Department of Computer Science and Engineering

Lab Manual

for

Object Oriented Programming Laboratory

CS3381

Regulation 2021
CS3381 – III Semester



ADHIPARASAKTHI ENGINEERING COLLEGE

Melmaruvathur, Chengalpet District- 603319.

INDEX

Expt. No.	Name of the Experiments	Page No.
1.	Sequential Search	1
2.	Binary Search	3
3.	Selection Sort	6
4.	Insertion Sort	9
5.	Stack and Queue	11
6.	Generation of Payslip using Inheritance	17
7.	Area of Shapes using Abstract Class	24
8.	Area of Shape using Interface	28
9.	Exception Handling	32
10.	User defined Exception	34
11.	Implementing Multithread Application	37
12.	File Operations – File Content Display	41
13.	File Operations – File Copy	43
14.	File Operations – Reverse File	46
15.	File Operations - File Content Case Conversion	49
16.	Generic Class	52
17.	Application using Javafx Controls Layouts and Menus	54

APEC

EX. NO: 1	SEQUENTIAL SEARCH

Aim:

To write a Java program to search element in sequential based order.

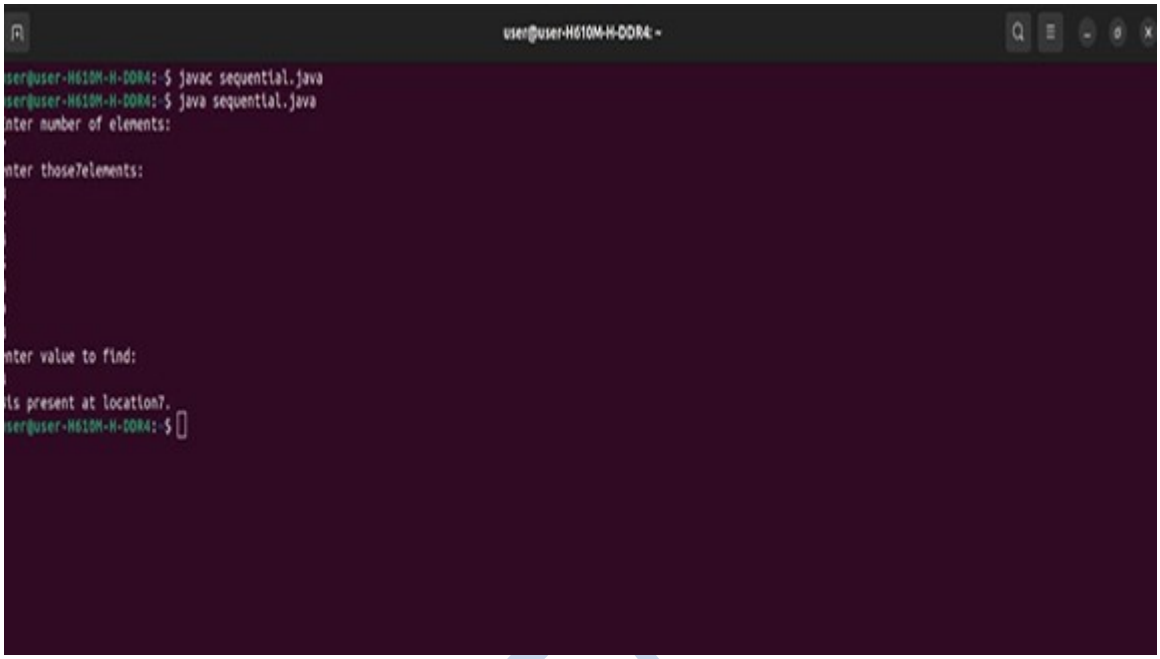
Algorithm:

Step 1: Start
Step 2: Get the array element
Step 3: Get the key element to search
Step 4: Traverse and match the key element with array elements
Step 5: If key element is found return index position
Step 6: If key element not found return 1
Step 7: Stop

Program:

```
import java.util.Scanner;
class sequential
{
        public static void main(String args[])
                {
                        int c, n, search, array[];
                        Scanner in = new Scanner(System.in);
                        System.out.println("Enter number of elements:");
                          n = in.nextInt();
                         array = new int[n];
                        System.out.println("Enter those " + n + " elements:");
                       for (c = 0; c < n; c++)
                        array[c] = in.nextInt();
                        System.out.println("Enter value to find:");
                       search = in.nextInt();
                       for (c = 0; c < n; c++)
                       {
                                if (array[c] == search)
                                {
                                        System.out.println(search + " is present at location " + (c + 1) + ".");
                                       break;
                                }
                         }
                 if (c == n)
                   System.out.println(search + " isn't present in array.");
                }
         }
```

Output:



Result:

Thus, the sequential program has been executed successfully.

EX. NO: 2	BINARY SEARCH

Aim:

To write a java program to demonstrate binary search.

Algorithm:

Step 1: Start

Step 2: Begin with the mid element of the whole array as a search key.

Step 3: If the value of the search key is equal to the item, then return an index of the

Search key.

Step 4: Or if the value of the search key is less than the item in the middle of the interval,

Narrow the interval to the lower half.

Step 5: Otherwise, narrow it to the upper half.

Step 6: Repeatedly check from the second point until the value is found or the interval is

Empty.

Step 7: Stop.

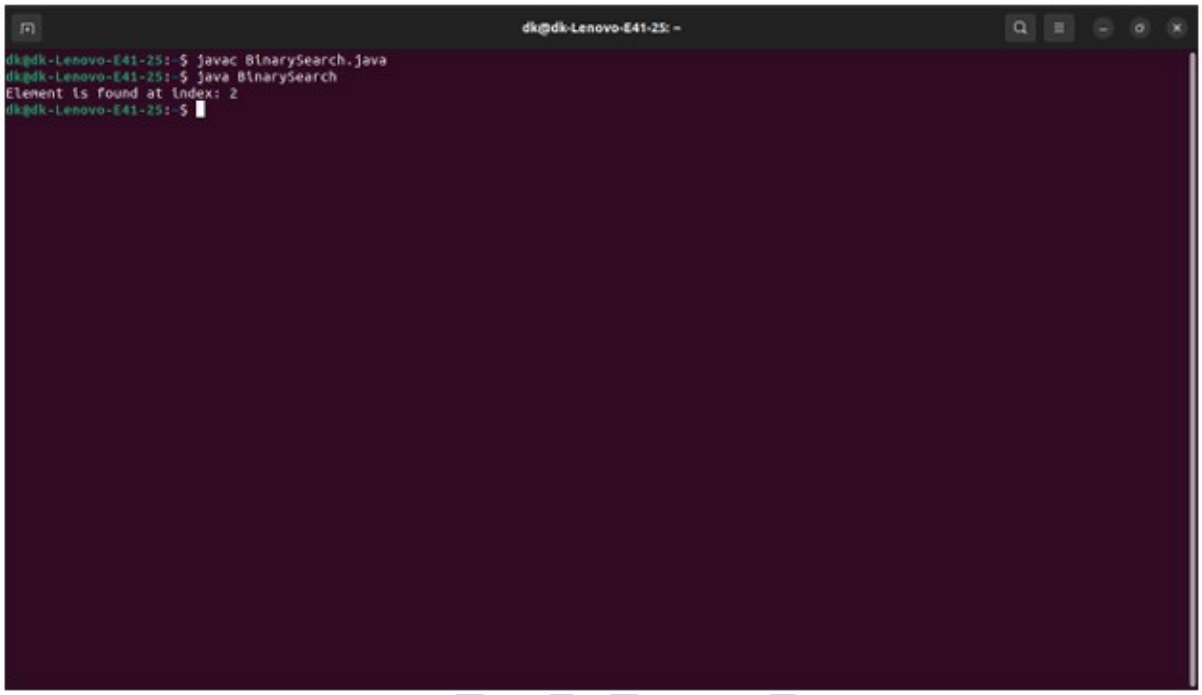
Program:

```
class BinarySearch
{
	public static void binarySearch(int arr[], int first, int last, int key)
		{
			int mid = (first + last)/2;
			while( first <= last )
			{
				if ( arr[mid] < key )
				{
					first = mid + 1;
				}
				else if ( arr[mid] == key )
				{
					System.out.println("Element is found at index: " + mid); break;
```

```
                }
                else
                {
                        last = mid - 1;
                }
                mid = (first + last)/2;
                }
                if ( first > last )
                        System.out.println("Element is not found!");
        }
        public static void main(String args[])
        {
                int arr[] = {10,20,30,40,50};
                int key = 30;
                int last=arr.length-1;
                binarySearch(arr,0,last,key);
        }
}
```

APEC

Output:



Result:

Thus, the binary search program has been executed successfully.

EX. NO: 3	SELECTION SORT

Aim:

To write a Java program for selection sort.

Algorithm:

Step 1: Start.

Step 2: Initialize an array arr with definite values.

Step 3: Assign the find element to a variable.

Step 4: Check whether the variable is greater than the next element.

Step 5: If greater swap the two values.

Step6: Repeat Step 3, 4&5 till n-1 times.

Step 7: Print the array.

Step 8: Stop.

Program:

```
import java.io.*;
class SelectionSort
{

    public static void SelectionSort(int[]arr)
    {
        for(int i=0;i<arr.length-1;i++)
        {
            int index=i;
            for(int j=i+1;j<arr.length;j++)
            {
            if(arr[j]<arr[index])
            {
```

