

Chetan Kumar G CH.SC.U4CSE24109

OBJECT ORIENTED PROGRAMMING (23CSE111)

LAB RECORD



**AMRITA VISHWA VIDYAPEETHAM AMRITA SCHOOL OF COMPUTING, CHENNAI**

**BONAFIDE CERTIFICATE**

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by ***CH.SC.U4CSE24109 – Chetan Kumar G*** in **“Computer**

**Science and Engineering”** is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

This Lab examination held on

Internal Examiner 1 Internal Examiner 2

INDEX

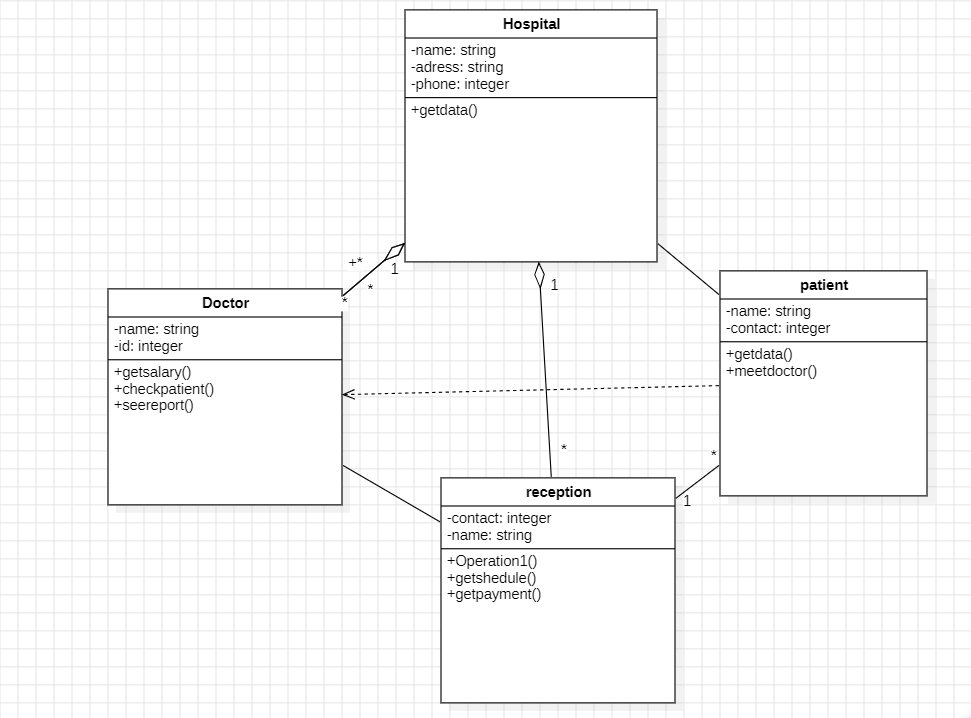
|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE.NO** |
| UML DIAGRAM | | |
| 1. | **HOSPITAL MANAGEMENT SYSTEM** | |
|  | a) Class Diagram | 6 |
|  | b) Use Case Diagram | 7 |
|  | c) Sequence Diagram | 8 |
|  | d) Object Diagram | 9 |
|  | e) State-Activity Diagram | 10 |
| 2. | **ATM MANAGEMENT SYSTEM** | |
|  | a) Class Diagram | 11 |
|  | b) Use Case Diagram | 12 |
|  | c) Sequence Diagram | 13 |
|  | d) Object Diagram | 14 |
|  | e) State-Activity Diagram | 15 |
| 3. | **BASIC JAVA PROGRAMS** | |
|  | 1. Check Even or Odd | 16 |
|  | 1. Largest of Three Numbers | 17 |
|  | 1. Reverse a Number | 19 |
|  | 1. Sum of Digits | 20 |
|  | 1. Print Fibonacci Series | 21 |
|  | 1. Prime Number Check | 22 |
|  | 1. Print a Pyramid Pattern | 23 |
|  | 1. Factorial of a Number | 24 |
|  | 1. Armstrong Number | 25 |
|  | 1. Number Guessing Game | 26 |

|  |  |  |
| --- | --- | --- |
|  | **INHERITANCE** |  |
| 4. | **SINGLE INHERITANCE PROGRAMS** |  |
|  | 4.a) Vehicle Management System | 27 |
|  | 4.b) Student Management System | 29 |
| 5. | **MULTILEVEL INHERITANCE PROGRAMS** |  |
|  | 5.a) Animals Information | 31 |
|  | 5.b) Shape Details | 33 |
| 6. | **HIERARCHICAL INHERITANCE PROGRAMS** |  |
|  | 6.a) Employee Management System | 35 |
|  | 6.b) School Management System | 37 |
| 7. | **HYBRID INHERITANCE PROGRAMS** |  |
|  | 7.a) Account Management System | 40 |
|  | 7.b) Car Management System | 43 |
|  | **POLYMORPHISM** |  |
| 8. | **CONSTRUCTOR PROGRAMS** |  |
|  | 8.a) Email Management System | 46 |
| 9. | **CONSTRUCTOR OVERLOADING PROGRAMS** |  |
|  | 9.a) Animals Information | 47 |
| 10. | **METHOD OVERLOADING PROGRAMS** |  |
|  | 10.a) Order Management System | 49 |
|  | 10.b) Phone Charge System | 51 |
| 11. | **METHOD OVERRIDING PROGRAMS** |  |
|  | 11.a) Chatbot System | 53 |
|  | 11.b) Vehicle Fare Management System | 54 |
|  | **ABSTRACTION** |  |
| 12. | **INTERFACE PROGRAMS** |  |
|  | 12.a) Sports Management System | 56 |
|  | 12.b) Music Management System | 57 |
|  | 12.c) Smart Home Management System | 58 |
|  | 12.d) Online Payment Management System | 60 |
| 13. | **ABSTRACT CLASS PROGRAMS** |  |
|  | 13.a) Fare Calculator System | 63 |
|  | 13.b) Employee Management System | 65 |
|  | 13.c) Bank Account Management System | 67 |
|  | 13.d) Food Management System | 69 |
|  | **ENCAPSULATION** |  |
| 14. | **ENCAPSULATION PROGRAMS** |  |
|  | 14.a) Student Management System | 72 |
|  | 14.b) Car Driving System | 73 |
|  | 14.c) Bank Management System | 75 |
|  | 14.d) Library Management System | 78 |
| 15. | **PACKAGES PROGRAMS** |  |
|  | 15.a) User Defined (Library Management System) | 80 |
|  | 15.b) User Defined (Shopping Management System) | 83 |
|  | 15.c) Built – in Package (IP Address System) | 85 |
|  | 15.d) Built – in Package (Draw A Rectangle) | 86 |
| 16. | **EXCEPTION HANDLING PROGRAMS** |  |
|  | 16.a) ATM Withdraw System | 87 |
|  | 16.b) Division By 0 | 89 |
|  | 16.c) Array Index Manager | 90 |
|  | 16.d) File Reading | 91 |
| 17. | **FILE HANDLING PROGRAMS** |  |
|  | 17.a) Writing/Reading A File | 92 |
|  | 17.b) Line Count | 93 |
|  | 17.c) Append Data To File | 94 |
|  | 17.d) Copy The File | 95 |

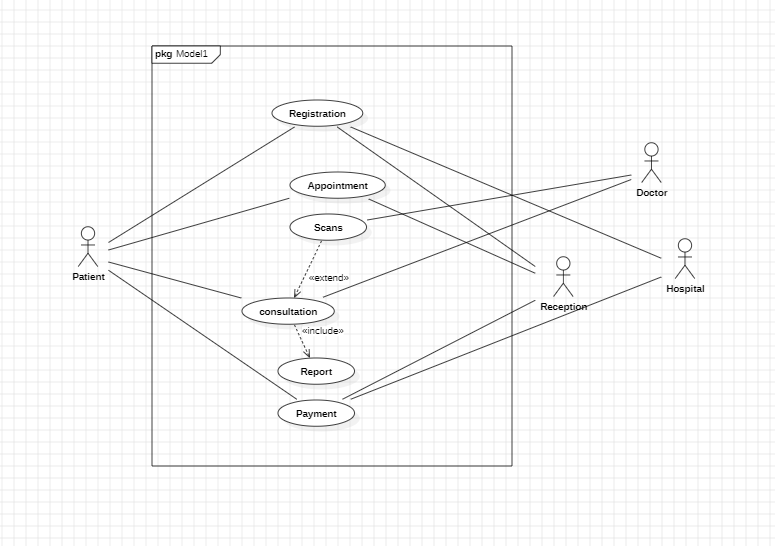
UML DIAGRAMS

# HOSPITAL MANAGEMENT SYSTEM

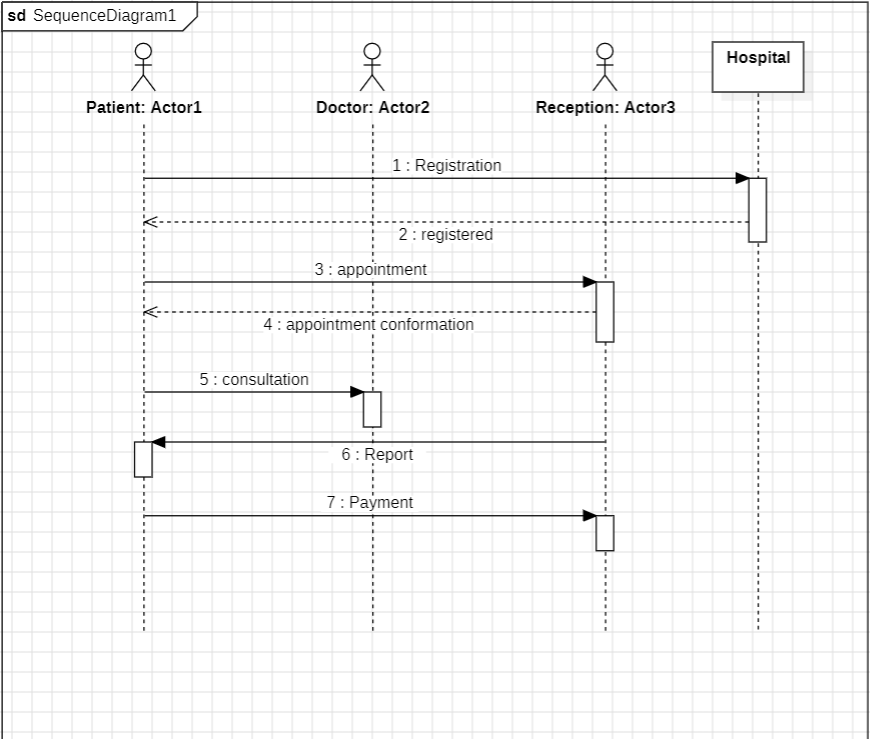
## Class Diagram



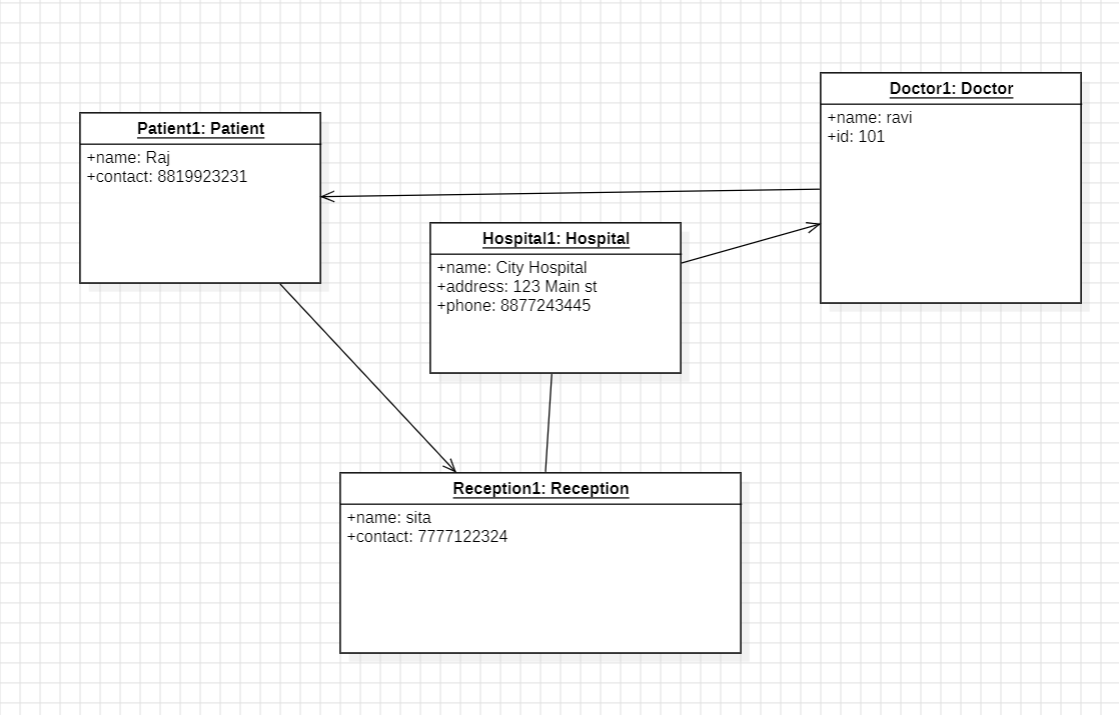
## Use Case Diagram:

****

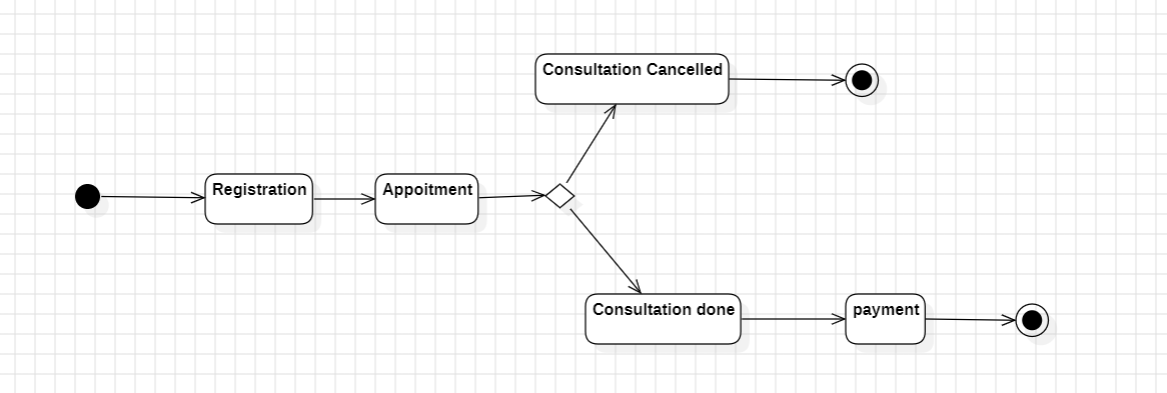
## Sequence Diagram:

****

## Object Diagram:

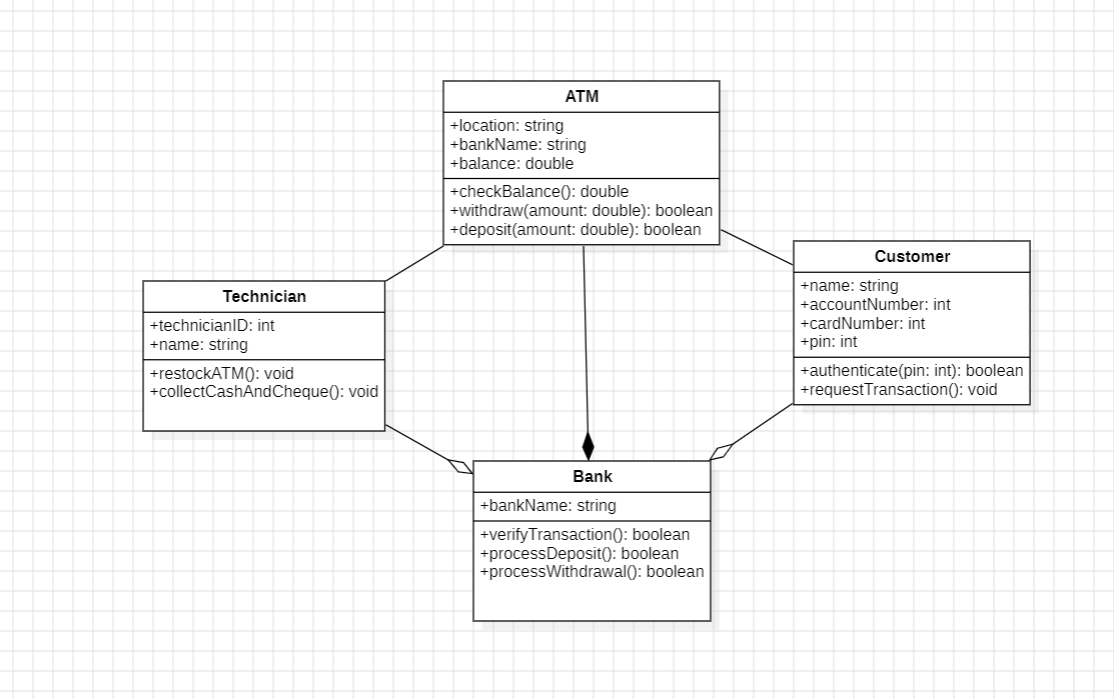
****

## State-Activity Diagram:

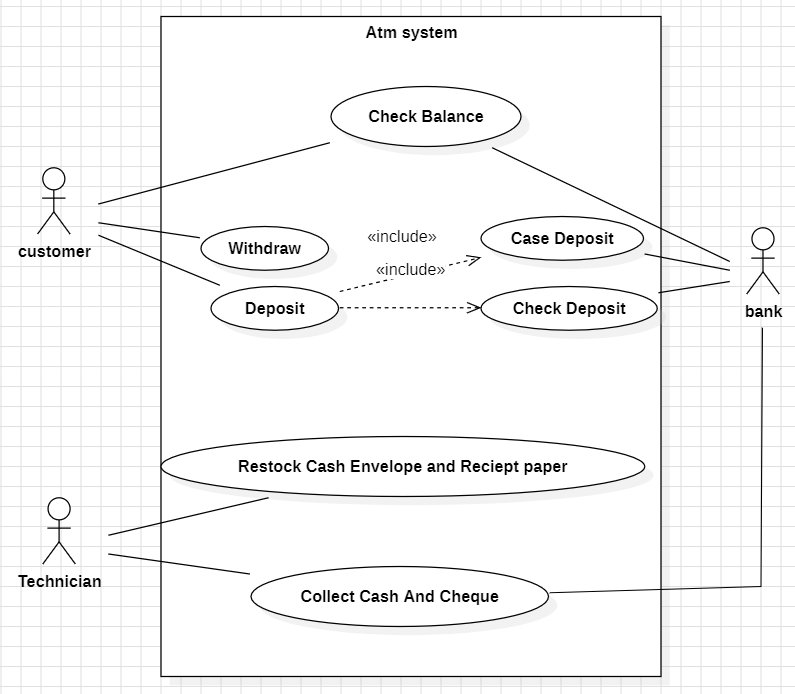
****

# 2.ATM Management System

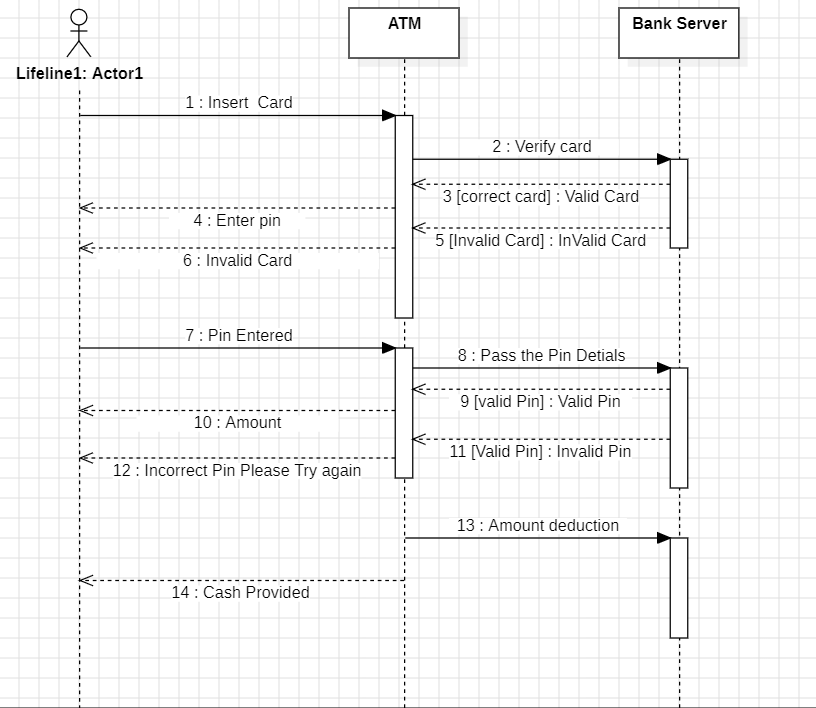
## Class Diagram:

****

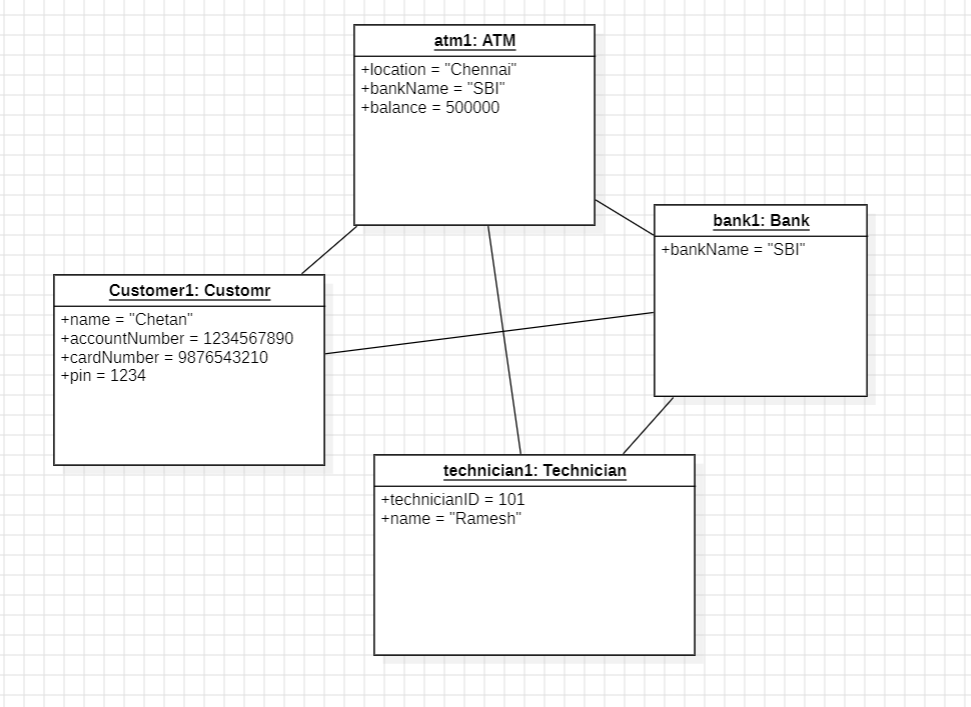
## Use Case Diagram:

****

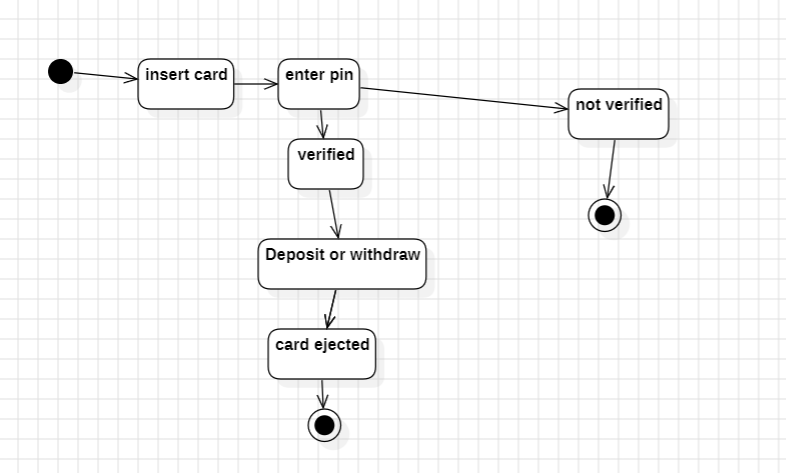
## Sequence Diagram:

****

## Object Diagram:

****

## State-Activity Diagram:

****

# 3.Basic Java Programs

## Check Even or Odd

Write a Java program that takes an integer input from the user and determines whether it is even or odd using an if-else statement.

**Code:**

import java.util.\*;

public class oddeven{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

if(num%2==0){

System.out.println("the number is even");

}

else{

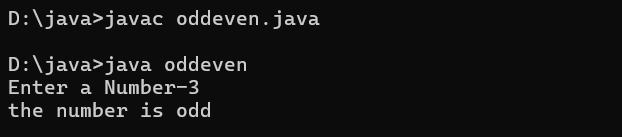
System.out.println("the number is odd");

}

}

}

**Output:**



## Largest of Three Numbers

Write a program that takes three integers as input and determines the largest among them using nested if-else statements.

**Code:**

import java.util.\*;

public class largest{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter first Number-");

int num1 = obj.nextInt();

System.out.print("Enter second Number-");

int num2 = obj.nextInt();

System.out.print("Enter third Number-");

int num3 = obj.nextInt();

if(num1>num2){

if(num1>num3){

System.out.println("the First number is Greatest" + num1);

}

else if(num3>num1){

System.out.println("the Third number is Greatest" + num3);

}

else{

System.out.println("Two numbers are same");

}

}

else if(num2>num1){

if(num2>num3){

System.out.println("the Second number is Greatest" + num2);

}

else if(num3>num2){

System.out.println("the Third number is Greatest" + num3);

}

else{

System.out.println("Two numbers are same");

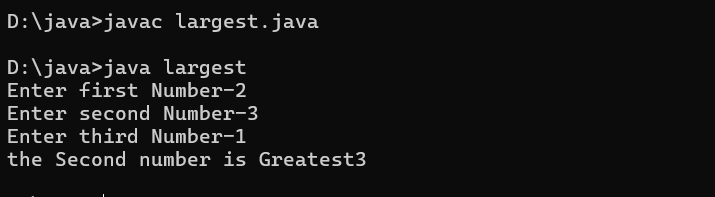
}

}

}

}

**Output:**



## Reverse a Number

Write a Java program that takes an integer input and reverses its digits using a while loop.  
**Example:**  
**Input:** 1234  
**Output:** 4321

**Code:**

import java.util.\*;

public class reverse{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

int reverse = 0;

while (num!=0){

int digit=num%10;

reverse=reverse\*10+digit;

num=num/10;

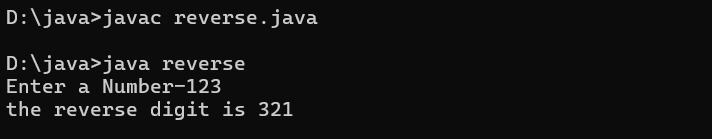
}

System.out.println("the reverse digit is " + reverse);

}

}

**Output:**



## Sum of Digits

Write a program to find the sum of the digits of a given number using a while loop.  
**Example:**  
**Input:** 145  
**Output:** 10 (1+4+5)

**Code:**

import java.util.\*;

public class sumOfDigits{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

int sum= 0;

while (num!=0){

int digit=num%10;

sum=sum+digit;

num=num/10;

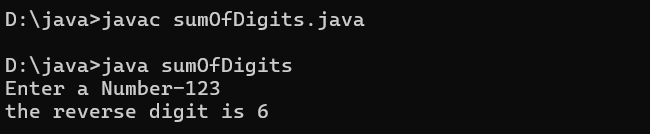
}

System.out.println("the reverse digit is " + sum);

}

}

**Output;**



## Print Fibonacci Series

Write a Java program to print the first n numbers of the Fibonacci series using a for loop.  
**Example:**  
**Input:** 5  
**Output:** 0 1 1 2 3

**Code:**

import java.util.\*;

public class fibonacciSeries{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

int a= 0;

int b= 1;

System.out.print(a + " ");

System.out.print(b + " ");

num=num-2;

while (num!=0){

int sum= a+b;

System.out.print(sum + " ");

a=b;

b=sum;

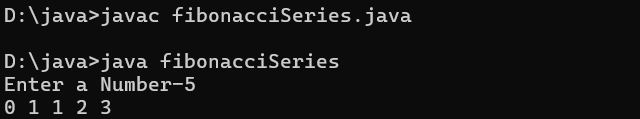
num=num-1;

}

}

}

**Output:**



## Prime Number Check

Write a Java program that takes an integer input and checks whether it is a prime number or not using a for loop.

**Code:**

import java.util.\*;

public class primeNumber{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

for (int i=2; i<num; i++){

if(num%i==0){

System.out.println("the number is not prime");

break;

}

else{

System.out.println("the number is prime");

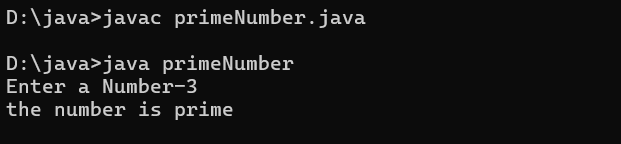
}

}

}

}

**Output:**



## Print a Pyramid Pattern

Write a program that prints a pyramid pattern of \* based on user input.  
**Example:**  
**Input:** 5  
**Output:**

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**Code:**

import java.util.\*;

public class pattern{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

for (int i=1; i<=num; i++){

for(int k=0; k<=num-i; k++){

System.out.print(" ");

}

for(int j=1; j<=(2\*i-1); j++){

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

}

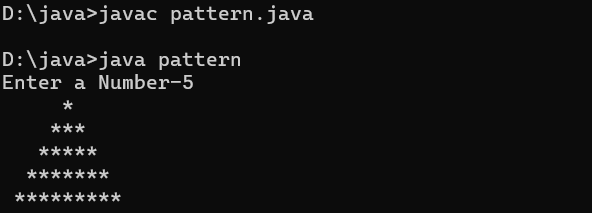
System.out.println();

}

}

}

**Output:**



## Factorial of a Number

Write a Java program that calculates the factorial of a given number using a for loop.  
**Example:**  
**Input:** 5  
**Output:** 120 (5! = 5 × 4 × 3 × 2 × 1)

**Code:**

import java.util.\*;

public class factorial{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

int factorial=1;

for (int i=1; i<=num; i++){

factorial=factorial\*i;

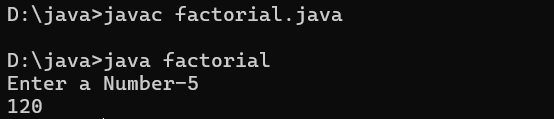
}

System.out.print(factorial);

}

}

**Output:**



## Armstrong Number

Write a Java program to check if a number is an Armstrong number (sum of cubes of digits equals the number).  
**Example:**  
**Input:** 153  
**Output:** Yes, it's an Armstrong number.  
(Because 1³ + 5³ + 3³ = 153)

**Code:**

import java.util.\*;

public class armstrongNumber{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter a Number-");

int num = obj.nextInt();

int num2=num;

int sum= 0;

String str = String.valueOf(num);

int len = str.length();

while (num!=0){

int digit=num%10;

sum=sum+(int) Math.pow(digit, len);

num=num/10;

}

if(sum==num2){

System.out.println("this is an Armstrong number");

}

else{

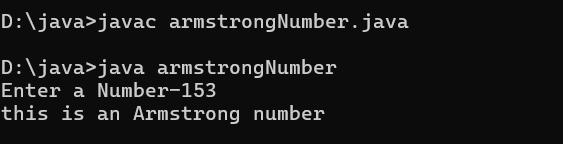
System.out.println("this is not an Armstrong number");

}

}

}

**Output:**



## Number Guessing Game

Write a simple number guessing game where the program randomly selects a number (1-100), and the user keeps guessing until they get it right.

