OOP Final Lab File

Contains all experiments as Listed

By RAJUL DUBEY

SAP 500069424  ROLLno: R171218081

Experiment-1

• Print table of a given number

• Show prime numbers till 1000.

• Calculate and print area of a Triangle, Square and Circle using different methods.

• Write a program to calculate the transpose, inverse and matrix multiplication.

**Code:**

import java.util.Scanner;

class Table  //To print the table.

{

    int n;

    Table(int n)

    {

        this.n=n;

    }

   void dspTable()

   {

       for(int i=1;i<=10;i++)

       {

           System.out.println(n+"x"+i+" = "+n\*i);

       }

   }

}

class Prime  //Print prime numbers

{

   boolean flag;

    void dspPrime()

    {

        for(int i=1;i<=1000;i++)

        {

            for(int j=2;j<i;j++)

            {

                if(i%j==0)

                {

                    flag=true;  // if the i is divisible by any other

                    break;      //number which is less than i

                                //it will not be a prime number and flag will be true

                }

            }

            if(flag==false)

            {

              System.out.println(i);

            }

            else

            {

                flag=false;

            }

        }

    }

}

class Area  //Area

{

    double Area;

    void areaS(double x)

    {

        Area = Math.pow(x,2);

        System.out.println("the area of the square is "+Area+" sq units");

    }

    void areaT(double base\_x, double hight\_y)

    {

        Area = ( base\_x\*hight\_y) / 2 ;

        System.out.println("the area of the Triangle is "+Area+" sq units");

    }

    void areaC(double x)

    {

        double z = 3.14 \* x \* x;

        System.out.println("the area of the circle is "+z+" sq units");

    }

}

class Matrix //Matrix

{

    int i,j,N;

    Matrix(int N)

    {

        this.N=N;

    }

    void transpose(int A[][])

    {

        int B[][]= new int[N][N];

        for (i=0; i<N;i++)

        {

            for (j=0;j<N;j++)

            {

                B[i][j] = A[j][i];

            }

        }

        System.out.print("Result matrix is \n");

        for (i = 0; i < N; i++)

        {

            for (j = 0; j < N; j++)

            System.out.print(B[i][j] + " ");

            System.out.print("\n");

        }

     }

}

public class Experiment\_1 {

 public static void main(String[] args)

 {

     int op,temp1,temp2;

     Scanner sc = new Scanner(System.in);

     System.out.println("1. Print table of a given number\r\n"

            + "2. Show prime numbers till 1000.\r\n"

            + "3. Calculate and print area of a Triangle, Square and Circle using different methods.\r\n"

            + "4. Program to calculate the transpose");

     System.out.println("\*\*\*");

     System.out.print("Enter Your Choise Here:>");

     op=sc.nextInt();

    System.out.println("");

     switch(op)

     {

     case 1:

        System.out.print("Enter the Number: ");

         temp1=sc.nextInt();

         new Table(temp1).dspTable();

         break;

     case 2:

         new Prime().dspPrime();

         break;

     case 3:

         System.out.print("Enter the x(base,edge1,redius): ");

         temp1=sc.nextInt();

        System.out.println(" ");

         System.out.print("Enter the y(hight,edge2): ");

         temp2=sc.nextInt();

         Area a=new Area();

         a.areaS(temp1);

         a.areaT(temp1,temp2);

         a.areaC(temp1);3

         break;

     case 4:

         int N=4;

         int A[][] =  { {1, 1, 1, 1},

                 {2, 2, 2, 2},

                  {3, 3, 3, 3},

                  {4, 4, 4, 4}};

         System.out.print("Transpose of matrix\n");

            for (int i=0; i< N; i++)

            {

                for (int j = 0; j < N; j++)

                System.out.print(A[i][j] + " ");

                System.out.print("\n");

            }

            System.out.println("is");

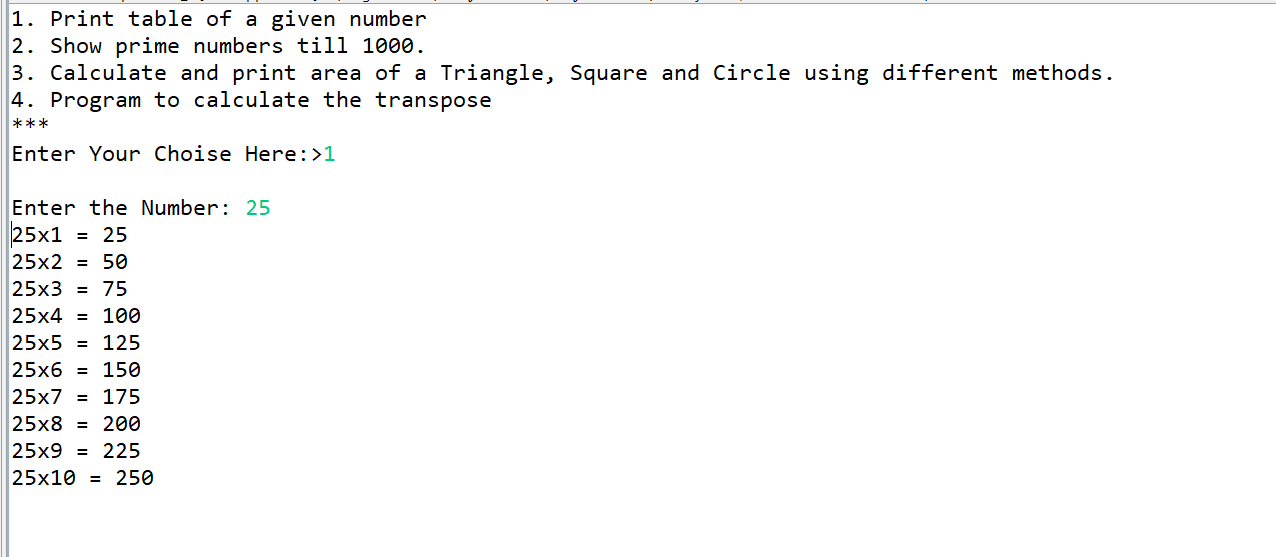
     new Matrix(N).transpose(A);

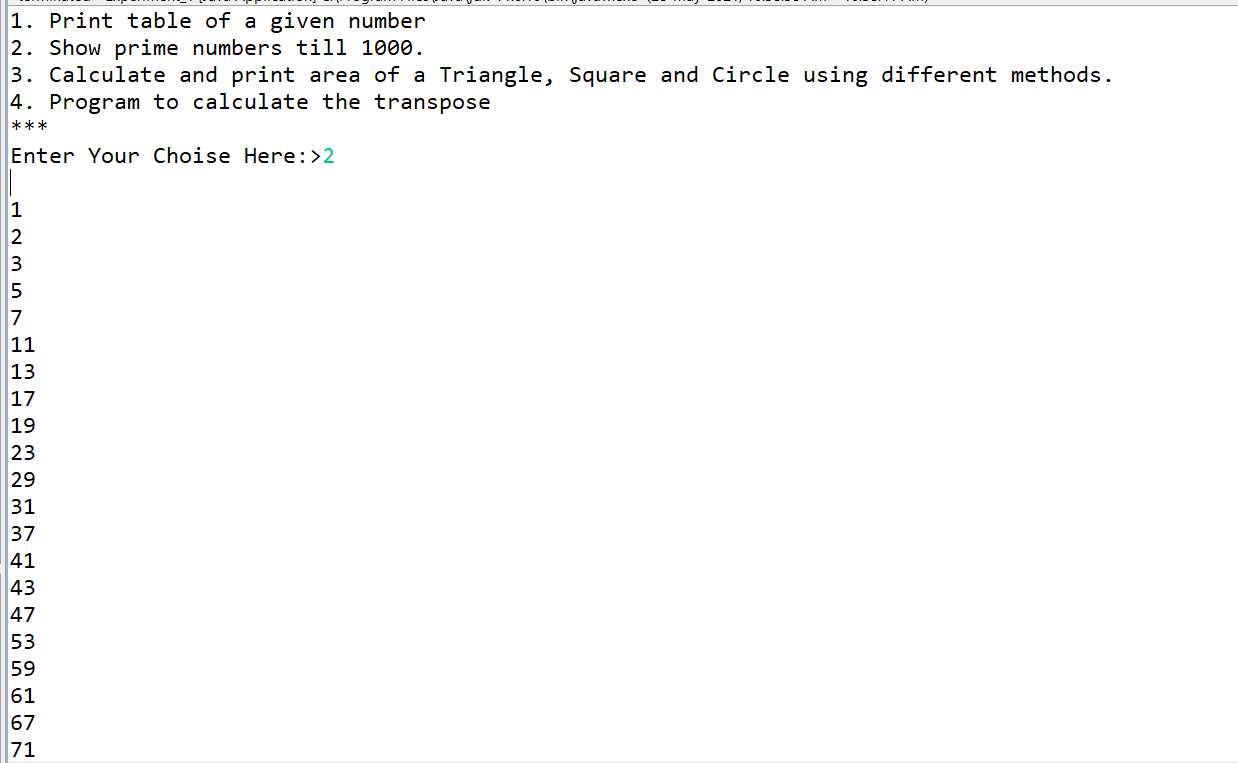
     }

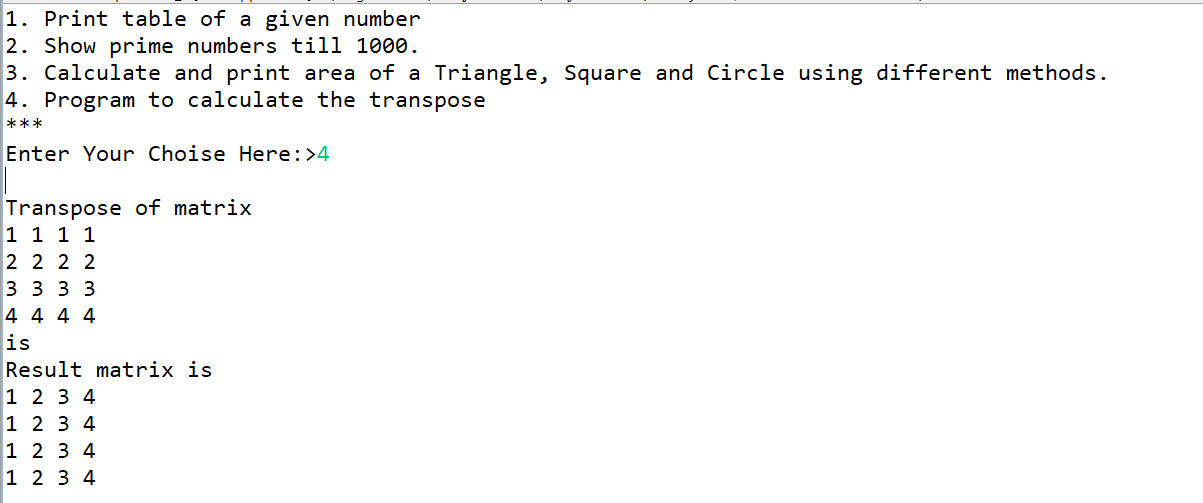
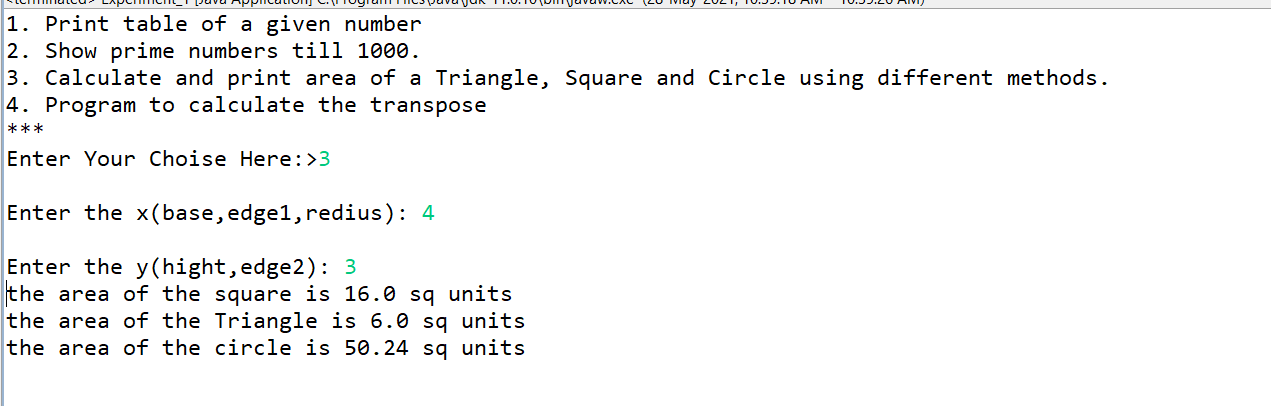
 }

}

**Execution:**

****

****

****

Experiment-2

• Write a program to perform aggregation.