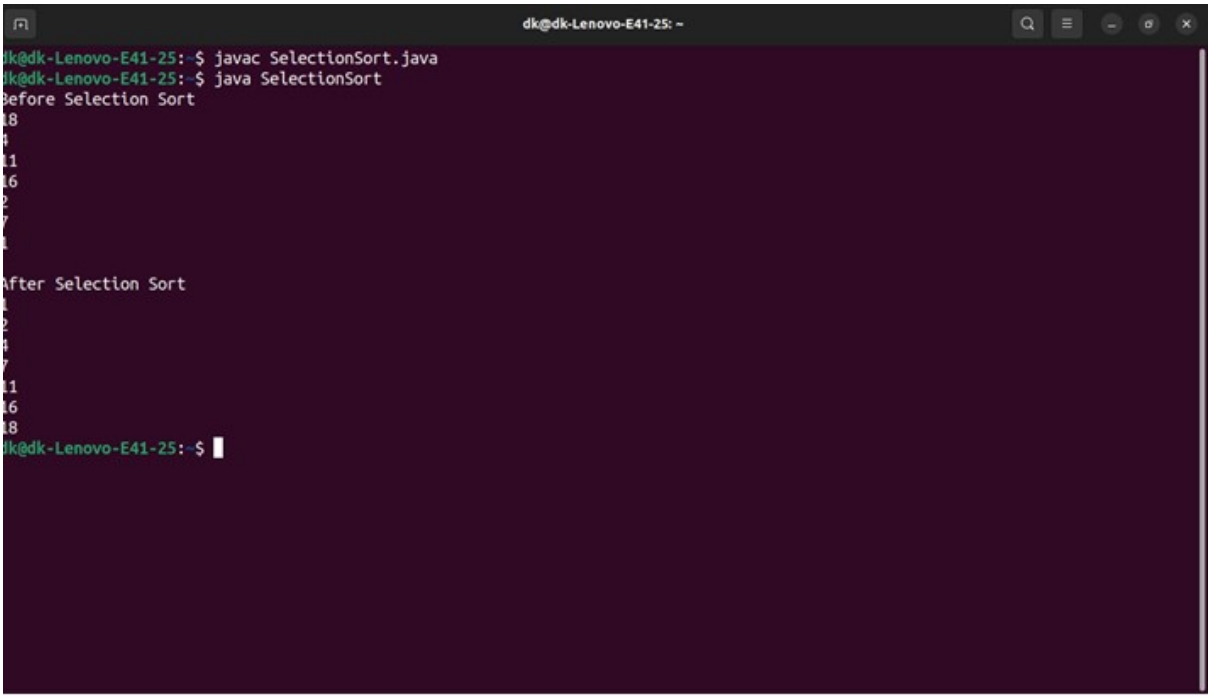
APEC

```
                    index=j;
                }

            }
        int smallernumber=arr[index]; arr[index]=arr[i]; arr[i]=smallernumber;
        }
}
        public static void main(String args[])
        {
            int[] arr1={18,4,11,16,2,7,1};
            System.out.println("Before Selection Sort"); for(int i:arr1)
            {
                System.out.println(i+" ");
            }
            System.out.println(); SelectionSort(arr1);
            System.out.println("After Selection Sort"); for(int i:arr1)
            {
                System.out.println(i+" ");
            }
            }
        }
```

APEC

Output:



Result:

Thus, the selection sort program has been executed successfully.

EX. NO: 4	INSERTION SORT

Aim:

To write a Java program to search element in insertion search-based order.

Algorithm:

Step 1: Start

Step 2: Iterate from array position 1 to array position n over the array.

Step 3: Compare the current element (key) to its predecessor.

Step 4: If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

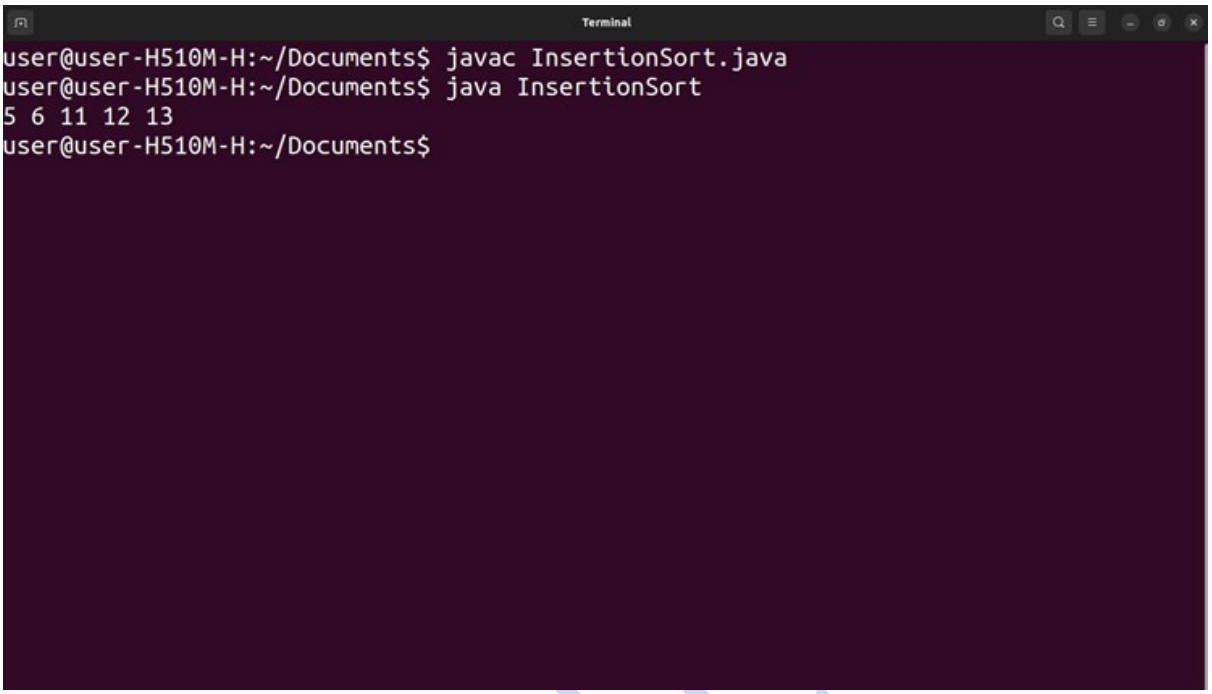
Step 5: Stop

Program:

```
class InsertionSort
{
void sort(int arr[])
{
        for (int i = 1; i < arr.length; ++i)
        {
                int key = arr[i]; int j = i - 1;
                while (j >= 0 && arr[j] > key)
                {
                        arr[j + 1] = arr[j]; j = j - 1;
                }
                arr[j + 1] = key;
        }
}
 static void printArray(int arr[])
{
        int n = arr.length;
        for (int i = 0; i < n; ++i) System.out.print(arr[i] + " "); System.out.println();
}
public static void main(String args[])
{
        int arr[] = { 12, 11, 13, 5, 6 };
        InsertionSort ob = new InsertionSort(); ob.sort(arr);
        printArray(arr);
}}
```

APEC

Output:



Result:

Thus, the Java program to search element in insertion search-based order has been executed successfully.

EX. NO: 5	STACK AND QUEUE

Aim:

To develop stack and queue data structures using classes and packages

Algorithm:

Step 1: Start

Step 2: Create two packages named Stack and Queue

Step 3: Define the required functions for stack and queue data structures

Step 4: Create a Demo file for initializing objects for stack and queue

Step 5: Get data from user through a menu driven output

Step 6: Execute the selected functions by the user

Step 7: Stop

Program:

StackApp.java:

```
package Stack;
public class StackApp
{
        private int maxSize; private long[] stackArray; private int top;
        public StackApp(int s)
        {
                maxSize=s;
                stackArray=new long[maxSize]; top=-1;
        }

/** this method pushes the sum into stack @param num the valueto be pushed */
         public void push(long j)
        {
                stackArray[++top]= j;
        }

        public long pop()
        {
        return stackArray[top--];
        }
```

APEC

```
        public long peek()
        {
        return stackArray[top];
        }
        public boolean isEmpty()
        {
        return(top==-1);
        }
        public boolean isFull()
        {
        return(top==maxSize-1);
        }
        public void display()
        {
        if(top==-1)
        System.out.println("Stack is empty"); for(int i=top;i>=0;i--)
        System.out.println(stackArray[i]);
        }
}
```

QueueApp.java:

```
package Queue;

/** this class demonstrates Queue class
@author meenakshi
@version 1.0
*/
public class QueueApp
{

private int maxSize;
private long[]queArray;
private int front,rear,items;
public QueueApp(int s)
{

maxSize= s;
queArray=new long[maxSize];front=0;
```

APEC

```
rear=0; items=0;
}

/** this method inserts the value into queue and increments the  number of items in queue@param j he value to be inserted */
public void insert(long j)
{
        if (rear==maxSize-1)
        {
                System.out.println("\n queue is full\n");return;
        }
        else
        {
                queArray[rear]=j;rear++;
                return;
        }
}
public long remove()
{
        long temp=queArray[front++];if(front==maxSize)
        front=0; items--; return temp;
}
public long peekfront()
{
        return queArray[front];
}
public boolean isEmpty()
{
        return(items==0);
}
public boolean isFull()
{
        return(items==maxSize);
}
```

APEC

```
public int size()
{
        return items;
}

public void display()
{
        if(front==-1)
        System.out.println("Empty queue");
        for(int i=front;i<rear;i++)
        System.out.print(queArray[i]+"\t");System.out.println("");
}
}
```

Demo.java:

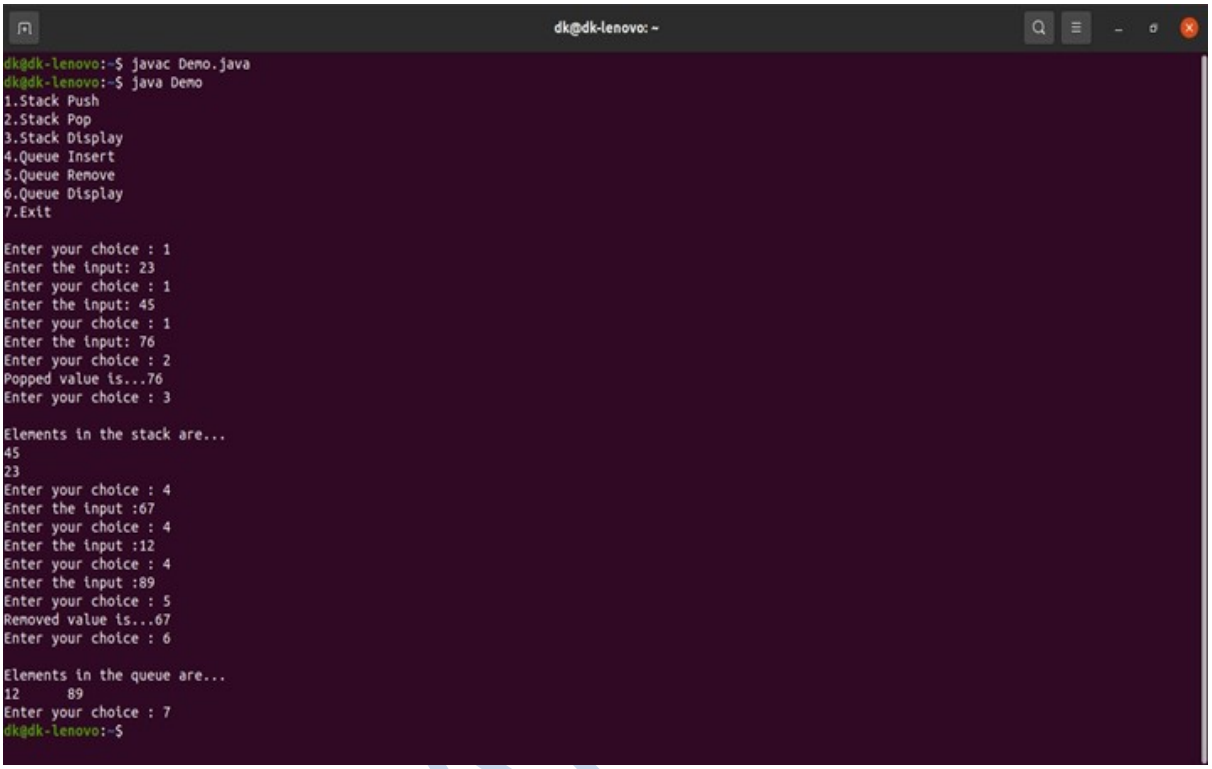
```
import java.util.*; import Stack.StackApp;
import Queue.QueueApp;class Demo
{
        public static void main (String args[])
        {
                int a,c=0;
                Scanner sc= new Scanner(System.in);
                StackApp s =new StackApp(10);
                QueueApp q =new QueueApp(5);
                System.out.print("\n1.Stack Push\n2.Stack Pop\n3.Stack Display\n4.Queue
                Insert\n5.Queue Remove\n6.Queue Display\n7.Exit\n");

        while(true)
        {
                System.out.print("\nEnter your choice : ");
                c=sc.nextInt();
                switch(c)
                {
                case 1:
                        System.out.print("Enter the input: ");
                        a=sc.nextInt();
```

```
                s.push(a);
                s.peek();
                break;
        case 2:
                long value=s.pop();
                System.out.println("Popped value is..."+value);
                break;
        case 3:
                System.out.println("Elements in the stack are...");
                s.display();
                break;
        case 4:
                System.out.print("Enter the input :");
                a=sc.nextInt();
                q.insert(a);break;
        case 5:
                long one=q.remove();
                System.out.println("Removed value is..."+one);
                break;
        case 6:
                System.out.println("Elements in the queue are...");
                q.display();
                break;
        case 7:
                System.exit(0);
        }
    }
  }
}
```

