Object Oriented Programming with Java

Lab Manual

P.E.S.College of Engineering, Mandya-571401

**Department of Information Science & Engineering**

Object Oriented Programming with Java (Integrated-Course)

Lab Manual/P21IS305

Vision

“The department strives to equip our graduates with Knowledge and Skills to contribute significantly to Information Science & Engineering and enhance quality research for the benefit of society”.

Mission

M1: To provide students with state of art facilities and tools of Information Science & Engineering to become productive, global citizens and life-long learners.

M2: To prepare students for careers in IT industry, Higher education and Research.

M3: To inculcate leadership qualities among students to make them competent Information Science & Engineering professionals or entrepreneurs.

Program Educational Objectives (PEOs)

Graduates of the program will be able to

PEO1: Establish a productive Information Science & Engineering career in industry, government or academia.

PEO2: Interact with their peers in other disciplines by exhibiting professionalism and team work to contribute to the economic growth of the country.

PEO3: Promote the development of solutions to the problems in Information Science using hardware and software integration.

PEO4: Pursue higher studies in Engineering, Management or Research.

Program Outcomes (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Prerequisite**

Instruction: 1. Need to install intellJ IDEA By visiting to [www.jetbrains.com](http://www.jetbrains.com)

2. Select Classic mode for download

3. Create New Project

4. Go to source code and select main then editor enables you to write the Code.

|  |
| --- |
| Course Outcomes (COs) :  1. Understand and explore the fundamental concepts of object oriented programming language  2. Apply the syntax and semantics of java for solving a given problem  3. Analyze the given Java code snippet to identify the bugs and correct the code  4. Conduct experiments using IDE to demonstrate the features of Java programming language |

**List of Experiments**

|  |  |
| --- | --- |
| Sl.No | Experiments Title |
| 1 | Write a program to find the sum of the series 1 + 1/ (2\*2) + 1/(3\*3) + 1/(4\*4) + ….. + 1/(n\*n). |
| 2 | Write a Java program for printing Pascals's Triangle ( 5 rows) using nested loops. |
| 3 | Write a program that accepts three numbers from the user and prints "increasing" if the numbers are in increasing order, "decreasing" if the numbers are in decreasing order, and "Neither increasing or decreasing order" otherwise. |
| 4 | Create a Java class called Complex with the following details and variables within it as (i) Real (ii) Imaginary. Develop a Java program to perform addition and subtraction of two complex numbers by using the method add() and subtract( ) respectively by passing object as parameter and display result using method display(). Initialize the real and imaginary values of the complex number using parameterized constructor. |
| 5 | A class called MyTime, which models a time instance with private instance variables: hour: between 0 to 23, minute: between 0 to 59, constructor shall invoke the setTime() method to set the instance variable (setTime(int hour, int minute): It shall check if the given hour and minute are valid before setting the instance variables). Define methods - getHour(), getMinute(),nextMinute( )Update this instance to the next minute and return this instance. Take note that the nextMinute() of 23:59 is 00:00 nextHour() is similar to the above. Write the code for the MyTime class. Also write a test program (called TestMyTime) to test all the methods defined in the MyTime class. |
| 6 | Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. Create a class Account that stores customer name, account number and type of account. From this derive the classes Curr-acct and Sav-acct to make them more specific to their requirements. The savings account provides compound interest and withdrawal facilities. The current account does not provide interest. Current account holders should also maintain a minimum balance (Rs 5000) and if the balance falls below this level, a service charge (Rs 100) is imposed. Include the necessary methods in order to achieve the following tasks: Accept deposit from customer and update the balance. Display the balance. Compute and deposit interest Permit withdrawal and update the balance Check for the minimum balance(only for Current account), impose penalty if necessary and update the balance |
| 7 | Design a base class Circle with member variables (radius of type double and color of type character), methods (getRadius(), getArea()) and constructors (Circle(radius), Circle(radius, color)). Derive subclass called Cylinder from the super class Circle with member variable (height) of type double, public methods (getHeight(), getVolume(), getArea()) and constructors(Cylinder(height),Cylinder(height,radius), Cylinder(height, radius, color)). Create the two instances of cylinder and print similar cylinders if the area, volume and color of cylinders are same. Demonstrate the code reuse and polymorphism properties of Object oriented programming by inheriting the constructors and methods of the base class. |
| 8 | Create two classes called HDFCAccount and StateBankAccount. that implements all the methods defined in interface Account.Declare the methods getBalance, deposit and withdraw in Account interface. HDFCAccount uses member variables deposits and withdrawals for maintaining the balance, where as State BankAccount uses only balance to maintain the balance. In the main method create objects of HDFCAccount and StateBankAccount, but assigned them to the reference of the interface Account. Also write an method to print balance in main which prints the balance amount. |
| 9 | Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class internals has an array that stores the internal marks scored in six courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array thatstores the SEE marks scored in six courses of the current semester of the student. Import the two packages in a file that declares the final marks of N students in all six courses. |
| 10. | Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number. |
| 11. | Write a java program to handle the following exceptions based on choice made by the user by writing suitable try and catch block.  i) ArithmeticException  ii) ArrayIndexOutOfBoundsException  iii) NumberFormatException  iv) StringIndexOutOfBoundException  v) NullPointerException |
| 12. | Define a class Sort with generic method by name Arrange(T[]) and Display(T[]). Write a program to sort array elements of different data types. |