• Write a program to pass object as parameters.

• Write a program to perform method overloading

**Code:**

class A

{

    int n;

    String msg;

    void dsp()

    {

        System.out.println("Inside Class A");

    }

    void dsp(int n)  //Overloading

    {

        this.n = n;

        System.out.println("Your number is: "+n);

    }

    void dsp(String msg) //Overloading

    {

        this.msg=msg;

        System.out.println("Your msg is: "+msg);

    }

    void hello()

    {

        System.out.println("Hello");

    }

}

class B

{

    A a1; //Aggregation

     void recieveObj(A a)  //method receiving the object

    {

      a1 = a;

      dsp();

    }

     void dsp()

     {

            a1.dsp();  //Calling overloaded methods

            a1.dsp(25); //

            a1.dsp("Hello"); //

     }

}

public class Experiment\_2 {

public static void main(String[] arg)

{

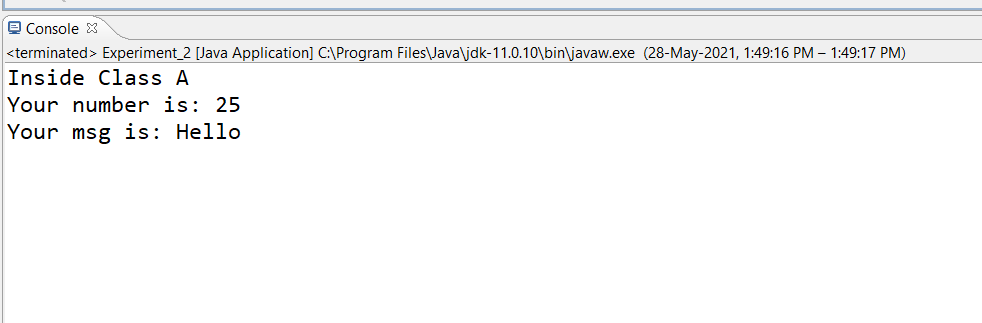
    A a1 =new A();

    new B().recieveObj(a1); // Passing the object to B

}

}

**Execution:**

****

Experiment-3

• Write a program to implement the returning of objects.

• Write a program for recursion.

**Code:**

class Calcus

{

int val;

 Calcus()

 {

     //Default constructor

 }

 Calcus(int x)

 {

     val=x;

 }

  Robj returnObj()

  {

      Robj R = new Robj(val); //This will get called after constructor

      R.call(); //This is a valid statement because its inside a method

   return R;

  }

}

class Robj

{

    int y,val;

    Robj(int x)

    {

        val=x;

    }

    int fabo(int x)  //Recursive Method

    {

        if(x==0)

        {

            return 0;

        }

        if(x==1)

        {

            return 1;

        }

        return fabo(x-1)+fabo(x-2);

    }

    void call()

    {

        y=fabo(val);

    }

}

public class Experiment\_3

{

public  static void main (String[] args) {

    Calcus C1 = new Calcus(3);  //Here you provide Step of Fibonacci Series  example: 3

    Robj R2;

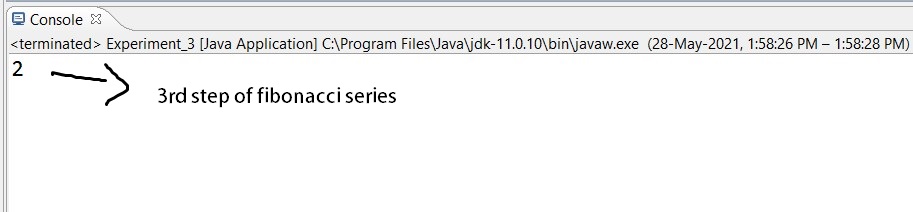
    R2=C1.returnObj();

     //Calling Object returning method of Calcus class

    System.out.println(R2.y);

}

}

**Execution: **

Experiment-4

• Write a program to show inheritance of classes.

• Write a program to calculate salary of an employee. The Employee class must contain name, id, basic salary, sa, hra, da. The class should also have a parameterised constructor to assign these values at the time of creation and a method named calculateSalary () to calculate the salary of employee using formula salary =basic+SA+HRA+DA. There should be subclass of Employee named Manager which has an additional property mainly sa. This class should override the calculateSalary () method and return salary with formula salary = basic + 2\*SA + HRA + DA. The manager class should have a subclass named project manager which also overrides the calculateSalary () method which calculates salary using the formula salary = basic + 2\*SA + 2\*HRA + da.

• Write a program to demonstrate Constructor calling sequence

**Code:**

// This program contains example of inheritance and constructor calling sequence also //

class Employee

{

    double basic,sa,hra,da;

    double salary;

    public Employee(double basic, double sa, double hra, double da) {

        System.out.println("Employee's constructor called");  //Calling constructor sequence printing

        this.basic = basic;

        this.sa = sa;

        this.hra = hra;

        this.da = da;

    }

    void calculateSalary()

    {

        salary=basic+sa+hra+da;

        System.out.println("Employee Salary: "+salary);

    }

}

class Manager extends Employee

{

    double basic,sa,hra,da;

    double salary;

    public Manager(double basic, double sa, double hra, double da) //Managers constructor

    {

        super(basic,sa,hra,da);

        System.out.println("Manager's constructor called");  //Calling constructor sequence printing

        this.basic = basic;

        this.sa = sa;

        this.hra = hra;

        this.da = da;

      }

    @Override

    void calculateSalary()

    {

        salary=basic+2\*sa+hra+da;

        System.out.println("Manager Salary: "+salary);

        super.calculateSalary(); //Calling the calculateSalary method of super class

    }

}

class ProjectManager extends Manager

{

    double basic,sa,hra,da;

    double salary;

    public ProjectManager(double basic, double sa, double hra, double da) //Project Manager constructor

    {

        super(basic, sa, hra, da);

        System.out.println("ProjectManager's constructor called");  //Calling constructor sequence printing

        this.basic = basic;

        this.sa = sa;

        this.hra = hra;

        this.da = da;

    }

    @Override

    void calculateSalary()

    {

        salary=basic+2\*sa+2\*hra+da;

        System.out.println("Project Manager Salary: "+salary);

        super.calculateSalary(); //Calling the calculateSalary method of super class

    }

}

public class Experiment\_4 {

 //Predefine basic salary,SA,HRA,DA

//bSalary = 23,000

// HRA = 15000

// SA = 12% = (23000/100)\*12

//DA=17% = (23000/100)\*17

static  double bSalary = 23000;

static  double hRA = 15000;

static  double sA = (23000/100)\*12;

static  double dA = (23000/100)\*17;

public static void main(String[] args)

{

double eSalary,mSalary,pmSalary;

System.out.println("\*\*\*Printing the sequence of constructors\*\*\*"); //Here printing constructor calling sequence

 Employee E = new ProjectManager(bSalary,sA,hRA,dA);  //Passing the value to the constructor of ProjectManager class

//All the constructor of parent and child classes will be called

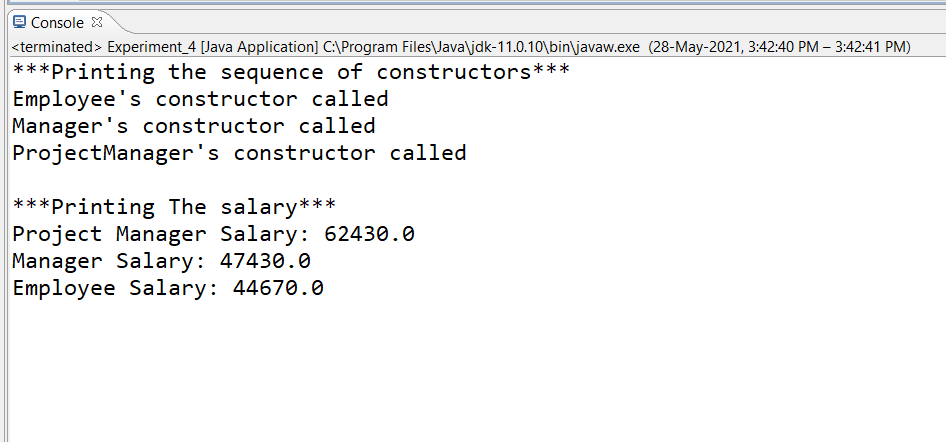
 System.out.println("\n\*\*\*Printing The salary\*\*\*");

 E.calculateSalary();

}

}

**Execution:**

****

Experiment-5

• Write the program to implement the abstract class.

• Write the program to demonstrate the use of final keyword.

**1.) Write the program to implement the abstract class.**

**Code:** abstract class A1

{

    double pi=3.14;

    abstract double Area();  //this a abstract method of abstract class

    A1()

    {

        System.out.println("this is the constructor of Abstract class A");

    }

    void dsp()

    {

        System.out.println("This is the concrete method of Abstract class");

    }

}

class B1 extends A1

{

    int r;

    B1(int r)

    {

        System.out.println("this is the constructor of class B");

        this.r=r;

    }

    @Override

    double Area()

    {

        return pi\*r\*r;

    }

}

public class Experimet\_5abstraction {

 public static void main(String arg[])

 {

  // A a = new A(); //Error: Abstract class can't have instance

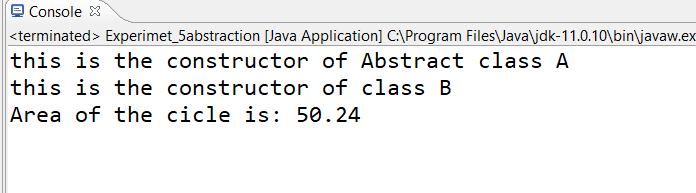
  A1 a = new B1(4); //valid

 System.out.println("Area of the cicle is: "+a.Area());

 }

}

**Execution:**

****

**2.) Write the program to demonstrate the use of final keyword.**

**2.1 final class**

This program will not compile.