APEC

Output:



Result:

Thus, the stack and queue has been implemented using packages successfully.

EX. NO: 6	GENERATION OF PAY SLIP USING INHERITANCE

Aim:

To create a java application for generating pay slip using inheritance.

Algorithm:

Step 1: Start.

Step 2: Create an Employee class and define the necessary variables.

Step 3: Create a method to calculate gross and net salaries.

Step 4: Create classes such as Programmer, Assistant Professor, Associate Professor, and Professor.

Step 5: Extend these classes to Employee class hierarchically.

Step 6: Give sample info for profession (inherited class).

Step 7: Pass the basic pay as a constructor argument in the main function.

Step 8: Use while along with switch to create or generate multiple pay slips.

Step 9: Stop.

Program:

```
import java.util.*;
class Employee {
   public String Emp_name, Address, Mail_id, Mobile_no;
   public int Emp_id;
   double netSalary, grossSalary, da, hra, pf, club;
   public String companyName = "Space Tech Solutions ";
   void calculationGross(double pay)
   {

      da = pay * 0.97;
      hra = pay * 0.1;
      pf = pay * 0.12;
      club = pay * 0.1;
      grossSalary = pay + da + hra + pf + club;
   }
   void calculationNet()
   {
```

APEC

```
        netSalary = grossSalary - pf - club;

    }
}
class Programmer extends Employee
{
    double basicPay;
    public Programmer(double pay)
    {

        Emp_name = "Ajithnath";
        Address = "Chennai";
        Mail_id = "ak@gmail.com";
        Mobile_no = "1234567890";
        Emp_id = 1234;
        basicPay = pay;
        super.calculationGross(pay);
        super.calculationNet();
    }
    public void show()
    {
        System.out.println("\n\n----------PAY SLIP   ");

        System.out.println("Name: " + Emp_name);
        System.out.println("Employee id : " + Emp_id);
        System.out.println("Address: " + Address);
        System.out.println("Mobile No : " + Mobile_no);
        System.out.println("");
        System.out.println("Basic Pay: " + basicPay);
        System.out.println("Dearness Allowances : " + da);
        System.out.println("House Rent allowances : " + hra);
        System.out.println("");
        System.out.println("Gross Salary : " + super.grossSalary);
```

APEC

```
        System.out.println("Net Salary: " + super.netSalary);
    }
}
class Assistant_Professor extends Employee
{
    double basicPay;
    public Assistant_Professor(double pay)
    {
        Emp_name = "Divyesh Prasad";
        Address = "Hyderabad";
        Mail_id = "dp@gmail.com";
        Mobile_no = "7362884927";
        Emp_id = 1235;
        basicPay = pay;
        super.calculationGross(pay);
        super.calculationNet();
    }
    public void show()
    {
        System.out.println("\n\n----------PAY SLIP   ");
        System.out.println("Name          : " + Emp_name);
        System.out.println("Employee id : " + Emp_id);
        System.out.println("Address                 : " + Address);
        System.out.println("Mobile No : " + Mobile_no);
        System.out.println("                                                ");
        System.out.println("Basic Pay                                       : " + basicPay);
        System.out.println("Dearness Allowances : " + da);
        System.out.println("House Rent allowances : " + hra);
        System.out.println("                                                ");
        System.out.println("Gross Salary : " + super.grossSalary);
        System.out.println("Net Salary                      : " + super.netSalary);
    }
}
class Associate_Professor extends Employee
```

APEC

```
{
   double basicPay;
   public Associate_Professor(double pay)
   {
      Emp_name = "Divyesh Prasad";
      Address = "Hyderabad";
      Mail_id = "dp@gmail.com";
      Mobile_no = "7362884927";
      Emp_id = 1235;
      basicPay = pay;
      super.calculationGross(pay);
      super.calculationNet();
   }
   public void show()
   {
      System.out.println("\n\n----------PAY SLIP   ");
      System.out.println("Name: " + Emp_name);
      System.out.println("Employee id : " + Emp_id);
      System.out.println("Address: " + Address);
      System.out.println("Mobile No : " + Mobile_no);
      System.out.println("");
      System.out.println("Basic Pay: " + basicPay);
      System.out.println("Dearness Allowances : " + da);
      System.out.println("House Rent allowances : " + hra);
      System.out.println("");
      System.out.println("Gross Salary : " + super.grossSalary);
      System.out.println("Net Salary   : " + super.netSalary);
   }
}
class Professor extends Employee
{
   double basicPay;
   public Professor(double pay)
   {
```

APEC

```
        Emp_name = "Divyesh Prasad";
        Address = "Hyderabad";
        Mail_id = "dp@gmail.com";
        Mobile_no = "7362884927";
        Emp_id = 1235;
        basicPay = pay;
        super.calculationGross(pay);
        super.calculationNet();
    }
    public void show() {
        System.out.println("\n\n----------PAY SLIP   ");
        System.out.println("Name: " + Emp_name);
        System.out.println("Employee id : " + Emp_id);
        System.out.println("Address: " + Address);
        System.out.println("Mobile No : " + Mobile_no);
        System.out.println("");
        System.out.println("Basic Pay: " + basicPay);
        System.out.println("Dearness Allowances : " + da);
        System.out.println("House Rent allowances : " + hra);
        System.out.println("");
        System.out.println("Gross Salary : " + super.grossSalary);
        System.out.println("Net Salary   : " + super.netSalary);
    }
}
public class Common
{
    public static void main(String[] args)
    {
        try (Scanner sc = new Scanner(System.in))
        {
            while (true)
            {
                System.out.print("1.Programmer \n2.Assistant Professor\
                    n3.Associate Professor\ n4.Professor\ n5.Exit\ nEnter the choice to generate payslip...");
```

```
            int choice = sc.nextInt();
            switch (choice) {
               case 1:
                  Programmer pr = new Programmer(34000);
                  pr.show();
                  break;
               case 2:
                  Assistant_Professor ap = new Assistant_Professor(45000);
                  ap.show();
                  break;
               case 3:
                  Associate_Professor asp = new Associate_Professor(54000);
                  asp.show();
                  break;
               case 4:
                  Professor p = new Professor(60000);
                  p.show();
                  break;
               case 5:
                  System.exit(choice);
               default:
                  System.out.println("Enter a valid choice    ");
            }

         }

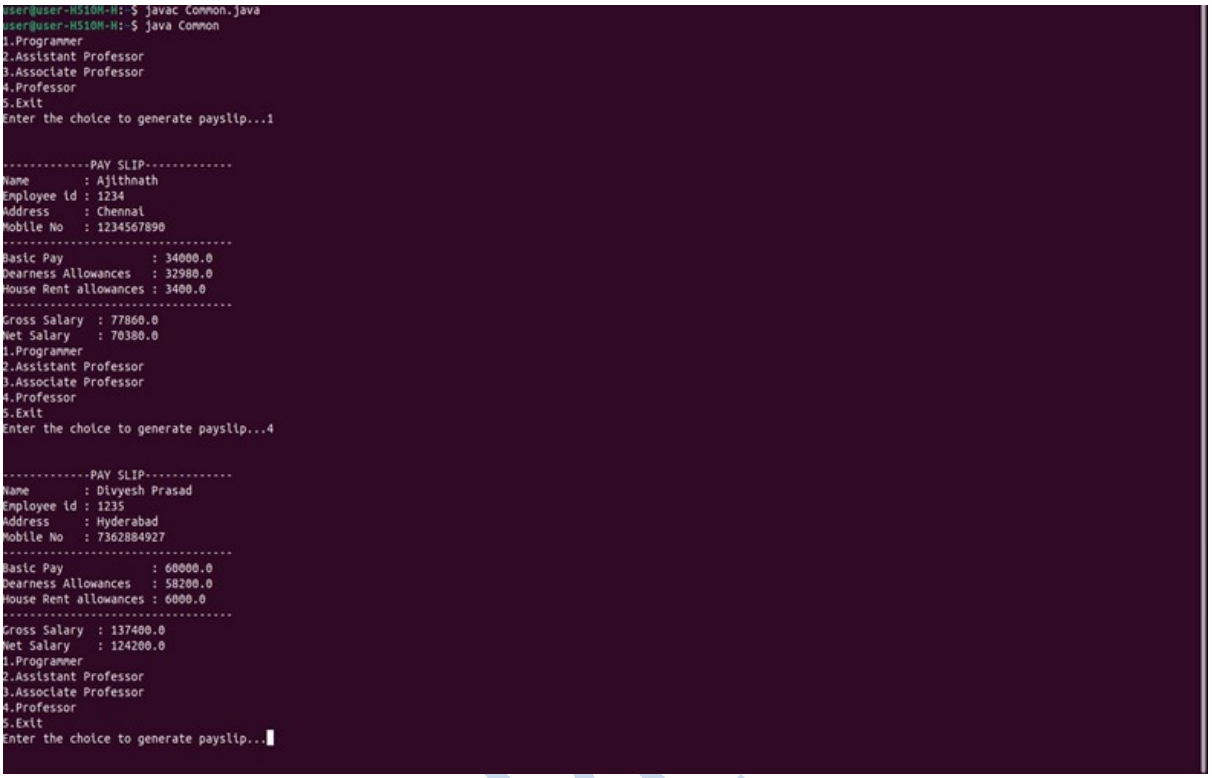
      }

   }

}
```

APEC

Output:



Result:

Thus, a java application for generating pay slips using classes has been executed successfully.

