

**LAB RECORD**

23CSE111 – Object Oriented Programming

***Submitted by***

CH.SC.U4CSE24060 – **PRODDUTURU JASHWANTH**

**BACHELOR OF TECHNOLOGY**

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

## 

**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.U4CSE24060 –PRODDUTURU JASHWANTH*** 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.

|  |  |
| --- | --- |
| Internal Examiner 1 | Internal Examiner 2 |

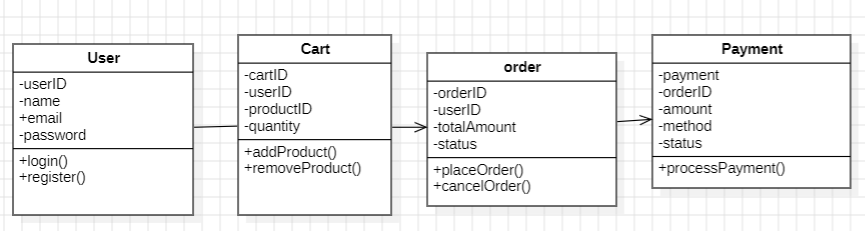
# **Index**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Experiment Name** | **Page Number** |
| 1. | UML DIAGRAM | 4 |
|  | Online Shopping   * Class Diagram * Use Case Diagram * Sequence Diagram * State Activity Diagram * Object Diagram | 4 |
| 2. | Library Management   * Class Diagram * Use Case Diagram * Sequence Diagram * State Activity Diagram * Object Diagram | 8 |
| 3. | Java Basic Programs | 12 |
| i) | Odd or Even | 12 |
| ii) | Factorial | 13 |
| iii) | Reverse a Number | 14 |
| iv) | Palindrome | 15 |
| v) | Multiples of 2 | 16 |
| vi) | Quadratic Equation | 17 |
| vii) | Hypotenuse of a Triangle | 18 |
| viii) | Square Root of a Number | 19 |
| xi) | Armstrong of a Number | 20 |
| x) | Greatest of a Numbers | 21 |
|  |  |  |
|  | **INHERITANCE:** |  |
| 4. | Single Inheritance |  |
| i) | Students Details | 23 |
| ii | Bank Details |  |
| 5. | **MULTILEVEL INHERITANCE PROGRAMS** |  |
| i) | General Details |  |
| ii) | Employee Salary |  |
| 6. | **HIERARCHICAL INHERITANCE PROGRAMS** |  |
| i) | Vehicle Model |  |
| ii) | Person Details |  |
| 7. | **HYBRID INHERITANCE PROGRAMS** |  |
| i) | Student Details |  |
| ii) | Vehicle type |  |
|  | **POLYMORPHISM** |  |
| 8. | **CONSTRUCTOR PROGRAMS** |  |
| i) | Book Details |  |
| 9. | **CONSTRUCTOR OVERLOADING PROGRAMS** |  |
| i) | Employee info |  |
| 10. | **METHOD OVERLOADING PROGRAMS** |  |
| i) | Employee details |  |
| ii) | Shape details |  |
| 11. | **METHOD OVERRIDING PROGRAMS** |  |
| i) | Vehicle car bike class |  |
| ii) | BankAccount interest Rate |  |
|  | **ABSTRACTION** |  |
| 12. | **INTERFACE PROGRAMS** |  |
| i) | Shape Area |  |
| ii) | Camara and Music Player class |  |
| iii) | Payment Type |  |
| iv) | FuelType |  |
| 13. | **ABSTRACT CLASS PROGRAMS** |  |
| i) | Mileage Calculator |  |
| ii) | Vehicle start and stop |  |
| iii) | Area of shape |  |
| iv) | Shape 2D |  |
|  | **ENCAPSULATION** |  |
| 14. | **ENCAPSULATION PROGRAMS** |  |
| i) | Patients Records |  |
| ii) | Product Details |  |
| iii) | House Details |  |
| iv) | Game Characters |  |
| 15. | **PACKAGES PROGRAMS** |  |
| i) | Button creation |  |
| ii) | Basic Calculation |  |
| iii) | Area calculation |  |
| iv) | Employee Details |  |
| 16. | **EXCEPTION HANDLING PROGRAMS** |  |
| i) | Division by 0 |  |
| ii) | Age limit with throws keyword |  |
| iii) | Age limit with throw |  |
| iv) | File missing error |  |
| 17. | **FILE HANDLING PROGRAMS** |  |
| i) | Reading the file |  |
| ii) | Writing the file |  |
| iii) | Reading N writing to a file |  |
| iv) | writing to file multiple try catch |  |