References:

1. The Complete Reference - Java , Herbert Schildt , 11th Edition , 2019, McGraw Hill Education

**Course Articulation Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Co/PO | PO 1 | PO 2 | PO31 | PO4 | PO5 | PO6 | PO7 | PO 8 | PO9 | PO10 | PO11 | PO12 |
| CO 1 | 2 |  |  |  |  |  |  |  |  |  |  |  |
| CO 2 | 2 | 2 | 2 |  |  |  |  |  |  |  |  |  |
| CO 3 | 2 | 2 | 1 |  |  |  |  |  |  |  |  |  |
| CO 4 | 2 | 2 | 2 |  | 2 |  |  |  | 1 |  |  | 1 |

Lab Program: 1

Aim of the Experiment: Write a program to find the sum of the series 1 + 1/ (2\*2) + 1/(3\*3) + 1/(4\*4) + ….. + 1/(n\*n).

Algorithm

Solution for Program 1. Write a Program to find sum of the series

1+1/(2\*2)+1/(3\*3)+1/(4\*4)....+1/(n\*n).

Code:

import java.io.\*;

class GFG {

    // Function to calculate the following series

    static float Series( float n)

    {

       float i;

       float sums = 0;

        for (i = 1; i <= n; i++)

            sums += 1/(i \* i);

        return sums;

    }

    // Driver Code

    public static void main(String[] args)

    {

         float n = 3;

         float res = Series(n);

        System.out.println(res);

    }

}

Out put: 1.36

Lab Program: 2

Program 2. Write a Program for printing Pascal’s Triangle (5Rows) Using Nested Loop

Theory: Pascal’s triangle is a pattern of the triangle which is based on nCr. below is the pictorial representation of Pascal’s triangle.

Example:

Input: N = 5

Output:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

**Algorithm:**

* Take a number of rows to be printed, assume it to be n
* Make outer iteration i from 0 to n times to print the rows.
* Make inner iteration for j from 0 to (N – 1).
* Print single blank space ” “
* Close inner loop (j loop) //it’s needed for left spacing
* Make inner iteration for j from 0 to i.
* Print nCr of i and j.
* Close inner loop.
* Print newline character (\n) after each inner iteration.