**Code:**

import java.util.\*;

import java.util.Random;

public class guessNumber{

public static void main(String[] args){

Random random= new Random();

int rannum = random.nextInt(100)+1;

Scanner obj = new Scanner(System.in);

Boolean a=true;

while(a){

System.out.print("Enter a Number-");

int num = obj.nextInt();

if (num==rannum){

System.out.print("Congratulation!!! Correct");

a=false;

}

else{

System.out.println("Oops..., try again");

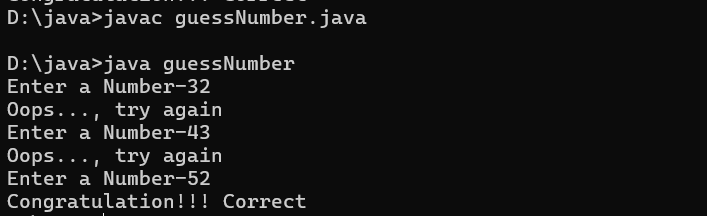
}

}

}

}

**Output:**



# INHERITANCE

# 4.SINGLE INHERITANCE

## Vehicle Management System

**Code:**

import java.util.\*;

class Vehicle{

    String model;

    int year;

    int speed;

    Vehicle(String model,int year,int speed){

        this.model=model;

        this.year=year;

        this.speed=speed;

    }

    void displayInfo(){

        System.out.println("model="+" "+model);

        System.out.println("year="+" "+year);

        System.out.println("speed="+" "+speed);

    }

}

class car extends Vehicle{

    String fuelType;

    car(String fuelType,String model,int year,int speed){

        super(model,year,speed);

        this.fuelType=fuelType;

    }

    void displayInfoFuel(){

        System.out.println("fuelType="+" "+fuelType);

    }

}

public class singleInheritance\_1{

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        System.out.print("enter model");

        String model=sc.nextLine();

        System.out.print("enter year");

        int year=sc.nextInt();

        System.out.print("enter speed");

        int speed=sc.nextInt();

        sc.nextLine();

        String fuelType=sc.nextLine();

        car obj = new car(fuelType,model,year,speed);

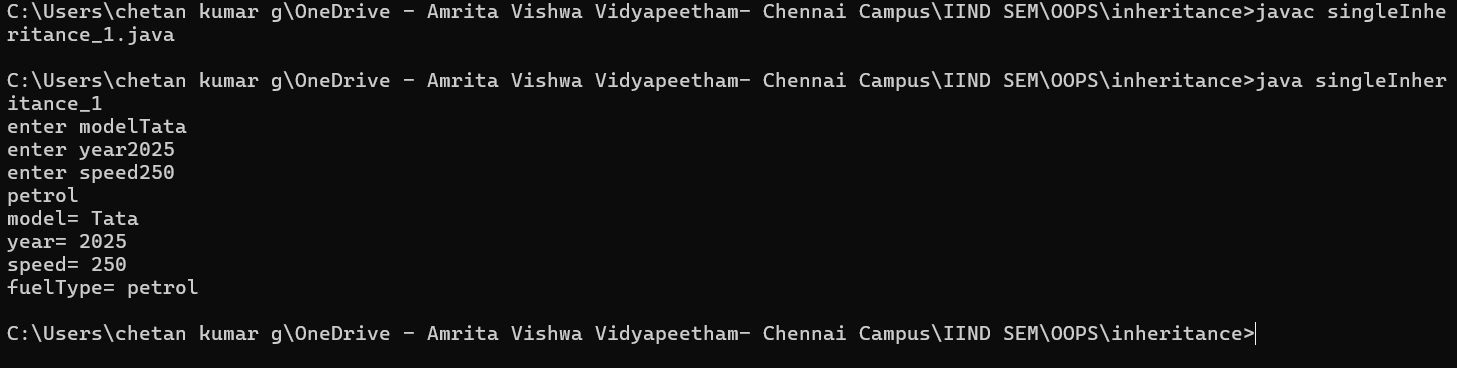
        obj.displayInfo();

        obj.displayInfoFuel();

    }

}

**OUTPUT:**



## Student Management System

**Code:**

import java.util.\*;

class Person {

    String name;

    int age;

    Person(String name, int age) {

        this.name = name;

        this.age = age;

    }

    void displayInfo() {

        System.out.println("Name: " + name);

        System.out.println("Age: " + age);

    }

}

class Student extends Person {

    String grade;

    Student(String name, int age, String grade) {

        super(name, age);

        this.grade = grade;

    }

    void displayStudentInfo() {

        displayInfo();

        System.out.println("Grade: " + grade);

    }

}

public class singleInheritance\_2 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        System.out.print("enter name - ");

        String name=sc.nextLine();

        System.out.print("enter grade - ");

        String grade=sc.nextLine();

        System.out.print("enter age - ");

        int age=sc.nextInt();

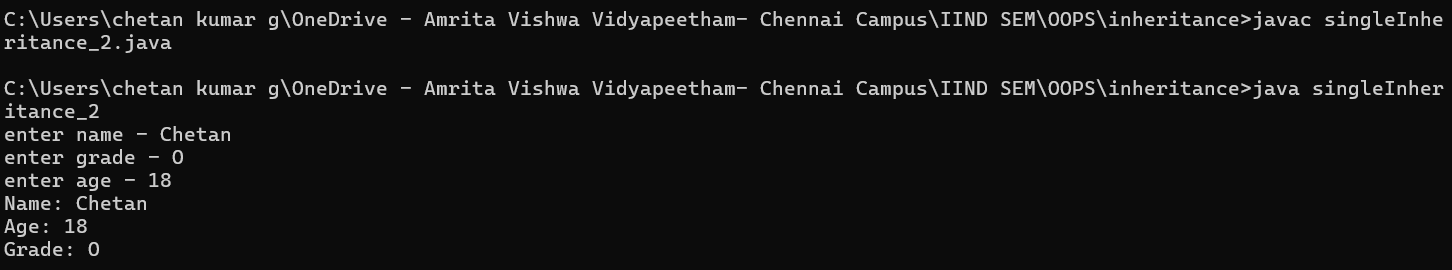
        Student student = new Student(name,age,grade);

        student.displayStudentInfo();

    }

}

**OUTPUT:**



# 5.MULTILEVEL INHERITANCE

## Animals Information

**Code:**

class Animal{

    String name;

    String species;

    Animal(String name,String species){

        this.name=name;

        this.species=species;

    }

    void displayInfo(){

        System.out.println("animal name="+" "+name);

        System.out.println("animal species="+" "+species);

    }

}

class mammal extends Animal{

    boolean hair;

    mammal(boolean hair,String name, String species){

        super(name,species);

        this.hair=hair;

    }

    void displayInfomammal(){

        if (hair){

            System.out.println("yes it has hair");

            displayInfo();

        }

        else{

            System.out.println("no it doesn't has hair");

            displayInfo();

        }

    }

}

public class multilevelInheritance\_1{

    public static void main(String[] args){

        mammal obj = new mammal(true,"bull","dog");

        obj.displayInfomammal();

    }

}

class dog extends mammal{

    String breed;

    dog(String breed,boolean hair,String name, String species){

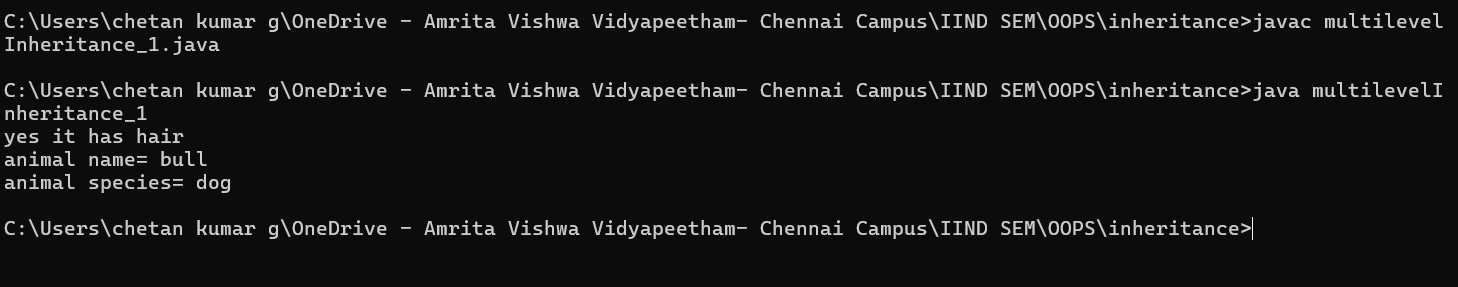
        super(hair,name,species);

        this.breed=breed;

    }

}

**OUTPUT:**



## Shape Details

**Code:**

class Shapes{

    String color;

    Shapes(String color){

        this.color=color;

    }

    void displayInfo(){

        System.out.println("color="+" "+color);

    }

}

class twoDShapes extends Shapes{

    twoDShapes(String color){

        super(color);

    }

    void displayArea(double area){

        System.out.println("the area is="+" "+area);

        displayInfo();

    }

}

class Circle extends twoDShapes{

    int radius;

    Circle(int radius,String color){

        super(color);

        this.radius=radius;

    }

    void calculateArea(){

        double area= 3.14\*radius\*radius;

        displayArea(area);

    }

}

public class multilevelInheritance\_2{

    public static void main(String[] args){

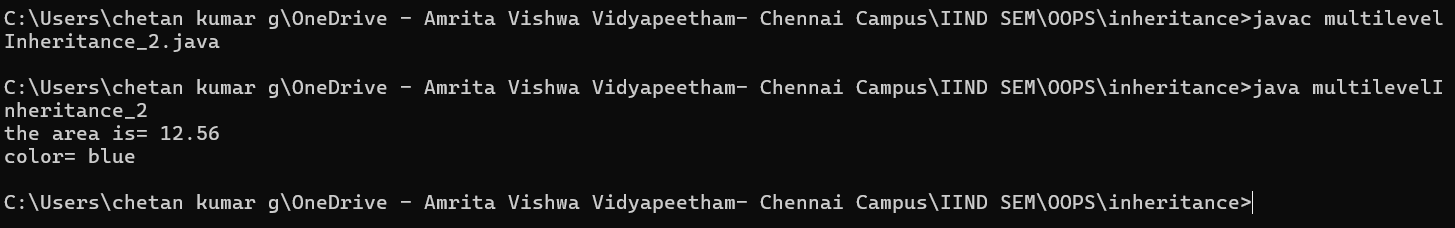
        Circle obj = new Circle(2,"blue");

        obj.calculateArea();

    }

}

**OUTPUT:**



# 6.HIERARCHICAL INHERITANCE

## Employee Management System

**Code:**

import java.util.Arrays;

class Employee{

    String name;

    String department;

    double salary;

    Employee(String name,String department,double salary){

        this.name=name;

        this.department=department;

        this.salary=salary;

    }

    void displayInfo(){

        System.out.println("Name - "+name);

        System.out.println("Department - "+department);

        System.out.println("Salary - "+salary);

    }

}

class manager extends Employee{

    String[] team = new String[4];

    manager(String name,String department,double salary,String[] team){

        super(name,department,salary);

        this.team=team;

    }

    void displayManagerInfo(){

        //System.out.println(Arrays.toString(team));

        for(String name:team){

            System.out.println(name);

        }

        displayInfo();

    }

}

class developer extends Employee{

    String[] languages;

    developer(String name,String department,double salary,String[] languages){

        super(name,department,salary);

        this.languages=languages;

    }

    void displayDeveloperInfo(){

       for(String name:languages){

            System.out.println(name);

        }

        displayInfo();

    }

}

public class hierarchicalInheritance\_1{

    public static void main(String[] args){

        String[] team = {"chetan","agneay","prasannaa","vivek","susendran"};

        String[] lang = {"java","python","c++","c"};

        manager obj1 = new manager("chetan","cse",1000000,team);

        System.out.println("------------Manager details-----------");

        obj1.displayManagerInfo();

        System.out.println("------------Developer details-----------");

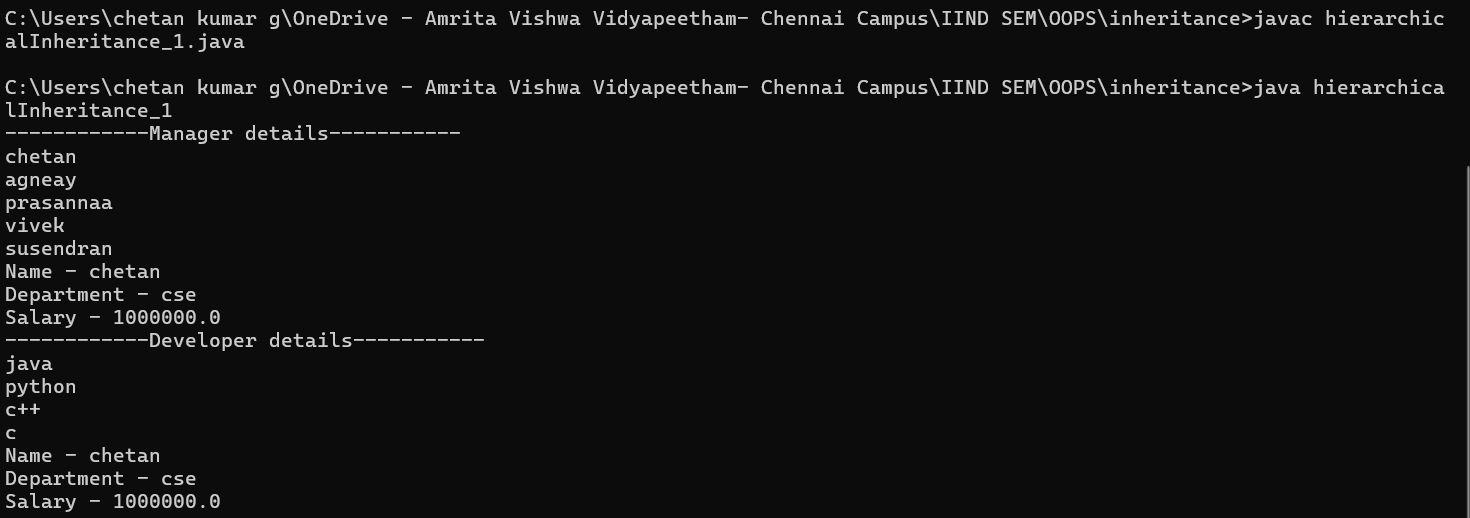
        developer obj2 = new developer("chetan","cse",1000000,lang);

        obj2.displayDeveloperInfo();

    }

}

**OUTPUT:**



## School Management System

**Code:**

import java.util.\*;

class person{

    String name;

    int age;

    String address;

    person(String name,int age, String address){

        this.name=name;

        this.age=age;

        this.address=address;

    }

    void displayInfo(){

        System.out.println("Name - "+name);

        System.out.println("age - "+age);

        System.out.println("address - "+address);

    }

}

class student extends person{

    String[] course = new String[5];

    int id;

    student(String name,int age, String address,String[] course,int id){

        super(name,age,address);

        this.course=course;

        this.id=id;

    }

    void displayStudentInfo(){

        //System.out.println(Arrays.toString(team));

        for(String name:course){

            System.out.print(name+", ");

        }

        System.out.println();

        System.out.println("Id - "+id);

        displayInfo();

    }

}

class teacher extends person{

    String subject;

    teacher(String name,int age, String address,String subject){

        super(name,age,address);

        this.subject=subject;

    }

    void displayTeacherInfo(){

        System.out.println("subject - "+subject);

        displayInfo();