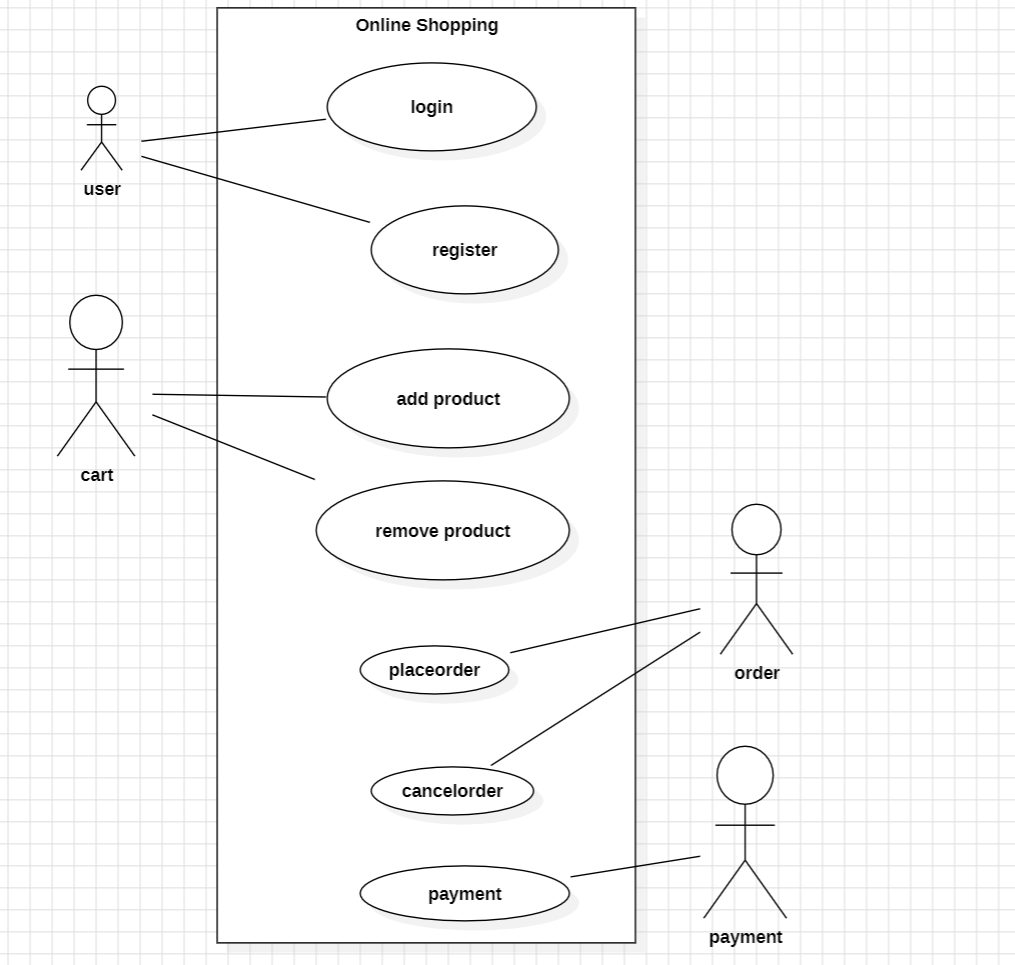
UML DIAGRAM

1.Online Shopping

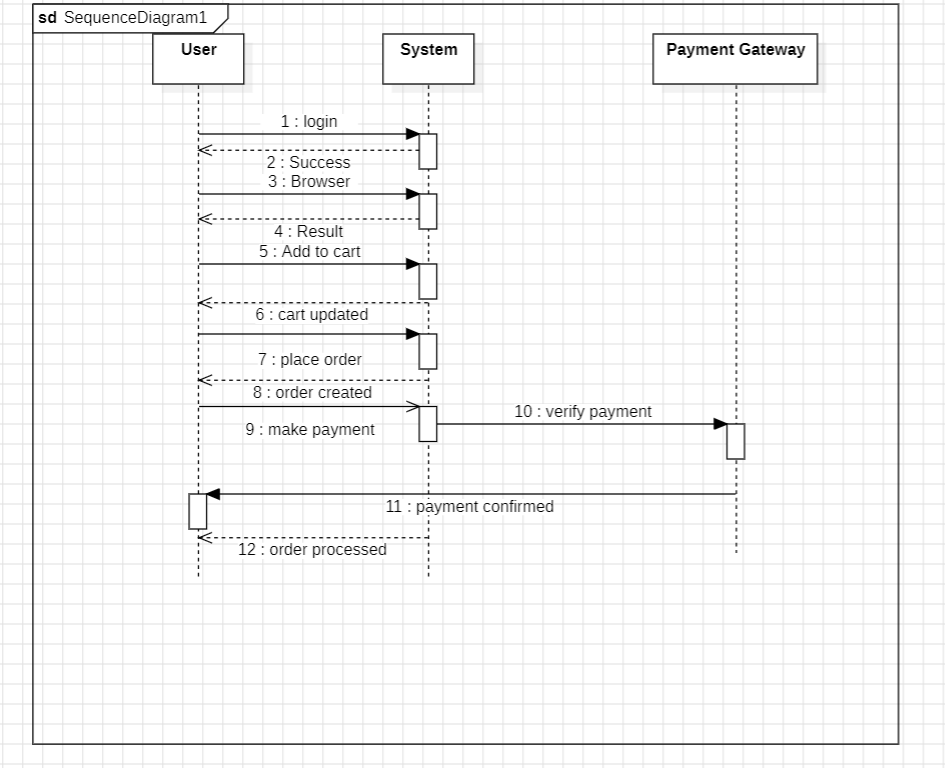
1. Class Diagram:



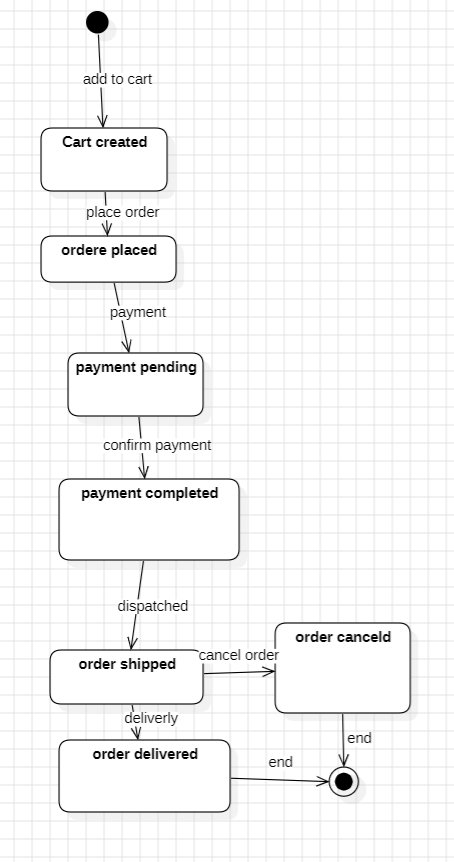
1. Use Case Diagram:



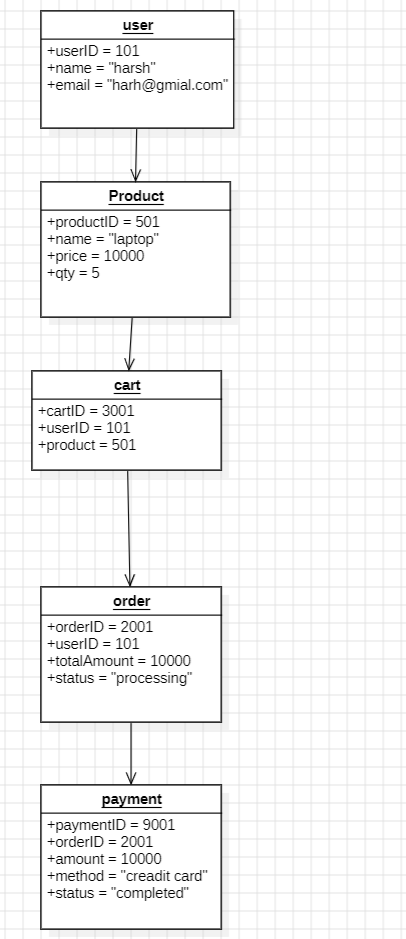
1. Sequence Diagram:



