**Program 1**   
**import java.io.\*;**   
**class GFG {**   
  **static int Series(int n) {**   
  **int i;**   
  **int sums = 0;**   
  **for (i = 1; i <= n; i++)**   
  **sums += 1 / (i \* i); // This will still use integer division**   **return sums;**   
  **}**   
  **public static void main(String[] args) {**   
  **int n = 3;**   
  **int res = Series(n);**   
  **System.out.println(res);**   
  **}**   
**}**

**Program 2**   
**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++) {**

**System.out.print(" ");**   
  **}**   
  **for (j = 0; j <= i; j++) {**   
  **System.out.print(" " + (g.factorial(i) / (g.factorial(j) \* g.factorial(i - j))));**   **}**   
  **System.out.println();**   
  **}**   
  **}**   
**}**

**Program 3**   
**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 nor decreasing order");**   **}**   
  **in.close();**   
  **}**

**}**

**Program 4**   
**import java.util.\*;**   
**class Complex {**   
  **int real, imaginary;**   
  **Complex() {**   
  **}**   
  **Complex(int tempReal, int tempImaginary) {**   
  **real = tempReal;**   
  **imaginary = tempImaginary;**   
  **}**   
  **Complex addComp(Complex C1, Complex C2) {**   **Complex temp = new Complex();**   
  **temp.real = C1.real + C2.real;**   
  **temp.imaginary = C1.imaginary + C2.imaginary;**   **return temp;**   
  **}**   
  **Complex subtractComp(Complex C1, Complex C2) {**   **Complex temp = new Complex();**   
  **temp.real = C1.real - C2.real;**   
  **temp.imaginary = C1.imaginary - C2.imaginary;**   **return temp;**   
  **}**

**void printComplexNumber() {**   
  **System.out.println("Complex number: " + real + " + " + imaginary + "i");**   **}**   
**}**   
**class GFG {**   
  **public static void main(String[] args) {**   
  **Complex C1 = new Complex(3, 2);**

**C1.printComplexNumber();**   
  **Complex C2 = new Complex(9, 5);**   
  **C2.printComplexNumber();**   
  **Complex C3 = new Complex();**   
  **C3 = C3.addComp(C1, C2);**   
  **System.out.print("Sum of ");**   
  **C3.printComplexNumber();**   
  **C3 = C3.subtractComp(C1, C2);**   
  **System.out.print("Difference of ");**   
  **C3.printComplexNumber();**   
  **}**   
**}**

**Program 5**   
**public class MyTime {**   
  **private int hour; // between 0 and 23**   
  **private int minute;// between 0 and 59**   
  **public MyTime(int hour, int minute) {**   
  **setTime(hour, minute);**   
  **}**   
  **public void setTime(int hour, int minute) {**   
  **setHour(hour);**   
  **setMinute(minute);**   
  **}**   
  **public void setHour(int hour) {**   
  **if (hour >= 0 && hour < 24) {**   
  **this.hour = hour;**   
  **} else {**   
  **throw new IllegalArgumentException("Invalid hour!");**   **}**   
  **}**

**public void setMinute(int minute) {**   
  **if (minute >= 0 && minute < 60) {**   
  **this.minute = minute;**   
  **} else {**   
  **throw new IllegalArgumentException("Invalid minute!");**   **}**   
 **}**   
 **public int getHour() {**   
  **return hour;**   
 **}**   
 **public int getMinute() {**   
  **return minute;**   
 **}**   
 **@Override**   
 **public String toString() {**   
  **return String.format("%02d:%02d", hour, minute);**   
 **}**   
 **public MyTime nextMinute() {**   
  **if (minute == 59) {**   
  **minute = 0;**   
  **nextHour();**   
  **} else {**   
  **minute++;**   
  **}**   
  **return this;**   
 **}**   
 **public MyTime nextHour() {**   
  **if (hour == 23) {**   
  **hour = 0;**   
  **} else {**   
  **hour++;**   
  **}**

**return this;**   
  **}**   
  **public static void main(String[] args) {**   
  **MyTime time = new MyTime(23, 59);**   
  **System.out.println("Current time: " + time);**   
  **System.out.println("Next minute: " + time.nextMinute());**   **System.out.println("Next hour: " + time.nextHour());**   **}**   
**}**

**Program 6**   
**import java.util.Scanner;**   
**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 += amt;**   
  **System.out.println("Balance after deposit: " + balance);**   
  **}**   
  **public void withdraw(double amt) {**   
  **if (amt > balance) {**