    }

}

public class hierarchicalInheritance\_2{

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        String[] arr= new String[4];

        for (int i=0 ; i<=3 ; i++){

            String arrele=sc.nextLine();

            arr[i]=arrele;

        }

        String name = sc.nextLine();

        String address = sc.nextLine();

        int age = sc.nextInt();

        int id = sc.nextInt();

        student obj1 = new student(name,age,address,arr,id);

        System.out.println("------------Students details-----------");

        obj1.displayStudentInfo();

        sc.nextLine();

        String tname = sc.nextLine();

        String taddress = sc.nextLine();

        String tsubject = sc.nextLine();

        int tage = sc.nextInt();

        teacher obj2 = new teacher(tname,tage,taddress,tsubject);

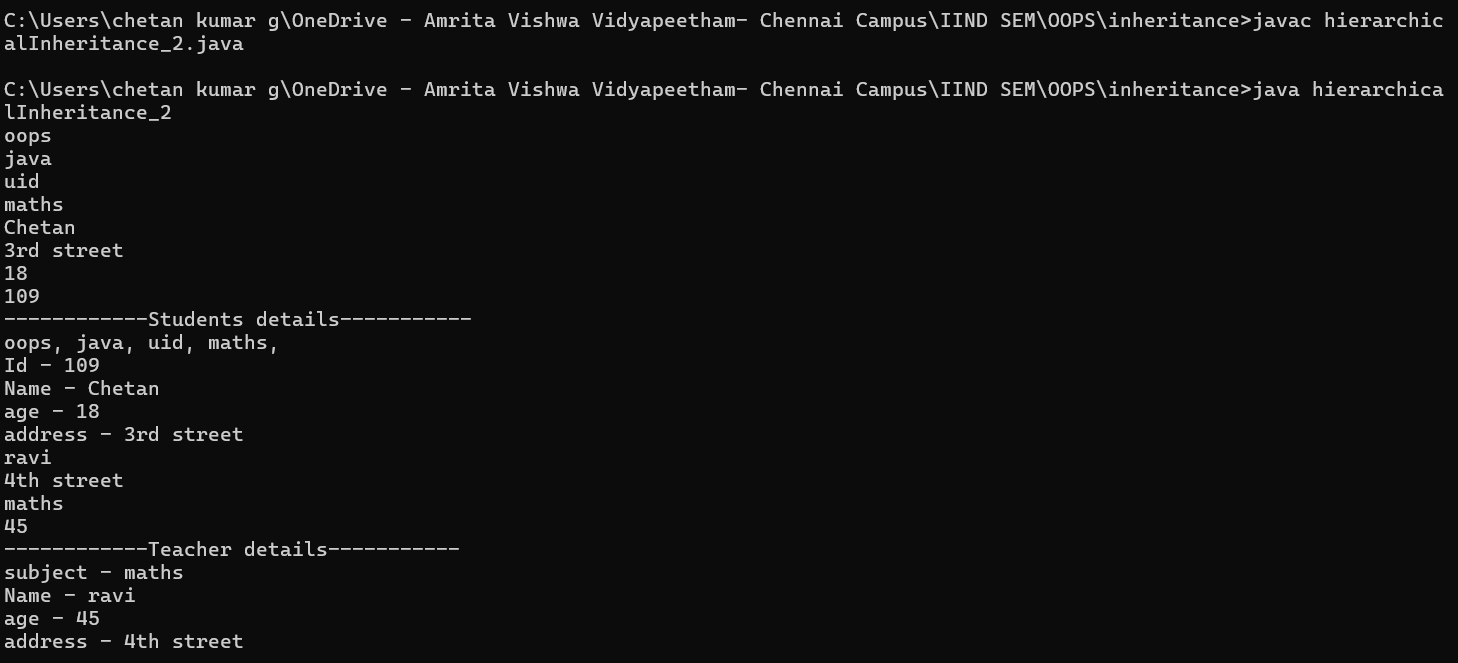
        System.out.println("------------Teacher details-----------");

        obj2.displayTeacherInfo();

    }

}

**OUTPUT:**



# 7.HYBRID INHERITANCE

## Account Management System

**Code:**

import java.util.\*;

class Bank{

    String bankname;

    Bank(String bankname){

        this.bankname=bankname;

    }

    void bankDetails(){

        System.out.println("Bank Name - "+bankname);

    }

}

class branch extends Bank{

    String branchLocation;

    branch(String bankname,String bankLocation){

        super(bankname);

        this.branchLocation=bankLocation;

    }

    void branchLocation(){

        bankDetails();

        System.out.println("Branch Location - "+branchLocation);

    }

}

class savingAccount extends branch{

    double minBalance;

    savingAccount(String bankname,String bankLocation,double minBalance){

        super(bankname, bankLocation);

        this.minBalance=minBalance;

    }

    void accountType(){

        branchLocation();

        System.out.println("Minimunm Balance - "+ minBalance);

    }

}

class currentAccount extends branch{

    double overDraftLimit;

    currentAccount(String bankname,String bankLocation,double overDraftLimit){

        super(bankname, bankLocation);

        this.overDraftLimit=overDraftLimit;

    }

    void accountType(){

        branchLocation();

        System.out.println("Over Draft Limit - "+ overDraftLimit);

    }

}

public class hybridInheritance\_1{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("enter bank name -");

        String name= sc.nextLine();

        System.out.print("enter bank location -");

        String address=sc.nextLine();

        System.out.print("enter minimum balance -");

        double minbal=sc.nextDouble();

        System.out.print("enter overdraft limit -");

        double overdraft=sc.nextDouble();

        savingAccount sa = new savingAccount(name,address,minbal);

        sa.accountType();

        System.out.println("------------------current account----------------");

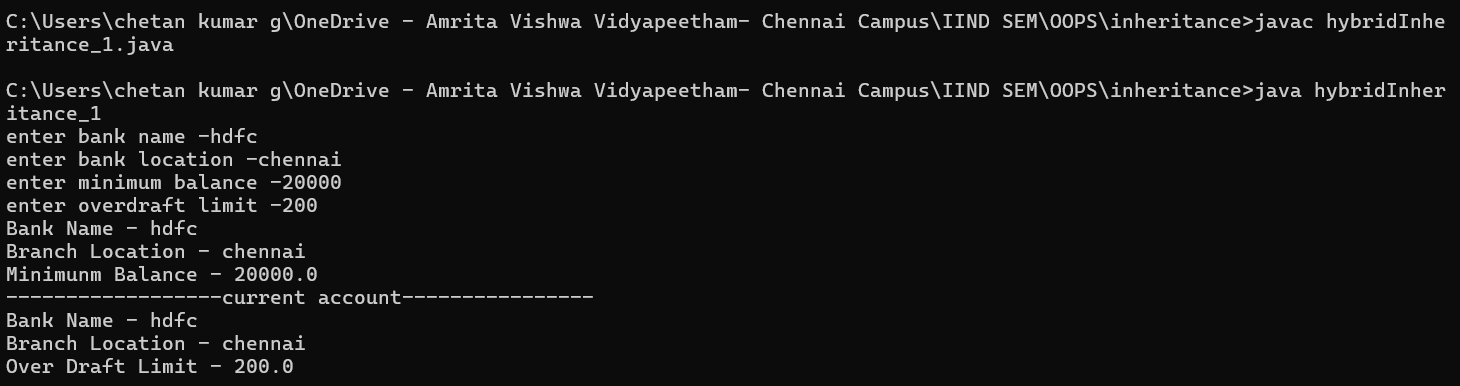
        currentAccount ca = new currentAccount(name,address,overdraft);

        ca.accountType();

    }

}

**OUTPUT:**



## Car Management System

**Code:**

import java.util.Scanner;

class Vehicle {

    String brand;

    Vehicle(String brand){

        this.brand=brand;

    }

    void displayInfo(){

        System.out.println("brand - "+brand);

    }

    void start() {

        System.out.println("Vehicle is starting...");

    }

}

class Car extends Vehicle {

    String fueltype ;

    Car(String brand,String fueltype){

        super(brand);

        this.fueltype=fueltype;

    }

    void displayFuelInfo(){

        System.out.println("Fuel Type - "+fueltype);

    }

    void drive() {

        System.out.println("Car is driving...");

    }

}

class Sedan extends Car {

    int trunksize;

    Sedan(String brand,String fueltype,int trunksize){

        super(brand, fueltype);

        this.trunksize=trunksize;

    }

    void sedanFeatures() {

        displayInfo();

        displayFuelInfo();

        start();

        drive();

        System.out.println("Sedan: Comfortable and fuel-efficient");

        System.out.println("Trunk Size: " + trunksize + " liters");

    }

}

class SUV extends Car {

    double groundClearance;

    SUV(String brand,String fueltype,double groundClearance){

        super(brand, fueltype);

        this.groundClearance=groundClearance;

    }

    void suvFeatures() {

        displayInfo();

        displayFuelInfo();

        start();

        drive();

        System.out.println("SUV: Powerful and spacious");

        System.out.println("Ground Clearance: " + groundClearance + " mm");

    }

}

public class hybridInheritance\_2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String brand = sc.nextLine();

        String fueltype = sc.nextLine();

        double groundClearance=sc.nextDouble();

        int trunksize=sc.nextInt();

        Sedan se = new Sedan(brand,fueltype,trunksize);

        se.sedanFeatures();

        System.out.println("---------------------SUV-------------------");

        SUV su = new SUV(brand,fueltype,groundClearance);

        su.suvFeatures();

    }

}

**OUTPUT:**



# POLYMORPHISM

# 8.CONSTRUCTOR

## Email Management System

**Code:**

import java.util.Scanner;

class user{

    String username;

    String email;

    user(String username, String email){

        this.username=username;

        this.email=email;

        System.out.println("name - "+username);

        System.out.println("Email - "+email);

    }

}

public class constructor\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String username=sc.nextLine();

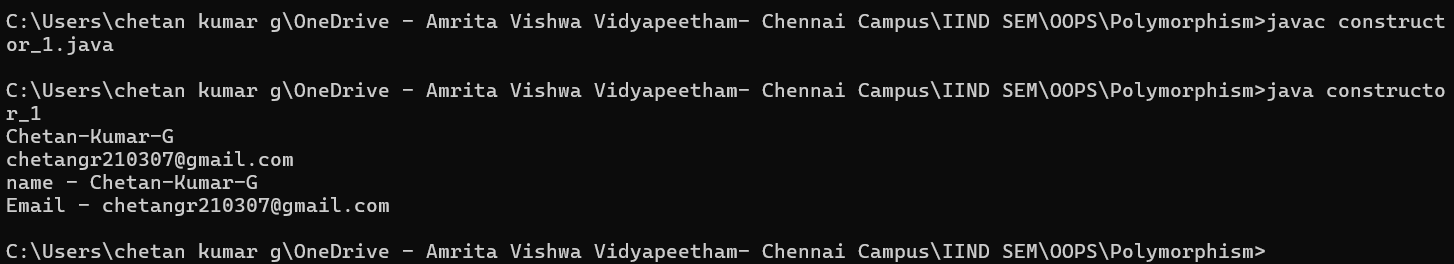
        String email = sc.nextLine();

        user obj = new user(username,email);

    }

}

**OUTPUT:**



# 9.CONSTRUCTOR OVERLOADING

## Animals Information

**Code:**

import java.util.Scanner;

class account{

    int accountNumber;

    double balance;

    String accountType;

    account(int accountNumber,double balance){

        this.accountNumber=accountNumber;

        this.balance=balance;

        System.out.println("accountnumber - "+accountNumber);

        System.out.println("balance - "+balance);

    }

    account(int accountNumber,double balance,String accountType){

        this.accountNumber=accountNumber;

        this.balance=balance;

        this.accountType=accountType;

        System.out.println("accountnumber - "+accountNumber);

        System.out.println("balance - "+balance);

        System.out.println("accountType - "+accountType);

    }

}

public class constructoroverloading\_1 {

    public static void main(String[] args) {

        Scanner sc =new Scanner(System.in);

        String accountType= sc.nextLine();

        int accountNumber= sc.nextInt();

        double balance= sc.nextDouble();

        account obj1 = new account(accountNumber, balance);

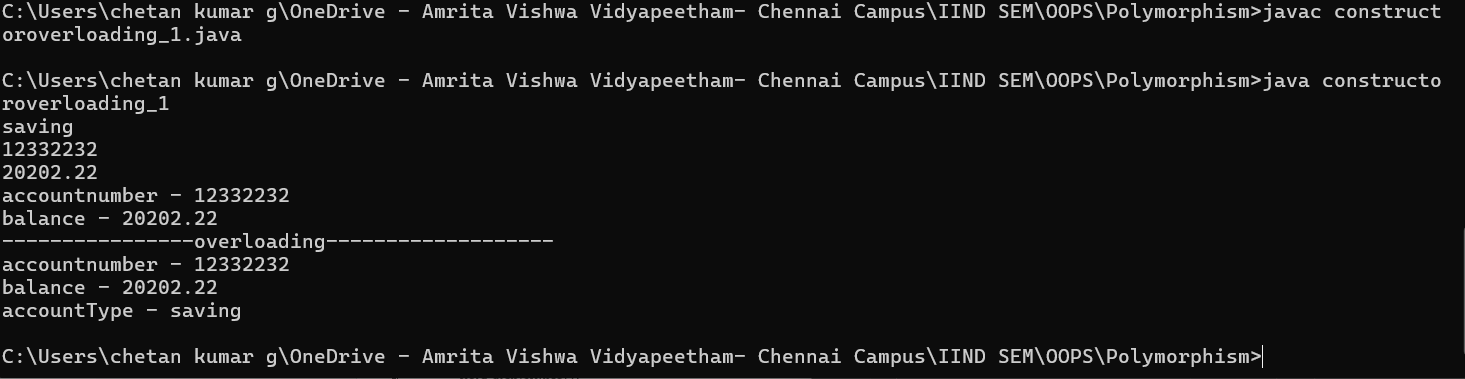
        System.out.println("----------------overloading-------------------");

        account obj2 = new account(accountNumber, balance, accountType);

    }

}

**OUTPUT:**



# 10.METHOD OVERLOADING

## Order Management System

**Code:**

import java.util.\*;

class order{

    int orderid;

    String discountCode;

    int price = 1000;

    void displayPrice(int orderid){

        System.out.println("you order id is "+orderid);

        System.out.println("your price is - "+price);

    }

    void displayPrice(int orderid,String discountCode){

        System.out.println("you order id is "+orderid);

        price = price-(price/2);

        System.out.println("your discount price is - "+price);

    }

}

public class methodoverloading\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String discountCode = sc.nextLine();

        int orderid = sc.nextInt();

        order o1 = new order();

        o1.displayPrice(orderid);

        System.out.println("second object");

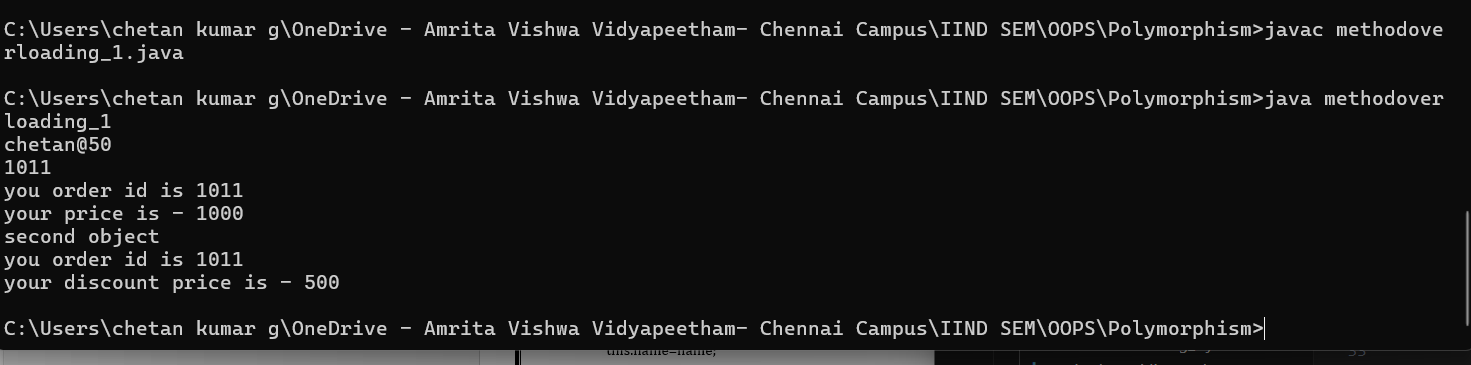
        order o2 = new order();

        o2.displayPrice(orderid,discountCode);