1. State Activity Diagram:

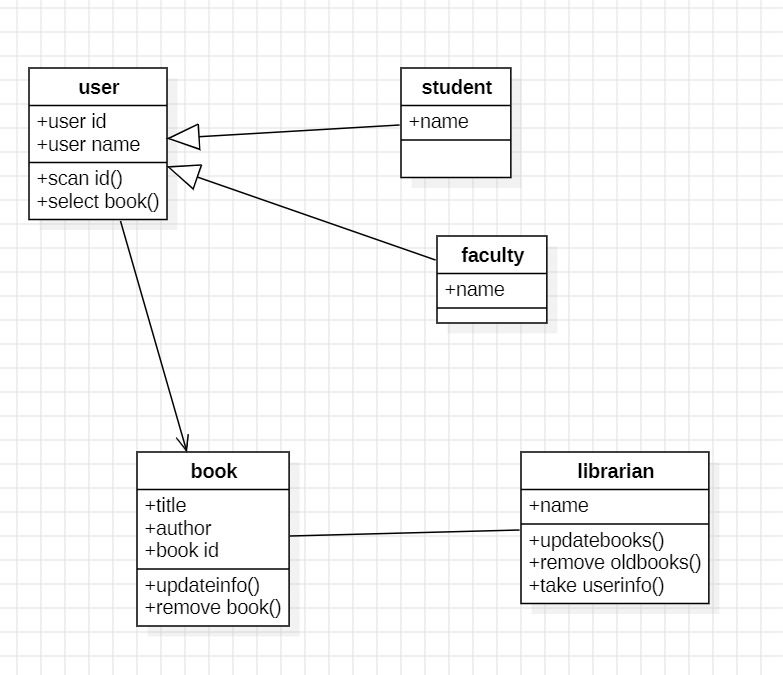


1. Object Diagram:

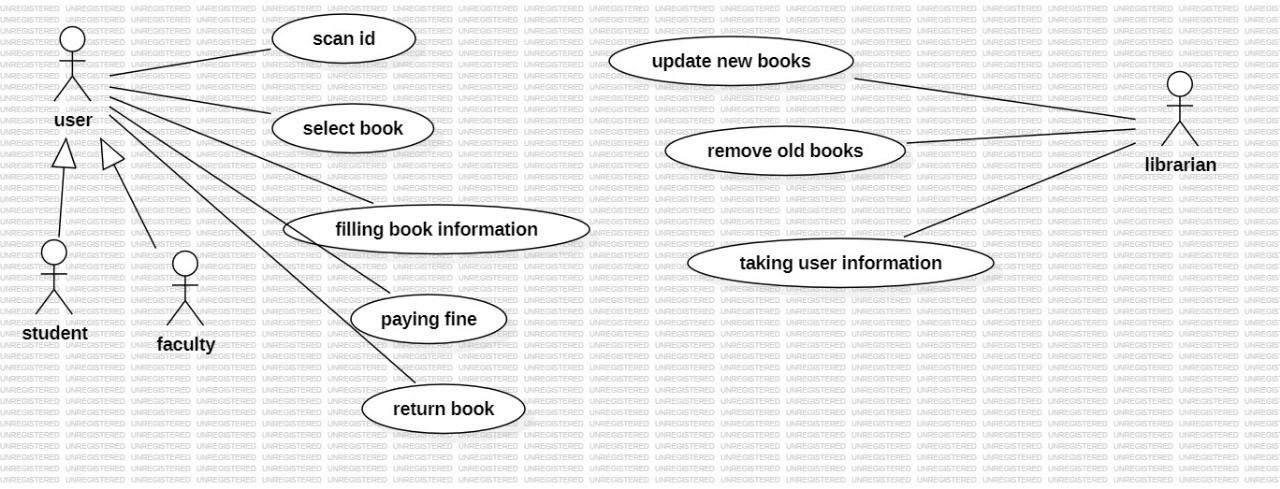


2.Library Management:

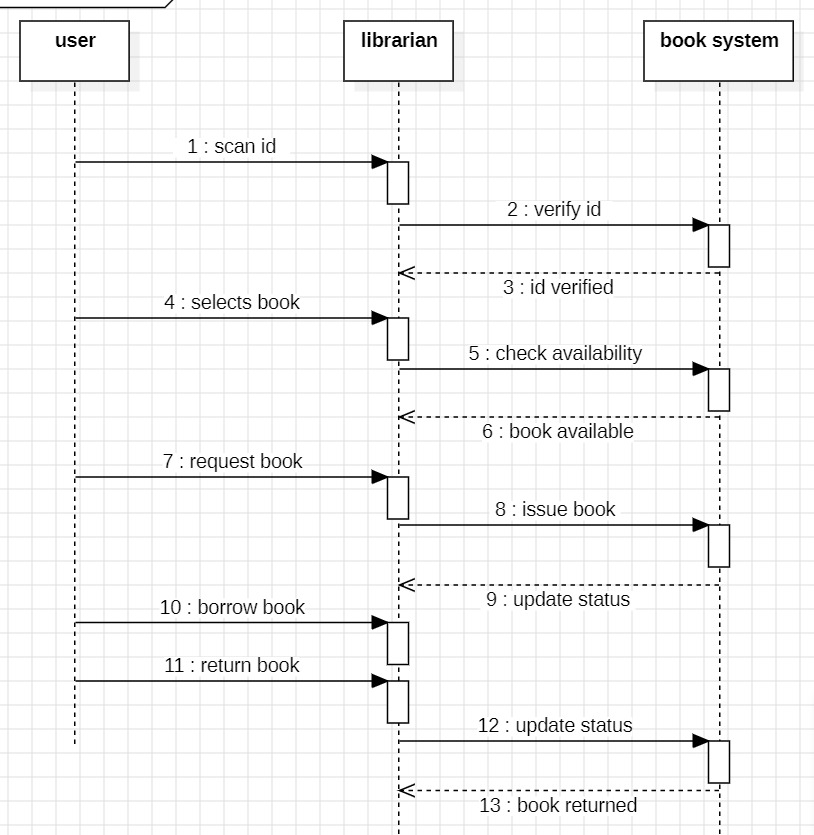
1. Class Diagram:



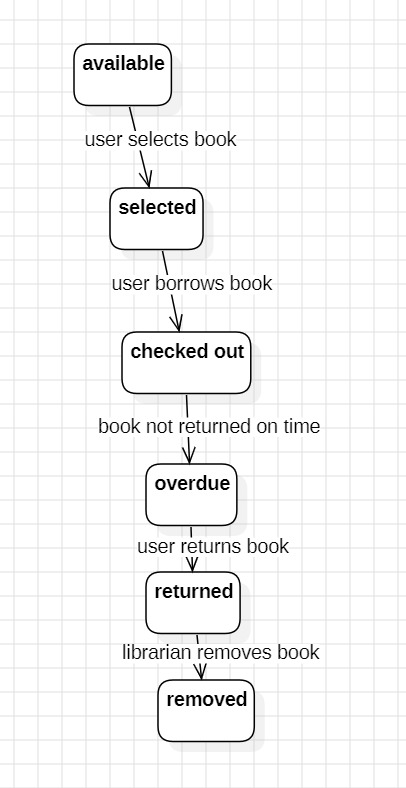
1. Use Case Diagram:



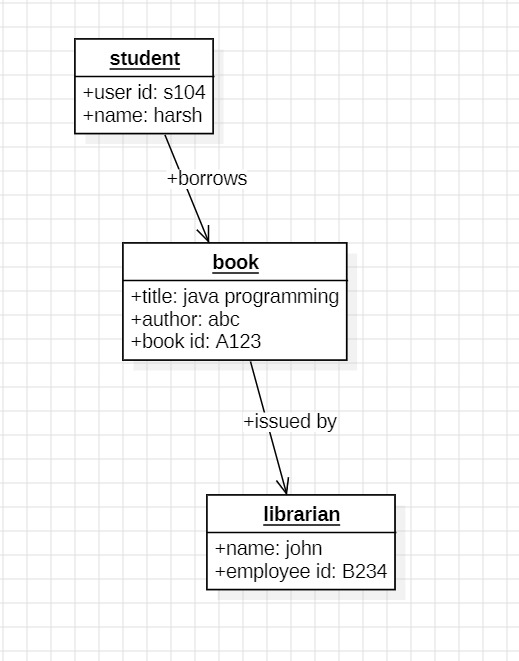
1. Sequence Diagram:



1. State Activity Diagram:



1. Object Diagram:



JAVA PROGRAMMES

1.Aim:Checkwhether a give number is odd or even.

Code:

import java.util.Scanner;

public class Main1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

if (number % 2 == 0) {

System.out.println(number + " is Even.");

} else {

System.out.println(number + " is Odd.");

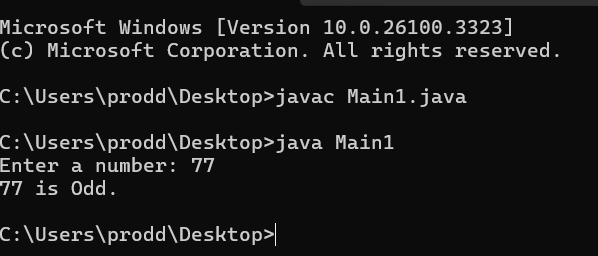
}

scanner.close();

}

}

Output:



2.Aim:Factorial of a given number.

Code:

import java.util.Scanner;

public class Main2 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

long factorial = 1;

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

factorial \*= i;

}

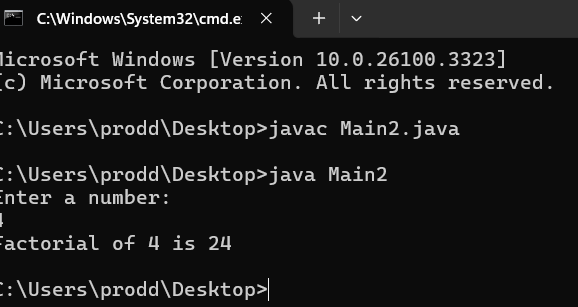
System.out.println("Factorial of " + number + " is " + factorial);

scanner.close();

}

}

Output:



3.AIM:Reverse a number.

Code:

import java.util.Scanner;

public class Main3 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

int reversed = 0;

while (number != 0) {

int digit = number % 10;

reversed = reversed \* 10 + digit;

number /= 10;

}

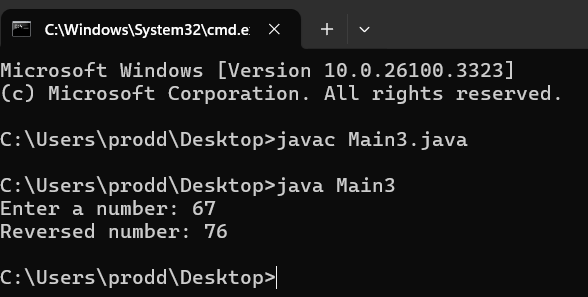
System.out.println("Reversed number: " + reversed);

scanner.close();

}

}

Output:



4.Aim:check whether a number is palindrome or not.

Code:

import java.util.Scanner;

public class Main4 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

int originalNumber = number;

int reversed = 0;

while (number != 0) {

int digit = number % 10;

reversed = reversed \* 10 + digit;

number /= 10;

}

if (originalNumber == reversed) {

System.out.println(originalNumber + " is a palindrome.");

} else {

System.out.println(originalNumber + " is not a palindrome.");

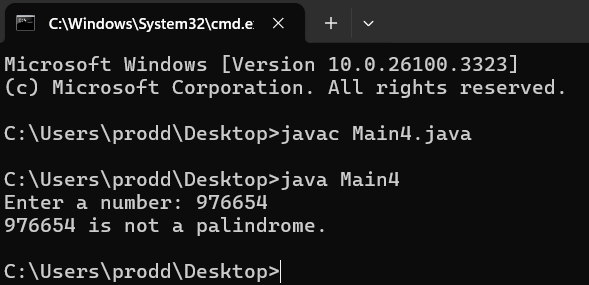
}

scanner.close();

}

}

Output:



5.Aim:Multiples of 2 upto 12.

Code:

import java.util.Scanner;

public class Main5 {

public static void main(String[] args) {

System.out.println("Multiples of 2 up to 12:");

for (int i = 1; i <= 12; i++) {

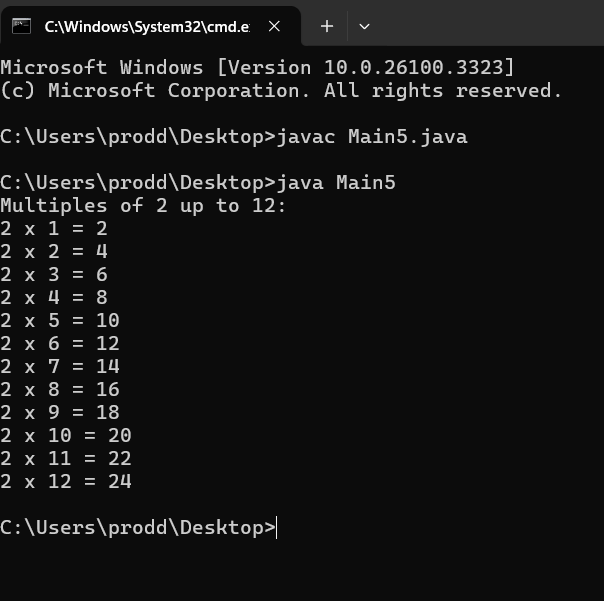
System.out.println("2 x " + i + " = " + (2 \* i));

}

}

}

Output:



6.Aim:Qudratic equation roots.

