      Run.add(run);
      Run.add(debug);
      mb.add(Run);
```

```
        f.setMenuBar(mb);
        f.setSize(300,200);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[]) {
        new MenuExample();
    }
}
```

Output:





Practical-7: WAP to create a Frame that display the student information.

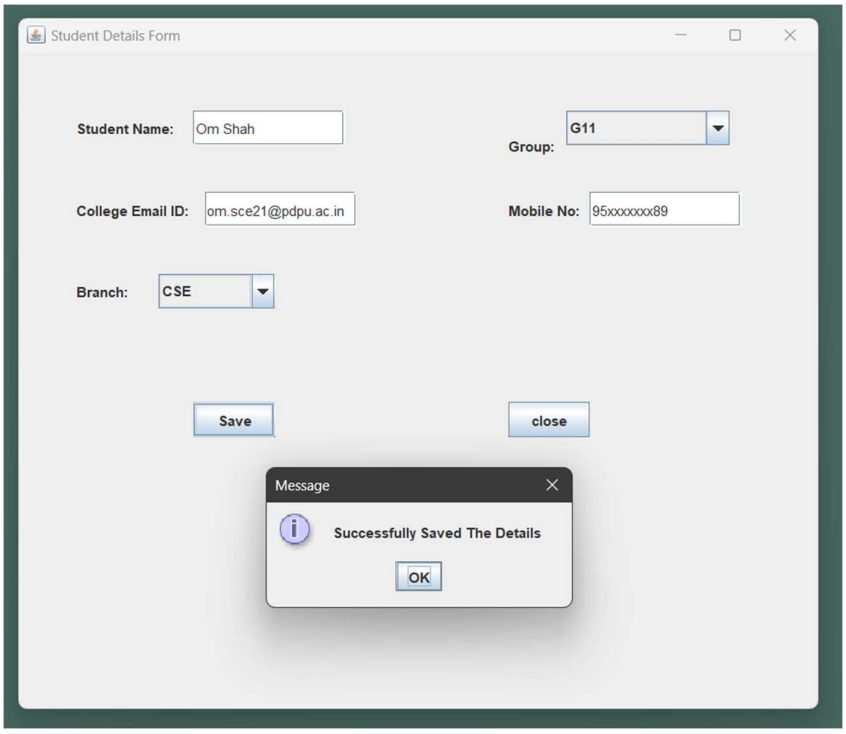
```
import javax.swing.*;
import java.awt.event.*;
import java.io.*;

public class SevenStudent {
   public static void StudentInfo() {
      JFrame f = new JFrame("Student Details Form");

      JLabel l1, l2, l3, l4, l5;
      JTextField t1, t2, t3;
      JComboBox j1, j2;
      JButton b1, b2;

      l1 = new JLabel("Student Name:");
      l1.setBounds(50, 50, 100, 30);
      l2 = new JLabel("College Email ID:");
      l2.setBounds(50, 120, 120, 30);
      l3 = new JLabel("Branch:");
      l3.setBounds(50, 190, 50, 30);
      l4 = new JLabel("Group:");
      l4.setBounds(420, 50, 70, 60);
      l5 = new JLabel("Mobile No:");
      l5.setBounds(420, 120, 70, 30);

      t1 = new JTextField();
      t1.setBounds(150, 50, 130, 30);
      t2 = new JTextField();
      t2.setBounds(160, 120, 130, 30);
      t3 = new JTextField();
      t3.setBounds(490, 120, 130, 30);
```

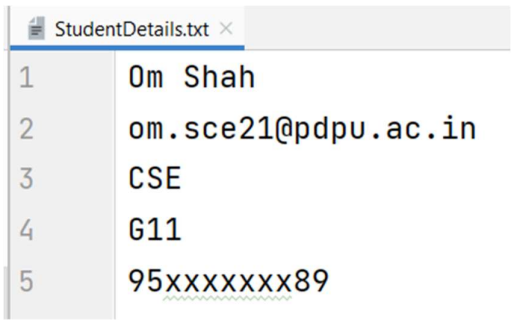
```
String s1[]  = { " ", "CSE", "ECE", "EEE", "CIVIL", "MEC", "Others" };
String s2[]  = { " ", "G1", "G2", "G3", "G4", "G5", "G6", "G7", "G8", "G9", "G10",
"G11", "G12" };