EX. NO: 7	AREA OF SHAPES USING ABSTRACT CLASS

Aim:

To write a java to computer area of major shapes using concepts of abstract class.

Algorithm:

Step 1: Start.

Step 2: Enter your choice.

Step 3: Compile the chosen choice of particular class

Step 4: Stop.

Program:

```
import java.util.*;
abstract class Shape
{
   int a, b;
   abstract void printArea(int a, int b);
}
class Rectangle extends Shape
{
   void printArea(int a, int b)
{
      int area = a * b;
      System.out.println("Area of rectangle: " + area + "\n\n");
   }
}
class Triangle extends Shape
{

   void printArea(int a, int b)
   {
```

APEC

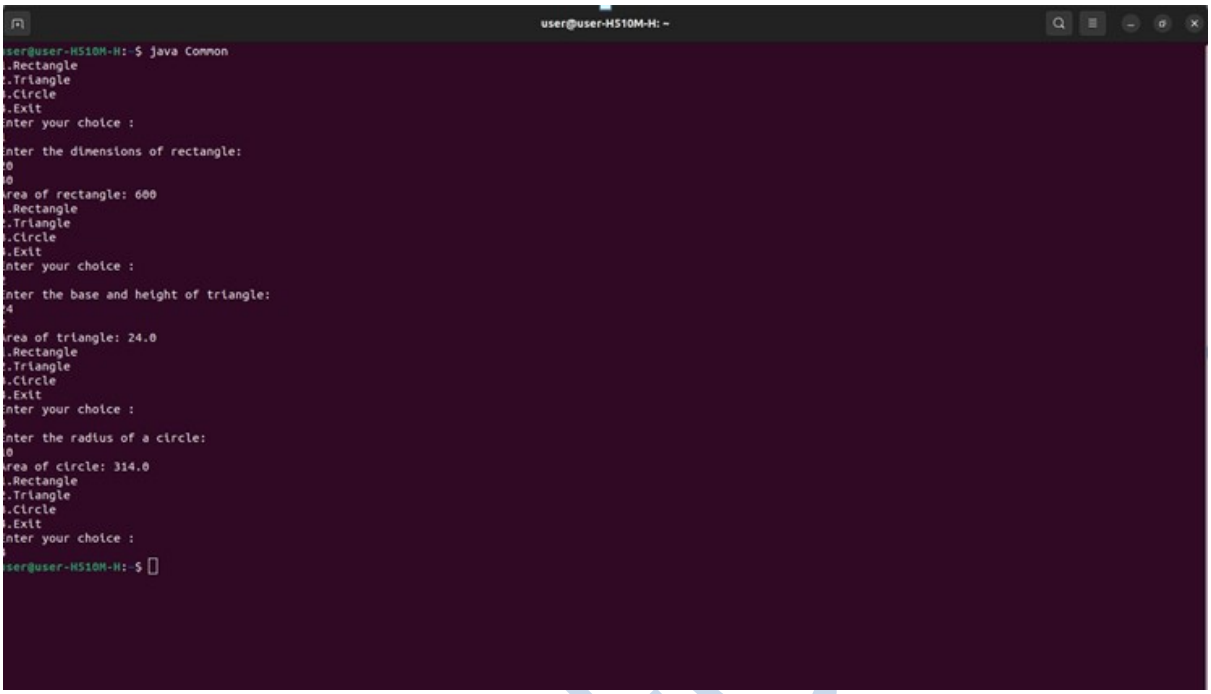
```
        double area = 0.5 * (double) a * (double) b;
        System.out.println("Area of triangle: " + (float) area + "\n\n");
    }
}
class Circle extends Shape
{
    void printArea(int a, int b)
{
        double area = 3.14 * a * a;
        System.out.println("Area of circle: " + (float) area + "\n\n");
    }
}
class Common
{
    public static void main(String args[])
{
        Scanner sc = new Scanner(System.in);
        while (true)
{
            System.out.print("1.Rectangle\n2.Triangle\n3.Circle\n4.Exit\nEnter your choice: ");
             int choice =sc.nextInt();
            switch(choice)
            {
                  case 1:
                     Rectangle r = new Rectangle();
                     System.out.println("Enter the dimensions of rectangle: ");
                     int a = sc.nextInt();
                     int b = sc.nextInt();
                     r.printArea(a, b);
                     break;
                  case 2:
                     Triangle t = new Triangle();
                     System.out.println("Enter base and height of a triangle: ");
                     int d = sc.nextInt();
```

APEC

```
                int e = sc.nextInt();
                t.printArea(d, e);
                break;
            case 3:
                Circle c = new Circle();
                System.out.println("Enter the radius of a circle: ");
                int f = sc.nextInt();
                c.printArea(f, 1);
                break;
            case 4:
                System.exit(choice);
            default:
                System.out.println("Enter a valid choice...\n\n");
        }
    }
}
```

APEC

Output:



Result:

Thus, the program for shape class using Abstract class has been executed successfully.

EX. NO: 8	AREA OF SHAPE USING INTERFACE

Aim:

To write a java to computer area of major shapes using concepts of overriding using interfaces.

Algorithm:

Step 1: Start.

Step 2: Create an interface with printArea method

Step 3: Classes such as rectangle, triangle and circle are implemented with the method

Step 4: Pass Numerical value to each class Step 5: Print area of the each shape

Step 6: Stop.

Program:

```
import java.io.*;
import java.util.*;
interface Shape
{
   int a = 0, b = 0;
   void printArea(int a, int b);
}
class Rectangle implements Shape
{
   public void printArea(int a, int b)
  {
      int area = a * b;
      System.out.println("Area of rectangle: " + area);
   }
}
class Triangle implements Shape
{

   public void printArea(int a, int b)
{
```

APEC

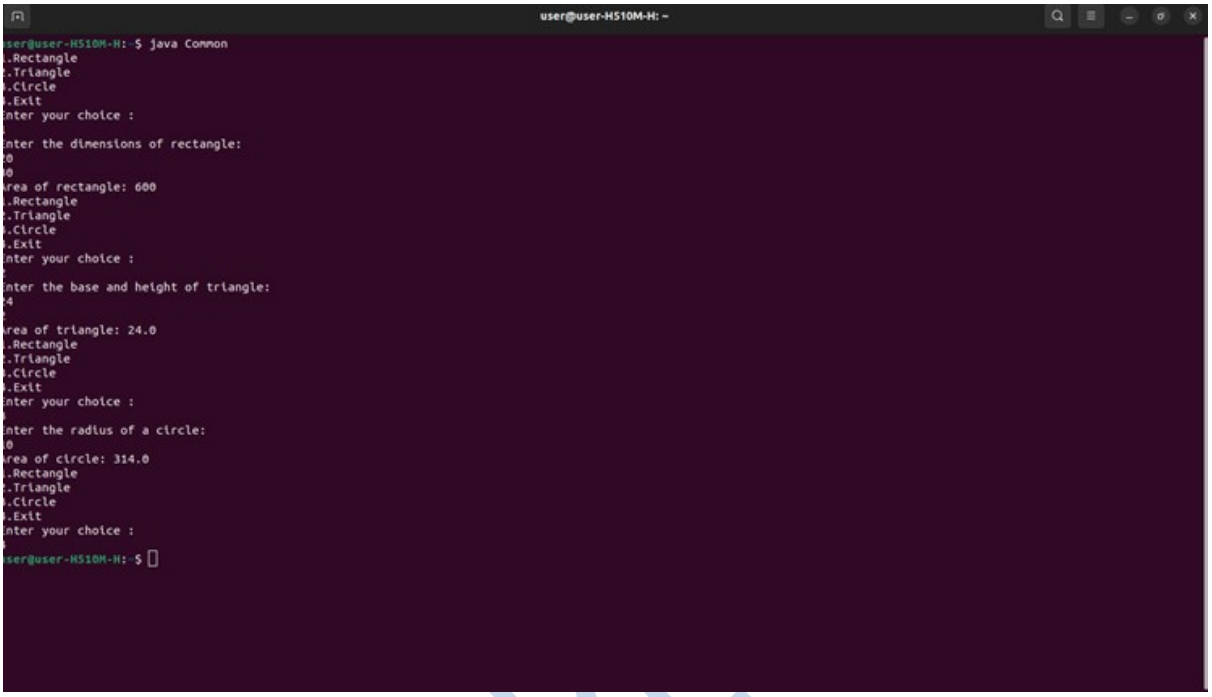
```
        double area = 0.5 * (double) a * (double) b;
        System.out.println("Area of triangle: " + area);
    }
}
class Circle implements Shape
{
    public void printArea(int a, int b)
{
        double area = 3.14 * a * a;
        System.out.println("Area of circle: " + area);
    }
}
class Common
{
    public static void main(String args[])
 {
        Scanner sc = new Scanner(System.in);
        while (true)
 {
            System.out.println("1.Rectangle");
            System.out.println("2.Triangle");
            System.out.println("3.Circle");
            System.out.println("4.Exit");
            System.out.println("Enter your choice :");
            int choice = sc.nextInt();
            switch (choice)
            {
            case 1:
                    Rectangle r = new Rectangle();
                    System.out.println("Enter the dimensions of rectangle: ");
                     int a = sc.nextInt();
                     int b = sc.nextInt();
                     r.printArea(a, b);
                     break;
```

APEC

```
case 2:
        Triangle t = new Triangle();
         System.out.println("Enter the base & height of triangle:");
         int d = sc.nextInt();
         int e = sc.nextInt();
        t.printArea(d, e);
        break;
 case 3:
        Circle c = new Circle();
        System.out.println("Enter the radius of a circle: ");
         int f = sc.nextInt();
        c.printArea(f, 1);
         break;
   case 4:
          System.exit(0);
          default:
                System.out.println("Enter a valid choice...");
                 }
          }
        }
}
```

APEC

OutPut:



Result:

Thus, the program for shape class using interfaces is executed successfully.

EX. NO: 9	EXCEPTION HANDLING

Aim:

To implement exception handling using a java program.

Algorithm:

Step 1: Start the program.

Step 2: Declare the variables a,b,c.

Step 3: Read the values a,b,c,.

Step 4: Divide two variables if num2!=0 and display or throw exception.

Step 5: Catch the exception and display the appropriate message.