        }

}

**OUTPUT:**



## b) Phone Charging System

**Code:**

import java.util.Scanner;

class mobilecharge{

    int min;

    boolean fastcharge;

    int charge = 40;

    void charge(int min){

        System.out.println("the charge will be"+ (charge+20));

    }

    void charge(int min,boolean fastcharge){

        if (fastcharge){

            System.out.println("the fastcharge will be"+ (charge+40));

        }

        else{

            System.out.println("the charge will be"+ (charge+20));

        }

    }

}

public class methodoverloading\_2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        boolean fastcharge = sc.nextBoolean();

        int min = sc.nextInt();

        mobilecharge obj1 = new mobilecharge();

        obj1.charge(min);

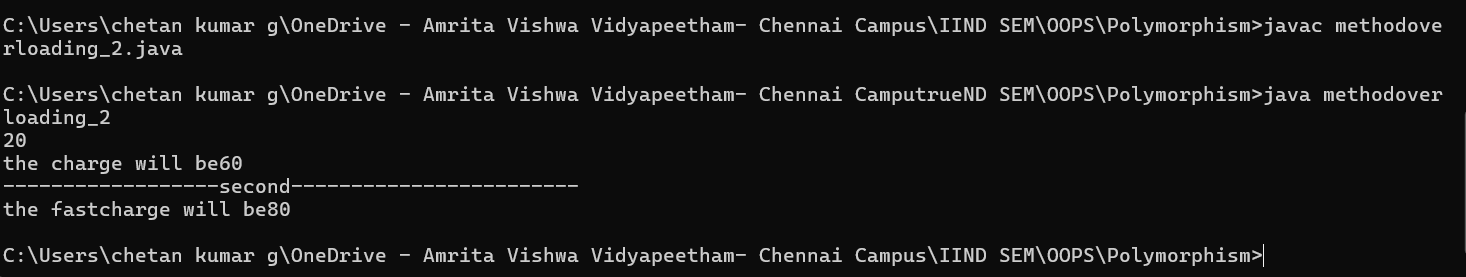
        System.out.println("------------------second------------------------");

        obj1.charge(min, fastcharge);

    }

}

**OUTPUT:**



# 11.METHOD OVERRIDING

## Chatbot System

**Code:**

class Chatbot {

    void reply(String message) {

        System.out.println("Chatbot: Processing message...");

    }

}

class WeatherBot extends Chatbot {

    void reply(String message) {

        System.out.println("WeatherBot: The current temperature is 30°C.");

    }

}

class SupportBot extends Chatbot {

    void reply(String message) {

        System.out.println("SupportBot: How can I assist you with your issue?");

    }

}

public class methodoverriding\_1 {

    public static void main(String[] args) {

        Chatbot bot1 = new WeatherBot();

        Chatbot bot2 = new SupportBot();

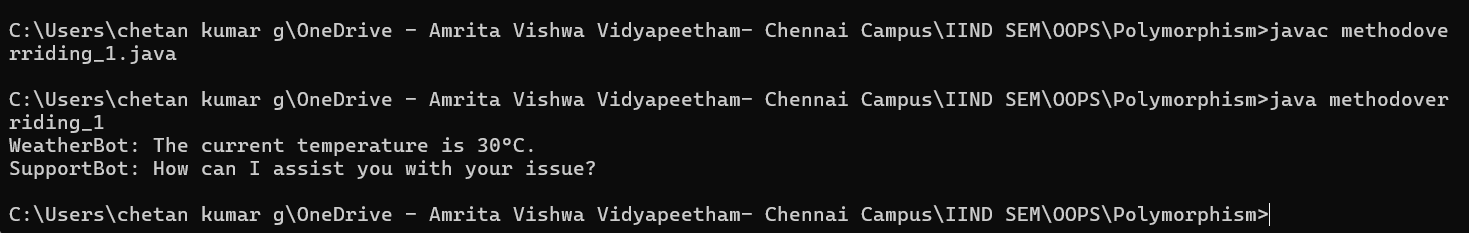
        bot1.reply("What's the weather?");

        bot2.reply("I need help with my order.");

    }

}

**OUTPUT:**



## Vehicle Fare Management System

**Code:**

import java.util.Scanner;

class Vehicle {

    void calculateFare(int distance) {

        System.out.println("Calculating fare for " + distance + " km.");

    }

}

class Car extends Vehicle {

    @Override

    void calculateFare(int distance) {

        System.out.println("Car Fare: " + (distance \* 10));

    }

}

class Bike extends Vehicle {

    @Override

    void calculateFare(int distance) {

        System.out.println("Bike Fare: " + (distance \* 5));

    }

}

class Auto extends Vehicle {

    @Override

    void calculateFare(int distance) {

        System.out.println("Auto Fare: " + (distance \* 7));

    }

}

public class methodoverriding\_2{

    public static void main(String[] args) {

        Vehicle car = new Car();

        Vehicle bike = new Bike();

        Vehicle auto = new Auto();

        Scanner sc = new Scanner(System.in);

        int distance = sc.nextInt();

        car.calculateFare(distance);

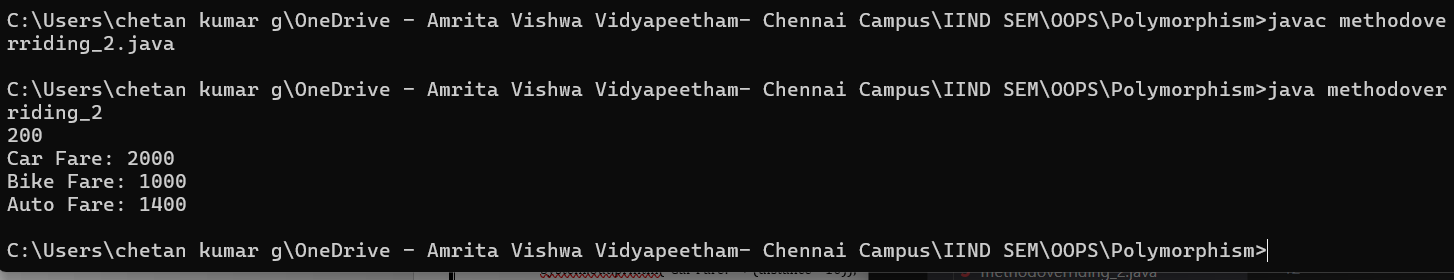
        bike.calculateFare(distance);

        auto.calculateFare(distance);

    }

}

**OUTPUT:**



# ABSTRACTION

# 12. INTERFACE

## Sports Management System

**Code:**

interface swimming{

    void swim();

}

interface running{

    void run();

}

class triathlete implements swimming,running{

    public void swim(){

        System.out.println("i am swimming........");

    }

    public void run(){

        System.out.println("i am running.............");

    }

}

public class interface\_1{

    public static void main(String[] args) {

        triathlete obj = new triathlete();

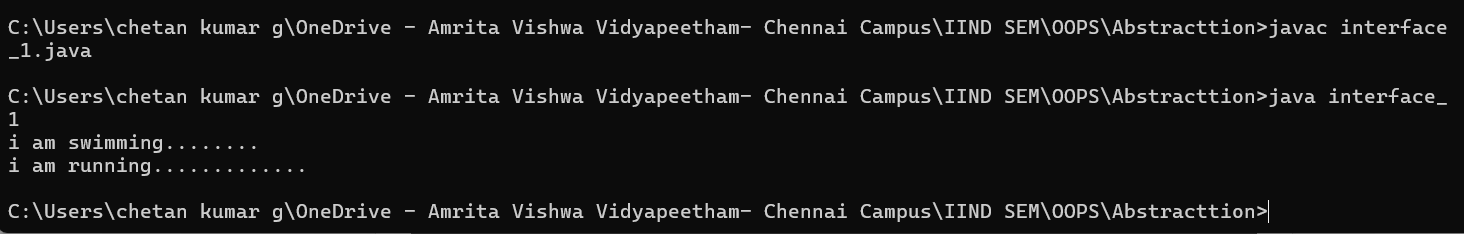
        obj.swim();

        obj.run();

    }

}

**OUTPUT:**



## Music Management System

**Code:**

interface MusicPlayer {

    void play();

    void stop();

}

class Smartphone implements MusicPlayer {

    public void play() {

        System.out.println("Playing music from Smartphone.");

    }

    public void stop() {

        System.out.println("Music stopped.");

    }

}

public class interface\_2 {

    public static void main(String[] args) {

        MusicPlayer myPhone = new Smartphone();

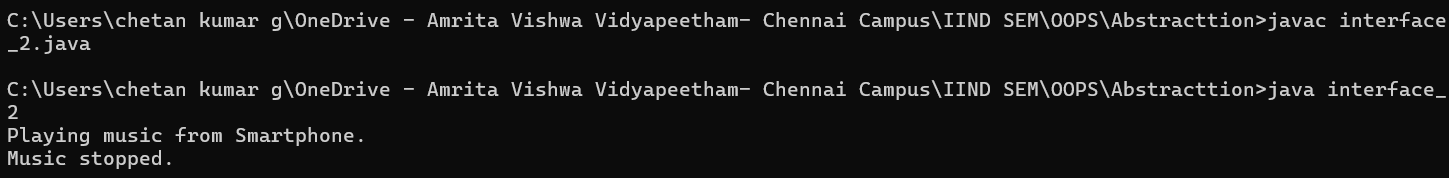
        myPhone.play();

        myPhone.stop();

    }

}

**OUTPUT:**



## c)Smart Home Management System

**Code:**

interface SmartDevice {

    void turnOn();

    void turnOff();

}

class Fan implements SmartDevice {

    public void turnOn() {

        System.out.println("Fan is now ON.");

    }

    public void turnOff() {

        System.out.println("Fan is now OFF.");

    }

}

class Light implements SmartDevice {

    public void turnOn() {

        System.out.println("Light is now ON.");

    }

    public void turnOff() {

        System.out.println("Light is now OFF.");

    }

}

class AC implements SmartDevice {

    public void turnOn() {

        System.out.println("AC is now ON.");

    }

    public void turnOff() {

        System.out.println("AC is now OFF.");

    }

}

public class interface\_3 {

    public static void main(String[] args) {

        SmartDevice[] devices = { new Fan(), new Light(), new AC() };

        for (SmartDevice device : devices) {

            device.turnOn();

        }

        for (SmartDevice device : devices) {

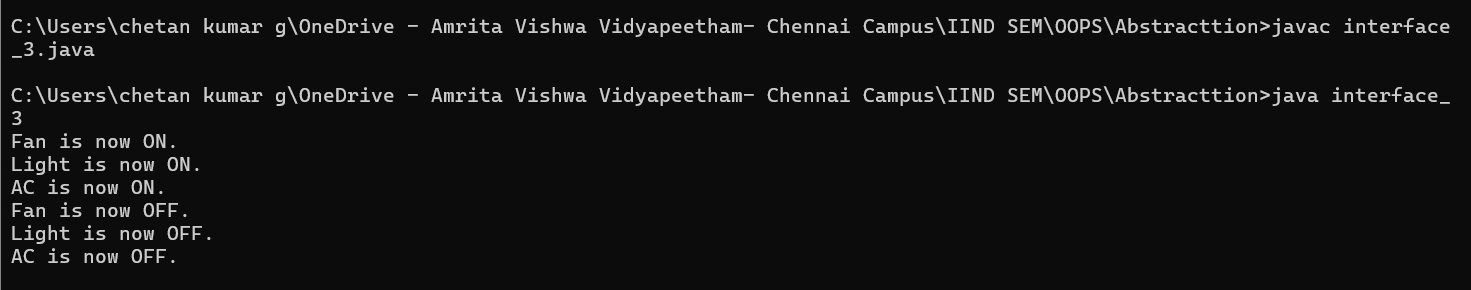
            device.turnOff();

        }

    }

}

**OUTPUT:**



## d)Online Payment Management System

**Code:**

import java.util.Scanner;

interface OnlinePayment {

    void pay(double amount);

    void refund(double amount);

}

class CreditCard implements OnlinePayment {

    public void pay(double amount) {

        System.out.println("Paid " + amount + " using Credit Card.");

    }

    public void refund(double amount) {

        System.out.println("Refunded " + amount + " to Credit Card.");

    }

}

class PayPal implements OnlinePayment {

    public void pay(double amount) {

        System.out.println("Paid " + amount + " using PayPal.");

    }

    public void refund(double amount) {

        System.out.println("Refunded " + amount + " to PayPal.");

    }

}

class UPI implements OnlinePayment {

    public void pay(double amount) {

        System.out.println("Paid " + amount + " using UPI.");

    }

     public void refund(double amount) {

        System.out.println("Refunded " + amount + " to UPI.");

    }

}

public class interface\_4{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Creditcard or Paypal or Upi-");

        String type = sc.nextLine();

        if (type.equalsIgnoreCase("creditcard")){

            OnlinePayment paymentMethod = new CreditCard();

            paymentMethod.pay(1000);

            paymentMethod.refund(500);

        }

        else if(type.equalsIgnoreCase("paypal"))

        {

            OnlinePayment paymentMethod = new PayPal();

            paymentMethod.pay(1000);

            paymentMethod.refund(500);

        }

        else if(type.equalsIgnoreCase("upi"))

        {

            OnlinePayment paymentMethod = new UPI();

            paymentMethod.pay(1000);

            paymentMethod.refund(500);

        }

        else{

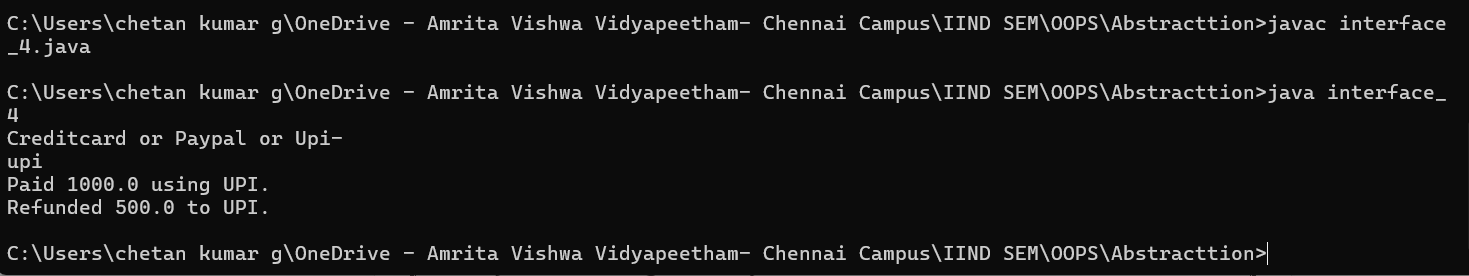
            System.out.println("invalid type");

        }

    }

}

**OUTPUT:**



# 13. ABSTRACT CLASS

## Fare Calculator System

**Code:**

import java.text.ListFormat.Style;

import java.util.Scanner;

abstract class Vehicle {

    String fuelType;

    int speedLimit;

    public Vehicle(String fuelType, int speedLimit) {

        this.fuelType = fuelType;

        this.speedLimit = speedLimit;

    }

    abstract void startEngine();

}

class Car extends Vehicle {

    public Car(String fuelType,int speedLimit) {

        super(fuelType,speedLimit);

    }

    void startEngine() {

        System.out.println("Car engine started with " + fuelType + ". Speed limit: " + speedLimit + " km/h.");