**Code:** final class FinalClass

{

    int value = 25;

    void dsp()

    {

        System.out.println("Inside the final class: "+value);

    }

}

class ChildClass extends FinalClass //as this will give error because final class can't be inherited

{

  ChildClass()

  {

      System.out.println("In child class ");

  }

}

public class Experiment\_5 {

 public static void main(String[] arg)

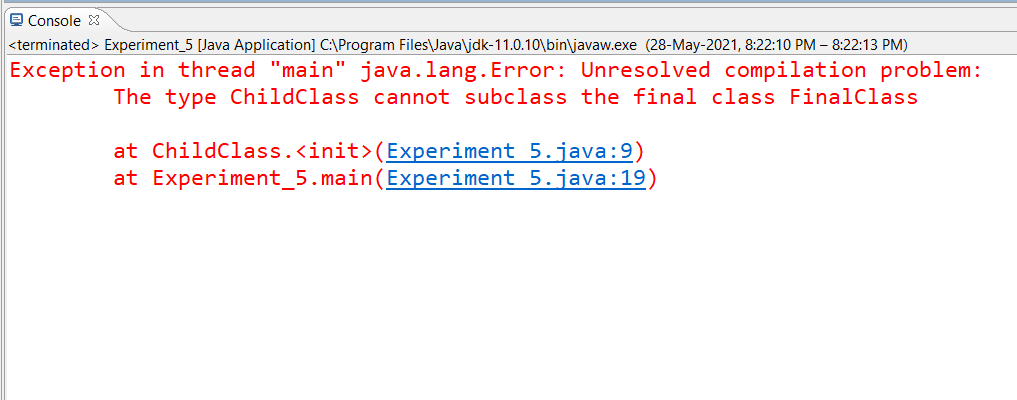
 {

     ChildClass ch = new ChildClass();

 }

}

**Execution:**

****

**2.2 final method**

This program will not compile.

 class FinalMethodClass

{

    int value = 25;

    final void dsp()

    {

        System.out.println("Inside the final dsp method of FinalMethodClass: "+value);

    }

}

class ChildClass extends FinalMethodClass

{

  @Override

  void dsp()  //we can not Override a final method this will give an error

  {

      System.out.println("Inside the dsp method of ChildClass");

  }

}

public class Experiment\_5 {

 public static void main(String[] arg)

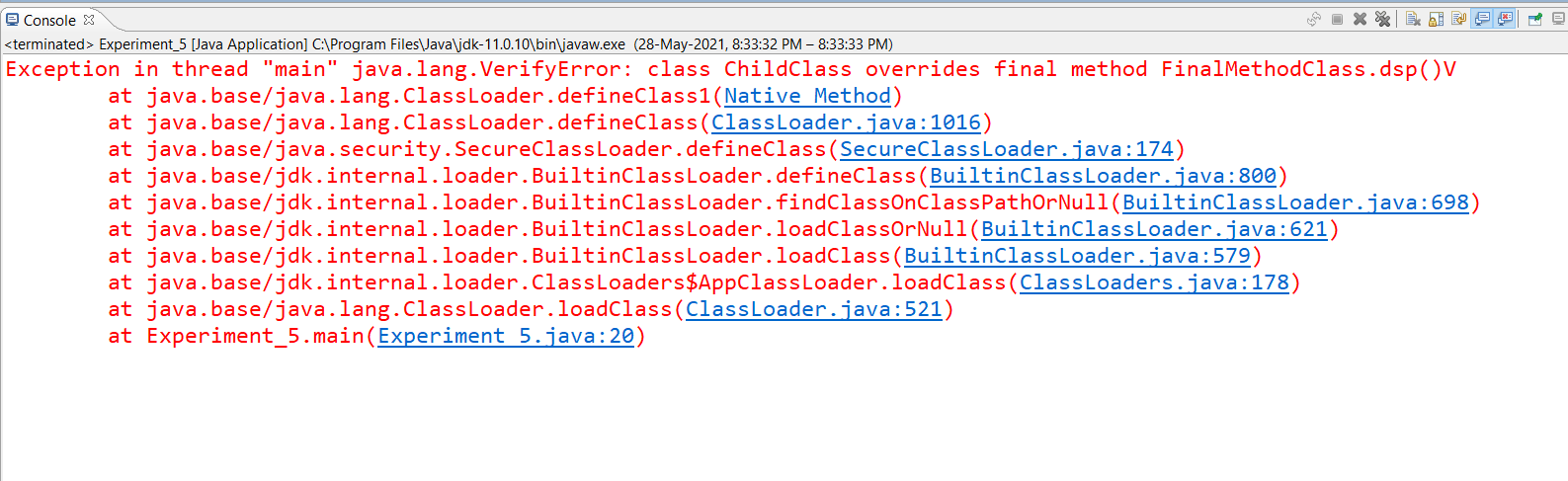
 {

     ChildClass ch = new ChildClass();

 }

}

**Execution:**

****

**2.3 final variable**

This program will not compile.

 class FinalMethodClass

{

    int value = 25;

    final void dsp()

    {

        System.out.println("Inside the final dsp method of FinalMethodClass: "+value);

    }

}

class ChildClass extends FinalMethodClass

{

  @Override

  void dsp()  //we can not Override a final method this will give an error

  {

      System.out.println("Inside the dsp method of ChildClass");

  }

}

public class Experiment\_5 {

 public static void main(String[] arg)

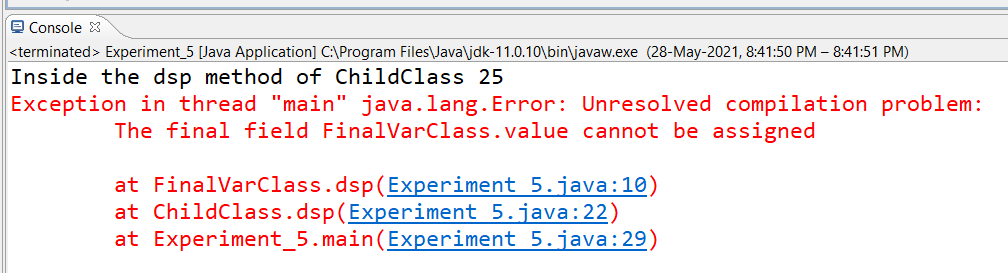
 {

     ChildClass ch = new ChildClass();

 }

}

**Execution**

****

Experiment-6

• Write a program to demonstrate all inner classes.

• Write a program to demonstrate the use of packages.

**1.) Write the program to implement the abstract class.**

Code:

import java.lang.\*;

abstract class Anonymous

{

    int x,y;

    Anonymous(int x,int y)

    {

        this.x=x;

        this.y=y;

    }

    abstract int sum();

    //abstract int mul(int x,int y);

    //abstract int div(int x,int y);

}

class MemberClass

{   int x,y;

    MemberClass(int x,int y)

    {

        this.x=x;

        this.y=y;

    }

    class InnerArea  //Inner Class //this class can access the content of member class

    {

        void areaOfcirecle()

        {

          float area = (float)(2\*x\*3.14f);  //taking x as radius

          System.out.println("Area of circle  is: "+area+" \*Printing From Inner class");

        }

    }

}

class Local

{

 int x,y;

 Local(int x,int y)

 {

     this.x=x;

     this.y=y;

 }

 void calcus()

 {

    class InnerArea

    {

      void cA() //area of circle

      {

          float area = (float)(2\*x\*3.14f);

          System.out.println("Area of circle: "+area+" \*Printing from Local inner class"); //This method is not get called yet

      }

      void rA() // area of rectangle

      {

          float area = (float)(x\*y);

          System.out.println("Area of Rectangle: "+area+" \*Printing From Local Inner Class");

      }

    }

    /\*class InnerPeri

    {

    }\*/

    InnerArea A = new InnerArea();

    A.rA(); //Calling rA method of Local inner class InnerArea from calcus method

  }

}

public class Experiment\_6

{

    public static void main(String arg[])

    {

        int val;

        MemberClass Mobj = new MemberClass(2,5);     //MemberClass

        MemberClass.InnerArea In = Mobj.new InnerArea();

        In.areaOfcirecle();

        Anonymous A = new Anonymous(2,5) //Anonymous class

        {

        int sum()  //defining abstract method of Anonymous class here

         {

           return x+y;

         }

        };                           //End Anonymous

      val= A.sum();  //Calling sum method of Anonymous class

      System.out.println("Sum of x and y is: "+val+" \*Printing from Anonymous class");

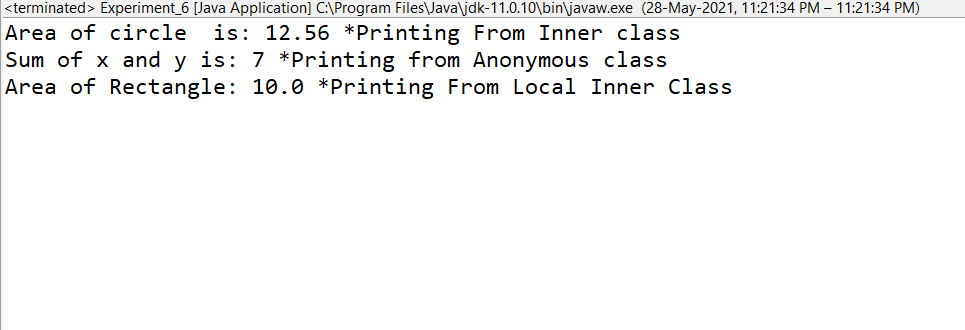
        Local L = new Local(2,5);

        L.calcus(); //calling calcus method of Local class which will call cA,rA method of inner class InnerArea

    }

}

Execution:



**2.) Write a program to demonstrate the use of packages.**

MyClass1 (package1)

package package1; //Current package

public class MyClass1 {

 void dsp1()

 {

   System.out.println("This is the display funtion of MyClass1 in package1");

 }

}

MyClass2 (package2)

package package2; //Current package

public class MyClass2 {

public void dsp2() {

    System.out.println("Inside package2 MyClass2");

}

}

MyInnerClass (package2.inner)

package package2.inner; //Current package

public class MyInnerClass {

public void dspInner()

{

    System.out.println("Inide packge2.inner MyInnerClass");

}

}

MainClass (package1)

package package1; //Current package name

import package2.\*; //Importing package2

import package2.inner.MyInnerClass; //imported MyInnerClass from inner package of package2

public class MainClass {

public static void main(String[] arg)

{

 MyClass1 mc1 = new MyClass1(); //This class is in the same package

 mc1.dsp1();

 MyClass2 mc2 = new MyClass2();  //This class is from package2  which is imported above

 mc2.dsp2();

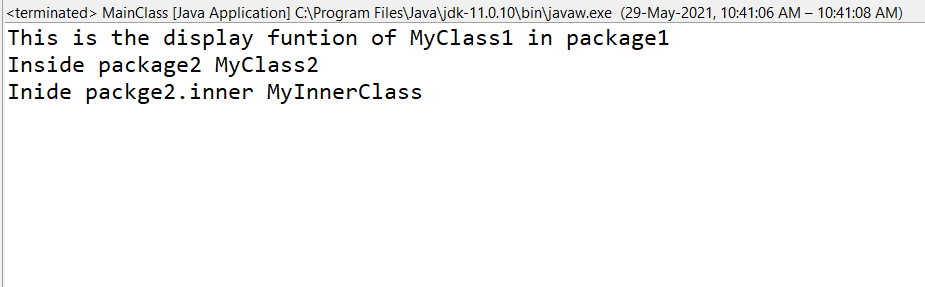
 MyInnerClass mic = new MyInnerClass(); //This class is from inner package  of package2

 mic.dspInner();