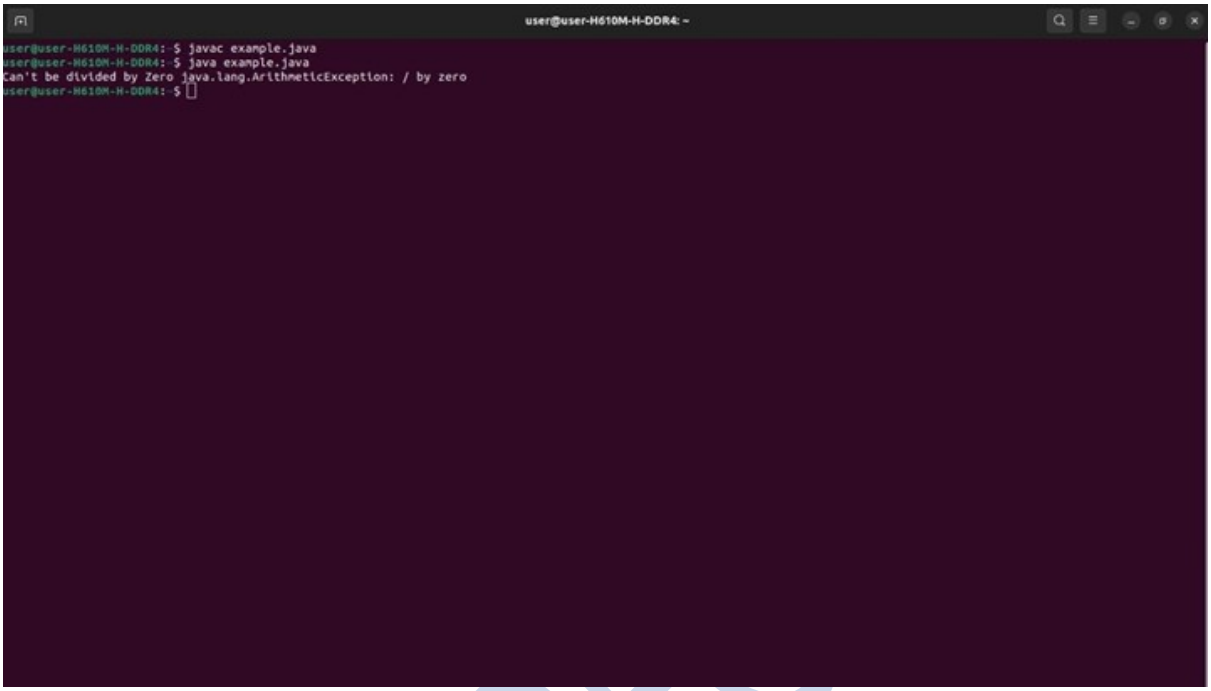
Step 6: Stop the program.

Program:

```
class example
{
  public static void main (String args[])
  {
    int num1 = 15, num2 = 0,
    result = 0;try
    {
       result = num1/num2;
       System.out.println("The result is"
       +result);
    }
    catch (ArithmeticException e)
    {
      System.out.println ("Can't be divided by Zero " + e);
    }
  }
}
```

APEC

Output:



Result:

Thus, the program for Exception handling has been implemented successfully.

EX. NO: 10	USER DEFINED EXCEPTION

Aim:

To implement user defined exception using a java program.

Algorithm:

Step 1: Start the program.

Step 2: Declare a parameterized exception with string str as a parameter.

Step 3: Implement the program to check for the user's age.

Step 4: Throw user defined exception if the given age is less than 18.

Step 5: Print the message from user defined Exception object.

Step 6: Stop the program.

Program:

InvalidAgeException.java:

```
class InvalidAgeException extends Exception
{
    public InvalidAgeException (String str)
    {
        super(str);
     }
}
```

Test.java

```
import
java.util.*;
public class
Test
{
```

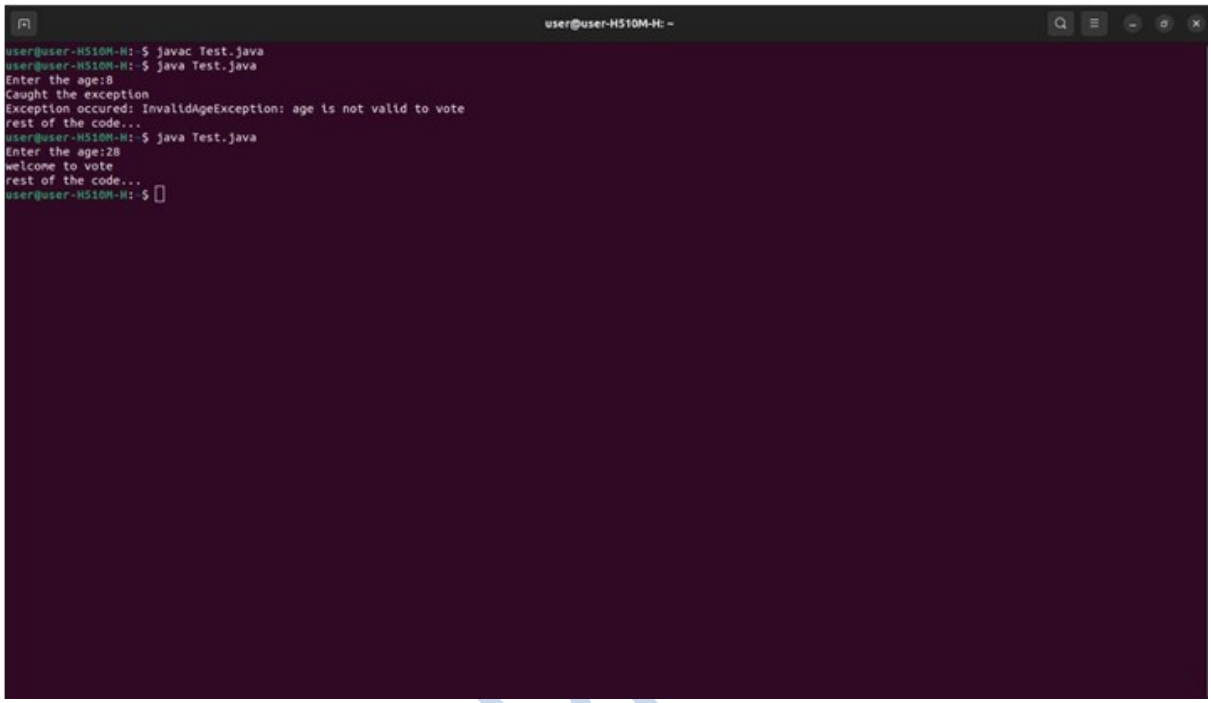
APEC

```
    static void validate (int age) throws
    InvalidAgeException{if(age < 18){
    throw new InvalidAgeException("age is not valid to vote");
}
 else
 {
        System.out.println("welcome to vote");

 }
}
public static void main(String args[])
{
    Scanner sc = new
    Scanner(System.in);try
    {
            System.out.print("Enter the
            age:");int age=sc.nextInt();
            validate(age);
    }
    catch (InvalidAgeException ex)
    {
       System.out.println("Caught the exception");
       System.out.println("Exception occured: " + ex);
    }
    System.out.println("rest of the code...");
  }
}
```

APEC

Output:



Result:

Thus, the program for User defined Exception has been implemented successfully.

EX. NO: 11	IMPLEMENTING MULTITHREAD APPLICATION

Aim:

To write a java program that implements Multithread application.

Algorithm:

Step 1: Start.

Step 2: Create a Thread class to create a random number with an interval of one second.

Step 3: If the number is divisible by 2 run square thread to print the square of that number.

Else invoke cube thread to print cube of that number.

Step 4: Stop.

Program:

```
import java.util.Random;
class RandomNumberThread extends Thread
{
   public void run() {
      Random random = new Random();
      for (int i = 0; i < 10; i++)
      {
         int randomInteger = random.nextInt(100);
         System.out.println("Random Integer generated : " + randomInteger);
         if ((randomInteger % 2) == 0)
         {

         } else
         {
         }
         try
         {
```

APEC

```
            }
            SquareThread sThread = new SquareThread(randomInteger);
            sThread.start();
            CubeThread cThread = new CubeThread(randomInteger);
            cThread.start();
            Thread.sleep(1000);
            catch (InterruptedException ex)
            {
               System.out.println(ex);
            }
        }
    }
}
class SquareThread extends Thread {
    int number;

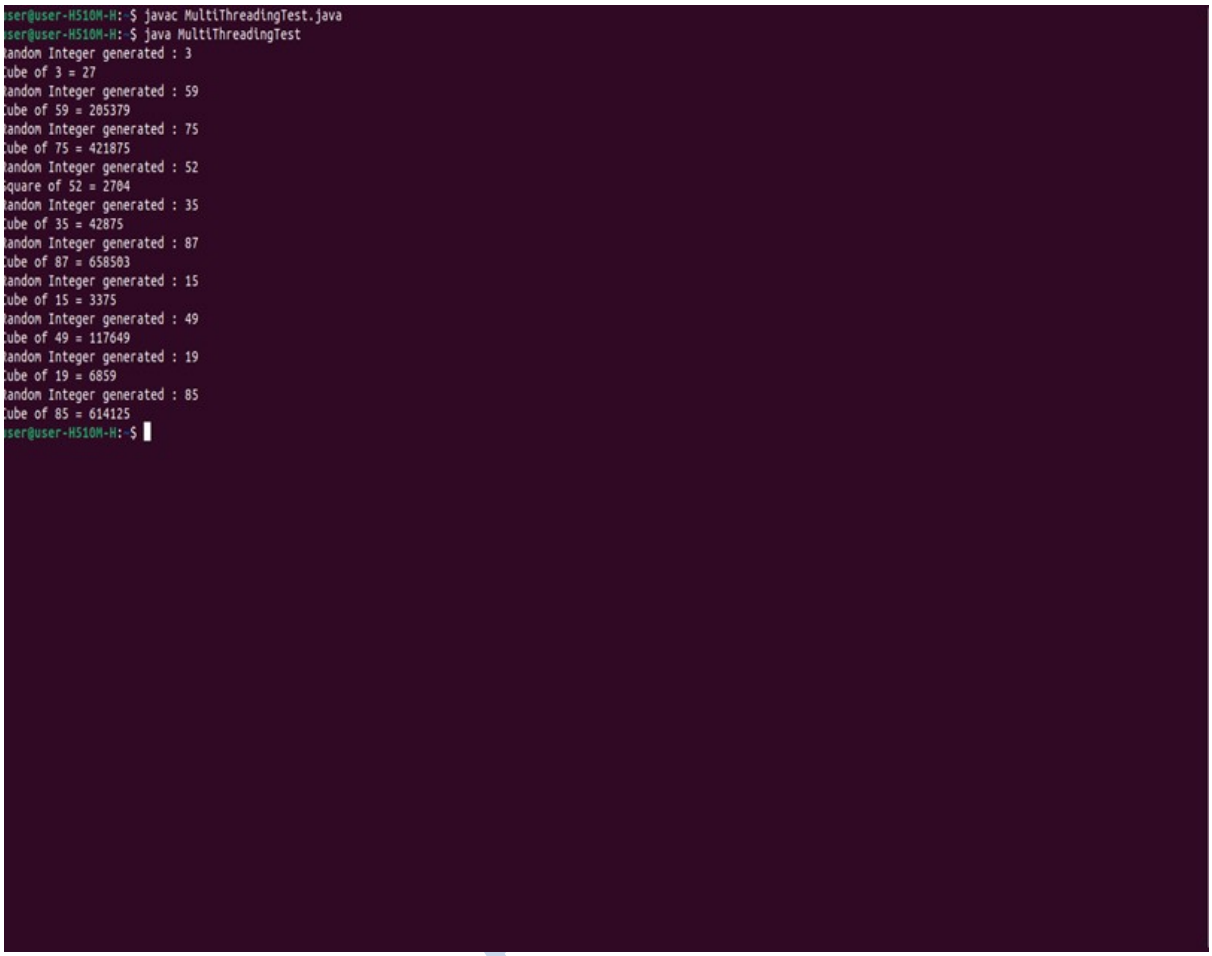
    SquareThread(int randomNumbern)
    {
        number = randomNumbern;
    }
    public void run()
    {
        System.out.println("Square of " + number + " = " + (number * number));
    }
}
class CubeThread extends Thread
{
    int number;
    CubeThread(int randomNumber)
    {
        number = randomNumber;
    }
    public void run()
    {
```