    }

}

class Bike extends Vehicle {

    public Bike(String fuelType,int speedLimit) {

        super(fuelType,speedLimit);

    }

    void startEngine() {

        System.out.println("Bike engine started with " + fuelType + ". Speed limit: " + speedLimit + " km/h.");

    }

}

class Bus extends Vehicle {

    public Bus(String fuelType,int speedLimit) {

        super(fuelType,speedLimit);

    }

    void startEngine() {

        System.out.println("Bus engine started with " + fuelType + ". Speed limit: " + speedLimit + " km/h.");

    }

}

public class abstraction\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String fuelType = sc.nextLine();

        int speedLimit=sc.nextInt();

        Vehicle[] vehicles = { new Car(fuelType,speedLimit), new Bike(fuelType,speedLimit), new Bus(fuelType,speedLimit) };

        for (Vehicle v : vehicles) {

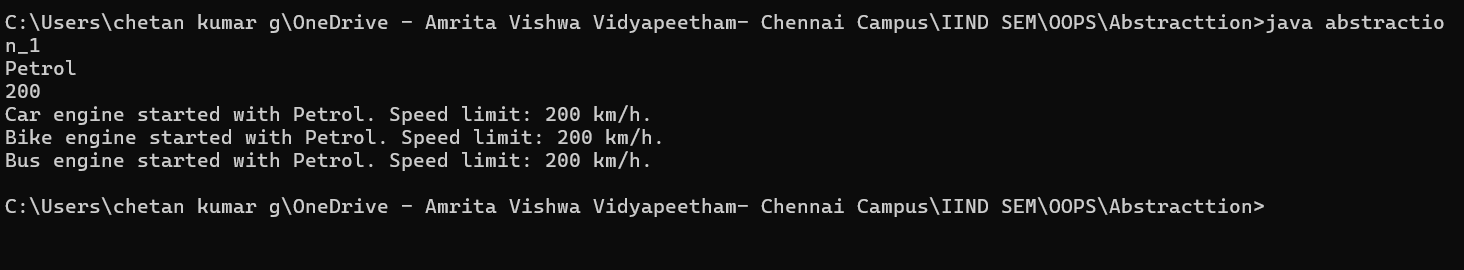
            v.startEngine();

        }

    }

}

**OUTPUT:**



## Employee Management System

**Code:**

abstract class Employee {

    String name;

    int id;

    public Employee(String name, int id) {

        this.name = name;

        this.id = id;

    }

    abstract double calculateSalary();

}

class SalariedEmployee extends Employee {

    double monthlySalary;

    public SalariedEmployee(String name, int id, double monthlySalary) {

        super(name, id);

        this.monthlySalary = monthlySalary;

    }

    double calculateSalary() {

        return monthlySalary;

    }

}

class HourlyEmployee extends Employee {

    double hourlyRate;

    int hoursWorked;

    public HourlyEmployee(String name, int id, double hourlyRate, int hoursWorked) {

        super(name, id);

        this.hourlyRate = hourlyRate;

        this.hoursWorked = hoursWorked;

    }

    double calculateSalary() {

        return hourlyRate \* hoursWorked;

    }

}

public class abstraction\_2{

    public static void main(String[] args) {

        Employee emp1 = new SalariedEmployee("Alice", 101, 50000);

        Employee emp2 = new HourlyEmployee("Bob", 102, 500, 160);

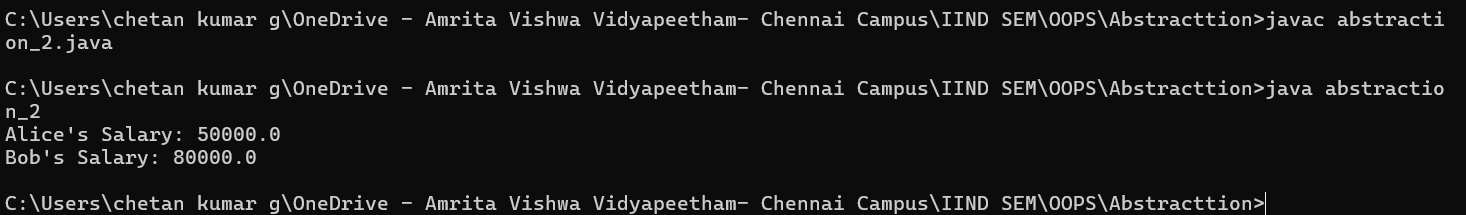
        System.out.println(emp1.name + "'s Salary: " + emp1.calculateSalary());

        System.out.println(emp2.name + "'s Salary: " + emp2.calculateSalary());

    }

}

**OUTPUT:**



## c)Bank Account Management System

**Code:**

abstract class BankAccount {

    String accountNumber;

    double balance;

    public BankAccount(String accountNumber, double balance) {

        this.accountNumber = accountNumber;

        this.balance = balance;

    }

    public void deposit(double amount) {

        balance += amount;

        System.out.println("Deposited " + amount + ". New balance: " + balance);

    }

    abstract void withdraw(double amount);

}

class SavingsAccount extends BankAccount {

    public SavingsAccount(String accountNumber, double balance) {

        super(accountNumber, balance);

    }

    void withdraw(double amount) {

        if (balance - amount >= 1000) {

            balance -= amount;

            System.out.println("Withdrawn " + amount + ". Remaining balance: " + balance);

        } else {

            System.out.println("Withdrawal denied! Minimum balance of 1000 must be maintained.");

        }

    }

}

class CurrentAccount extends BankAccount {

    public CurrentAccount(String accountNumber, double balance) {

        super(accountNumber, balance);

    }

    void withdraw(double amount) {

        if (balance >= amount) {

            balance -= amount;

            System.out.println("Withdrawn " + amount + ". Remaining balance: " + balance);

        } else {

            System.out.println("Insufficient balance!");

        }

    }

}

public class abstraction\_3 {

    public static void main(String[] args) {

        BankAccount savings = new SavingsAccount("SA12345", 5000);

        BankAccount current = new CurrentAccount("CA67890", 10000);

        savings.deposit(2000);

        savings.withdraw(6000);

        savings.withdraw(2000);

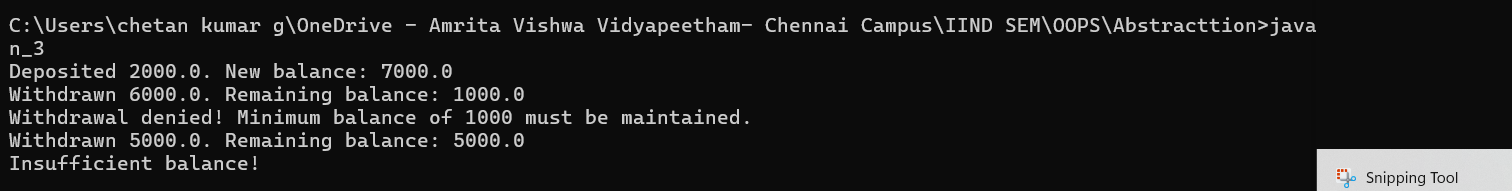
        current.withdraw(5000);

        current.withdraw(6000);

    }

}

**OUTPUT:**



## d)Food Management System

**Code:**

import java.util.Scanner;

abstract class FoodItem {

    String name;

    double price;

    public FoodItem(String name, double price) {

        this.name = name;

        this.price = price;

        System.out.println(name + "price-" +price);

    }

    abstract void prepare();

}

class Pizza extends FoodItem {

    public Pizza(double price) {

        super("Pizza", price);

    }

    void prepare() {

        System.out.println("Preparing Pizza: baking dough, adding toppings, and baking again.");

    }

}

class Burger extends FoodItem {

    public Burger(double price) {

        super("Burger", price);

    }

    void prepare() {

        System.out.println("Preparing Burger: grilling patty, assembling with bun and toppings.");

    }

}

class IceCream extends FoodItem {

    public IceCream(double price) {

        super("Ice Cream", price);

    }

    void prepare() {

        System.out.println("Preparing Ice Cream: freezing mixture and adding flavors.");

    }

}

public class abstraction\_4 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        double pp = sc.nextDouble();

        double bp = sc.nextDouble();

        double ip = sc.nextDouble();

        FoodItem pizza = new Pizza(pp);

        pizza.prepare();

        FoodItem burger = new Burger(bp);

        burger.prepare();

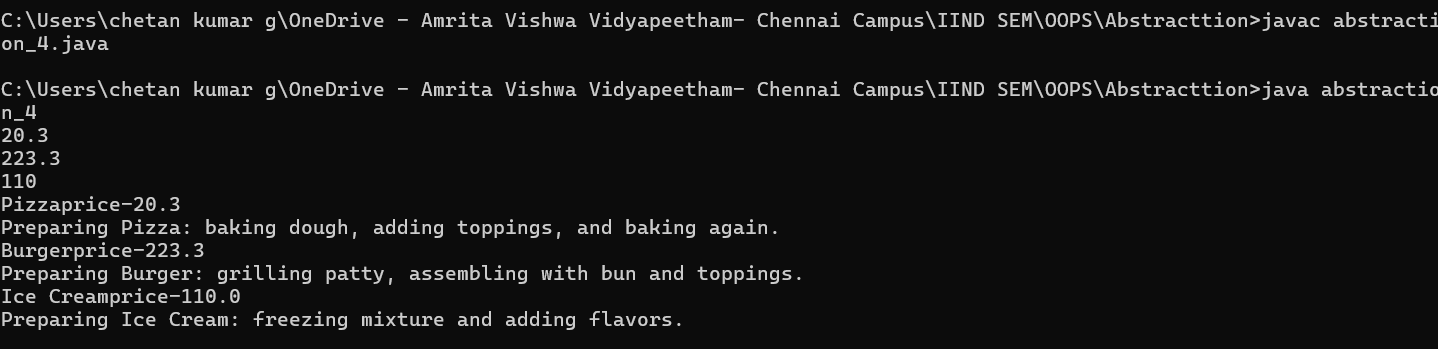
        FoodItem icecream = new IceCream(ip);

        icecream.prepare();

    }

}

**OUTPUT:**



# ENCAPSULATION

# 14. ENCAPSULATION

## Student Management System

**Code:**

import java.util.Scanner;

class students{

    private String name;

    private int roll;

    private String grade;

    public void setname(String name1){

        name=name1;

    }

    public void setgrade(String grade1){

        grade=grade1;

    }

    public void setroll(int roll1){

        roll=roll1;

    }

    public String getname(){

        return name;

    }

    public String getgrade(){

        return grade;

    }

    public int getroll(){

        return roll;

    }

}

public class encapsulation\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("name - ");

        String name1 = sc.nextLine();

        System.out.println("");

        System.out.print("grade - ");

        String grade1 = sc.nextLine();

        System.out.println("");

        System.out.print("roll - ");

        int roll1 = sc.nextInt();

        students obj = new students();

        obj.setname(name1);

        obj.setgrade(grade1);

        obj.setroll(roll1);

        System.out.println("name="+" "+obj.getname());

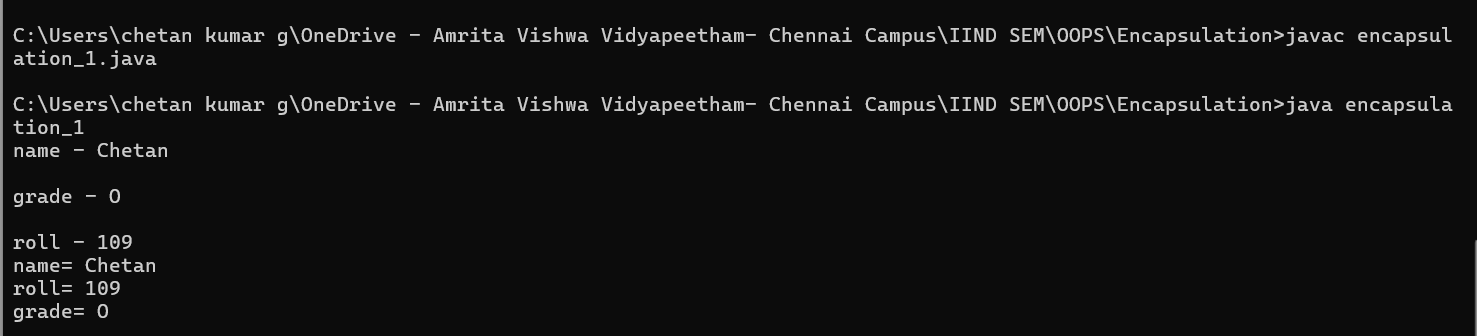
        System.out.println("roll="+" "+obj.getroll());

        System.out.println("grade="+" "+obj.getgrade());

    }

}

**OUTPUT:**



## Car Driving System

**Code:**

import java.util.Scanner;

class car{

    private String model;

    private int speed;

    public void setmodel(String model1){

        model = model1;

    }

    public void setspeed(int speed1){

        speed = speed1;

    }

    public String getmodel(){

        return model;

    }

    public int getspeed(){

        return speed;

    }

    public void acceleration(int amount){

        if(speed + amount<=200){

            speed=speed+amount;

            System.out.println("the acceleration amount="+" "+ amount +", current speed " + speed);

        }

        else{

            System.out.println("maximum speed is 200");

        }

    }

}

public class encapsulation\_2 {

    public static void main(String[] args) {

        Scanner sc =new Scanner(System.in);

        System.out.print("enter model");

        String model=sc.nextLine();

        System.out.print("enter speed");

        int speed=sc.nextInt();

        System.out.print("enter acceleration");

        int acceleration=sc.nextInt();

        car obj =new car();

        obj.setmodel(model);

        obj.setspeed(speed);

        System.out.println("the model is = "+obj.getmodel());

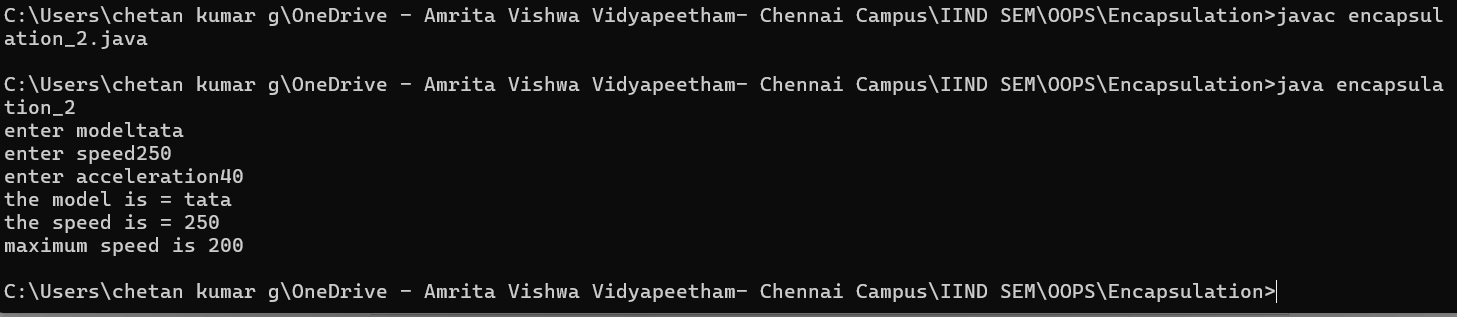
        System.out.println("the speed is = "+obj.getspeed());

        obj.acceleration(acceleration);

    }

}

**OUTPUT:**



## c)Bank Management System

**Code:**

import java.util.Scanner;

class BankAccount {

    private String accountNumber;

    private double balance;

    private int pin;

    public BankAccount(String accountNumber, double initialBalance, int pin) {

        this.accountNumber = accountNumber;

        this.balance = initialBalance;

        this.pin = pin;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Deposited: " + amount + ". New balance: " + balance);

        } else {

            System.out.println("Invalid deposit amount.");

        }

    }

    public void withdraw(double amount, int enteredPin) {

        if (enteredPin == pin) {

            if (amount > 0 && amount <= balance) {

                balance -= amount;

                System.out.println("Withdrawn: " + amount + ". Remaining balance: " + balance);