j1 = new JComboBox(s1);
j1.setBounds(120, 190, 100, 30);
j2 = new JComboBox(s2);
j2.setBounds(470, 50, 140, 30);

b1 = new JButton("Save");
b1.setBounds(150, 300, 70, 30);
b2 = new JButton("close");
b2.setBounds(420, 300, 70, 30);

b1.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      String s1 = t1.getText();
      String s2 = t2.getText();
      String s3 = j1.getSelectedItem() + "";
      String s4 = j2.getSelectedItem() + "";
      String s5 = t3.getText();
      if (e.getSource() == b1) {
         try {
            FileWriter w= new FileWriter("StudentDetails.txt", true);
            w.write(s1 + "\n");
            w.write(s2 + "\n");
            w.write(s3 + "\n");
            w.write(s4 + "\n");
            w.write(s5 + "\n");
            w.close();
         }
         catch (Exception ae) {
```

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

        JOptionPane.showMessageDialog(f,"Successfully Saved" + " The Details");
    }
});

b2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e)
    {
        f.dispose();
    }
});

f.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
});

f.add(l1);
f.add(t1);
f.add(l2);
f.add(t2);
f.add(l3);
f.add(j1);
f.add(l4);
f.add(j2);
f.add(l5);
f.add(t3);
```

```
        f.add(b1);
        f.add(b2);
        f.setLayout(null);
        f.setSize(700, 600);
        f.setVisible(true);
    }
    public static void main(String args[]) {
        StudentInfo();
    }
}
```

Output:

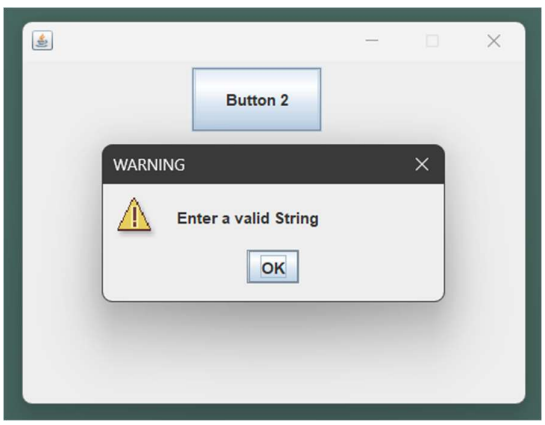




Practical-8: WAP to create a Dialogbox.

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

class DialogueClass extends JFrame implements ActionListener {
   JButton b1;
   DialogueClass() {
      this.setLayout(null);
      b1 = new JButton("Button 2");
      b1.setBounds(130, 05, 100, 50);
      this.add(b1);
      b1.addActionListener(this);
   }

   public void actionPerformed(ActionEvent evt) {
      if (evt.getSource() == b1) {
         JOptionPane.showMessageDialog(this, "Enter a valid String",
"WARNING", JOptionPane.WARNING_MESSAGE);
      }
   }
}

class EightDialogue {
   public static void main(String args[]) {
      DialogueClass f = new DialogueClass();
      f.setBounds(200, 200, 400, 300);
      f.setResizable(false);
      f.setVisible(true);
   }
}
```

Output



Practical-9: WAP to implement the FlowLayout and BorderLayout.

// Flow Layout

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

class Layout extends JFrame {
   JLabel l1, l2, l3, l4, l5;
   public Layout() {
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      FlowLayout layout = new FlowLayout();
      this.setLayout(layout);
      l1 = new JLabel("Label 1 ");
      l2 = new JLabel("Label 2 ");
      l3 = new JLabel("Label 3 ");
      l4 = new JLabel("Label 4 ");
      l5 = new JLabel("Label 5 ");
      this.add(l1);
      this.add(l2);
      this.add(l3);
      this.add(l4);
      this.add(l5);
   }
}

class NineFlowLayout {
   public static void main(String[] args) {
      Layout f = new Layout();
      f.setTitle("Example of FlowLayout");
      f.setBounds(200, 100, 600, 400);
      f.setVisible(true);
   }
}
```