Code:

import java.util.Scanner;

public class Main6 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter coefficient a: ");

double a = scanner.nextDouble();

System.out.print("Enter coefficient b: ");

double b = scanner.nextDouble();

System.out.print("Enter coefficient c: ");

double c = scanner.nextDouble();

double discriminant = b \* b - 4 \* a \* c;

if (discriminant > 0) {

double root1 = (-b + Math.sqrt(discriminant)) / (2 \* a);

double root2 = (-b - Math.sqrt(discriminant)) / (2 \* a);

System.out.println("Roots are real and distinct: " + root1 + " and " + root2);

} else if (discriminant == 0) {

double root = -b / (2 \* a);

System.out.println("Roots are real and equal: " + root);

} else {

double realPart = -b / (2 \* a);

double imaginaryPart = Math.sqrt(-discriminant) / (2 \* a);

System.out.println("Roots are imaginary: " + realPart + " ± " + imaginaryPart + "i");

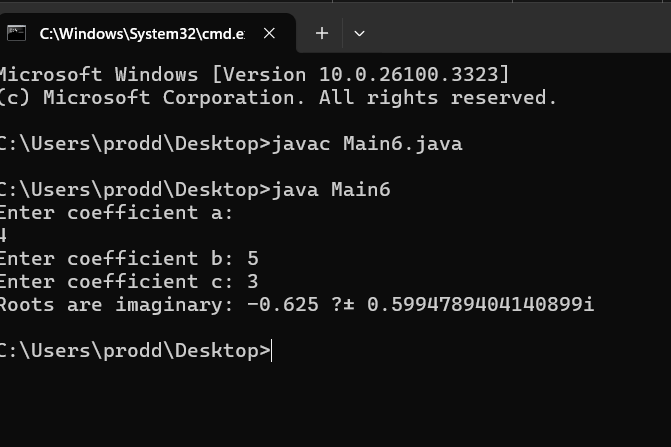
}

scanner.close();

}

}

Output:



7.Aim: To Find Hypotenuse of a Triangle

Code:

import java.util.Scanner;

public class Main7

{

public static void main(String[] args) {

double x;

double y;

double z;

Scanner scanner = new Scanner(System.in);

System.out.println("Enter side x: ");

x =scanner.nextDouble();

System.out.println("Enter side y: ");

y =scanner.nextDouble();

z = Math.sqrt((x\*x)+(y\*y));

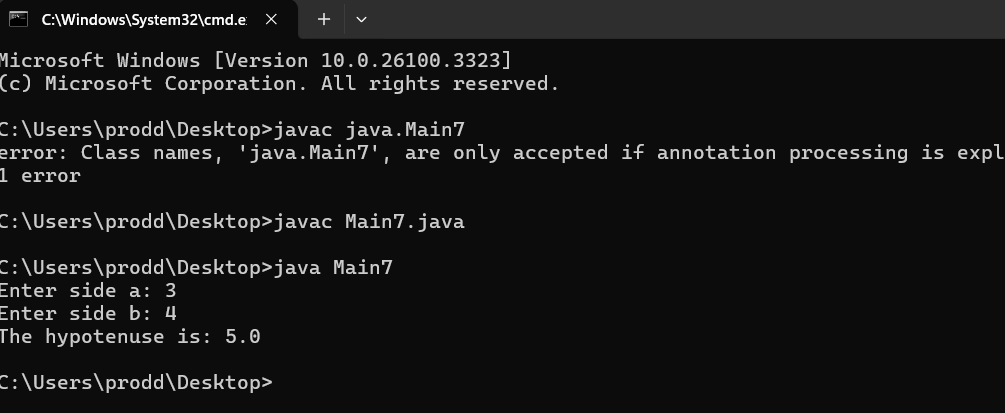
System.out.println("The hypotenuse is: "+z);

scanner.close();

}

}

Output:



8.AIM:Square root of a number.

Code:

import java.util.Scanner;

public class SquareRootCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double number = scanner.nextDouble();

if (number < 0) {

System.out.println("Square root of a negative number is not real.");

} else {

double squareRoot = Math.sqrt(number);

System.out.println("The square root of " + number + " is: " + squareRoot);

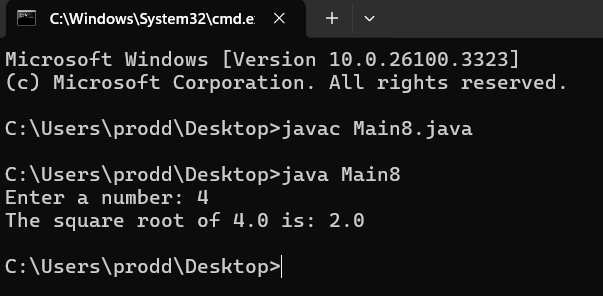
}

scanner.close();

}

}

Output:



9.Aim:Armstrong of a number.

Code:

import java.util.Scanner;

public class Main9 {

public static boolean isArmstrong(int num) {

int originalNum = num, sum = 0, digits = 0;

int temp = num;

while (temp > 0) {

temp /= 10;

digits++;

}

temp = num;

while (temp > 0) {

int digit = temp % 10;

sum += Math.pow(digit, digits);

temp /= 10;

}

return sum == originalNum;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

if (isArmstrong(number)) {

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

} else {

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

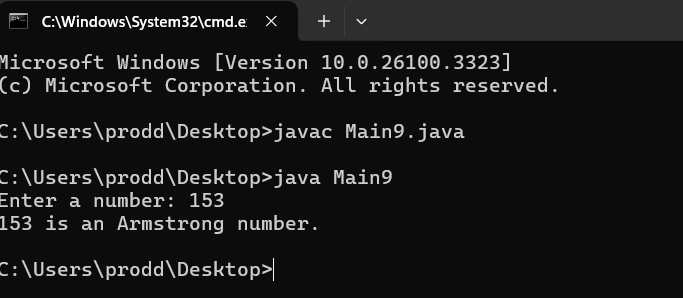
}

scanner.close();

}

}

Output:



10.Aim:. greastest of three numbers

Code:

import java.util.Scanner;

public class Main10 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int a = scanner.nextInt();

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

int b = scanner.nextInt();

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

int c = scanner.nextInt();

int greatest;

if (a >= b && a >= c) {

greatest = a;

} else if (b >= a && b >= c) {

greatest = b;

} else {

greatest = c;

}

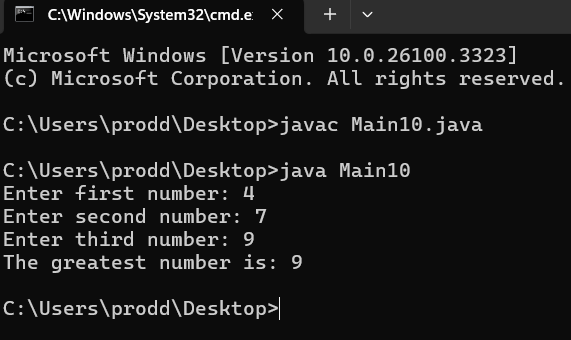
System.out.println("The greatest number is: " + greatest);

scanner.close();

}

}

Output:



**INHERITANCE**

**4)**

**i) Students Details:**

**CODE:**

class College {

String collegeName = "AMRITA";