Code:

// Print Pascal's Triangle in Java

import java.io.\*;

class GFG {

    public int factorial(int i)

    {

        if (i == 0)

            return 1;

        return i \* factorial(i - 1);

    }

    public static void main(String[] args)

    {

        int n = 4, i, j;

        GFG g = new GFG();

        for (i = 0; i <= n; i++) {

            for (j = 0; j <= n - i; j++) {

                // for left spacing

                System.out.print(" ");

            }

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

                // nCr formula

                System.out.print(

                    " "

                    + g.factorial(i)

                          / (g.factorial(i - j)

                             \* g.factorial(j)));

            }

            // for newline

            System.out.println();

        }

    }

}

**Output**

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Lab Program: 3

Write a Program That accept three numbers from user and prints “increasing “ if the numbers from the user are in increasing order, “decreasing” if the numbers are in increasing order, and “neither” increasing or decreasing order” otherwise.

Code:

import java.util.Scanner;

class Exercise31 {

public static void main(String[] args)

{

Scanner in = new Scanner(System.in);

System.out.print("Input first number: ");

double x = in.nextDouble();

System.out.print("Input second number: ");

double y = in.nextDouble();

System.out.print("Input third number: ");

double z = in.nextDouble();

if (x < y && y < z)

{

System.out.println("Increasing order");

}

else if (x > y && y > z)

{

System.out.println("Decreasing order");

}

else

{

System.out.println("Neither increasing or decreasing order");

}

}

}

Output:

Input first number: 1524

Input second number: 2345

Input third number: 3321

Increasing order

Lab Program: 4

Create Java class called complex with the following details and variables within it as (i) Real (ii) Imaginary.

Develop a Java program to perform addition and subtraction of two complex numbers to perform addition and subtraction of two complex numbers by using add() and subtract() methods by passing object as parameter and display result using method display(). Initialize the real and imaginary values of the complex numbers using parameterized constructer.

Code:

import java.util.\*;

// User Defined Complex class

class Complex {

    // Declaring variables

    int real, imaginary;

    // Empty Constructor

    Complex()

    {

    }

    // Constructor to accept

    // real and imaginary part

    Complex(int tempReal, int tempImaginary)

    {

        real = tempReal;

        imaginary = tempImaginary;

    }

    // Defining addComp() method

    // for adding two complex number

    Complex addComp(Complex C1, Complex C2)

    {

        // creating temporary variable

        Complex temp = new Complex();

        // adding real part of complex numbers

        temp.real = C1.real + C2.real;

        // adding Imaginary part of complex numbers

        temp.imaginary = C1.imaginary + C2.imaginary;

        // returning the sum

        return temp;

    }

    // Defining subtractComp() method

    // for subtracting two complex number

    Complex subtractComp(Complex C1, Complex C2)

    {

        // creating temporary variable

        Complex temp = new Complex();

        // subtracting real part of complex numbers

        temp.real = C1.real - C2.real;

        // subtracting Imaginary part of complex numbers

        temp.imaginary = C1.imaginary - C2.imaginary;

        // returning the difference

        return temp;

    }

    // Function for printing complex number

    void printComplexNumber()

    {

        System.out.println("Complex number: "

                           + real + " + "

                           + imaginary + "i");

    }

}

// Main Class

class GFG {

    // Main function

    public static void main(String[] args)

    {

        // First Complex number

        Complex C1 = new Complex(3, 2);

        // printing first complex number

        C1.printComplexNumber();

        // Second Complex number

        Complex C2 = new Complex(9, 5);

        // printing second complex number

        C2.printComplexNumber();

        // for Storing the sum

        Complex C3 = new Complex();

        // calling addComp() method

        C3 = C3.addComp(C1, C2);

        // printing the sum

        System.out.print("Sum of ");

        C3.printComplexNumber();

        // calling subtractComp() method

        C3 = C3.subtractComp(C1, C2);

        // printing the difference

        System.out.print("Difference of ");

        C3.printComplexNumber();

    }

}

Out Put:

Complex number: 3 + 2i

Complex number: 9 + 5i

Sum of Complex number: 12 + 7i

Difference of Complex number: -6 + -3i

Lab Program 5

|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  | A class called MyTime, which models a time instance, is designed as shown in the class diagram. |
|  | It contains the following private instance variables: |
|  | hour: between 0 to 23. |
|  | minute: between 0 to 59. |
|  | Second: between 0 to 59. |
|  | The constructor shall invoke the setTime() method (to be described later) to set the instance variable. |
|  | It contains the following public methods: |
|  | setTime(int hour, int minute, int second): It shall check if the given hour, minute and second are valid before setting the instance variables. |
|  | (Advanced: Otherwise, it shall throw an IllegalArgumentException with the message "Invalid hour, minute, or second!".) |
|  | Setters setHour(int hour), setMinute(int minute), setSecond(int second): It shall check if the parameters are valid, similar to the above. |
|  | Getters getHour(), getMinute(), getSecond(). |
|  | toString(): returns "HH:MM:SS". |
|  | nextSecond(): Update this instance to the next second and return this instance. Take note that the nextSecond() of 23:59:59 is 00:00:00. |
|  | nextMinute(), nextHour(), previousSecond(), previousMinute(), previousHour(): similar to the above. |
|  | Write the code for the MyTime class. Also write a test program (called TestMyTime) to test all the methods defined in the MyTime class. |
|  | \*/ |
|  | package oneseven; |
|  |  |
|  | public class TestMyTime { |
|  | public static void main(String[] args) { |
|  | MyTime a = new MyTime(10,59,34); |
|  | System.out.println(a.toString() + " one hour later is " + (a.nextHour()).toString()); |
|  | System.out.println(a.toString() + " one minute later is " + (a.nextMinute()).toString()); |
|  | MyTime b = new MyTime(0,0,0); |
|  | System.out.println("hour of " + b.toString() + " is " + b.getHour()); |
|  | System.out.println(b.toString() + " one second before is " + (b.previousSecond()).toString()); |
|  | } |
|  | } |

Lab Program -6

Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. The saving account provides compound interest ad withdrawal facilities but not cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintains a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and opening balance. From this derive the classes Current and Saving to make them more specific to their requirements. Include necessary member functions in order to achieve the following tasks:

(i) deposit an amount for a customer and update the balance

(ii) display the account details

(iii) compute and deposit interest

(iv) withdraw amount for a customer after checking the balance and update the balance.

(v) check for the minimum balance (for current account holders), impose penalty, if necessary, and update the balance.

Implement these without using any constructor

Code:

import java.util.\*;

import java.lang.\*;

class Account

{

public String acc\_name;

public double acc\_no;

public int acc\_type;

public double balance;

public void getdata(String name,double no,int type,double bal)

{

acc\_name=name;

acc\_no=no;

acc\_type=type;

balance=bal;

}

}

class Savings extends Account

{

public void deposit(double amt)

{

balance=balance+amt;

System.out.println(balance);

}

public void withdraw(double amt)

{

balance=balance-amt;

System.out.println(balance);

}

public void interest(int time,int no)

{

double intr=balance\*(1+6/no);

intr=Math.pow(intr,(time\*no));

System.out.println("Intertest calculated is"+intr);

balance=balance+intr;

System.out.println("The new balance is"+balance);

}

}

class Current extends Account

{

public void deposit(double amt)

{

balance=balance+amt;

System.out.println(balance);

}

public void withdraw(double amt)

{

balance=balance-amt;

System.out.println(balance);

check(balance);

}

public void check(double amt)

{

if(amt<10000)

{

balance =balance-500;

System.out.println("Insufficient Balance"+balance);

}

}

}

class Main

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int temp=1;

while(temp==1)

{

double amt=0;

System.out.println("Enter name");

sc.next();

String name=sc.nextLine();

System.out.println("Enter acc\_no");

double no=sc.nextDouble();