**System.out.println("Insufficient balance.");**   
  **} else {**   
  **balance -= amt;**   
  **System.out.println("Balance after withdrawal: " + balance);**   **}**   
  **}**   
  **public void interest(int time, int no) {**   
  **double rate = 0.06; // Assuming 6% interest rate**   
  **double intr = balance \* Math.pow(1 + rate / no, time \* no) - balance;**   **System.out.println("Interest calculated: " + intr);**   
  **balance += intr;**   
  **System.out.println("The new balance is: " + balance);**   
  **}**   
**}**   
**class Current extends Account {**   
  **public void deposit(double amt) {**   
  **balance += amt;**   
  **System.out.println("Balance after deposit: " + balance);**   
  **}**

**public void withdraw(double amt) {**   
  **if (amt > balance) {**   
  **System.out.println("Insufficient balance.");**   
  **} else {**   
  **balance -= amt;**   
  **System.out.println("Balance after withdrawal: " + balance);**   **check(balance);**   
  **}**   
 **}**   
 **public void check(double amt) {**   
  **if (amt < 10000) {**   
  **balance -= 500;**

**System.out.println("Penalty applied. Insufficient balance: " + balance);**   **}**   
  **}**   
**}**   
**class Main {**   
  **public static void main(String[] args) {**   
  **Scanner sc = new Scanner(System.in);**   
  **int temp = 1;**   
  **while (temp == 1) {**   
  **System.out.println("Enter name:");**   
  **String name = sc.next();**   
  **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();**

**System.out.println("Enter initial balance:");**   
 **double amt = sc.nextDouble();**   
 **if (type == 0) {**   
 **Savings s = new Savings();**   
 **s.getData(name, no, type, amt);**   
 **System.out.println("\n1. Deposit\n2. Withdraw\n3. Interest"); int temp3 = sc.nextInt();**   
 **switch (temp3) {**   
  **case 1:**   
  **System.out.println("Enter Amount:");**   
  **double amt1 = sc.nextDouble();**   
  **s.deposit(amt1);**   
  **break;**   
  **case 2:**   
  **System.out.println("Enter Amount:");**   
  **amt1 = sc.nextDouble();**

**s.withdraw(amt1);**   
  **break;**   
  **case 3:**   
  **System.out.println("Enter time period:");**   
  **int tp = sc.nextInt();**   
  **System.out.println("Enter number of times interest is compounded per year:");**   
  **int nof = sc.nextInt();**   
  **s.interest(tp, nof);**   
  **break;**   
  **default:**   
  **System.out.println("Invalid option.");**   
  **}**

**} else if (type == 1) {**   
 **Current c = new Current();**   
 **c.getData(name, no, type, amt);**   
 **System.out.println("\n1. Deposit\n2. Withdraw"); int temp3 = sc.nextInt();**   
 **switch (temp3) {**   
  **case 1:**   
  **System.out.println("Enter Amount:");**   
  **double amt1 = sc.nextDouble();**   
  **c.deposit(amt1);**   
  **break;**   
  **case 2:**   
  **System.out.println("Enter Amount:");**   
  **amt1 = sc.nextDouble();**   
  **c.withdraw(amt1);**   
  **break;**   
  **default:**   
  **System.out.println("Invalid option.");**

**}**   
  **} else {**   
  **System.out.println("Invalid account type.");**   
  **}**   
  **System.out.println("To continue, enter 1; to exit, enter 0:");**   **temp = sc.nextInt();**   
  **}**   
  **sc.close();**   
  **}**   
**}**

**Program 7**   
**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;**   
  **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 getHeight() {**   
  **return height;**   
 **}**   
 **@Override**   
 **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 " + radius + ", Height is " + height + ", Color is " + 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];**   
  **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();**   
  **double r = s.nextDouble();**   
  **s.nextLine(); // Consume newline**   
  **String st = s.nextLine();**   
  **c[3] = new Cylinder(h, r, st);**   
  **for (int i = 0; i < 4; i++) {**   
  **System.out.println("The dimensions of cylinder " + (i + 1) + ":");**   
  **c[i].display();**   
  **}**   
  **for (int i = 0; i < 4; i++) {**

**for (int j = i + 1; j < 4; j++) {**   
  **c[i].check(c[i], c[j], i, j);**   
  **}**   
  **}**   
  **s.close();**   
  **}**   
**}**

**Program 8**   
**interface Account {**   
  **double getBalance();**   
  **void deposit(double amount);**   
  **void withdraw(double amount);**   
**}**   
**class HDFCAccount implements Account {**   
  **private double deposits = 0.0;**   
  **private double withdrawals = 0.0;**   
  **@Override**   
  **public double getBalance() {**   
  **return deposits - withdrawals;**   
  **}**   
  **@Override**   
  **public void deposit(double amount) {**   
  **deposits += amount;**   
  **}**   
  **@Override**   
  **public void withdraw(double amount) {**   
  **if (amount <= getBalance()) {**   
  **withdrawals += amount;**   
  **} else {**   
  **System.out.println("Insufficient balance");**