String address = "CHENNAI, India";

void showCollegeDetails() {

System.out.println("College Name: " + collegeName);

System.out.println("Address: " + address);

}

}

class Student extends College {

String studentName;

int rollNumber;

Student(String studentName, int rollNumber) {

this.studentName = studentName;

this.rollNumber = rollNumber;

}

void showStudentDetails() {

System.out.println("Student Name: " + studentName);

System.out.println("Roll Number: " + rollNumber);

}

}

public class SingleInheritanceExample1 {

public static void main(String[] args) {

Student s1 = new Student("Rahul", 101);

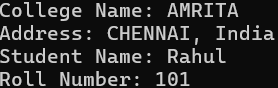
s1.showCollegeDetails();

s1.showStudentDetails();

}

}

**OUTPUT:**

****

**ii)Bank Details**

**CODE:**

class BankAccount {

String accountHolder;

double balance;

BankAccount(String accountHolder, double balance) {

this.accountHolder = accountHolder;

this.balance = balance;

}

void showBalance() {

System.out.println("Account Holder: " + accountHolder);

System.out.println("Balance: $" + balance);

}

}

class SavingsAccount extends BankAccount {

double interestRate = 5.0;

SavingsAccount(String accountHolder, double balance) {

super(accountHolder, balance);

}

void calculateInterest() {

double interest = (balance \* interestRate) / 100;

System.out.println("Annual Interest: $" + interest);

}

}

public class SingleInheritanceExample2 {

public static void main(String[] args) {

SavingsAccount acc1 = new SavingsAccount("John Doe", 5000);

acc1.showBalance();

acc1.calculateInterest();

}

}

**MULTILEVEL INHERITANCE**

**5.**

**i)General Details**

**CODE:**

class LivingBeing {

void breathe() {

System.out.println("Living beings breathe.");

}

}

class Human extends LivingBeing {

void speak() {

System.out.println("Humans can speak.");

}

}

class Student extends Human {

String name;

int studentID;

Student(String name, int studentID) {

this.name = name;

this.studentID = studentID;

}

void study() {

System.out.println(name + " is studying.");

}

void showDetails() {

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

System.out.println("Student ID: " + studentID);

}

}

public class MultilevelExample1 {

public static void main(String[] args) {

Student s1 = new Student("Rahul", 101);

s1.breathe();

s1.speak();

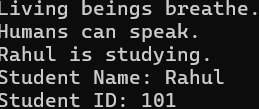
s1.study();

s1.showDetails();

}

}

**OUTPUT:**

****

**ii)Employee**

**CODE:**

class Person {

String name;

Person(String name) {

this.name = name;

}

void showPerson() {

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

}

}

class Employee extends Person {

int employeeID;

double salary;

Employee(String name, int employeeID, double salary) {

super(name);

this.employeeID = employeeID;

this.salary = salary;

}

void showEmployee() {

System.out.println("Employee ID: " + employeeID);

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

}

}

class Manager extends Employee {

String department;

Manager(String name, int employeeID, double salary, String department) {

super(name, employeeID, salary);

this.department = department;

}

void showManager() {

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

System.out.println("Role: Manager");

}

}

public class MultilevelExample2 {

public static void main(String[] args) {

Manager m1 = new Manager("Alice", 2001, 75000, "HR");

m1.showPerson();

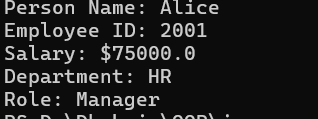
m1.showEmployee();

m1.showManager();

}

}

**OUTPUT:**

****

**HIERARCHICAL INHERITANCE PROGRAMS**

**6.**

**i)Code:**

class Vehicle {

    private String brand;

    private String model;

     public Vehicle(String brand, String model) {

        this.brand = brand;

        this.model = model;

    }

    public void start() {

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

    }

    public void stop() {

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

    }

    public String getBrand() {

        return brand;

    }

  public String getModel() {

        return model;

    }

}

class Car extends Vehicle {

    private int numberOfDoors ;

   public Car(String brand, String model, int numberOfDoors) {

        super(brand, model);

        this.numberOfDoors = numberOfDoors;

    }

  public void drive() {

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

    }

    public int getNumberOfDoors() {

        return numberOfDoors;

    }

}

class ElectricCar extends Car {

    private int batteryCapacity;

   public ElectricCar(String brand, String model, int numberOfDoors, int batteryCapacity) {

        super(brand, model, numberOfDoors);

        this.batteryCapacity = batteryCapacity;

    }

  public void charge() {

        System.out.println("Electric car is charging.");

    }

 public int getBatteryCapacity() {

        return batteryCapacity;

    }

}

class Truck extends Vehicle {

    private double cargoCapacity;

    public Truck(String brand, String model, double cargoCapacity) {

        super(brand, model);

        this.cargoCapacity = cargoCapacity;

    }

 public void loadCargo() {

        System.out.println("Truck is loading cargo.");

    }

  public double getCargoCapacity() {

        return cargoCapacity;

    }

}

public class Main {

    public static void main(String[] args) {

        Car car = new Car("Toyota", "Corolla", 4);

        car.start();

        car.drive();

        car.stop();

        System.out.println("Car doors: " + car.getNumberOfDoors());

        ElectricCar electricCar = new ElectricCar("Tesla", "Model S", 4, 100);

        electricCar.start();

        electricCar.drive();

        electricCar.charge();

        System.out.println("Battery capacity: " + electricCar.getBatteryCapacity());

        Truck truck = new Truck("Ford", "F-150", 2000.5);

        truck.start();

        truck.loadCargo();

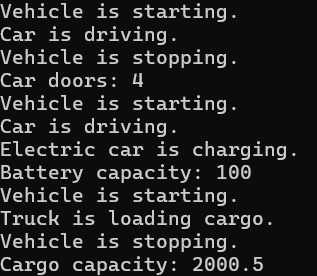
        truck.stop();

        System.out.println("Cargo capacity: " + truck.getCargoCapacity());

    }

}

**OUTPUT:**

****

**ii)**

**Code:**

class Person {

    private String name;

    private int age;

    public Person(String name, int age) {

        this.name = name;

        this.age = age;

    }

    public void displayDetails() {

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

    }

}

class Student extends Person {

    private int studentId;

    private String major;

    public Student(String name, int age, int studentId, String major) {

        super(name, age);

        this.studentId = studentId;

        this.major = major;

    }

    public void study() {

        System.out.println("Student is studying " + major);

    }

  public void displayDetails() {

        super.displayDetails();

        System.out.println("Student ID: " + studentId + ", Major: " + major);

    }

}

class Professor extends Person {

    private String department;

    private String researchArea;

  public Professor(String name, int age, String department, String researchArea) {

        super(name, age);

        this.department = department;

        this.researchArea = researchArea;