}

}

**Execution:**

****

Experiment-7

• Write a program to demonstrate the access specifiers (all the cases)

**Package Name:** mypack2 classes

1.) DefaultClass

package mypack2;

class DefaultClass {

public DefaultClass()

{

     System.out.println("This is the Defaults class constructor");

 }

 void Display()

 {

     System.out.println("This is the Display method of Default class");

 }

}

2.) PrivateClass

package mypack2;

class PrivateClass {

 int k =3; //All the data members will also be private

 private PrivateClass()

 {

     System.out.println("Private Class Constructor");

 }

 {

     System.out.println("Default const Private class mypack1");

 }

private void Display()

 {

     System.out.println("This is the Display method of Priavate class");

 }

}

3.) ProtectedClass

package mypack2;

public class ProtectedClass {

    public ProtectedClass()

 {

     System.out.println("This is the constructor of Protected class");

 }

    protected void Display()

     {

         System.out.println("This is the Display method of Protected class");

     }

}

4.) PublicClass

package mypack2;

public class PublicClass {

 public PublicClass()

   {

    System.out.println("This is the constructor of Public class");

   }

 public void Display()

 {

     System.out.println("This is the Display method of Public class");

 }

}

5.) AccessClass

**(This program will not compile due to private class)**

package mypack2;

//import mypack1;

public class AcessClass {

 public static void main(String arg[])

 {

     PrivateClass Prc = new PrivateClass (); //Accessing private class

     DefaultClass Dc = new DefaultClass();

     ProtectedClass Pc = new ProtectedClass();

     PublicClass Pubc = new PublicClass();

     System.out.println("\*\*\*\*\*\*Execution\*\*\*\*\*\*");

     Prc.Display(); //This is the display of private //Cant be accessed

     Dc.Display(); //------------------------Default

     Pc.Display(); //------------------------Protected

     Pubc.Display(); //----------------------public

 }

}

(This will comile )

package mypack2;

//import mypack1;

public class AcessClass {

 public static void main(String arg[])

 {

    // PrivateClass Prc = new PrivateClass (); //Accessing private class

     DefaultClass Dc = new DefaultClass();

     ProtectedClass Pc = new ProtectedClass();

     PublicClass Pubc = new PublicClass();

     System.out.println("\*\*\*\*\*\*Execution\*\*\*\*\*\*");

     //Prc.Display(); //This is the display of private //Cant be accessed

     Dc.Display(); //------------------------Default

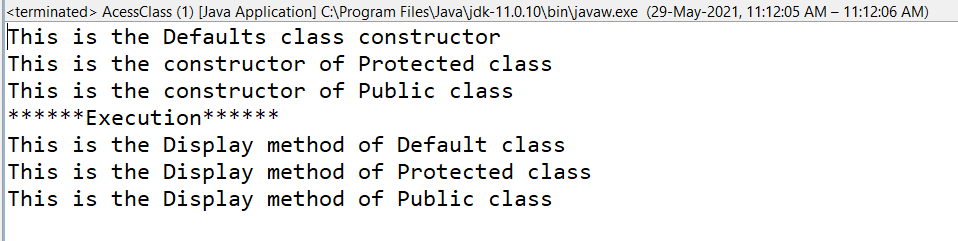
     Pc.Display(); //------------------------Protected

     Pubc.Display(); //----------------------public

 }

}

Execution:



**Package Name:** mypack3 classes

1.) Access Class

(This program will not compile)

package mypack3;

import mypack2.\*;

class A extends ProtectedClass

{

    void displayA()

    {

        System.out.println("Thsi is DisplayA method of class A");

        Display();

    }

}

public class AcessClass {

 public static void main(String arg[])

 {

     PrivateClass Prc = new PrivateClass (); //Accessing private class

     DefaultClass Dc = new DefaultClass();     //Accessing Default Class

     ProtectedClass Pc = new ProtectedClass();  //Accessing Protected Class

     PublicClass Pubc = new PublicClass();      //Accessing Public Class

     A a = new A();

     System.out.println("\*\*\*\*\*\*Execution\*\*\*\*\*\*");

     Prc.Display(); //This is the display of private //Cant be accessed

     Dc.Display(); //------------------------Default //Can't be accessed

     Pc.Display(); //----------------Protected  //Can't be accessed directly

     Pubc.Display(); //----------------------public

    // a.Display(); // Won't work

     a.displayA(); // Will work

 }

}

(This program will compile)

package mypack3;

import mypack2.\*; //Importing package2

class A extends ProtectedClass

{

    void displayA()

    {

        System.out.println("Thsi is DisplayA method of class A");

        Display();

    }

}

public class AcessClass {

 public static void main(String arg[])

 {

    // PrivateClass Prc = new PrivateClass (); //Accessing private class

    // DefaultClass Dc = new DefaultClass();

     ProtectedClass Pc = new ProtectedClass();

     PublicClass Pubc = new PublicClass();

     A a = new A();

     System.out.println("\*\*\*\*\*\*Execution\*\*\*\*\*\*");

     //Prc.Display(); //This is the display of private //Cant be accessed

    // Dc.Display(); //------------------------Default //Can't be accessed

    // Pc.Display(); //--------------Protected  //Can't be accessed directly

     Pubc.Display(); //----------------------public

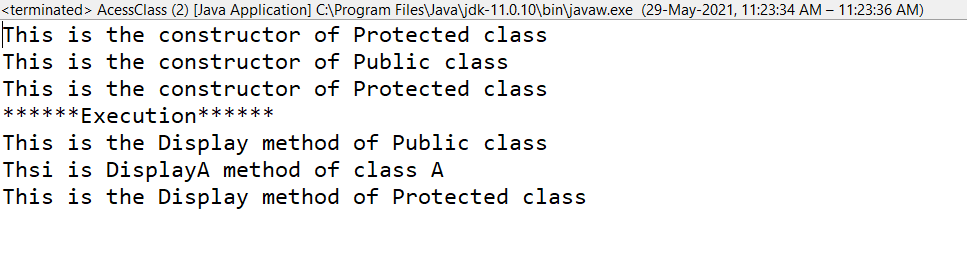
    // a.Display(); // Won't work

     a.displayA(); // Will work

 }

}

**Execution:**



Experiment-8

• Show an example for checked and unchecked exception.

• Implement a program for nested try catch block.

• Implement a program for use of finally of keyword.

• Write a program to demonstrate manual exception handling using the bank management system.

1.) Show an example for checked and unchecked exception.

**1.1 Checked Exception example**

public class CheckedException {

 public static void main(String arg[]) throws InterruptedException //Handling the Exception

 {

     for(int i=0;i<10;i++)

     {

         Thread.sleep(1000);  //Throwing InterruptedException exception

         System.out.println("Value of i: "+i);

     }

     // Alternative way to handle exception

     /\*try

     {

         for(int i=0;i<10;i++)

         {

             Thread.sleep(1000);  //Throwing InterruptedException exception

             System.out.println("Value of i: "+i);

         }

     }

     catch(InterruptedException e)

     {

         System.out.println("Thread Interrupted ");

         e.printStackTrace();

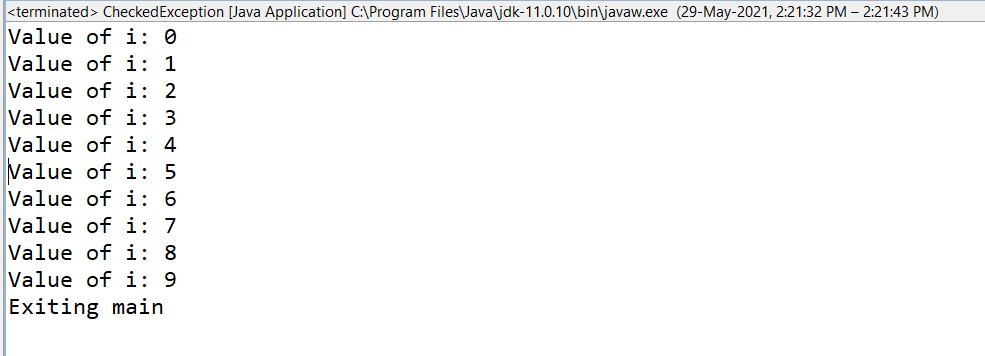
     }\*/

System.out.println("Exiting Main ");

 }

}

**Execution:**

****

**1.2 Unchecked Exception example**

public class UncheckedException {

public static void main(String[] arg) //throws ArithmeticException

{

    int i=0,j=5,div;

    try

    {

    div=j/i;

    }

    catch(ArithmeticException e)

    {

        System.out.println("Arithmetic Exception occured");

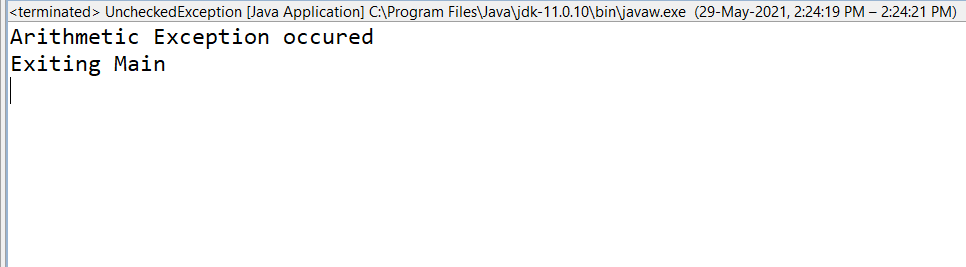
    }

    System.out.println("Exiting Main");

}

}

**Execution**

****

**2.) Implement a program for nested try catch block.**

**3.) Implement a program for use of finally of keyword.**

public class NestedTryCatch {

public static void main(String[] arg)

{

    int i=0,j=5,div;

    int arr[] = new int[5];

    try

    {

        try

        {

            div=j/i;

        }

        catch(ArrayIndexOutOfBoundsException e)

        {

          System.out.println("ArrayIndexOutOfBoundsException catched: ");

        }

    }

    catch(ArithmeticException Ae)

    {

        System.out.println("Arithmetic Exception catched: ");

      try

      {

          arr[7]=25;

      }

      catch(ArrayIndexOutOfBoundsException e)

      {

          System.out.println("ArrayIndexOutOfBoundsException catched: ");

      }

    }

    finally  //finally block

    {

        System.out.println("this is finally Block Executing");

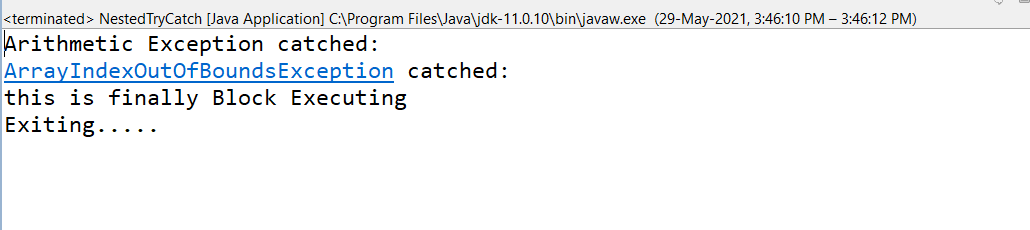
        System.out.println("Exiting.....");