APEC

```
        System.out.println("Cube of " + number + " = " + number * number * number);
    }
}
public class MultiThreadingTest
{
    public static void main(String args[])
    {
        RandomNumberThread rnThread = new RandomNumberThread();
        rnThread.start();
    }
}
```

APEC

Output:



Result:

Thus, the program for multithread has been executed successfully.

EX. NO: 12	FILE OPERATIONS - FILE CONTENT DISPLAY

Aim:

To write a java program to read the content from the file.

Algorithm:

Step 1: Start.

Step 2: Create a file (hello.txt) with the content for reading.

Step 3: Create file input stream object fis from read hello.txt

Step 4: Inside try block create a variable c to display the character.

Step 5: While ((c=fis.read())!=-1) then display the content of the file.

Step 6: If (fis! =NULL), then close the file by fis.close().

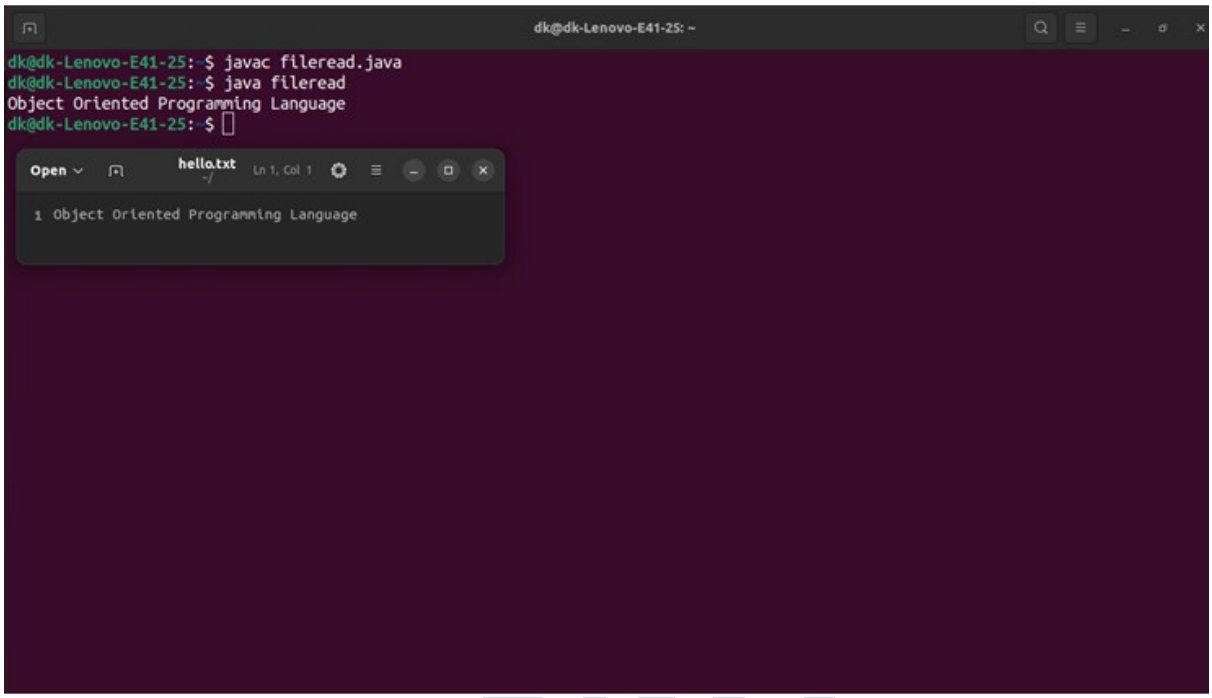
Step 7: Stop.

Program:

```
import java.io.*;
class fileread
{
        public static void main(String args[]) throws IOException
        {
                FileInputStream fis=new
                FileInputStream("hello.txt");
                try
                {
                        int c;
                        while((c=fis.read())!=-
                        1)
                        {
                                System.out.print((char)c);
                        }
                }
                finally
                {
                If(fis!=null)
                fis.close();
                 }
               }
   }
```

APEC

Output:



Result:

Thus, the java program for read content from file has been executed successfully.

EX. NO: 13	FILE OPERATIONS – FILE COPY

Aim:

Java program to copy the content of one file to another file.

Algorithm:

Step 1: Start.

Step 2: Create two files one file with content and the other file to copy the content.

Step 3: Create file input stream object fis from read hello.txt and Create file input stream object fos to write hello.txt (“output.txt”)

Step 4: Inside try block create a variable c to display the character.

Step 5: WHILE((c=fis.read())!=-1) then copy the content to the another file using fos.write(c)

Step 6: IF(fis!=NULL),close the file by fis.close().

Step 7: IF(fos!=null),close the file by fos.close().

Step 8: Stop.

Program:

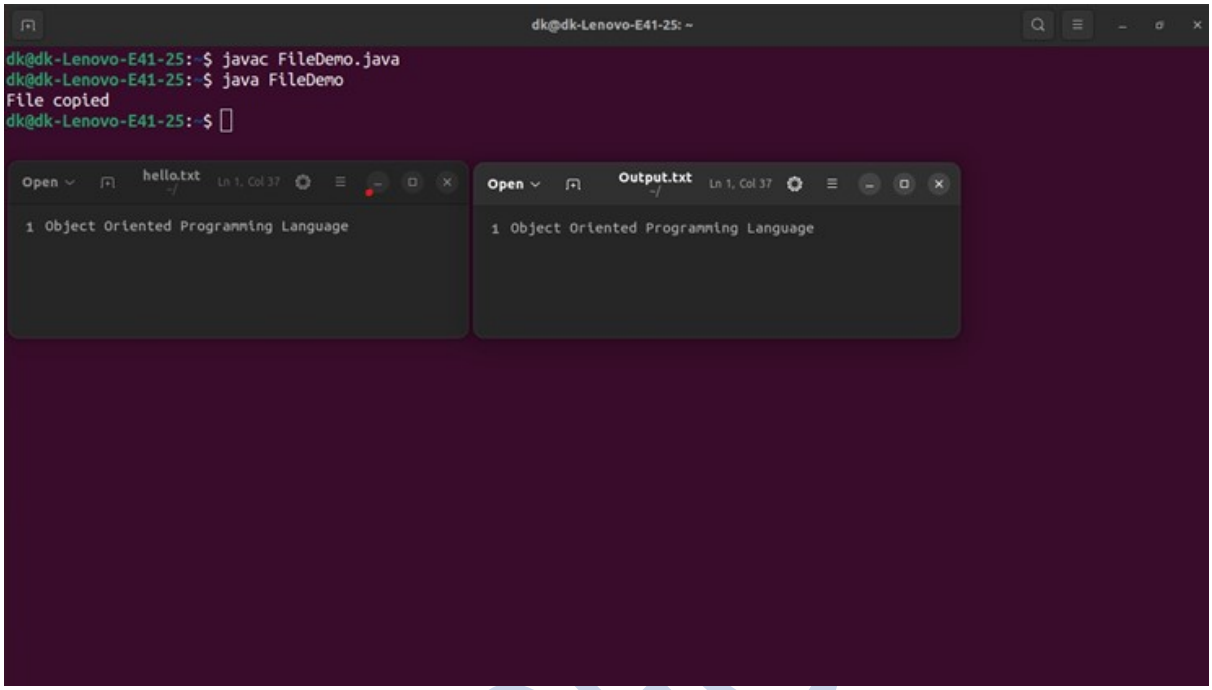
```
import java.io.*;
class FileDemo
{
   public static void main(String args[]) throws IOException
{
      FileInputStream fis = new FileInputStream("hello.txt");
      FileOutputStream fos = new FileOutputStream("Output.txt");
      try
         {
         int c;
         while ((c = fis.read()) != -1)
         {
            fos.write(c);
```

APEC

```
            }
        }
            finally
            {
            if (fis != null)
              fis.close();
            if (fos != null)
             fos.close();
            System.out.println("File copied");
            }
    }
}
```

APEC

Output:



Result:

Thus, the java program for file copy has been executed successfully.

EX. NO: 14	FILE OPERATIONS –REVERSE FILE

Aim:

To write a java program to reverse the text content of a file.

Algorithm:

Step 1: Start.

Step 2: import required packages.

Step 3: Open a new or already existing file in read mode.

Step 4: Create an output file.

Step 5: Create an object fis and fos for input and output streams.

Step 6: Using the reverse(),reverse the content in the file by FOR loop.

Step 7: Close the files after the operations.

Step 8: Stop.