            } else {

                System.out.println("Insufficient balance or invalid amount.");

            }

        } else {

            System.out.println("Incorrect PIN! Access denied.");

        }

    }

    public void checkBalance(int enteredPin) {

        if (enteredPin == pin) {

            System.out.println("Current balance: " + balance);

        } else {

            System.out.println("Incorrect PIN! Cannot display balance.");

        }

    }

}

public class encapsulation\_3 {

    public static void main(String[] args) {

        Scanner sc =new Scanner(System.in);

        BankAccount myAccount = new BankAccount("123456789",2000.2,1234);

        System.out.println("1: deposit ");

        System.out.println("2: withdraw ");

        System.out.println("3: balance ");

        int check = sc.nextInt();

        if (check==1){

            System.out.print("enter deposit amount : ");

            double depamo = sc.nextDouble();

            myAccount.deposit(depamo);

        }

        else if(check==2){

            System.out.print("enter withdraw amount : ");

            double witamo = sc.nextDouble();

            System.out.print("enter pin : ");

            int wpin = sc.nextInt();

            myAccount.withdraw(witamo, wpin);

        }

        else if(check==3){

            System.out.print("enter pin : ");

            int pin = sc.nextInt();

            myAccount.checkBalance(pin);

        }

        else{

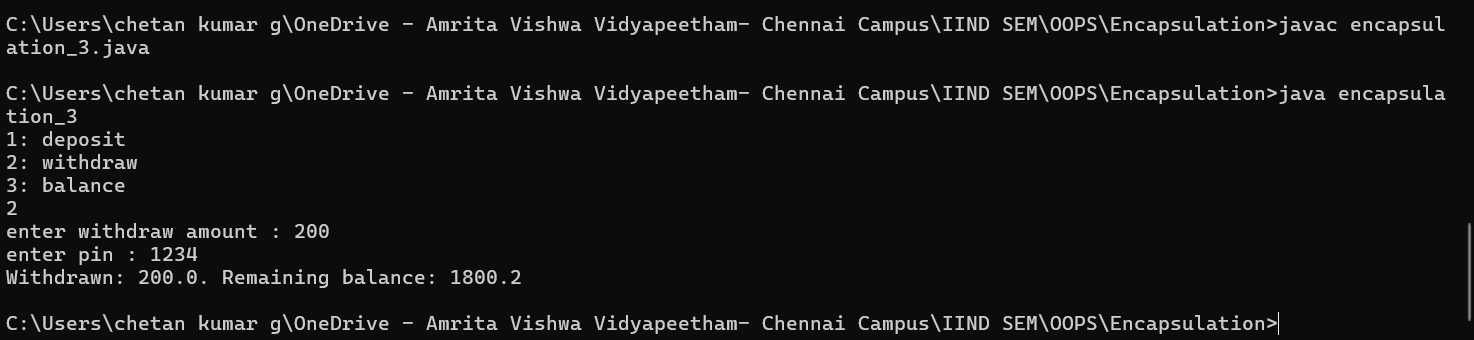
            System.out.println("invalid option");

        }

    }

}

**OUTPUT:**



## d)Library Management System

**Code:**

class Book {

    private String title;

    private String author;

    private int copies;

    public Book(String title, String author, int copies) {

        this.title = title;

        this.author = author;

        this.copies = copies;

    }

    public void borrowBook() {

        if (copies > 0) {

            copies--;

            System.out.println("You borrowed \"" + title + "\" by " + author + ". Copies left: " + copies);

        } else {

            System.out.println("Sorry, \"" + title + "\" is out of stock.");

        }

    }

    public void returnBook() {

        copies++;

        System.out.println("You returned \"" + title + "\". Copies available now: " + copies);

    }

    public void getDetails() {

        System.out.println("Book: " + title + " | Author: " + author + " | Copies available: " + copies);

    }

}

public class encapsulation\_4 {

    public static void main(String[] args) {

        Book myBook = new Book("The Alchemist", "Paulo Coelho", 3);

        myBook.getDetails();

        myBook.borrowBook();

        myBook.borrowBook();

        myBook.borrowBook();

        myBook.borrowBook();

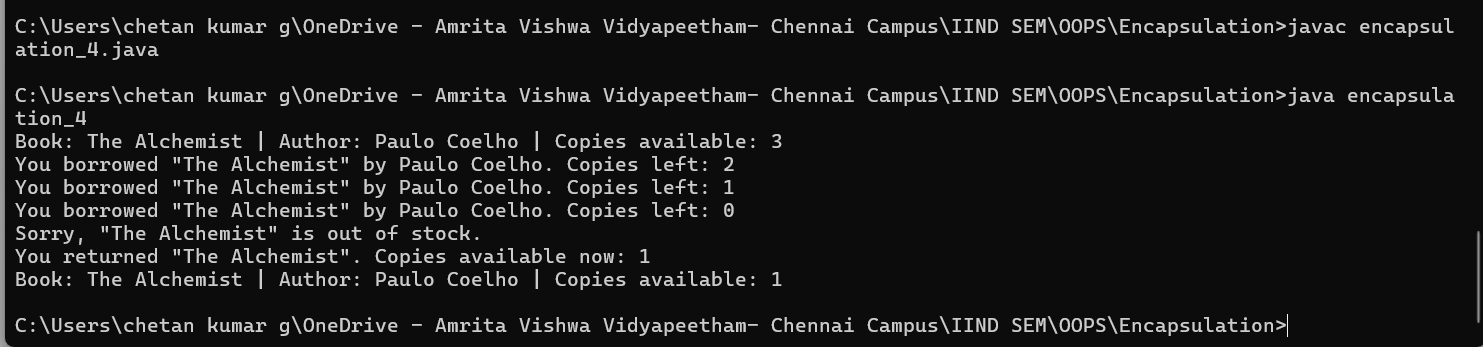
        myBook.returnBook();

        myBook.getDetails();

    }

}

**OUTPUT:**



# 15. PACKAGES

## Library Management System

**Code:**

**Book.java:**

package library.books;

public class Book {

    private int bookID;

    private String title;

    private String author;

    private boolean available;

    public Book(int bookID, String title, String author) {

        this.bookID = bookID;

        this.title = title;

        this.author = author;

        this.available = true;

    }

    public void displayBookDetails() {

        System.out.println("Book ID: " + bookID + ", Title: " + title + ", Author: " + author + ", Available: " + available);

    }

    public boolean isAvailable() {

        return available;

    }

    public void borrowBook() {

        if (available) {

            available = false;

        }

    }

    public void returnBook() {

        available = true;

    }

    public String getTitle() {

        return title;

    }

}

**Member.java:**

package library.members;

public class Member {

    private int memberID;

    private String name;

    private String email;

    public Member(int memberID, String name, String email) {

        this.memberID = memberID;

        this.name = name;

        this.email = email;

    }

    public void displayMemberDetails() {

        System.out.println("Member ID: " + memberID + ", Name: " + name + ", Email: " + email);

    }

    public String getName() {

        return name;

    }

}

**Transaction.java:**

package library.transactions;

import library.books.Book;

import library.members.Member;

public class Transaction {

    public static void borrowBook(Member member, Book book) {

        if (book.isAvailable()) {

            book.borrowBook();

            System.out.println(member.getName() + " borrowed the book: " + book.getTitle());

        } else {

            System.out.println("Book is not available for borrowing.");

        }

    }

    public static void returnBook(Member member, Book book) {

        book.returnBook();

        System.out.println(member.getName() + " returned the book: " + book.getTitle());

    }

}

**LibraryMain.java**

import library.books.Book;

import library.members.Member;

import library.transactions.Transaction;

public class LibraryMain {

    public static void main(String[] args) {

        // Creating books

        Book book1 = new Book(101, "Harry Potter", "J.K. Rowling");

        Book book2 = new Book(102, "The Hobbit", "J.R.R. Tolkien");

        Member member1 = new Member(1, "Chetan Kumar", "chetan@example.com");

        book1.displayBookDetails();

        book2.displayBookDetails();

        member1.displayMemberDetails();

        Transaction.borrowBook(member1, book1);

        Transaction.borrowBook(member1, book1); // Should show "not available"

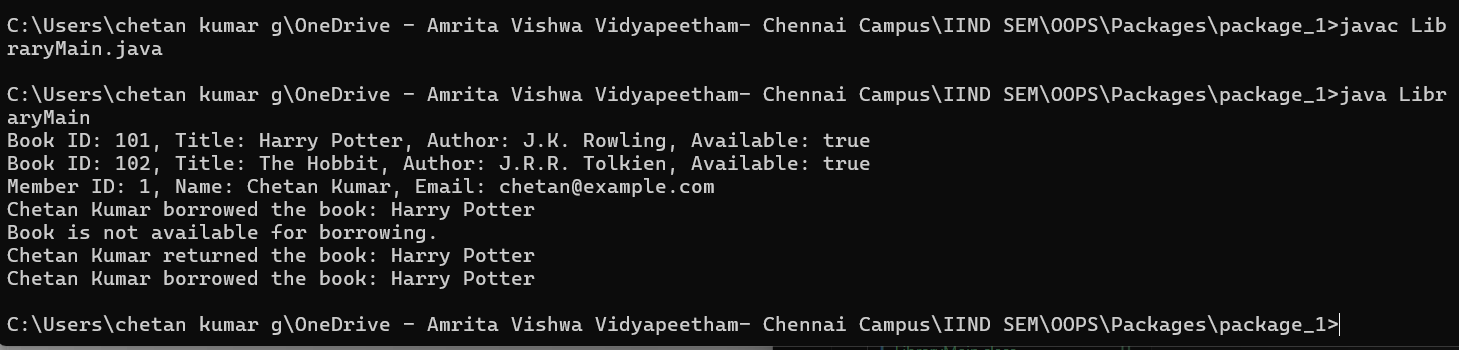
        Transaction.returnBook(member1, book1);

        Transaction.borrowBook(member1, book1); // Now it should work

    }

}

**OUTPUT:**



## Shopping Management System

**Code:**

**Customer.java:**

package shopping.customers;

public class Customer {

    private int customerID;

    private String name;

    public Customer(int customerID, String name) {

        this.customerID = customerID;

        this.name = name;

    }

    public String getName() {

        return name;

    }

}

**Product.java:**

package shopping.products;

public class Product {

    private int productID;

    private String name;

    private double price;

    public Product(int productID, String name, double price) {

        this.productID = productID;

        this.name = name;

        this.price = price;

    }

    public void displayProduct() {

        System.out.println("Product ID: " + productID + ", Name: " + name + ", Price: $" + price);

    }

    public double getPrice() {

        return price;

    }

    public String getName() {

        return name;

    }

}

**Order.java:**

package shopping.orders;

import shopping.products.Product;

import shopping.customers.Customer;

public class Order {

    public static void placeOrder(Customer customer, Product product) {

        System.out.println(customer.getName() + " ordered: " + product.getName() + " for $" + product.getPrice());

    }

}

**ShoppingMain.java:**

import shopping.products.Product;

import shopping.customers.Customer;

import shopping.orders.Order;

public class ShoppingMain {

    public static void main(String[] args) {

        Product p1 = new Product(201, "Laptop", 75000);

        Customer c1 = new Customer(101, "Amit Sharma");

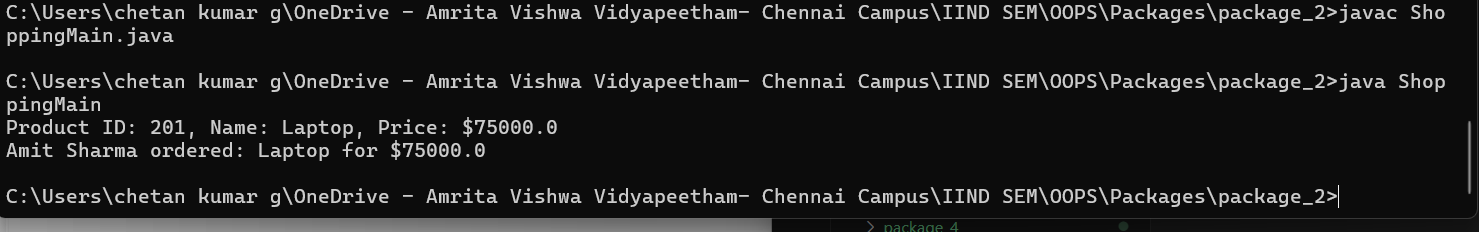
        p1.displayProduct();

        Order.placeOrder(c1, p1);

    }

}

**OUTPUT:**



## c)IP Address System

**Code:**

package systeminfo.demo;

import java.net.InetAddress;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.security.SecureRandom;

public class SystemInfo {

    public static void main(String[] args) {

        try {

            InetAddress ip = InetAddress.getLocalHost();

            System.out.println("Your IP Address: " + ip.getHostAddress());

            LocalDateTime now = LocalDateTime.now();

            DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

            System.out.println("Current Date & Time: " + now.format(format));

            SecureRandom random = new SecureRandom();

            int randomNumber = random.nextInt(1000);

            System.out.println("Secure Random Number: " + randomNumber);

        } catch (Exception e) {

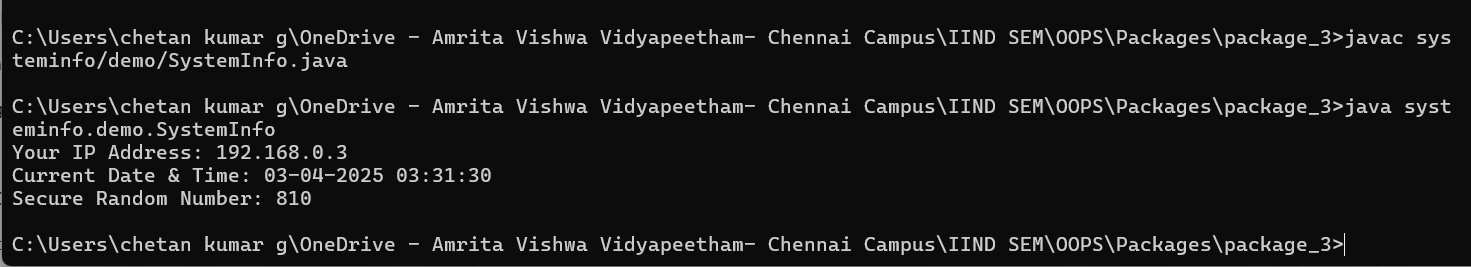
            System.out.println("Error: " + e.getMessage());

        }

    }

}

**OUTPUT:**



## d)Draw A Rectangle

**Code:**

package graphics.demo;

import java.awt.\*;

import javax.swing.\*;

public class DrawRectangle extends JPanel {

    public void paintComponent(Graphics g) {

        super.paintComponent(g);

        g.setColor(Color.RED);

        g.fillRect(50, 50, 200, 100);

    }

    public static void main(String[] args) {

        JFrame frame = new JFrame();

        frame.add(new DrawRectangle());

        frame.setSize(400, 300);

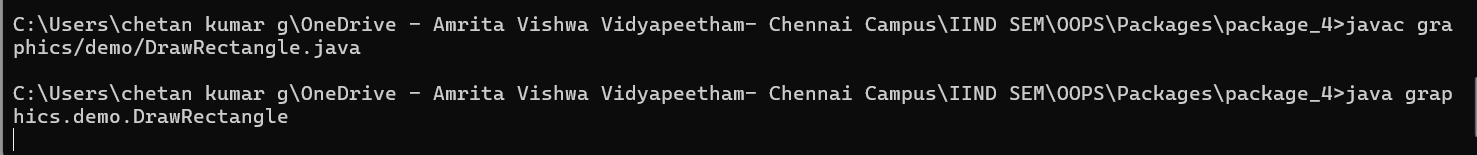
        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setVisible(true);

    }

}

**OUTPUT:**

****



# 16. EXCEPTIONAL HANDELING

## ATM Withdraw System

**Code:**

import java.util.Scanner;

class InsufficientBalanceException extends Exception {

    public InsufficientBalanceException(String message) {

        super(message);