    }

}

}

**Execution:**

****

**4.) Write a program to demonstrate manual exception handling using the bank management system.**

package JavaLab;

import java.util.Scanner;

class LowBalanceException extends RuntimeException

{

    LowBalanceException(String s)

    {

        super(s); //Passing to default handler if nobody wil catch this exception

    }

}

class AgeLimitException extends RuntimeException

{

    AgeLimitException(String s)

    {

        super(s); //Passing to default handler if nobody will catch this exception

    }

}

class BankAccount //Bank Class

{

  int aGe;

  String acNo;

  double baLance=100000; //Default balance

  double aMnt;

  BankAccount(int aMnt,int aGe)

  {

    this.aMnt=aMnt;

    this.aGe=aGe;

    //acNo = "1234567 " + age.toSring();

  }

  void withdraw() throws AgeLimitException,LowBalanceException

  {

      if(aMnt>10000 && aGe<15) //age and amount condition

      {

          throw new  AgeLimitException("You're under age"); //throwing exception

      }

      else if(aMnt>50000 && aGe<45)

      {

          throw new  AgeLimitException("You're under age"); //throwing exception

      }

      else if(baLance<aMnt) //balance condition

      {

          throw new LowBalanceException("You have insufficient balance"); //throwing exception

      }

      else

      {

          baLance-=aMnt; //if withdraw successful deduct the amount

          System.out.println("withdraw successful Current Balance is: "+baLance);

      }

  } //End of withdraw

  void deposit()

  {

      baLance+=aMnt;

  }

}

public class Bank

{

     public static void main(String arg[])

     {

         int i,amount,j;

         Scanner sc = new Scanner(System.in);

         System.out.println("\*RULES");

         System.out.println("Age<15 withdraw limit 10000");

         System.out.println("Age>15 and Age<45 withdraw limit 50000");

         System.out.println("Age>45 no withdraw limit\n");

         System.out.println("Chose one account:");

         System.out.println("1.Denis\nage:10\naBalance:100000 \n\n"

                + "2.Robert\nage:25\naBalance:100000\n\n3.Jerry\nAge:45\naBalance:100000");  //Predefined

         System.out.println("\nEnter Sequence Number:>");

         i = sc.nextInt();

         switch(i)

         {

         case 1:

         {

             System.out.println("Enter your choice: ");

             System.out.println("1. withdraw\n2.deposit ");

             j=sc.nextInt();

             if(j==1)

             {

                 System.out.println("Enter amount to withdraw: ");

                 amount=sc.nextInt();

                // System.out.println("Debug 2");

                 BankAccount BA = new BankAccount(amount,10);

               //  System.out.println("Debug 1");

                 BA.withdraw();

             }

             else if(j==2)

             {

                 System.out.println("Enter amount to deposit: ");

                 amount=sc.nextInt();

                 BankAccount BA = new BankAccount(amount,10);

                 BA.deposit();

             }

             else

             {

                 System.out.println("Invalid input");

             }

         }

         break;

         case 2:

         {

             System.out.println("Enter your choice: ");

             System.out.println("1. withdraw\n2.deposit ");

             j=sc.nextInt();

             if(j==1)

             {

                 System.out.println("Enter amount to withdraw: ");

                 amount=sc.nextInt();

                 BankAccount BA = new BankAccount(amount,25);

                 BA.withdraw();

             }

             else if(j==2)

             {

                 System.out.println("Enter amount to deposit: ");

                 amount=sc.nextInt();

                 BankAccount BA = new BankAccount(amount,25);

                 BA.deposit();

             }

             else

             {

                 System.out.println("Invalid input");

             }

         }

         break;

         case 3:

         {

             System.out.println("Enter your choice: ");

             System.out.println("1. withdraw\n2.deposit ");

             j=sc.nextInt();

             if(j==1)

             {

                 System.out.println("Enter amount to withdraw: ");

                 amount=sc.nextInt();

                 BankAccount BA = new BankAccount(amount,45);

                 BA.withdraw();

             }

             else if(j==2)

             {

                 System.out.println("Enter amount to deposit: ");

                 amount=sc.nextInt();

                 BankAccount BA = new BankAccount(amount,45);

                 BA.deposit();

             }

             else

             {

                 System.out.println("Invalid input");

             }

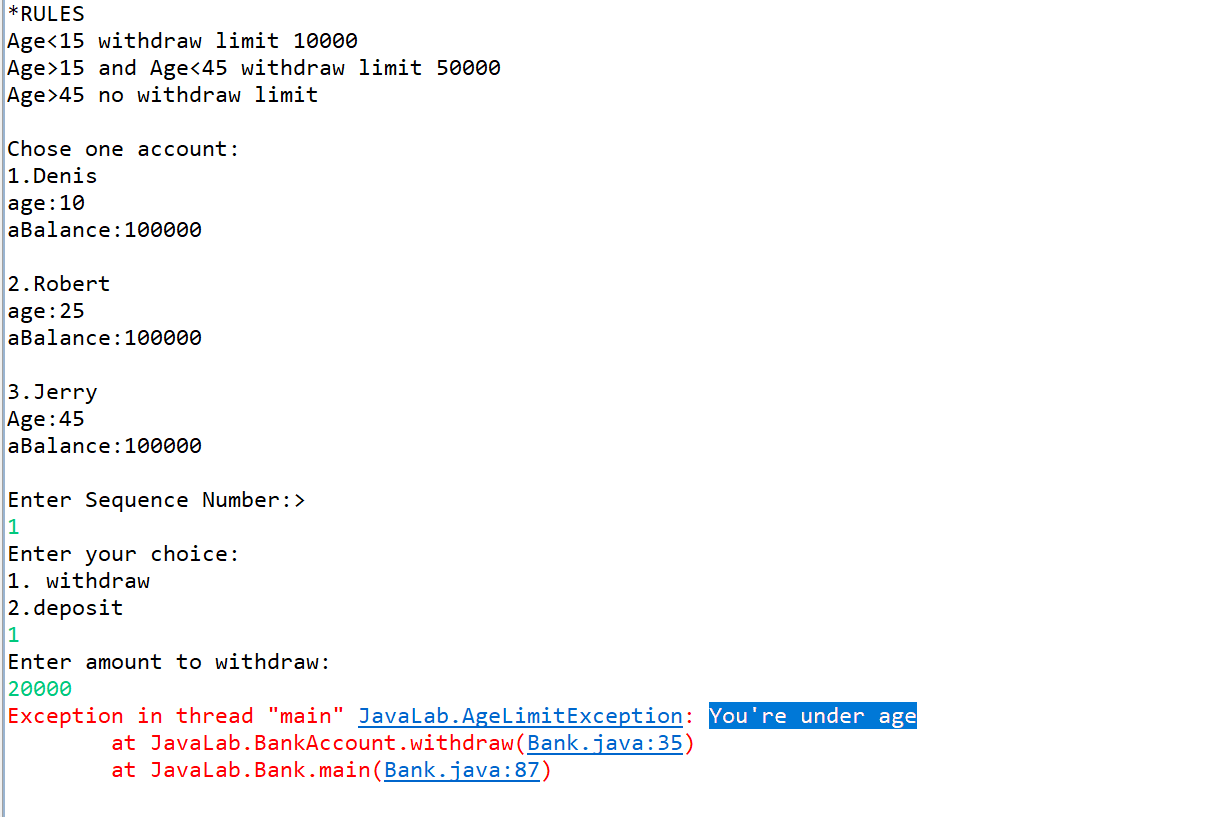
         }

         break;