    }

 public void teach() {

        System.out.println("Professor is teaching in " + department);

    }

   public void displayDetails() {

        super.displayDetails();

        System.out.println("Department: " + department + ", Research Area: " + researchArea);

    }

}

class TeachingAssistant extends Student {

    private String course;

    public TeachingAssistant(String name, int age, int studentId, String major, String course) {

        super(name, age, studentId, major);

        this.course = course;

    }

    public void assist() {

        System.out.println("Teaching assistant is assisting in " + course);

    }

public void displayDetails() {

        super.displayDetails();

        System.out.println("Course: " + course);

    }

}

public class Main2 {

    public static void main(String[] args) {

        Student student = new Student("Alice", 20, 101, "Computer Science");

        student.displayDetails();

        student.study();

         Professor professor = new Professor("Dr. Smith", 45, "Computer Science", "AI");

        professor.displayDetails();

        professor.teach();

        TeachingAssistant ta = new TeachingAssistant("Bob", 25, 102, "Mathematics", "Calculus");

        ta.displayDetails();

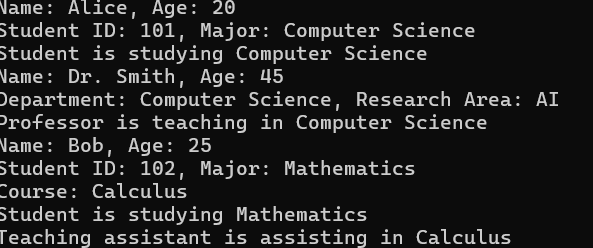
        ta.study();

        ta.assist();

    }

}

**OUTPUT:**

****

**HYBRID INHERITANCE PROGRAMS**

**7.**

**i)**

**CODE:**

class Person {

    String name;

    Person(String name) {

        this.name = name;

    }

    void showDetails() {

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

    }

}

class Student extends Person {

    int studentID;

    Student(String name, int studentID) {

        super(name);

        this.studentID = studentID;

    }

    void study() {

        System.out.println(name + " is studying.");

    }

}

class Teacher extends Person {

    String subject;

    Teacher(String name, String subject) {

        super(name);

        this.subject = subject;

    }

    void teach() {

        System.out.println(name + " is teaching " + subject + ".");

    }

}

interface Assistant {

    void assist();

}

class TeachingAssistant extends Student implements Assistant {

    TeachingAssistant(String name, int studentID) {

        super(name, studentID);

    }

    public void assist() {

        System.out.println(name + " is assisting in a lab session.");

    }

}

public class HybridInheritanceExample1 {

    public static void main(String[] args) {

        TeachingAssistant ta = new TeachingAssistant("Alex", 101);

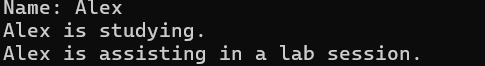
        ta.showDetails();

        ta.study();

        ta.assist();

    }

}

**OUTPUT:  
**

**ii)**

**CODE:**

class Vehicle {

    void startEngine() {

        System.out.println("Vehicle engine started.");

    }

}

class Car extends Vehicle {

    void drive() {

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

    }

}

class Boat extends Vehicle {

    void sail() {

        System.out.println("Boat is sailing.");

    }

}

interface Amphibious {

    void switchMode();

}

class AmphibiousCar extends Car implements Amphibious {

    public void switchMode() {

        System.out.println("Switching between land and water mode.");

    }

    void sail() {

        System.out.println("Amphibious car is sailing on water.");

    }

}

public class HybridInheritanceExample2 {

    public static void main(String[] args) {

        AmphibiousCar ac = new AmphibiousCar();

        ac.startEngine();

        ac.drive();

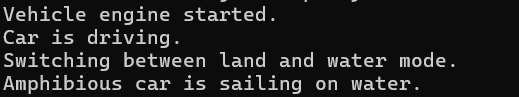
        ac.switchMode();

        ac.sail();

    }

}

**OUTPUT:**

****

**POLYMORPHISM**

**CONSTRUCTOR PROGRAMS**

**8**

**i)**

**CODE:**

class Book {

    String title;

    int pages;

    Book(String t, int p) {

        title = t;

        pages = p;

    }

    Book(Book b) {

        title = b.title;

        pages = b.pages;

    }

    void display() {

        System.out.println("Book: " + title + ", Pages: " + pages);

    }

}

 public class ConstructorExample {

    public static void main(String[] args) {

        Book b1 = new Book("Java Programming", 500);

        Book b2 = new Book(b1);

        b1.display();

        b2.display();

    }

}

**OUTPUT:**

****

**CONSTRUCTOR OVERLOADING PROGRAMS**

**9.**

**i)**

**CODE:**

class Employee {

    String name;

    int age;

    double salary;

    Employee() {

        name = "Unknown";

        age = 18;

        salary = 30000;

    }

    Employee(String n, int a) {

        name = n;

        age = a;

        salary = 40000;

    }

    Employee(String n, int a, double s) {

        name = n;

        age = a;

        salary = s;

    }

    void display() {

        System.out.println("Name: " + name + ", Age: " + age + ", Salary: $" + salary);

    }

}

public class ConstructorOverloadingExample {

    public static void main(String[] args) {

        Employee e1 = new Employee();

        Employee e2 = new Employee("John", 25);

        Employee e3 = new Employee("Alice", 30, 60000);

        e1.display();

        e2.display();

        e3.display();

    }

}

**OUTPUT:**

****

**METHOD OVERLOADING PROGRAMS**

**10.**

**i)**

**CODE:**

class Employee {

    private String name;

    private int id;

    private double salary;

    void setDetails(String name, int id) {

        this.name = name;

        this.id = id;

    }

    void setDetails(String name, int id, double salary) {

        this.name = name;

        this.id = id;

        this.salary = salary;

    }

    void setDetails(String name) {

        this.name = name;

    }

    void displayDetails() {

        System.out.println("Name: " + name + ", ID: " + id + ", Salary: " + salary);

    }

}

public class Main {

    public static void main(String[] args) {

        Employee emp1 = new Employee();

        emp1.setDetails("Alice", 101); // Calls first method

        emp1.displayDetails();

        Employee emp2 = new Employee();

        emp2.setDetails("Bob", 102, 50000.0); // Calls second method

        emp2.displayDetails();

        Employee emp3 = new Employee();

        emp3.setDetails("Charlie"); // Calls third method

        emp3.displayDetails();

    }

} **OUTPUT:**



**ii)**

**CODE:**

class Shape {

    void draw(double radius) {

        System.out.println("Drawing a circle with radius: " + radius);

    }

 void draw(double length, double width) {

        System.out.println("Drawing a rectangle with length: " + length + " and width: " + width);

    }

    void draw(double side1, double side2, double side3) {

        System.out.println("Drawing a triangle with sides: " + side1 + ", " + side2 + ", " + side3);