**}**   
  **}**   
**}**   
**class StateBankAccount implements Account {**   **private double balance = 0.0;**   
  **@Override**   
  **public double getBalance() {**   
  **return balance;**   
  **}**   
  **@Override**   
  **public void deposit(double amount) {**   
  **balance += amount;**   
  **}**   
  **@Override**   
  **public void withdraw(double amount) {**   
  **if (amount <= balance) {**   
  **balance -= amount;**   
  **} else {**   
  **System.out.println("Insufficient balance");**   **}**   
  **}**   
**}**

**public class Main {**   
  **public static void main(String[] args) {**   **Account hdfc = new HDFCAccount();**   **Account sbi = new StateBankAccount();**   **hdfc.deposit(1000);**   
  **hdfc.withdraw(200);**   
  **sbi.deposit(2000);**   
  **sbi.withdraw(500);**   
  **printBalance(hdfc);**

**printBalance(sbi);**   
  **}**   
  **public static void printBalance(Account account) {**   
  **System.out.println("Balance: " + account.getBalance());**   **}**   
**}**

**Program 9**   
**// File: CIE/Internals.java**   
**package CIE;**   
**public class Internals extends Student {**   
  **public int[] internalMarks = new int[6];**   
  **public Internals(String usn, String name, int sem, int[] internalMarks) {**   **this.usn = usn;**   
  **this.name = name;**   
  **this.sem = sem;**   
  **this.internalMarks = internalMarks;**   
  **}**   
**}**   
**// File: CIE/Student.java**   
**package CIE;**   
**public class Student {**   
  **public String usn;**   
  **public String name;**   
  **public int sem;**   
**}**   
**// File: SEE/External.java**   
**package SEE;**   
**import CIE.Student;**   
**public class External extends Student {**   
  **public int[] seeMarks = new int[6];**

**public External(String usn, String name, int sem, int[] seeMarks) {**   
  **this.usn = usn;**   
  **this.name = name;**   
  **this.sem = sem;**   
  **this.seeMarks = seeMarks;**   
  **}**   
**}**   
**// File: Main.java**   
**import CIE.Internals;**   
**import SEE.External;**   
**public class Main {**   
  **public static void main(String[] args) {**   
  **int N = 5; // Example number of students**   
  **Internals[] internalStudents = new Internals[N];**   
  **External[] externalStudents = new External[N];**   
  **for (int i = 0; i < N; i++) {**   
  **internalStudents[i] = new Internals("USN" + (i + 1), "Student" + (i + 1), 3, new int[]{80, 85, 75, 90, 88, 92});**   
  **externalStudents[i] = new External("USN" + (i + 1), "Student" + (i + 1), 3, new int[]{70, 75, 65, 80, 78, 82});**   
  **}**   
  **for (int i = 0; i < N; i++) {**   
  **System.out.println("Student: " + internalStudents[i].name);**   
  **System.out.println("USN: " + internalStudents[i].usn);**   
  **System.out.println("Semester: " + internalStudents[i].sem);**   
  **int totalMarks = 0;**   
  **for (int j = 0; j < 6; j++) {**   
  **int finalMarks = internalStudents[i].internalMarks[j] +**   
**externalStudents[i].seeMarks[j];**   
  **totalMarks += finalMarks;**   
  **System.out.println("Course " + (j + 1) + " Final Marks: " + finalMarks);**   **}**   
  **System.out.println("Total Marks: " + totalMarks + "\n");**

**}**   
  **}**   
**}**

**Program 10**   
**import java.util.Random;**   
**class GenerateNumber implements Runnable {**   
  **public void run() {**   
  **Random random = new Random();**   
  **while (true) {**   
  **int number = random.nextInt(100);**   
  **System.out.println("Generated Number: " + number);**   **if (number % 2 == 0) {**   
  **new Thread(new SquareNumber(number)).start();**   **} else {**   
  **new Thread(new CubeNumber(number)).start();**   **}**   
  **try {**   
  **Thread.sleep(1000); // Sleep for 1 second**   
  **} catch (InterruptedException e) {**   
  **System.out.println(e);**   
  **}**   
  **}**   
  **}**   
**}**   
**class SquareNumber implements Runnable {**   
  **private int number;**   
  **SquareNumber(int number) {**   
  **this.number = number;**   
  **}**