Output

// Border Layout

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

class BoderLayoutDemo extends JFrame {
    BoderLayoutDemo() {
        JPanel pa = new JPanel();
        pa.setLayout(new BorderLayout());
        pa.add(new JButton("Welcome"), BorderLayout.NORTH);
        pa.add(new JButton("OOP"), BorderLayout.SOUTH);
        pa.add(new JButton("Layout"), BorderLayout.EAST);
        pa.add(new JButton("Border"), BorderLayout.WEST);
        pa.add(new JButton("Java"), BorderLayout.CENTER);
        add(pa);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 300);
        setVisible(true);
    }
}

class NineBorderLayout {

    // Driver code
    public static void main(String[] args) {
        new BoderLayoutDemo();
    }
}
```

Output

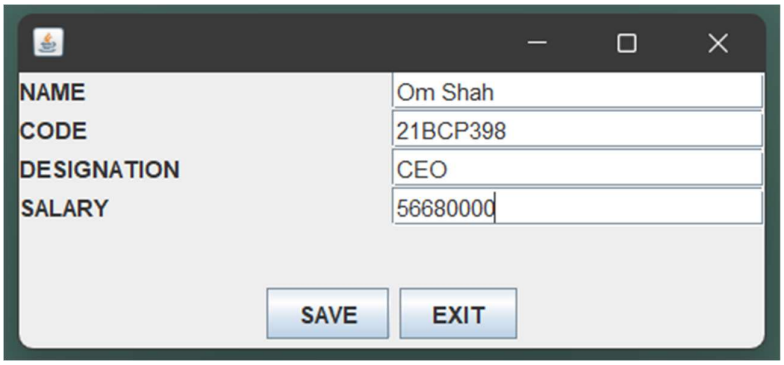
Practical-10: WAP to implement the GridLayout and CardLayout.

// Grid Layout

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

public class TenGridLayout extends JFrame {
   TenGridLayout() {
      JPanel p1 = new JPanel();
      p1.setLayout(new GridLayout(4, 2));
      FlowLayout layout = new FlowLayout();
      JPanel p2 = new JPanel();
      p2.setLayout(layout);
      JLabel one, two, three, four;
      JTextField tname, tsalary, tcode, tdesig;
      JButton buttonSave, buttonExit;

      one = new JLabel("NAME");
      tname = new JTextField(20);
      two = new JLabel("CODE");
      tcode = new JTextField(20);
      three = new JLabel("DESIGNATION");
      tdesig = new JTextField(20);
      four = new JLabel("SALARY");
      tsalary = new JTextField(20);
      buttonSave = new JButton("SAVE");
      buttonExit = new JButton("EXIT");
      p1.add(one);
      p1.add(tname);
      p1.add(two);
      p1.add(tcode);
      p1.add(three);
      p1.add(tdesig);
```

```
        p1.add(four);
        p1.add(tsalary);
        p2.add(buttonSave);
        p2.add(buttonExit);
        add(p1, "North");
        add(p2, "South");
        setVisible(true);
        this.setSize(400, 180);
    }
    public static void main(String args[]) {
        new TenGridLayout();
    }
}
```

Output:

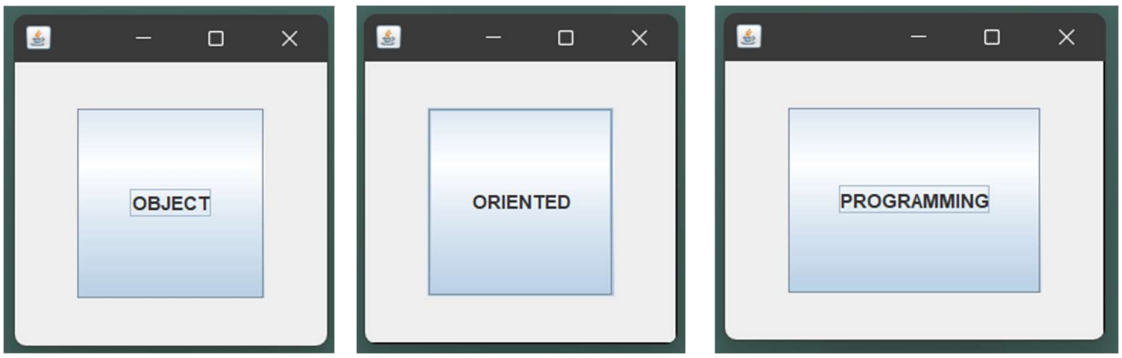


// Card Layout

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

public class TenCardLayout extends JFrame implements ActionListener {
    CardLayout card;
    JButton b1, b2, b3;
    Container c;
```