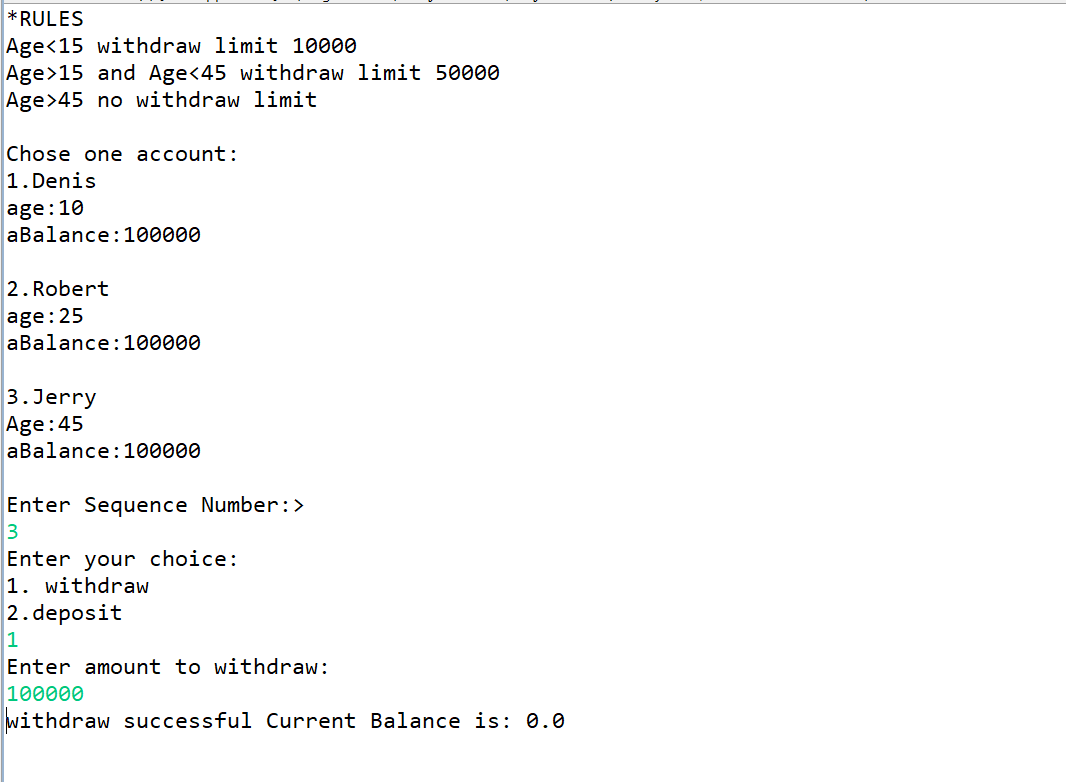
         } //end swithc

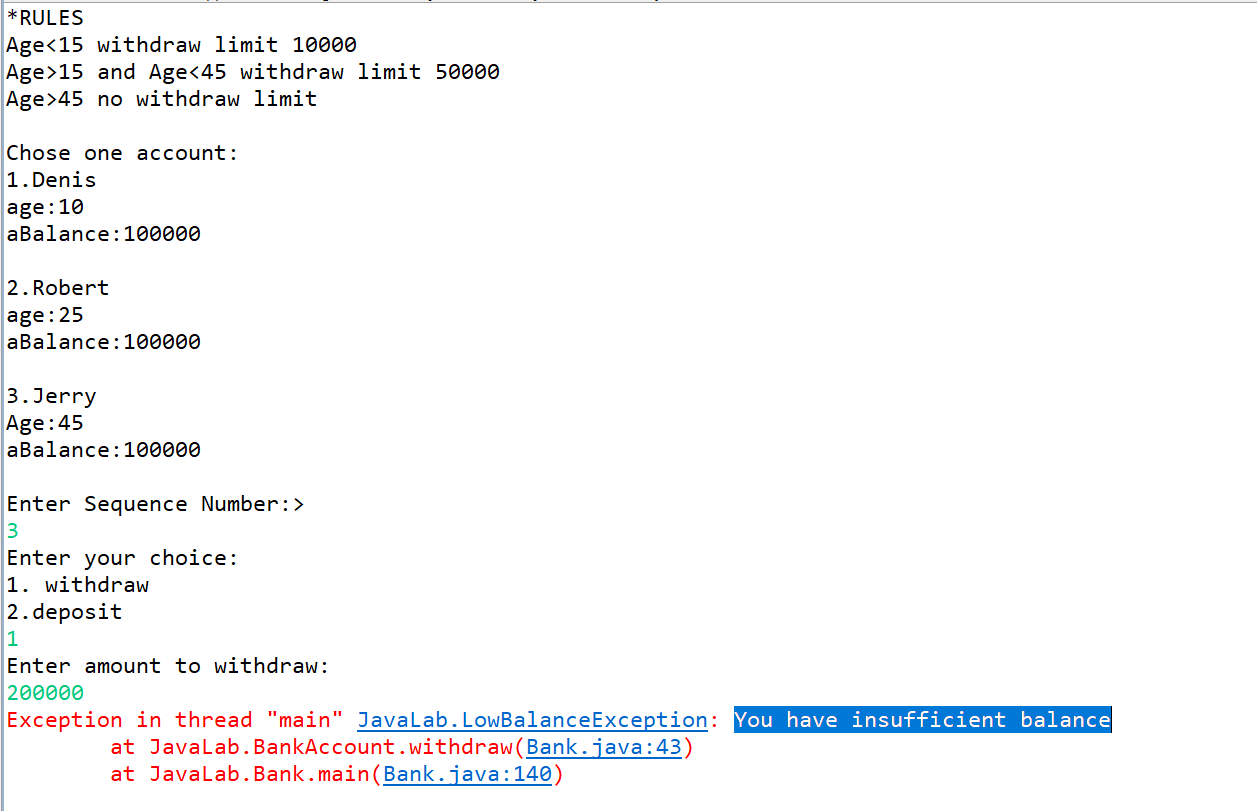
     } //End of main method

}

**Execution:1**

**Execution:2**

****

**Execution:3 **

Experiment-9

• Write a program to demonstrate thread priority.

• Write a program to demonstrate the join method.

• Write a program to demonstrate multi-threading

**1.) Write a program to demonstrate thread priority.**

class NewThread implements Runnable

{

    String name;

    int pri;

    int counter=0;

    Thread t;

    boolean flag=true;  //flag initialized as true

    NewThread(String n,int val)

    {

        name=n;

        pri=val;

        t = new Thread(this,name);

        t.start();

        t.setPriority(pri); //setting the priority of the Thread

    }

    public void run()

    {

        while(flag)  //while loop will run until the flag is true

        {

            counter++;

        }

    }

}

public class CounterThread {

 public static void main(String[] arg)

 {

 NewThread T1 = new NewThread("Thread-1",1);

 NewThread T2 = new NewThread("Thread-2",7);

 Thread.currentThread().setPriority(9); //Setting priority to Main Thread to  avoid a Crash

 try

 {

     Thread.sleep(1000);

    T1.flag=false; //To stop the Thread-1 from running // but not working ?

    T2.flag=false; //To stop the Thread-2 from running  // but not working ?

    //T1.t.stop();

 }

 catch (InterruptedException e)

 {

     System.out.println("Interruption occured");

 }

 System.out.println("T1 flag "+T1.flag);

 System.out.println("T2 flag "+T2.flag);

 System.out.println("Counter of first Thread 1 Priority:1 -> "+ T1.counter );

 System.out.println("Counter of first Thread 2 Priority:7-> "+ T2.counter );

 System.out.println("Thread 1:"+T1.t.isAlive());

 System.out.println("Thread 1:"+T2.t.isAlive());

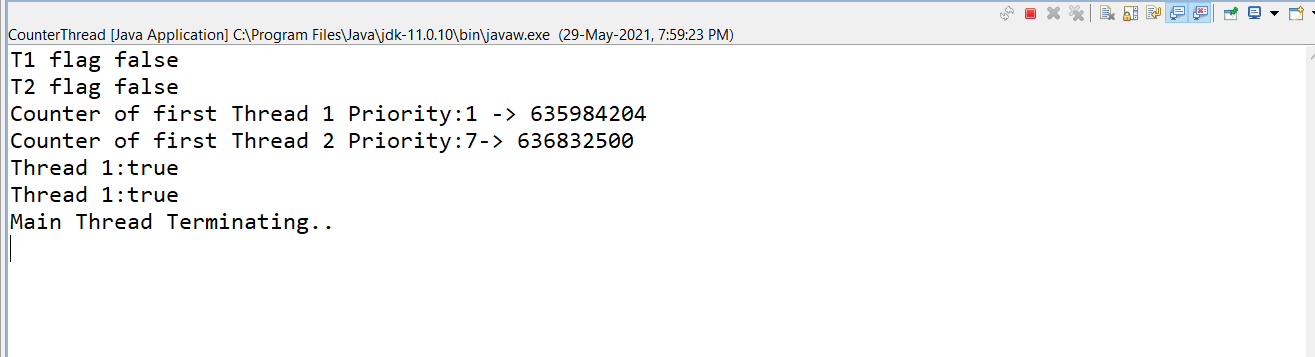
 System.out.println("Main Thread Terminating"+ "..");

 }

}

//Problem Threads are running until we stop the program manually

**Execution:**

****

**2.) Write a program to demonstrate the join method.**

**3.) Write a program to demonstrate multi-threading**

class NewThreadx implements Runnable

{

 String name;

 int value;

 Thread t;

 NewThreadx(String name,int n)

 {

     this.name=name;

     value=n;

     t = new Thread(this,name);

     System.out.println("Thread name: "+t+" Started");

     t.start();

 }

 public void run()

 {

    try

    {

        for(int i=value;i<value+5;i++)

        {

            Thread.sleep(500);

            System.out.println(name+" "+i);

        }

    }

    catch (InterruptedException e)

    {

        System.out.println("Thread Interrupted");

    }

 }

}

public class isAliveJoin

{

    public static void main(String arg[])

    {

        System.out.println("Main Thread Started: ");

        //Creating the threads.

        NewThreadx x1=new NewThreadx("Thread-01",10);

        NewThreadx x2=new NewThreadx("Thread-02",20);

        NewThreadx x3=new NewThreadx("Thread-03",30);

        System.out.println("Thread:-> "+x1.t.getName()+": isAlive-> "+x1.t.isAlive());

        System.out.println("Thread:-> "+x2.t.getName()+": isAlive-> "+x2.t.isAlive());

        System.out.println("Thread:-> "+x3.t.getName()+": isAlive-> "+x3.t.isAlive());

        try

        {

          x1.t.join();

          x2.t.join();

          x3.t.join();

        }

        catch (InterruptedException e)

        {

            System.out.println("Main Thread Interruped...");

        }

        System.out.println("Thread:-> "+x1.t.getName()+": isAlive-> "+x1.t.isAlive());

        System.out.println("Thread:-> "+x2.t.getName()+": isAlive-> "+x2.t.isAlive());

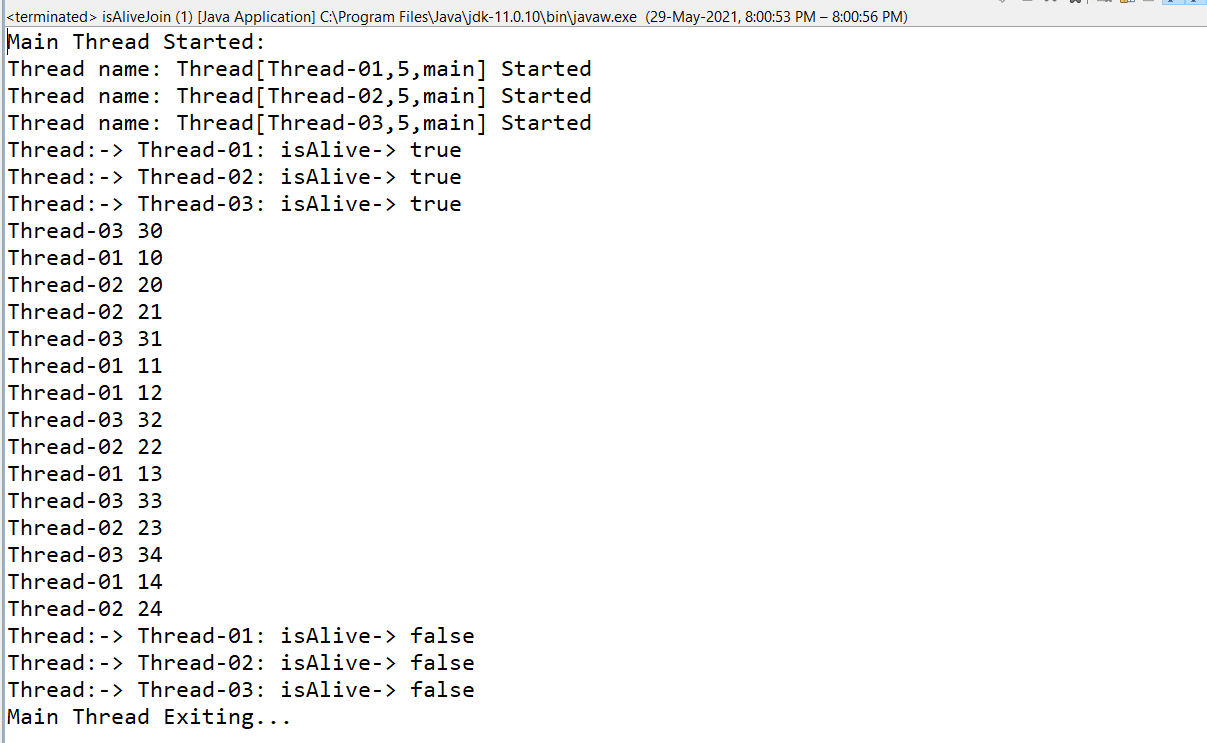
        System.out.println("Thread:-> "+x3.t.getName()+": isAlive-> "+x3.t.isAlive());

      System.out.println("Main Thread Exiting...");

    }

}

**Execution:**

****

Experiment-10

• Write a program to demonstrate deadlock.

• Write a program to demonstrate thread synchronization using wait () and notify () using the student and certificate example.

• Write a program to demonstrate all the synchronization methods.