System.out.println("Enter acc\_type\n0 for Savings\n1 for Current");

int type=sc.nextInt();

do

{

System.out.println("Enter balance");

amt=sc.nextDouble();

}while(type==1 && amt<10000);

if(type==0)

{

Savings s=new Savings();

s.getdata(name,no,type,amt);

System.out.println("\n1.Deposit\n\2.Withdraw\n3.Interest");

int temp3=sc.nextInt();

if(temp3==1)

{

System.out.println("Enter Amoumt");

double amt1=sc.nextDouble();

s.deposit(amt1);

}

else if(temp3==2)

{

System.out.println("Enter Amoumt");

double amt1=sc.nextDouble();

s.withdraw(amt1);

}

else if(temp3==3)

{

System.out.println("Enter time period");

int tp=sc.nextInt();

System.out.println("Enter no of times");

int nof=sc.nextInt();

s.interest(tp,nof);

}

}

else if(type==1)

{

Current c=new Current();

c.getdata(name,no,type,amt);

System.out.println("\n1.Deposit\n\2.Withdraw");

int temp3=sc.nextInt();

if(temp3==1)

{

System.out.println("Enter Amoumt");

double amt1=sc.nextDouble();

c.deposit(amt1);

}

else if(temp3==2)

{

System.out.println("Enter Amoumt");

double amt1=sc.nextDouble();

c.withdraw(amt1);

}

}

System.out.println("To continue 1 else 0");

temp=sc.nextInt();

}

}

}

Lab Program: 7

Design a base class Circle with member variables (radius and color) of type double, methods (getRadius(), getArea()) and constructors (Circle(radius), Circle(radius, color)). Derive subclass called Cylinder from the superclass Circle with member variable (height) of type double, public methods (getHeight(), getVolume(), getArea()) and its constructors(Cylinder(height, radius), Cylinder(height, radius,color)). Create the two instances of cylinder and print similar cylinders if the area, volume and color of cylinders are same. Demonstrate the code reuse and polymorphism properties of Object oriented programming by inheriting the constructors and methods of the base class.

import java.util.Scanner;

class circle

{

double radius;

String color;

circle()

{

radius=1.0;

color="blue";

}

circle(double radius)

{

this.radius=radius;

color="blue";

}

circle(double radius,String color)

{

this.radius=radius;

this.color=color;

}

double getarea()

{

return Math.PI\*radius\*radius;

}

double getradius()

{

return radius;

}

String getcolor(){return color;}

}

class cylinder extends circle{

double height;

double getheight()

{ return height;

}

cylinder()

{

super();

height=2.0;

}

cylinder(double height)

{

super();

this.height=height;

}

cylinder(double height, double radius)

{

super(radius);

this.height=height;

}

cylinder(double height,double radius, String color)

{

super(radius,color);

this.height=height;

}

double getarea()

{

return ((2\* Math.PI\*radius\*height)+(2\* Math.PI\*radius\*radius));

}

double getvolume()

{

return (super.getarea()\*height);

}

void display()

{

System.out.println("\nRadius is "+super.radius+",Height is "+height+", Color is "+super.color+",Area is "+getarea()+",Volume is "+getvolume());

}

void check (cylinder c1,cylinder c2,int i,int j){

if((c1.radius==c2.radius)&& (c1.height==c2.height)&&(c1.color.equalsIgnoreCase(c2.color)))

System.out.println("The cylinders "+(i+1)+" and "+(j+1)+"are similar");

}

}

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

cylinder[] c = new cylinder[4];

int i;

c[0] = new cylinder();

c[1] = new cylinder(3.0);

c[2] = new cylinder(3.0, 4.0, "Green");

System.out.println("Enter the details of cylinder 4 (height , radius and color)");

double h = s.nextDouble();

//s.nextLine();

double r = s.nextDouble();

s.nextLine();