```
    TenCardLayout(){
        c = getContentPane();
        card = new CardLayout(40, 30);
        c.setLayout(card);
        b1 = new JButton("OBJECT");
        b2 = new JButton("ORIENTED");
        b3 = new JButton("PROGRAMMING");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        c.add("a", b1);
        c.add("b", b2);
        c.add("c", b3);
    }
    public void actionPerformed(ActionEvent e) {
        card.next(c);
    }
    public static void main(String[] args) {
        TenCardLayout cl = new TenCardLayout();
        cl.setSize(400, 400);
        cl.setVisible(true);
        cl.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Output:



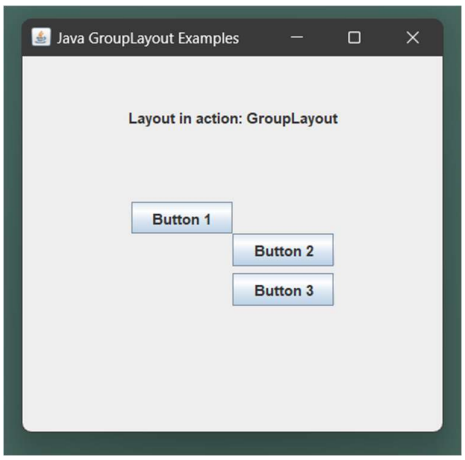
Practical-11: WAP to implement the GroupLayout and BoxLayout.

// Group Layout

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

public class ElevenGroupLayout {
   private JFrame mainFrame;
   private JLabel headerLabel, statusLabel, msglabel;
   private JPanel controlPanel;
   public ElevenGroupLayout(){
      prepareGUI();
   }
   public static void main(String[] args){
      ElevenGroupLayout GroupLayoutDemo = new ElevenGroupLayout();
      GroupLayoutDemo.showGroupLayoutDemo();
   }
   private void prepareGUI(){
      mainFrame = new JFrame("Java GroupLayout Examples");
      mainFrame.setSize(400, 400);
      mainFrame.setLayout(new GridLayout(3, 1));
      headerLabel = new JLabel("", JLabel.CENTER);
      statusLabel = new JLabel("", JLabel.CENTER);
      statusLabel.setSize(350, 100);
      mainFrame.addWindowListener(new WindowAdapter(){
         public void windowClosing(WindowEvent windowEvent)
         {
            System.exit(0);
         }
      });
      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());
```

```
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }
    private void showGroupLayoutDemo(){
        headerLabel.setText("Layout in action: GroupLayout");
        JPanel panel = new JPanel();
        panel.setSize(200, 200);
        GroupLayout layout = new GroupLayout(panel);
        layout.setAutoCreateGaps(true);
        layout.setAutoCreateContainerGaps(true);
        JButton btn1 = new JButton("Button 1");
        JButton btn2 = new JButton("Button 2");
        JButton btn3 = new JButton("Button 3");
        layout.setHorizontalGroup(layout.createSequentialGroup()
                .addComponent(btn1)
                .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                                .addComponent(btn2)
                                .addComponent(btn3))));