**1.) Write a program to demonstrate deadlock.**

**Code:**

class Ax

{

    synchronized void methodAx(Bx b)

    {

   System.out.println("Method1Ax called by"+Thread.currentThread().getName());

        try {Thread.sleep(1000);}

        catch(InterruptedException e) {System.out.println("Interrupted");}

        System.out.println("Calling Bx class bxDump method");

        b.bxDump();

    }

    synchronized void axDump()

    {

        System.out.println("axDump called by"+Thread.currentThread().getName());

    }

}

class Bx

{

    synchronized void methodBx(Ax a)

    {

   System.out.println("Method1Bx called by"+Thread.currentThread().getName());

        try {Thread.sleep(2000);}

        catch(InterruptedException e) {System.out.println("Interrupted");}

        System.out.println("Calling Ax class axDump method");

        a.axDump();

    }

    synchronized void bxDump()

    {

      System.out.println("bxDump called by"+Thread.currentThread().getName());

    }

}

abstract class References // Reference class

{

    static Ax a = new Ax();

    static Bx b = new Bx();

    //It is necessary to to declare class objects as static to perform synchronization

    //by taking abject from Reference class

}

class Thread1 extends References  implements Runnable

{

 Thread t;

// Ax a;

// Bx b;

    Thread1()

    {

        //this.a=a;

        //this.b=b;

        t = new Thread(this,"Thread1");

        t.start();

    }

    public void run()

    {

      a.methodAx(b);

    }

}

class Thread2 extends References  implements Runnable

{

 Thread t;

 //Ax a;

 //Bx b;

    Thread2()

    {

    //  this.a=a;

        //this.b=b;

        t = new Thread(this,"Thread2");

        t.start();

    }

    public void run()

    {

     b.methodBx(a);

    }

}

public class DeadLock {

 public static void main(String[] arg)

 {

    // Ax a = new Ax(); //Creating Ax's object  //wihtout References class and pass in threads

     //Bx b = new Bx(); //Creating Bx's objext // wihtout References class and pass in threads

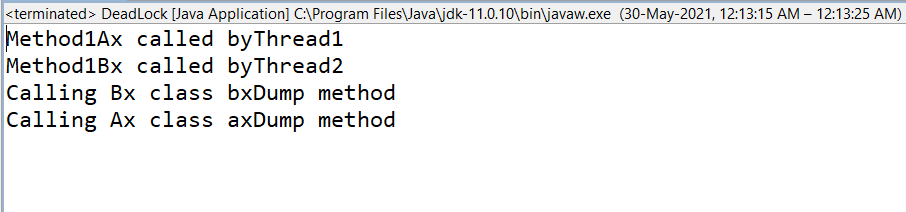
    new Thread1(); //

    new Thread2();  //

 }

}

**Execution:**

****

**2.) Write a program to demonstrate thread synchronization using wait () and notify () using the student and certificate example.**

**Code:**

class Threader implements Runnable

{

    boolean flag = false;

    static int  number=0;

    int remainder;

    static Object obj=new Object();

    Threader(int remainder)

    {

        this.remainder=remainder;

    }

    @Override

    public void run() {

        while (true) {

             synchronized (obj)  //Certificate Producing Block

              {

                 while (number%3!= remainder) {

                     try {

                         obj.wait();

                     } catch (InterruptedException e) {

                         System.out.println("Thread Interrupted");

                     }

                 }

   System.out.println("Ceritficate Generated for Student: "+((++number%3)+1));

                obj.notifyAll();

               }

            synchronized (obj) { //Certificate Consuming block

                while (number%3!= remainder) {

                    try {

                        obj.wait();

                    } catch (InterruptedException e) {

                        System.out.println("Thread Interrupted");

                    }

                }

  System.out.println("Certificate Taken by"+Thread.currentThread().getName());

            }

        }

    }

}

public class StudentCertificate {

    public static void main(String[] args) {

        Threader p1=new Threader(0);

        Threader p2=new Threader(1);

        Threader p3=new Threader(2);

        Thread t1=new Thread(p1,"Student-1");

        Thread t2=new Thread(p2,"Student-2");

        Thread t3=new Thread(p3,"Student-3");

        t3.start();

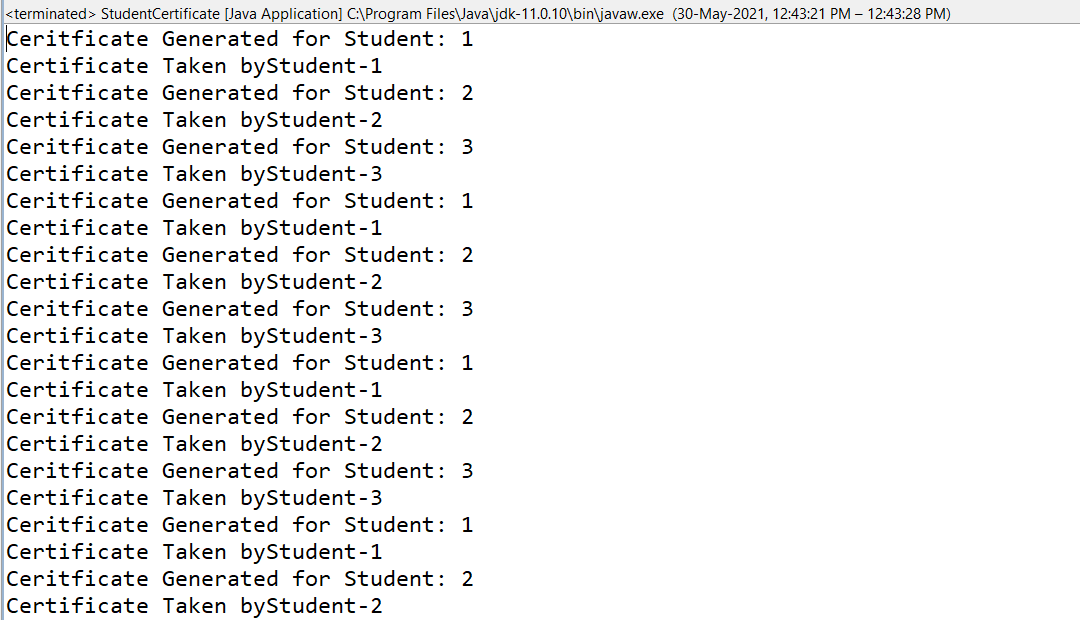
        t1.start();

        t2.start();

    }

}

**Execution:**

****

**3.) Write a program to demonstrate all the synchronization methods.**

**Code:**

class A

{

    synchronized void producer() //Synchronization method

    {

        System.out.println("Produced by "+Thread.currentThread().getName());

    }

    synchronized void consumer() //Synchronization method

    {

        System.out.println("Consumed by"+Thread.currentThread().getName());

    }

}

class Thread1 implements Runnable

{

 Thread t1;

 A a;

    Thread1(A a)

    {

     this.a=a;

      t1=new Thread(this,"Thread1");

      t1.start();

    }

    public void run()

    {

        synchronized(a) //Synchronization block

        {

            a.producer();

        }

        synchronized(a) //Synchronization block

        {

            a.consumer();

        }

    }

}

class Thread2 implements Runnable

{

 Thread t1;

 A a;

    Thread2(A a)

    {

      this.a=a;

      t1=new Thread(this,"Thread2");

      t1.start();

    }

    public void run()

    {

        synchronized(a)  //Synchronization block

        {

            a.producer();

        }

        synchronized(a) //Synchronization block

        {

            a.consumer();

        }

    }

}

class ProducerConsumer

{

    public static void main(String[] arg)

    {

        A a = new A();

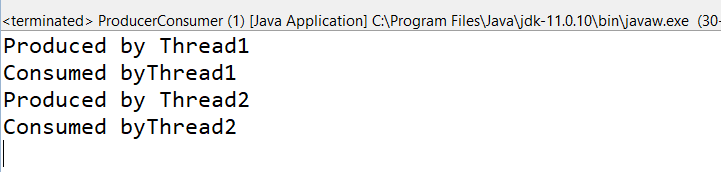
        new Thread1(a);

        new Thread2(a);

    }

}

**Execution:**

****

Experiment-11

• Write a program for JDBC as follows: Make an employee table with id, name, salary, phone, allowances.

1. Insert 5 records dynamically, mean get input from user at runtime 2. Increase the allowances by 10% for employees having salary greater than 15000

3. show the employee records with salary less than 15000 Note. You need to show records in last to first manner

1. Insert 5 records dynamically, mean get input from user at runtime.

**Code:**

import java.sql.\*;

import java.util.Scanner;

public class EmployeeInsertion {

public static void main(String arg[]) throws Exception

{

    Class.forName("com.mysql.cj.jdbc.Driver");

    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/OOP","root","bantu123");

    /\*

     User: root

     Password: bantu123

     Database name: OOP

     \*/

    PreparedStatement smt = con.prepareStatement("INSERT INTO employee values(?,?,?,?,?)"); //Created a prepared statement

    Statement smt1 = con.createStatement();

    Scanner sc = new Scanner(System.in);

    for(int i=1;i<=5;i++) //Loop for 5 enteries

    {

   System.out.println("Enter Details for Employee: "+i);

    System.out.print("Enter Employee ID: ");

    int dno=sc.nextInt();

    System.out.print("Enter Employee Name: ");

    String dname = sc.next();

    System.out.print("Enter Employee PhoneNumber");

    String dphone = sc.next();

    System.out.print("Enter Employee salary: ");

    double salary = sc.nextDouble();

    System.out.print("Enter Employee Allowance:");

    double alw = sc.nextDouble();

    smt.setInt(1,dno); //inserting the values in table

    smt.setString(2,dname); //

    smt.setString(3,dphone); //

    smt.setDouble(4,salary);//

    smt.setDouble(5,alw);//

    smt.executeUpdate(); //Executing the update to the table

    }

    System.out.println("Insertion process done! Exiting......");

    ResultSet rs = smt1.executeQuery("SELECT \* FROM employee"); //insert query here

    System.out.println("\n\*\*Printing Employee Table\*\*\n");

    while(rs.next())                //printing the table

     {

         System.out.print("Employee ID: "+rs.getInt(1)+" ");  //Retrieving the data from table

         System.out.print("Employee Name: "+rs.getString(2)+" "); //

         System.out.println("Employee PhoneNumber: "+rs.getString(3)); //

         System.out.println("Employee salary: "+rs.getDouble(4)); //

         System.out.println("Employee Allowance: "+rs.getInt(5)); //

     }

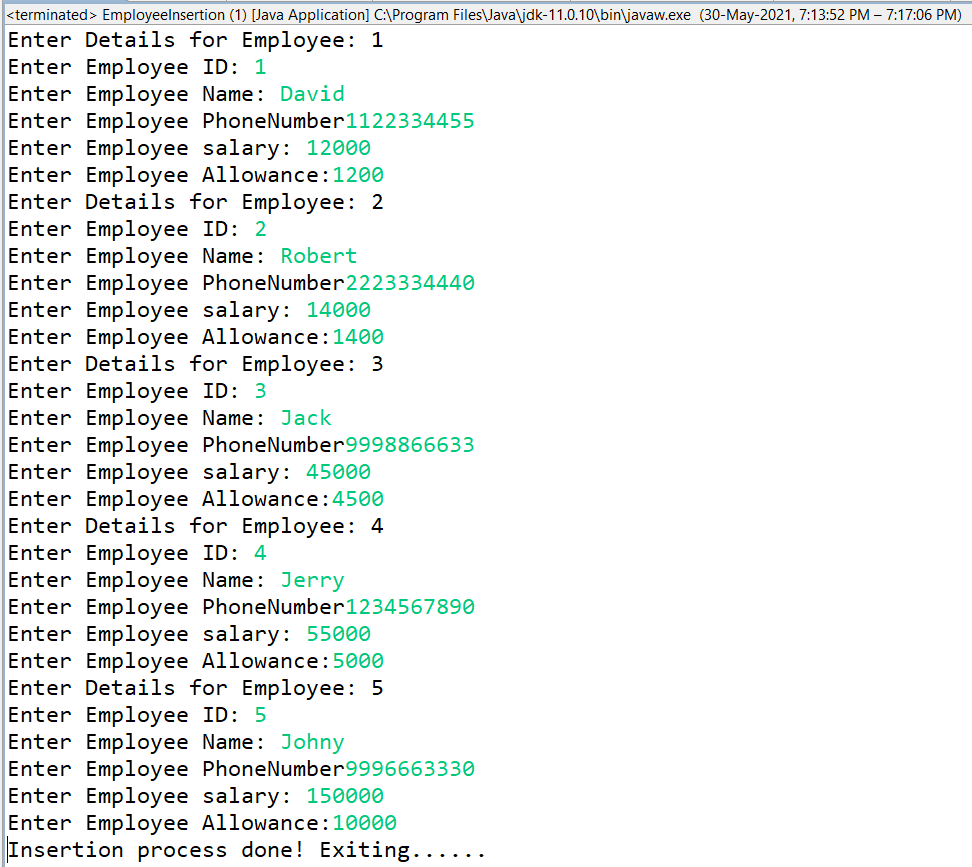
    smt.close(); //Closing the statement