String st = s.nextLine();

c[3] = new cylinder(h, r, st);

for (i = 0; i < 4; i++) {

System.out.println("The dimensions of cylinder " + (i + 1) + " is ");

c[i].display();

}

for (i = 0; i < 4; i++) {

int j;

for (j = i + 1; j < 4; j++) {

c[i].check(c[i], c[j], i, j);

}

}

}

}

Program-8

Create two classes called HDFCAccount & StatebankAccount. Those implements all methods defined in interface account. Declare the methods getBalence, Deposit & withdraw in account interface. HDFCAccount uses member variables deposits and withdrawals for maintaining the balance, whereas StatebankAccount uses only balance to maintain the balance. In the main method create objects of HDFCAccount, StatebankAccount Bit assigned them to the reference of the interface Account, Also write an method to print balance of interface Account. Also write an method to print balance in main which prints balance amount.

Code:

class TestAccountInterface  
{  
    public static void main(String s[])  
    {  
        IAccount account = new HDFCAccount();  
      
        System.out.println("Transacting using HDFC Account");  
        transactOnAccount(account);  
        System.out.println();  
      
        account = new StateBankAccount();  
      
        System.out.println("Transacting using State Bank Account");  
        transactOnAccount(account);  
    }  
      
    public static void transactOnAccount(IAccount account)  
    {  
        System.out.println("------------------------------");  
        account.deposit(10000.0);  
        printBalance("depositing 10,000.0", account);  
        account.withdraw(2500.0);  
        printBalance("withdrawing 2,500.0", account);  
        account.withdraw(4100.0);  
        printBalance("withdrawing 4,100.0", account);  
        account.deposit(5000.0);  
        printBalance("depositing 5,000.0", account);  
        System.out.println("------------------------------");  
    }  
      
    public static void printBalance(String message, IAccount account)  
    {  
        System.out.println("The balance after " + message + " is " + account.getBalance() +".");  
    }  
  
}  
  
interface IAccount  
{  
    double getBalance();  
  
    void deposit(double amount);  
  
    void withdraw(double amount);  
}  
  
class HDFCAccount implements IAccount  
{  
    double deposits;  
    double withdrawals;  
  
  
    public double getBalance()  
    {  
        return deposits - withdrawals;  
    }  
  
    public void deposit(double amount)  
    {  
        deposits += amount;  
    }  
  
    public void withdraw(double amount)  
    {  
        withdrawals += amount;  
    }  
}  
  
class StateBankAccount implements IAccount  
{  
    double balance;  
  
    public double getBalance()  
    {  
        return balance;  
    }  
  
    public void deposit(double amount)  
    {  
        balance += amount;  
    }  
  
    public void withdraw(double amount)  
    {  
        balance -= amount;  
    }  
}

OUTPUT

Transacting using HDFC Account  
------------------------------  
The balance after depositing 10,000.0 is 10000.0.  
The balance after withdrawing 2,500.0 is 7500.0.  
The balance after withdrawing 4,100.0 is 3400.0.  
The balance after depositing 5,000.0 is 8400.0.  
------------------------------  
  
Transacting using State Bank Account  
------------------------------  
The balance after depositing 10,000.0 is 10000.0.  
The balance after withdrawing 2,500.0 is 7500.0.  
The balance after withdrawing 4,100.0 is 3400.0.  
The balance after depositing 5,000.0 is 8400.0.  
------------------------------

Lab Program: 9

Create a Package CIE which has two classes –Student & Internals. The class Student has members like usn, name, sem. The class Internals has an array that stores the internal marks scored in six courses of the current semester of the student. Create another package SEE which has class External which is a derived class of student. This class has an array that stores the SEE marks scored in six courses of the current semester of the student. Import the two packages in a file that declares the final marks of N students in all six courses.

package CIE;

public class Student {

public String usn, name;

public int sem;