        layout.setVerticalGroup(layout.createSequentialGroup()
                .addComponent(btn1)
                .addComponent(btn2)
                .addComponent(btn3));
        panel.setLayout(layout);
        controlPanel.add(panel);
        mainFrame.setVisible(true);
    }
}
```

Output

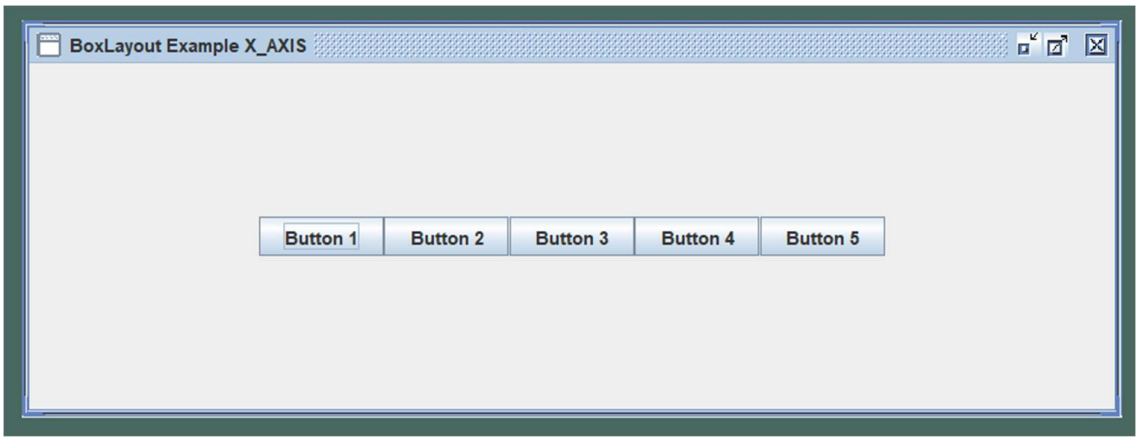


// Box Layout

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.BoxLayout;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import java.awt.Insets;

public class ElevenBoxLayout {
   public static void main(String[] args){
      JFrame.setDefaultLookAndFeelDecorated(true);
      JFrame frame = new JFrame("BoxLayout Example X_AXIS");
      JButton jbtn1, jbtn2, jbtn3, jbtn4, jbtn5;
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JPanel panel = new JPanel();
      BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.X_AXIS);

      panel.setLayout(boxlayout);
      panel.setBorder(new EmptyBorder(new Insets(100, 150, 100, 150)));

      jbtn1 = new JButton("Button 1");
      jbtn2 = new JButton("Button 2");
      jbtn3 = new JButton("Button 3");
      jbtn4 = new JButton("Button 4");
      jbtn5 = new JButton("Button 5");

      panel.add(jbtn1);
      panel.add(jbtn2);
      panel.add(jbtn3);
      panel.add(jbtn4);
      panel.add(jbtn5);
```

```
        frame.add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Output:



Practical-12: Write a program that demonstrates the life cycle of an applet.

```
import java.applet.Applet;
import java.awt.Graphics;
public class TwelveApplet extends Applet {
   public void init() {
      System.out.println("In init()");
   }
   public void start() {
      System.out.println("In start()");
   }
   public void paint(Graphics g) {
      System.out.println("In paint()");
   }
   public void stop() {
      System.out.println("In stop()");
   }
   public void destroy() {
      System.out.println("In destroy()");
   }
}
```

Practical-13: WAP to demonstrate System clock.

```
import java.time.*;
public class ThirteenClock {
   public static void main(String[] args) {
      Clock c = Clock.systemDefaultZone();
      System.out.println(c.getZone());
      System.out.println(c.instant());
   }
}
```

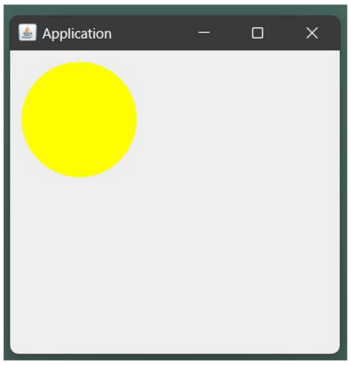
Output:

```
Asia/Calcutta
2022-11-16T13:01:07.564089900Z
```

Practical-14: WAP to demonstrate Painting in applet.

```
import java.awt.*;
import javax.swing.*;
class FourteenPaint extends JPanel {
   JButton jb;
   JTextField jt;
   FourteenPaint() {
      JFrame app = new JFrame("Application");
      app.add(this);
      app.setSize(300,300);
      app.setLocationRelativeTo(null);
      app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      app.setVisible(true);
   }
   public void paintComponent(Graphics g){
      super.paintComponent(g);
      g.setColor(Color.YELLOW);
      g.fillOval(10,10,100,100);

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

Output:



Practical-15: WAP to demonstrate Graphics in applet.

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

public class FifteenGraphics extends JPanel {
   FifteenGraphics() {
      JFrame app = new JFrame("Shapes in Graphics in Java");
      app.add(this);
      app.setSize(400,400);
      app.setLocationRelativeTo(null);
      app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      app.setVisible(true);
   }
   public void paint(Graphics g){
      g.setColor(Color.red);
      g.drawString("Welcome",50, 50);
      g.drawLine(20,30,20,300);
      g.drawRect(70,100,30,30);
      g.fillRect(170,100,30,30);
      g.drawOval(70,200,30,30);

      g.setColor(Color.pink);
      g.fillOval(170,200,30,30);
      g.drawArc(90,150,30,30,30,270);
      g.fillArc(270,150,30,30,0,180);
   }
   public static void main(String[] args) {
      new FifteenGraphics();
   }
}
```

Output