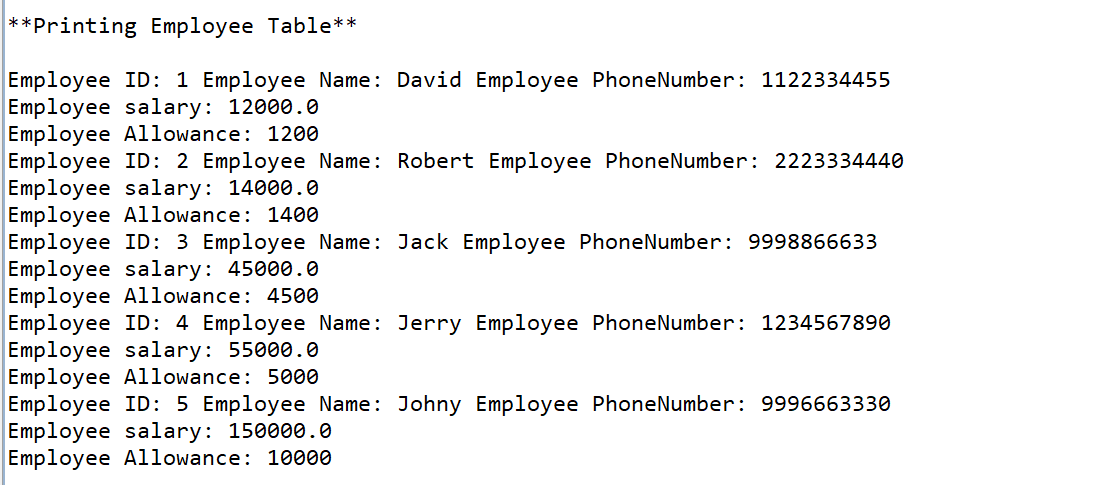
    con.close(); //Closing the connection

}

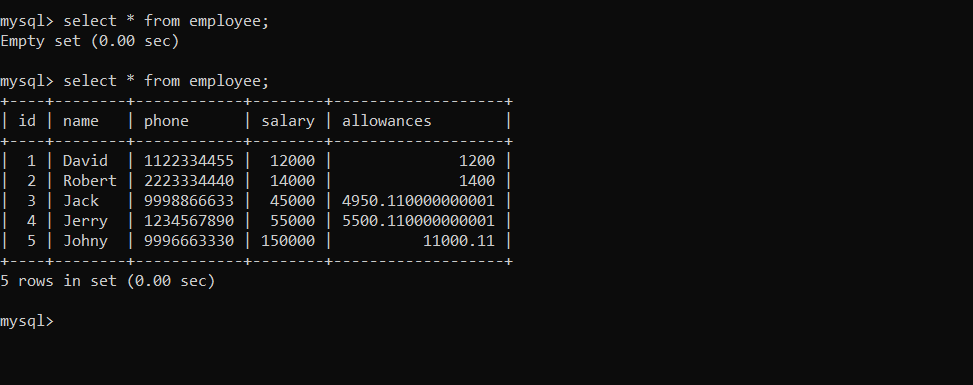
}

**Execution:**

****

****

**SQL TABLE:**

****

2. Increase the allowances by 10% for employees having salary greater than 15000 .

**Code:**

import java.sql.\*;

import java.util.Scanner;

public class AllowanceUpdate {

public static void main(String arg[]) throws Exception

{

    Class.forName("com.mysql.cj.jdbc.Driver");

    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/OOP","root","bantu123"); //Connection

    /\*

     User: root

     Password: bantu123

     Database name: OOP

     \*/

  Statement smt = con.createStatement();  //Crating Statement

  smt.executeUpdate("update employee set allowances=allowances+(allowances\*0.1) where salary>15000 "); // to update sql query

  ResultSet rs = smt.executeQuery("select \* from employee");

   while(rs.next())

   {

         System.out.println("Employee ID: "+rs.getInt(1)+" ");  //Retrieving the data from table

         System.out.println("Employee Name: "+rs.getString(2)+" "); //

         System.out.println("Employee PhoneNumber: "+rs.getString(3)); //

         System.out.println("Employee salary: "+rs.getDouble(4)); //

         System.out.println("Employee Allowance: "+rs.getInt(5)); //

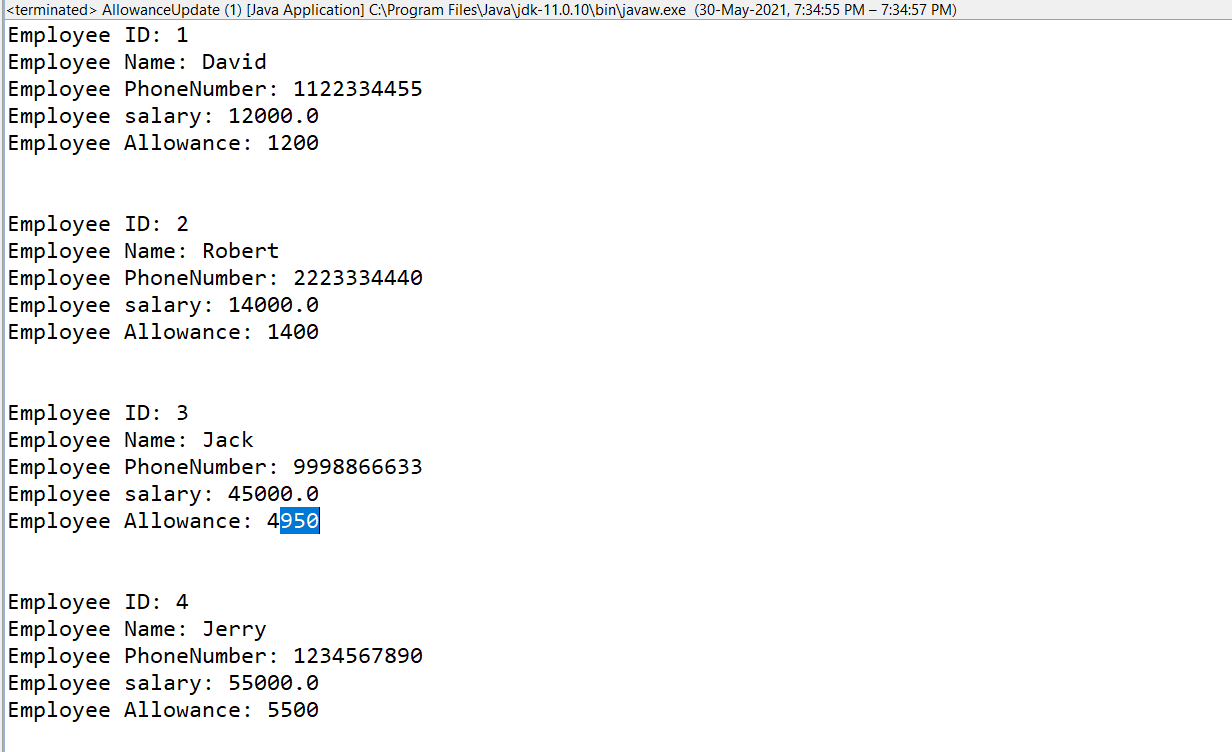
         System.out.println("\n");

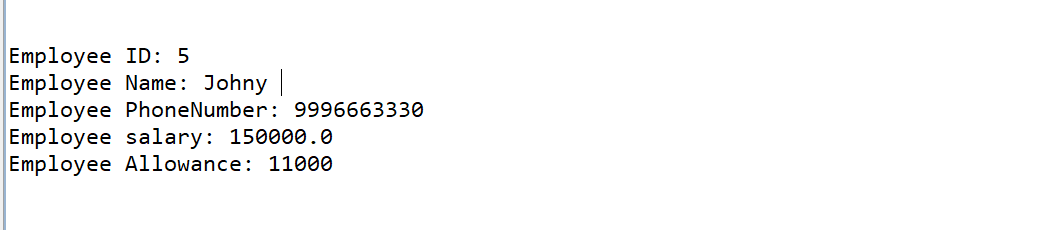
   }

}

}

**Execution:**

****

****

3. show the employee records with salary less than 15000 Note. You need to show records in last to first manner.

**Code:**

import java.sql.\*;

import java.util.Scanner;

public class LessthenSalary {

public static void main(String arg[]) throws Exception

{

    Class.forName("com.mysql.cj.jdbc.Driver");

    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/OOP","root","bantu123");

    /\*

     User: root

     Password: bantu123

     Database name: OOP

     \*/

  Statement smt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_READ\_ONLY);

  // Statement smt = con.createStatement();

  ResultSet rs = smt.executeQuery("SELECT \* FROM employee WHERE salary<=15000");

  //where employee `id` is the `PRIMARY KEY` so no need of `ORDER BY` specification

  rs.afterLast();   //Retrieving the data last to first manner from table

   while(rs.previous())

   {

       System.out.println("Employee ID: "+rs.getInt(1)+" ");  //Retrieving the data from table

         System.out.println("Employee Name: "+rs.getString(2)+" "); //

         System.out.println("Employee PhoneNumber: "+rs.getString(3)); //

         System.out.println("Employee salary: "+rs.getDouble(4)); //

         System.out.println("Employee Allowance: "+rs.getInt(5)); //

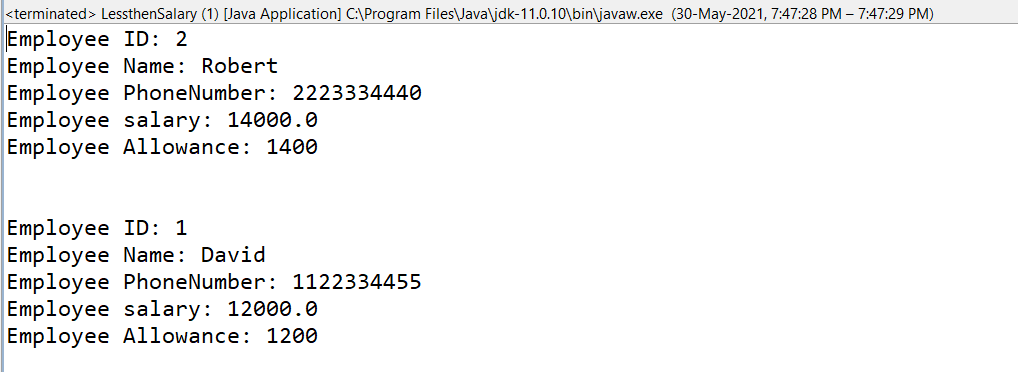
         System.out.println("\n");

   }

}

}

**Execution:**

****