}

public class Internals extends Student {

public int[] internalMarks = new int[5];

package CIE;

public class Student {public String usn, name;public int sem;}

public class Internals extends Student {

public int[]

internalMarks = new int[5];

}

package SEE;

public class External extends Student {

public int[] externalMarks = new int[5];

}

import CIE.\*;

import SEE.\*;

public class FinalMarks {package SEE;

public class External extends Student {

public int[] externalMarks = new int[5];}

import CIE.\*;

import SEE.\*;

public class FinalMarks {

public static void main(String[] args) {

int n = 3; // number of students

Internals[] cie = new Internals[n];

External[] see = new External[n];

// input the marks for each student

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

cie[i] = new Internals();

see[i] = new External();

public static void main(String[] args)

{int n = 3;

// number of students

Internals[] cie = new Internals[n];

External[] see = new External[n];

// input the marks for each student

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

{cie[i] = new Internals();

see[i] = new External();

for (int j = 0; j < 5; j++) {

cie[i].internalMarks[j] = // input value for internal mark

for course j

see[i].externalMarks[j] = // input value for external mark

for course j

}

}

// print the final marks for each student

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

System.out.println("Final marks for student " + (i + 1));

for (int j = 0; j < 5; j++)

{cie[i].internalMarks[j] =

// input value for internal mark

for course j

see[i].externalMarks[j] =

// input value for external markfor course j}}

// print the final marks for each student

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

System.out.println("Final marks for student " + (i + 1));

for (int j = 0; j < 5; j++) {

int finalMark = cie[i].internalMarks[j] +

see[i].externalMarks[j];

System.out.println("Course " + (j + 1) + ": " + finalMark);

}

}

}

for (int j = 0; j < 5; j++)

{int finalMark = cie[i].internalMarks[j] +see[i].externalMarks[j];

System.out.println("Course " + (j + 1) + ": " + finalMark);

}

}

}

Lab Program: 10

Write a Java Program that implements a multi thread application that has three threads. First thread generates random integer every 1 second and if value is even. Second thread computes the squre of the number and prints. If the value is odd the third thread will print the value of the cube of the number.

Code:

import java.util.Random;  
  
class Square extends Thread  
  
{  
  
 int x;  
  
 Square(int n)  
  
 {  
  
 x = n;  
  
 }  
  
 public void run()  
  
 {  
  
 int sqr = x \* x;  
  
 System.*out*.println("Square of " + x + " = " + sqr );  
  
 }  
  
}  
  
class Cube extends Thread  
  
{  
  
 int x;  
  
 Cube(int n)  
  
 {x = n;  
  
 }  
  
 public void run()  
  
 {  
  
 int cub = x \* x \* x;  
  
 System.*out*.println("Cube of " + x + " = " + cub );  
  
 }  
  
}  
  
class Number extends Thread  
  
{  
  
 public void run()  
  
 {  
  
 Random random = new Random();  
  
 for(int i =0; i<5; i++)  
  
 {  
  
 int randomInteger = random.nextInt(100);  
  
 System.*out*.println("Random Integer generated : " + randomInteger);  
  
 Square s = new Square(randomInteger);  
  
 s.start();  
  
 Cube c = new Cube(randomInteger);  
  
 c.start();  
  
 try {  
  
 Thread.*sleep*(1000);  
  
 } catch (InterruptedException ex) {  
  
 System.*out*.println(ex);  
  
 }  
  
 }  
  
 }  
  
}  
  
 class Thr {  
  
 public static void main(String args[])  
  
 {  
  
 Number n = new Number();  
  
 n.start();  
  
 }  
  
}

Lab Program: 11

Write a java program to handle the following exceptions based on choice made by the user by writing suitable try and catch block.