Program:

```
import java.io.*;
import java.util.*;
class Main
{
   public static void main(String args[]) throws IOException
{
      FileInputStream fis = new FileInputStream("DK.txt");
      FileOutputStream fos = new FileOutputStream("PK.txt");
      try {
         int ch;
         String str = "", das = "";
         while ((ch = fis.read()) != -1)
         {
            str += (char) ch;
         }
         StringBuilder i1 = new StringBuilder();
```

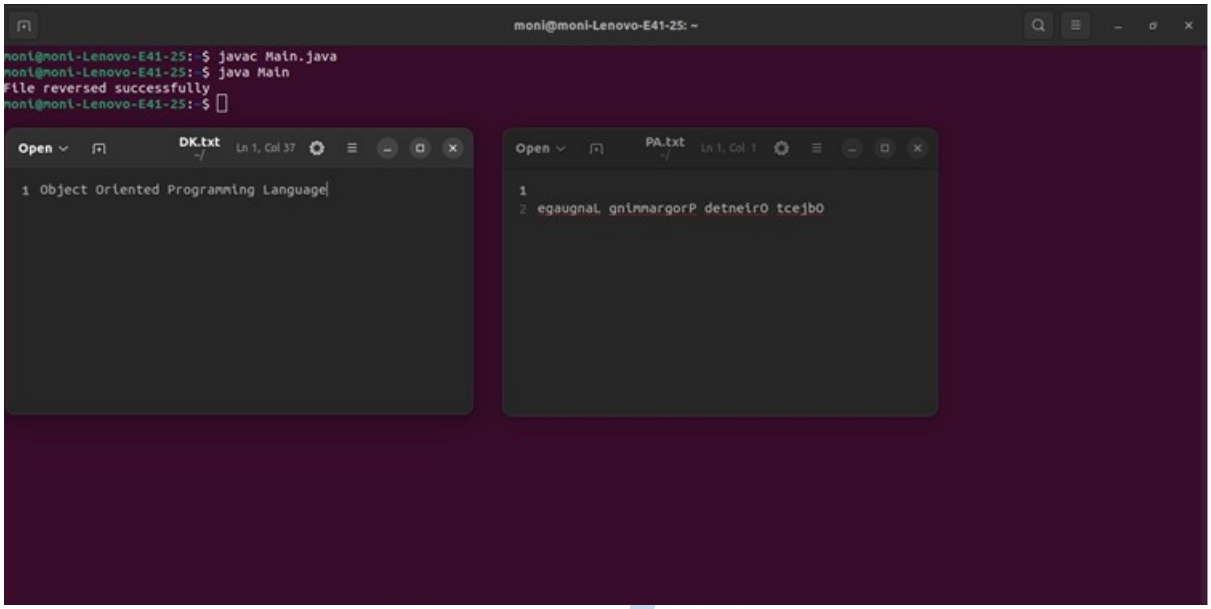
APEC

```
            i1.append(str);
            i1.reverse();
            das = i1.toString();
            for (int i = 0; i < das.length(); i++)
            {
               fos.write(das.charAt(i));
            }
            System.out.println("File reversed successfully");

        }
            finally
        {
                    if (fis != null)
                            fis.close();
                    if (fos != null)
                             fos.close();
        }
    }
}
```

APEC

Output:



Result:

Thus, the java program for file reverse has been executed successfully.

EX. NO: 15	FILE OPERATIONS - FILE CONTENT CASE CONVERSION

Aim:

To write a java program to convert the content of the file lowercase to uppercase.

Algorithm:

Step 1: Start.

Step 2: Import required packages.

Step 3: Open a new or already existing file in read mode.

Step 4: Create an output file.

Step 5: Create an object fis and fos for input and output streams.

Step 6: Use try and finally to reverse the contents of the file.

Step 7: Convert the contents using .ToUpper() method.

Step 8: Close the files when the operations are completed.

Step 9: Stop.

Program:

```
import java.io.*;
import java.util.*;
class main {
   public static void main(String args[]) throws IOException {
      FileInputStream fis = new FileInputStream("DK.txt");
      FileOutputStream fos = new FileOutputStream("PA.txt");
      try {
         int ch;
         String str = "";
         while ((ch = fis.read()) != -1) {
            str += (char) ch;
         }
         for (int i = 0; i < str.length(); i++)
         {
            fos.write(str.toUpperCase().charAt(i));
         }
```

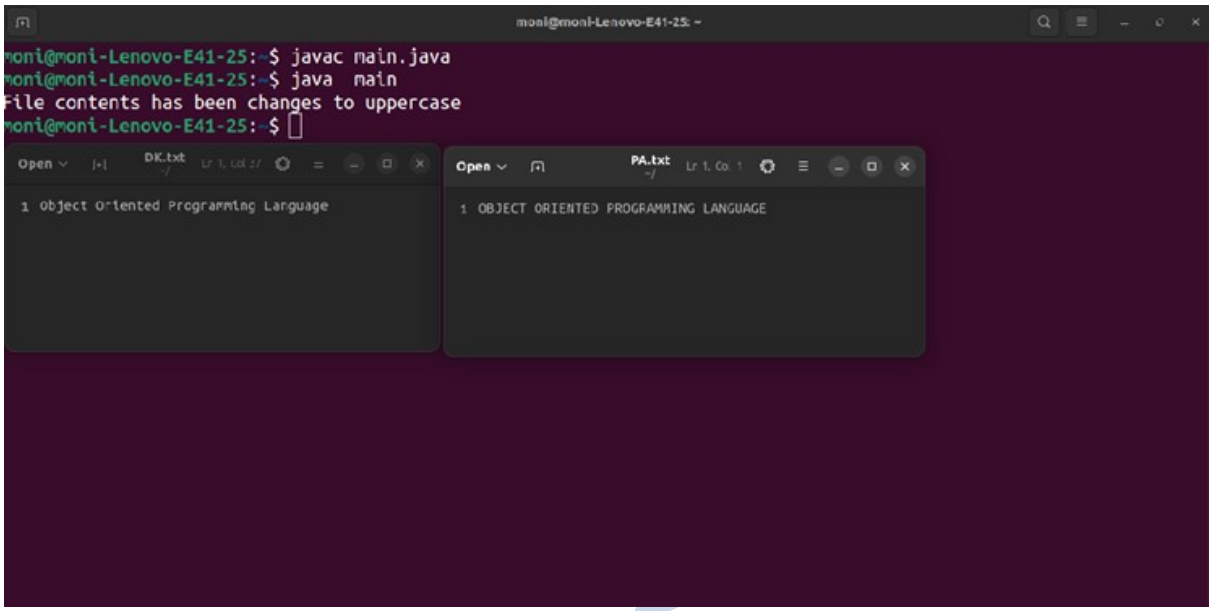
APEC

```
            System.out.println("File contents has been changes to uppercase");

        }
            finally
            {
            if (fis != null)
                    fis.close();
            if (fos != null)
                    fos.close();
             }
    }
}
```

APEC

Output:



Result:

Thus, the java program for file content case conversion has been executed successfully.

EX. NO: 16	GENERIC CLASS

Aim:

To develop an application to demonstrate the features of generic class.

Algorithm:

Step 1: Start.

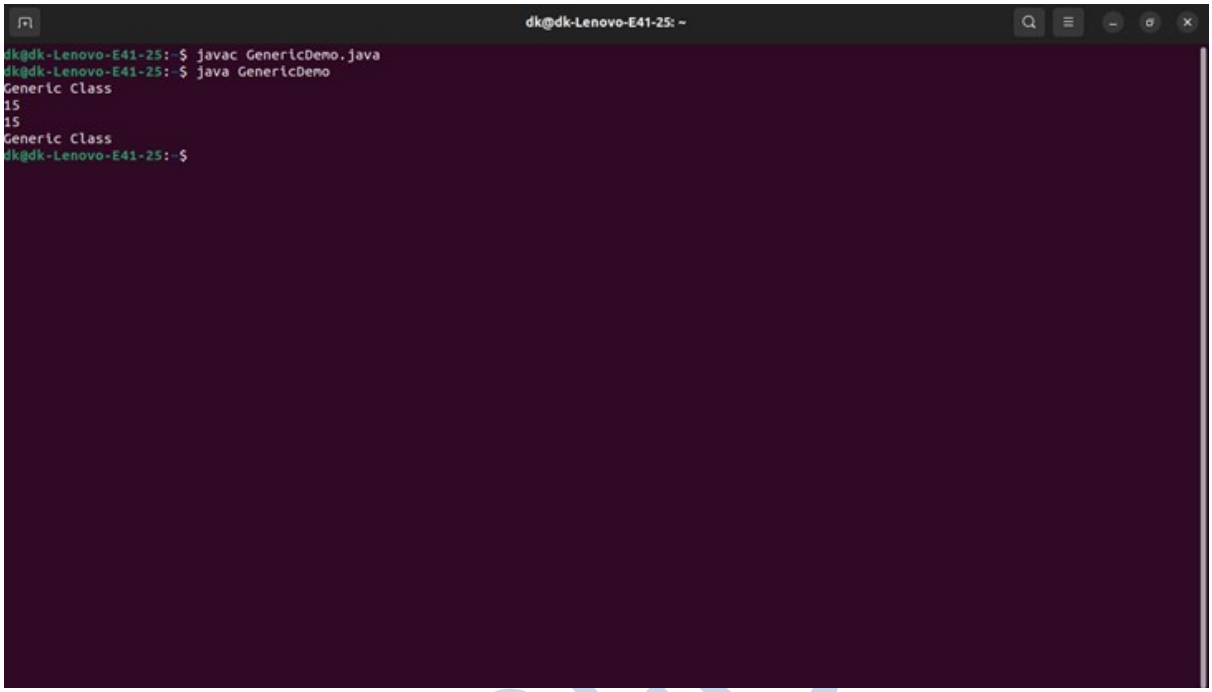
Step 2: Create a class with generic data Type.
Step 3: Implement this keyword to refer the members of a class.
Step 4: Implement get() method and return the object.
Step 5: Create an object of the generic class by specifying its data type and get the values.
Step 6: Stop.

Program:

```
class Test<T, U>
{
        T obj1; // An object of type T
        U obj2;  // An object of ype U
        // constructor
        Test(T obj1, U obj2)
        {
                this.obj1 = obj1;
                this.obj2 = obj2;
        }
        public void print()
        {
                System.out.println(obj1);
                System.out.println(obj2);
        }
}
public class GenericDemo
{
        public static void main (String[] args)
        {
                Test <String, Integer> obj = new Test<String, Integer>("Generic Class", 15);
                Test <String, Integer> obj = new Test<String, Integer>(15, “Generic Class”);
                obj.print();
                obj1.print();
        }
}
```

APEC

Output:



Result:

Thus, the generic class program has been executed successfully.

EX. NO: 17	APPLICATION USING JAVAFX CONTROLS LAYOUTS AND MENUS

Aim:

To develop registration form using JavaFX controls, layouts and menus.

Algorithm:

Step 1: Create a registration form class.

Step 2: Create a add uicontrols method and add the controls.

Step 3: Create label for name and add to gridpane

Step 4: Create text field for user enter the name and add this control to gridpane

Step 5: Create label for emailid and add to gridpane

Step 6: Create text field for user enter the email and add this control to gridpane

Step 7: Create label for mobileno and add to gridpane

Step 8: Create text field for user enter the mobileno and add this control to gridpane

Step 9: Create label for gender and add to gridpane

Step10: Create radio button field for gender and add totogglegroup.