**public void run() {**   
  **System.out.println("Square of " + number + " is " + (number \* number));**   **}**   
**}**   
**class CubeNumber implements Runnable {**   
  **private int number;**   
  **CubeNumber(int number) {**   
  **this.number = number;**   
  **}**   
  **public void run() {**   
  **System.out.println("Cube of " + number + " is " + (number \* number \* number));**   **}**   
**}**   
**public class MultiThreadedApp {**   
  **public static void main(String[] args) {**   
  **Thread generateThread = new Thread(new GenerateNumber());**   
  **generateThread.start();**   
  **}}**

**Program 11**   
**import java.util.Scanner;**   
**public class ExceptionHandling {**   
  **public static void main(String[] args) {**   
  **Scanner scanner = new Scanner(System.in);**   
  **System.out.println("Choose an exception to handle:");**   **System.out.println("1. ArithmeticException");**   
  **System.out.println("2. ArrayIndexOutOfBoundsException");**   **System.out.println("3. NumberFormatException");**   
  **System.out.println("4. StringIndexOutOfBoundsException");**   **System.out.println("5. NullPointerException");**   
  **int choice = scanner.nextInt();**

**switch(choice) {**   
  **case 1:**   
  **try {**   
  **System.out.println("Enter numerator and denominator:");**   
  **int numerator = scanner.nextInt();**   
  **int denominator = scanner.nextInt();**   
  **int result = numerator / denominator;**   
  **System.out.println("Result: " + result);**   
  **} catch (ArithmeticException e) {**   
  **System.out.println("Error: Division by zero is not allowed.");**   **}**   
  **break;**   
  **case 2:**   
  **try {**   
  **int[] array = {1, 2, 3};**   
  **System.out.println("Enter index:");**   
  **int index = scanner.nextInt();**   
  **System.out.println("Element at index " + index + ": " + array[index]);**   **} catch (ArrayIndexOutOfBoundsException e) {**   
  **System.out.println("Error: Array index is out of bounds.");**   
  **}**   
  **break;**   
  **case 3:**   
  **try {**   
  **System.out.println("Enter a number:");**   
  **String input = scanner.next();**   
  **int number = Integer.parseInt(input);**   
  **System.out.println("Number: " + number);**   
  **} catch (NumberFormatException e) {**   
  **System.out.println("Error: Invalid number format.");**   
  **}**   
  **break;**

**case 4:**   
  **try {**   
  **System.out.println("Enter a string:");**   
  **String str = scanner.next();**   
  **System.out.println("Enter index:");**   
  **int index = scanner.nextInt();**   
  **char ch = str.charAt(index);**   
  **System.out.println("Character at index " + index + ": " + ch);**   **} catch (StringIndexOutOfBoundsException e) {**   
  **System.out.println("Error: String index is out of bounds.");**   **}**   
  **break;**   
  **case 5:**   
  **try {**   
  **String str = null;**   
  **System.out.println("Length of the string: " + str.length());**   **} catch (NullPointerException e) {**   
  **System.out.println("Error: Null pointer exception.");**   
  **}**   
  **break;**   
  **default:**   
  **System.out.println("Invalid choice.");**   
  **}**   
  **scanner.close();**   
  **}**   
**}**

**12class Sort {**   
  **public <T extends Comparable<T>> void Arrange(T[] array) {**   **Arrays.sort(array);**   
  **}**   
  **public <T> void Display(T[] array) {**   
  **for (T element : array) {**   
  **System.out.print(element + " ");**   
  **}**   
  **System.out.println();**   
  **}**   
**}**   
**public class GenericSortExample {**   
  **public static void main(String[] args) {**   
  **Sort sorter = new Sort()**   
  **Integer[] intArray = {5, 3, 8, 1, 9};**   
  **System.out.println("Original Integer Array: ");**   
  **sorter.Display(intArray);**   
  **sorter.Arrange(intArray);**   
  **System.out.println("Sorted Integer Array: ");**   
  **sorter.Display(intArray);**   
  **String[] strArray = {"Banana", "Apple", "Cherry", "Date"};**   **System.out.println("\nOriginal String Array: ");**   
  **sorter.Display(strArray);**   
  **sorter.Arrange(strArray);**   
  **System.out.println("Sorted String Array: ");**   
  **sorter.Display(strArray);**   
  **Double[] doubleArray = {2.5, 3.7, 1.2, 4.8, 0.9};**   
  **System.out.println("\nOriginal Double Array: ");**   
  **sorter.Display(doubleArray);**   
  **sorter.Arrange(doubleArray);**   
  **System.out.println("Sorted Double Array: ");**   
  **sorter.Display(doubleArray);**

**}**   
**}**