i) ArithmeticException

ii) ArrayIndexOutOfBoundsException

iii) NumberFormatException

iv) StringIndexOutOfBoundException

v) NullPointerException

**Example 1: Arithmetic exception**

This exception occurs when the result of a [division operation](https://beginnersbook.com/2022/09/arithmetic-operators-in-java-with-examples/) is undefined. When a number is divided by zero, the result is undefined and that is when this exception occurs.

class JavaExample

{

public static void main(String args[])

{

try{

int num1=30, num2=0;

int output=num1/num2;

System.out.println ("Result: "+output);

}

catch(ArithmeticException e){

System.out.println ("You Shouldn't divide a number by zero");

}

}

}

**Example 2: ArrayIndexOutOfBounds Exception**

This exception occurs when you try to access an array index that doesn’t exist. For example, If array is having only 5 elements and you are trying to display 7th element then it would throw this exception.

class JavaExample

{

public static void main(String args[])

{

try{

int a[]=new int[10];

// This will throw exception as Array has

// only 10 elements and we are trying to access

// 12th element.

a[11] = 9;

}

catch(ArrayIndexOutOfBoundsException e){

System.out.println ("ArrayIndexOutOfBounds Exception occurred");

System.out.println ("System Message: "+e);

}

}

}

**Example 3: NumberFormat Exception**

This exception occurs when a string is parsed to any numeric variable. For example, the statement int num=Integer.parseInt ("XYZ");

Would throw NumberFormatException because String “XYZ” cannot be parsed to int.

class JavaExample

{

public static void main(String args[])

{

try{

int num=Integer.parseInt ("XYZ") ;

System.out.println(num);

}catch(NumberFormatException e){

System.out.println("Number format exception occurred");

}

}

}

**Example 4: StringIndexOutOfBound Exception**

Class: Java.lang.StringIndexOutOfBoundsException

* A string is nothing but an [array of string](https://beginnersbook.com/2022/08/string-array-in-java/) type. This exception occurs when you try to access an index that doesn’t exist, similar to what we have seen in ArrayIndexOutOfBoundsException.
* Each character of a string object is stored in a particular index starting from 0. For example: In the string “beginnersbook”, the char ‘b’ is stored at index 0, char ‘e’ at index 1 and so on.
* To get a character present in a particular index of a string we can use a [method charAt(int)](https://beginnersbook.com/2013/12/java-string-charat-method-example/) of [java.lang.String](https://beginnersbook.com/2013/12/java-strings/" \t "_blank) where int argument is the index.

In the following example, the scope of the string “beginnersbook” is from index 0 to 12, however we are trying to access the char at index 40, which doesn’t exist, hence the exception occurred.

class JavaExample

{

public static void main(String args[])

{

try{

String str="beginnersbook";

System.out.println(str.length());

char c = str.charAt(40);

System.out.println(c);

}catch(StringIndexOutOfBoundsException e){

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

}

}

}

**Example 5: NullPointer Exception**

Class: Java.lang.NullPointer Exception  
This exception occurs when you are trying to perform some operation on an object that references to the null value.

class JavaExample

{

public static void main(String args[])

{

try{

String str=null;

System.out.println (str.length());

}

catch(NullPointerException e){

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

}

}

}

Lab Program: 12

Define a class Sort with generic method by name Arrange(T[]) and Display(T[]). Write a program to sort array elements of different data types.

class Sort

{

void Arrange(int t[])

{

int pos;

int temp;

for (int i = 0; i < t.length; i++)

{

pos = i;

for (int j = i+1; j < t.length; j++)

{

if (t[j] < t[pos]) //find the index of the minimum element

{

pos = j;

}

}

temp = t[pos]; //swap the current element with the minimum element

t[pos] = t[i];

t[i] = temp;

}

}

void display(int t[]) //display the array

{

for (int i=0; i<t.length; i++)

{

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

}

}

public static void main(String args[])

{

Sort ob = new Sort();

int arr[] = {64,25,12,22,11};

ob.Arrange(arr);

ob.display(arr);

}

}

Output:

11 12 22 25 64