Step11: Add toggle group field to the gridpane.

Step 12: Create label for country and add to gridpane

Step 13: Create combo box field for list of the country and add to gridpane.

Step14: Create label for date of birth and add to gridpane

Step 15: Create date picker field for user select the dob and add to gridpane.

Step 16: Create submit button field for user submit their data and add to gridpane.

Step 17: Create the validation method for validate the mandatory fields

Step 18: Create alert method for show the error message to user.

Step 19: Create start method and create the gridpane object, call the add uicontrols method andcreatethe scene object.

Step 20: Call the primary class show method.

Step 21: Create main method and call the launch method.

Step 22: Launch method invokes the start method.

APEC

Program:

```
package application;
import javafx.application.Application;import javafx.event.ActionEvent; import
javafx.event.EventHandler; import javafx.geometry.HPos;
import javafx.geometry.Insets;import javafx.geometry.Pos; import javafx.scene.Scene; import
javafx.scene.control.*; import javafx.scene.layout.*; import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;import javafx.stage.Stage;
import javafx.stage.Window;import javafx.collections.*;
public class RegistrationFormApplication extends Application
{
	public void start(Stage primaryStage) throws Exception
	{
		primaryStage.setTitle("Registration Form JavaFX Application");
		// Create the registration form grid pane
		GridPane gridPane = createRegistrationFormPane();
		// Add UI controls to the registration form grid paneaddUIControls(gridPane);
		// Create a scene with registration form grid pane as the root nodeScene scene = new
		Scene(gridPane, 800, 500);
		// Set the scene in primary stageprimaryStage.setScene(scene);
		primaryStage.show();
	}
	private GridPane createRegistrationFormPane()
	{
		// Instantiate a new Grid Pane GridPane gridPane = new GridPane();
		// Position the pane at the center of the screen, both vertically and horizontally
		gridPane.setAlignment(Pos.CENTER);
		// Set a padding of 20px on each side gridPane.setPadding(new Insets(40, 40, 40, 40));
		// Set the horizontal gap between columnsgridPane.setHgap(10);
		// Set the vertical gap between rowsgridPane.setVgap(10);
		// Add Column Constraints
		// columnOneConstraints will be applied to all the nodes placed in column one.
		ColumnConstraints columnOneConstraints = new ColumnConstraints(100, 100,
		Double.MAX_VALUE);
		columnOneConstraints.setHalignment(HPos.RIGHT);
		// columnTwoConstraints will be applied to all the nodes placed in column two.
		ColumnConstraints columnTwoConstrains = new ColumnConstraints(200,200,
		Double.MAX_VALUE); columnTwoConstrains.setHgrow(Priority.ALWAYS);
		gridPane.getColumnConstraints().addAll(columnOneConstraints,columnTwo
		Constrains);
		return gridPane;
```

```
}
private void addUIControls(GridPane gridPane)
{
        // Add Header
        Label headerLabel = new Label("Registration Form");
        headerLabel.setFont(Font.font("Arial", FontWeight.BOLD, 24));
        gridPane.add(headerLabel, 0,0,2,1); GridPane.setHalignment(headerLabel,
        HPos.CENTER); GridPane.setMargin(headerLabel, new Insets(20, 0,20,0));
        // Add Name Label
        Label nameLabel = new Label("Full Name : ");gridPane.add(nameLabel, 0,1);
        // Add Name Text Field
        TextField nameField = new TextField();nameField.setPrefHeight(40);
        gridPane.add(nameField, 1,1);
        // Add Email Label
        Label emailLabel = new Label("Email ID : ");gridPane.add(emailLabel, 0, 2);
        // Add Email Text Field
        TextField emailField = new TextField();emailField.setPrefHeight(40);
        gridPane.add(emailField, 1, 2);
        // Add Password Label
        /*Label passwordLabel = new Label("Password : ");gridPane.add(passwordLabel, 0, 3);
        // Add Password Field
        PasswordField passwordField = new PasswordField();passwordField.setPrefHeight(40);
        gridPane.add(passwordField, 1, 3);*/
        Label MobileLabel = new Label("Mobile No : ");gridPane.add(MobileLabel, 0, 3);
        // Add Email Text Field
        TextField MobileNo = new TextField();MobileNo.setPrefHeight(40);
        gridPane.add(MobileNo, 1, 3);
        TilePane r = new TilePane();
        Label GenderLabel = new Label("Gender : ");ToggleGroup tg = new ToggleGroup();
        gridPane.add(GenderLabel, 0, 4);
        RadioButton MaleButton = new RadioButton("Male"); RadioButton FemaleButton =
        new RadioButton("Female");RadioButton OthersButton = new RadioButton("Others");
        MaleButton.setToggleGroup(tg); FemaleButton.setToggleGroup(tg);
        OthersButton.setToggleGroup(tg);
        //gridPane.add(GenderLabel); r.getChildren().add(MaleButton);
        r.getChildren().add(FemaleButton);r.getChildren().add(OthersButton); gridPane.add(r,
        1, 4);
        //Birth Date
        Label DobLabel = new Label("Date of Birth : ");gridPane.add(DobLabel, 0, 5);
        DatePicker dob = new DatePicker();gridPane.add(dob, 1, 5);
        // Country Label
```

```
        Label CountryLabel = new Label("Country");gridPane.add(CountryLabel, 0, 6);
        String CountryList[] = { "Afghanistan”, "Brazil
        ","Brunei","Bulgaria","Cambodia","Canada”,"Chad","Chile","China","Colombia",
        "Cuba", "Denmark","Egypt","Fiji","Finland","France","Germany","Iceland
        ","India
        ","Indonesia","Iran","Iraq","Ireland","Israel","Italy","Japan","Maldives","Mauritius","
        Mexico","Netherlands","New
        Zealand","Philippines","Poland","Portugal","Qatar","Russia","Thailand","United Arab
        Emirates","United Kingdom","United States of America"};
        ComboBox Countrycombo_box = new
        ComboBox(FXCollections.observableArrayList(CountryList));
        gridPane.add(Countrycombo_box, 1, 6);
// Add Submit Button
Button submitButton = new Button("Submit"); submitButton.setPrefHeight(40);
submitButton.setDefaultButton(true); submitButton.setPrefWidth(100);
gridPane.add(submitButton, 0, 7, 2, 1); GridPane.setHalignment(submitButton,
HPos.CENTER); GridPane.setMargin(submitButton, new Insets(20, 0,20,0));
submitButton.setOnAction(new EventHandler<ActionEvent>()
{
        @Override
        public void handle(ActionEvent event)
        {
                if(nameField.getText().isEmpty())
                {
                        showAlert(Alert.AlertType.ERROR, gridPane.getScene().getWindow(),
                "FormError!", "Please enter your name");
                        return;
                }
                if(emailField.getText().isEmpty())
                {
                        showAlert(Alert.AlertType.ERROR, gridPane.getScene().getWindow(),
                "FormError!", "Please enter your email id");
                return;
                }
                if(MobileNo.getText().isEmpty())
                {
                        showAlert(Alert.AlertType.ERROR, gridPane.getScene().getWindow(),
                "FormError!", "Please enter a Mobile No");
                return;
                }
                if(!MaleButton.isSelected() && ! FemaleButton.isSelected() &&
```

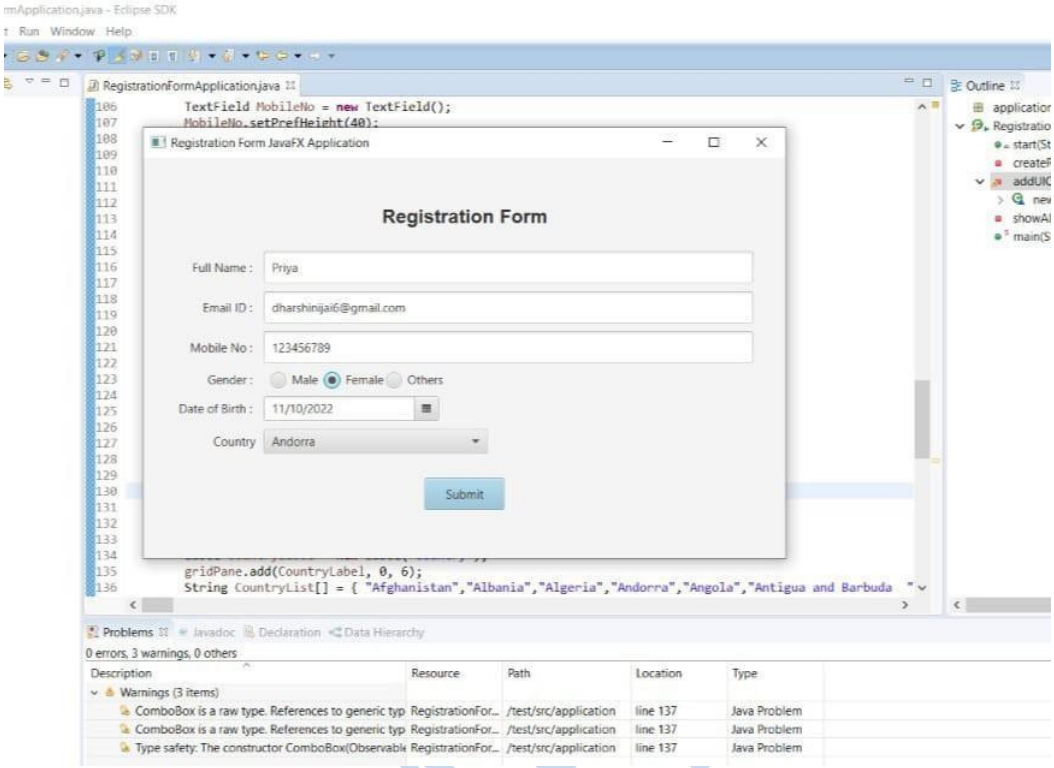
```
                !OthersButton.isSelected())
                {
                        showAlert(Alert.AlertType.ERROR, gridPane.getScene().getWindow(),
                "Form Error!", "Please select the Gender");
                        return;
                }
                        showAlert(Alert.AlertType.CONFIRMATION,
                gridPane.getScene().getWindow(),"Registration Successful!", "Welcome " +
                nameField.getText());
                }
                }
        }

private void showAlert(Alert.AlertType alertType, Window owner, String title, Stringmessage)
{
Alert alert = new Alert(alertType);alert.setTitle(title); alert.setHeaderText(null);
alert.setContentText(message); alert.initOwner(owner); alert.show();
}

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

APEC

Output:



Result:

Thus, the program for registration form using JavaFx has been implemented successfully.