    }

}

class BankAccount {

    private double balance;

    public BankAccount(double balance) {

        this.balance = balance;

    }

    public void withdraw(double amount) throws InsufficientBalanceException {

        if (amount > balance) {

            throw new InsufficientBalanceException("Insufficient balance! Available balance: " + balance);

        }

        balance -= amount;

        System.out.println("Withdrawal successful! Remaining balance: " + balance);

    }

}

public class errorhandeling\_1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        double amount = sc.nextDouble();

        BankAccount account = new BankAccount(5000);

        try {

            account.withdraw(amount);

        } catch (InsufficientBalanceException e) {

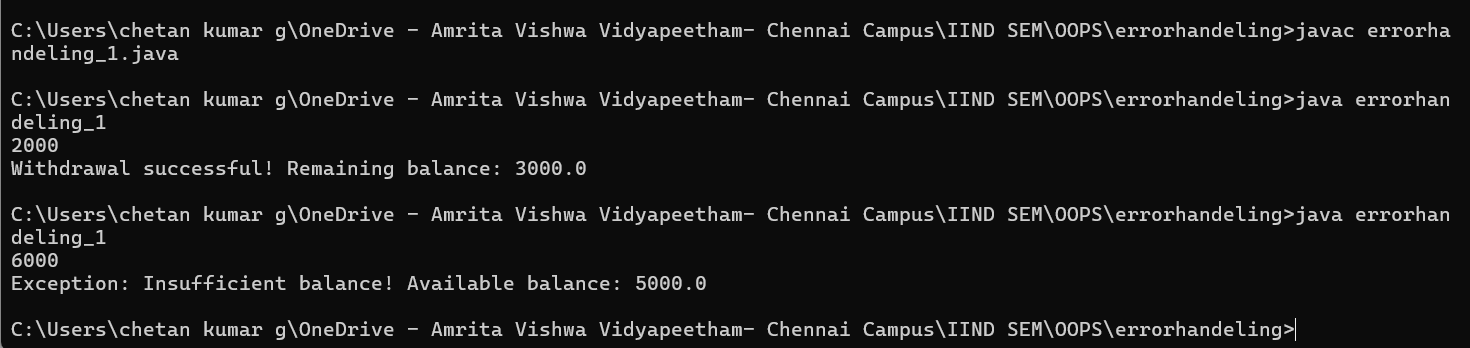
            System.out.println("Exception: " + e.getMessage());

        }

    }

}

**OUTPUT:**



## Division By 0

**Code:**

import java.util.Scanner;

class Calculator {

    public double divide(int num1, int num2) {

        return num1 / num2;

    }

}

public class errorhandeling\_2 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        int num1 = scanner.nextInt();

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

        int num2 = scanner.nextInt();

        Calculator calc = new Calculator();

        try {

            double result = calc.divide(num1, num2);

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

        } catch (ArithmeticException e) {

            System.out.println("Error: Cannot divide by zero!");

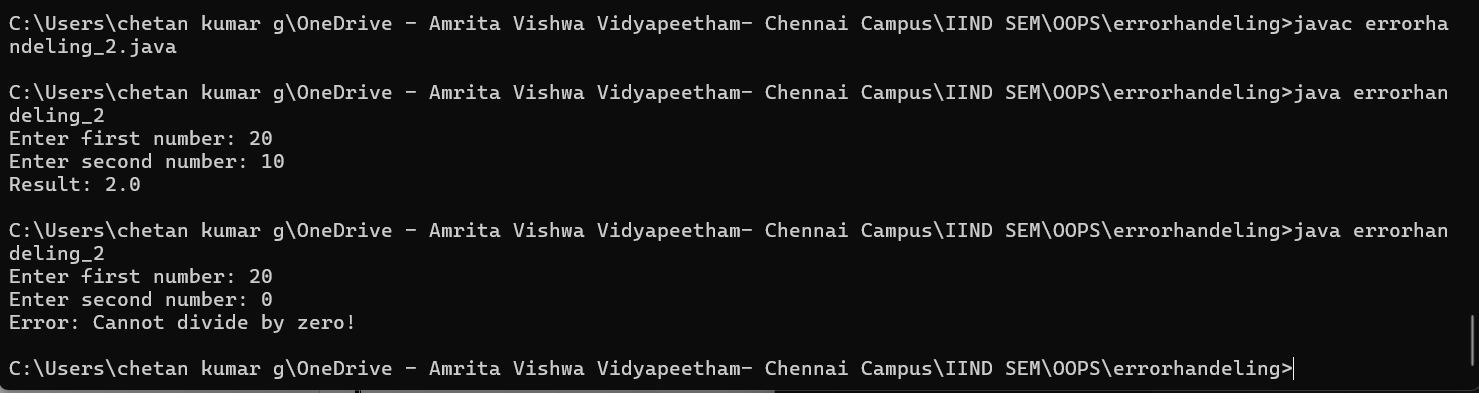
        }

        scanner.close();

    }

}

**OUTPUT:**



## c)Array Index Manager

**Code:**

import java.util.Scanner;

class ArrayHandler {

    private int[] numbers = {10, 20, 30, 40, 50};

    public void printValueAtIndex(int index) {

        System.out.println("Value at index " + index + ": " + numbers[index]);

    }

}

public class errorhandeling\_3 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        ArrayHandler arrayHandler = new ArrayHandler();

        System.out.print("Enter an index (0-4): ");

        int index = scanner.nextInt();

        try {

            arrayHandler.printValueAtIndex(index);

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Error: Invalid index entered!");

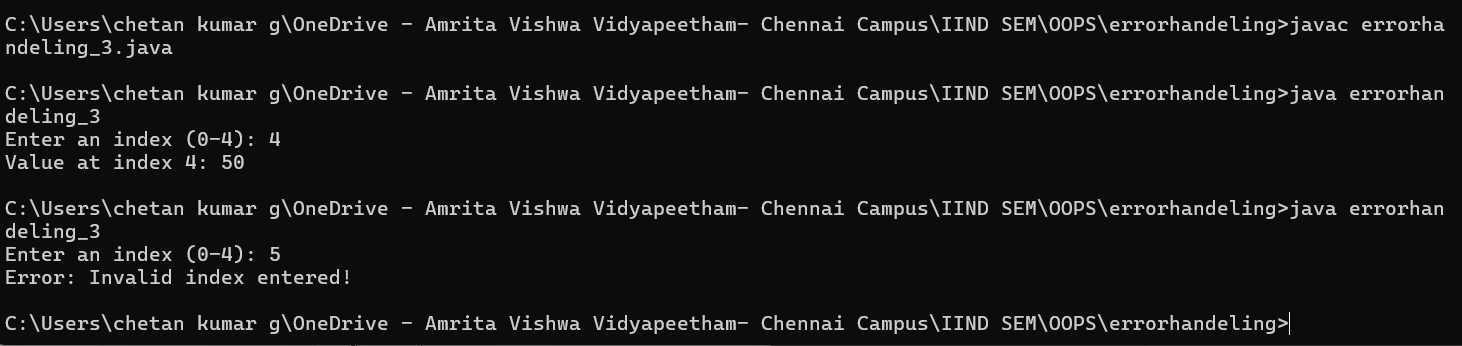
        }

        scanner.close();

    }

}

**OUTPUT:**



## d)File Reading

**Code:**

import java.io.\*;

class FileHandler {

    public void readFile(String filename) throws IOException {

        File file = new File(filename);

        BufferedReader br = new BufferedReader(new FileReader(file));

        String line;

        while ((line = br.readLine()) != null) {

            System.out.println(line);

        }

        br.close();

    }

}

public class errorhandeling\_4{

    public static void main(String[] args) {

        FileHandler fileHandler = new FileHandler();

        String filename = "sample.txt";

        try {

            fileHandler.readFile(filename);

        } catch (FileNotFoundException e) {

            System.out.println("Error: File not found!");

        } catch (IOException e) {

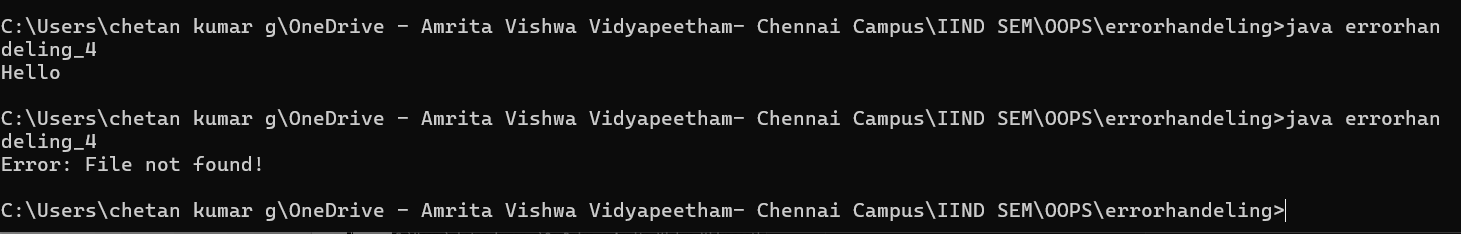
            System.out.println("Error: Issue reading the file!");

        }

    }

}

**OUTPUT:**



# 17. FILE HANDELING

## Writing/Reading A File

**Code:**

import java.io.\*;

public class filehandeling\_1 {

    public static void main(String[] args) {

        String fileName = "output.txt";

        String content = "Hello, this is a sample text file.\nJava file handling is interesting!";

        try (FileWriter writer = new FileWriter(fileName)) {

            writer.write(content);

            System.out.println("File written successfully.");

        } catch (IOException e) {

            System.out.println("An error occurred while writing to the file.");

            e.printStackTrace();

        }

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

            String line;

            System.out.println("\nReading from file:");

            while ((line = reader.readLine()) != null) {

                System.out.println(line);

            }

        } catch (IOException e) {

            System.out.println("An error occurred while reading the file.");

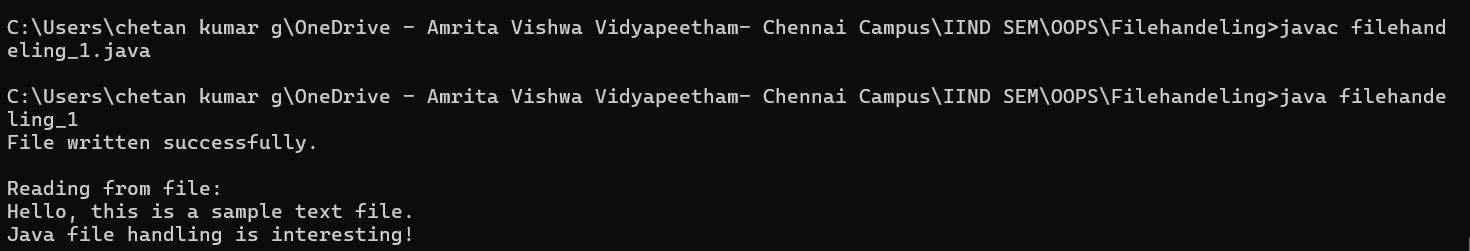
            e.printStackTrace();

        }

    }

}

**OUTPUT:**



## Line Count

**Code:**

import java.io.\*;

public class filehandeling\_2{

    public static void main(String[] args) {

        String fileName = "output.txt";

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

            int lineCount = 0, wordCount = 0;

            String line;

            while ((line = reader.readLine()) != null) {

                lineCount++;

                wordCount += line.split("\\s+").length;

            }

            System.out.println("\nTotal Lines: " + lineCount);

            System.out.println("Total Words: " + wordCount);

        } catch (IOException e) {

            System.out.println("An error occurred while processing the file.");

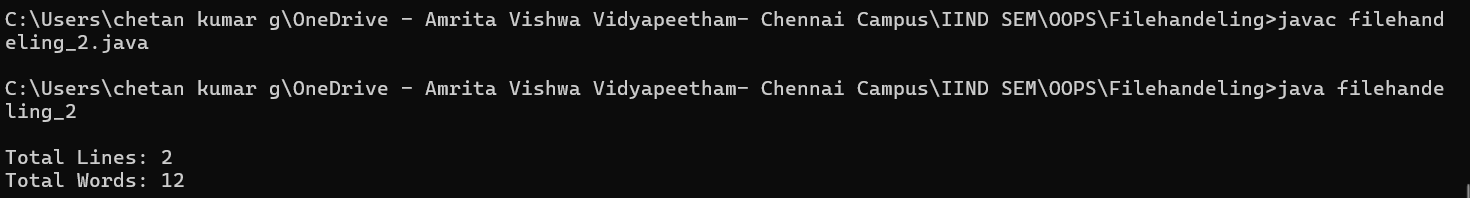
            e.printStackTrace();

        }

    }

}

**OUTPUT:**



## c)Append Data To File

**Code:**

import java.io.\*;

public class filehandeling\_3{

    public static void main(String[] args) {

        String fileName = "output.txt";

        try (FileWriter writer = new FileWriter(fileName, true);

             BufferedWriter bufferedWriter = new BufferedWriter(writer)) {

            bufferedWriter.newLine();

            bufferedWriter.write("Appending new data to the file.");

            System.out.println("\nData appended successfully.");

        } catch (IOException e) {

            System.out.println("An error occurred while appending data.");

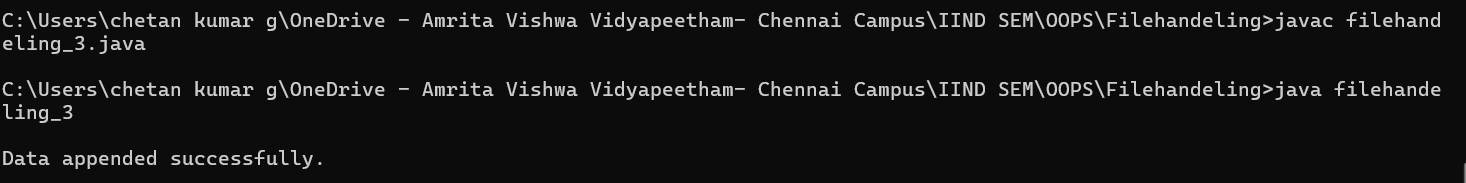
            e.printStackTrace();

        }

    }

}

**OUTPUT:**



output.txt:

Hello, this is a sample text file.

Java file handling is interesting!

Appending new data to the file.

## d)Copy The File

**Code:**

import java.io.\*;

public class filehandeling\_4{

    public static void main(String[] args) {

        String sourceFile = "output.txt";

        String destinationFile = "destination.txt";

        try (BufferedReader reader = new BufferedReader(new FileReader(sourceFile));

             BufferedWriter writer = new BufferedWriter(new FileWriter(destinationFile))) {

            String line;

            while ((line = reader.readLine()) != null) {

                writer.write(line);

                writer.newLine();

            }

            System.out.println("\nFile copied successfully to " + destinationFile);

        } catch (IOException e) {

            System.out.println("An error occurred while copying the file.");

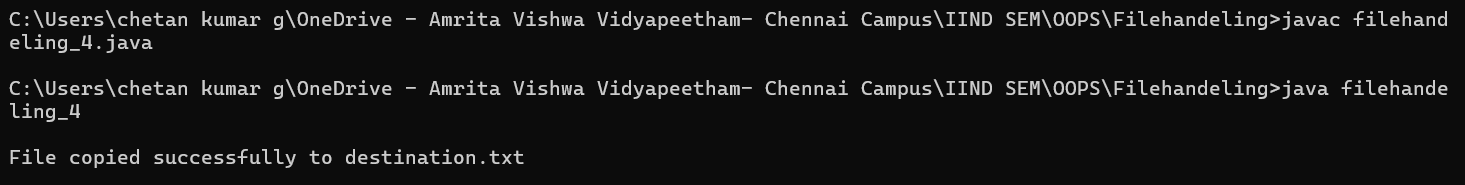
            e.printStackTrace();

        }

    }

}

**OUTPUT:**



destination.txt:

Hello, this is a sample text file.

Java file handling is interesting!

Appending new data to the file.