    }

}

public class  Main2{

    public static void main(String[] args) {

        Shape shape = new Shape();

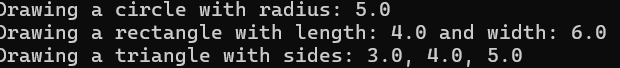
        shape.draw(5.0);

        shape.draw(4.0, 6.0);

        shape.draw(3.0, 4.0, 5.0);     }

}

**OUTPUT:**

****

**METHOD OVERRIDING PROGRAMS**

**11**

**i)**

**CODE:**

class Vehicle {

void speed() {

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

}

}

class Car extends Vehicle {

void speed() {

System.out.println("Car moves at 80 km/h");

}

}

class Bike extends Vehicle {

void speed() {

System.out.println("Bike moves at 60 km/h");

}

}

public class Main {

public static void main(String[] args) {

Vehicle v;

v = new Car();

v.speed();

v = new Bike();

v.speed();

}

} **OUTPUT:**

****

**ii)**

**CODE:**

class Bank {

    double getInterestRate() {

        return 0;

    }

}

class SBI extends Bank {

    double getInterestRate() {

        return 5.5;

    }

}

class ICICI extends Bank {

    double getInterestRate() {

        return 6.7;

    }

}

class HDFC extends Bank {

    double getInterestRate() {

        return 7.2;

    }

}

public class Main2 {

    public static void main(String[] args) {

        Bank b;

        b = new SBI();

        System.out.println("SBI Interest Rate: " + b.getInterestRate() + "%");

        b = new ICICI();

        System.out.println("ICICI Interest Rate: " + b.getInterestRate() + "%");

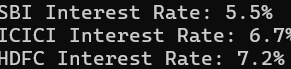
        b = new HDFC();

        System.out.println("HDFC Interest Rate: " + b.getInterestRate() + "%");

    }

}

**OUTPUT:**

****

**INTERFACE PROGRAMS**

**12**

**i)**

**CODE:**

interface Shape{

void getArea();

}

class Rectangle implements Shape{

    double lenght,breadth;

    Rectangle(double lenght, double breadth) {

        this.lenght = lenght;

        this.breadth = breadth;

    }

    public void getArea(){

       System.out.println("The area of rectangle "+(lenght\*breadth));

    }

}

class Circle implements Shape{

    double radius;

    Circle(double radius){

        this.radius=radius;

    }

    public void getArea(){

        System.out.println("The are of circle is "+(3.14\*(radius\*radius)));

    }

}

public class Infferface1 {

    public static void main(String[] args) {

        Rectangle r1=new Rectangle(12, 12);

        r1.getArea();

        Circle c1=new Circle(4);

        c1.getArea();

    }

}

**OUTPUT:**

****

**ii)**

**CODE:**

interface Camara{

    void takePhoto();

}

interface MusicPlayer{

    void canPlayMusic();

}

class SmartPhone implements Camara,MusicPlayer{

    public void takePhoto(){

        System.out.println("SmartPhone can take photo");

    }

    public void canPlayMusic(){

        System.out.println("SmartPhone can play music");

    }

}

public class Infferface2 {

    public static void main(String[] args) {

        SmartPhone s1=new SmartPhone();

        s1.canPlayMusic();

        s1.takePhoto();

    }

}

**OUTPUT:**

****

**iii)**

**CODE:**

interface Payment {

    void initiatePayment(double amount);

    void getPaymentStatus();

}

class CreditCard implements Payment {

    public void initiatePayment(double amount) {

        System.out.println("Processing Credit Card payment of $" + amount);

    }

    public void getPaymentStatus() {

        System.out.println("Credit Card payment successful!");

    }

}

class PayPal implements Payment {

    public void initiatePayment(double amount) {

        System.out.println("Processing PayPal payment of $" + amount);

    }

    public void getPaymentStatus() {

        System.out.println("PayPal payment successful!");

    }

}

public class Infferface3 {

    public static void main(String[] args) {

        Payment creditCardPayment = new CreditCard();

        Payment paypalPayment = new PayPal();

        creditCardPayment.initiatePayment(50.0);

        creditCardPayment.getPaymentStatus();

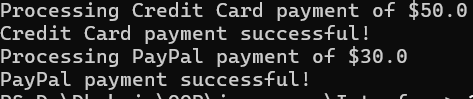
        paypalPayment.initiatePayment(30.0);

        paypalPayment.getPaymentStatus();

    }

}

**OUTPUT:**

****

**iv)**

**CODE:**

interface Vehicle{

    void start();

    void stop();

}

interface ElectricVehicle{

    void chargeBattery();

}

interface FuelVehicle{

    void refuel();

}

class HybridCar implements Vehicle,ElectricVehicle,FuelVehicle{

    public void start(){

        System.out.println("Vehicle Started ");

    }

    public void stop(){

        System.out.println("Vehicle Stoped ");

    }

    public void chargeBattery(){

        System.out.println("Vehicle charging battery ");

    }

    public void refuel(){

        System.out.println("Vehicle refilling the fuel tank ");

    }

}

public class Infferface4 {

    public static void main(String[] args) {

        HybridCar v1=new HybridCar();

        v1.chargeBattery();

        v1.start();

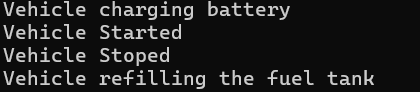
        v1.stop();

        v1.refuel();

    }

}

**OUTPUT:**

****

**ABSTRACT CLASS PROGRAMS**

**13**

**i)**

**Code:**

public class Main{

public static void main(String[] args){

Car c1=new Lam();

c1.carName("lambogini");

c1.carSpeed(122,"lambogini");

}

}

abstract class Car{

abstract void carName(String name);

abstract void carSpeed(int speed,String name);

abstract void Mileage(int fuel,double mileage);

}

class Lam extends Car{

void carName(String name){

System.out.println("Your Car name is "+name);

}

void carSpeed(int speed,String name){

System.out.println(name+" can travel at the speed of "+speed+" km/h");

}

void Mileage(int fuel,double mileage){

System.out.println("lam "+"can travel range of "+(fuel\*mileage));

}

}

**Output:**

****

**ii)**

**Code:**

abstract class Vehicle{

abstract void start(String name);

abstract void stop(String name);

}

class Car extends Vehicle{

void start(String name){

System.out.println( name+" is Starting");

}

void stop(String name){

System.out.println(name+" is stopping");

}

}

class Bike extends Vehicle{

void start(String name){

System.out.println( name+" is Starting");

}

void stop(String name){

System.out.println(name+" is stopping");

}

}

public class Main2{

public static void main(String[] args){

Car v1=new Car();

 v1.start("Lambo");

 v1.stop("lambo");

Bike v2=new Bike();

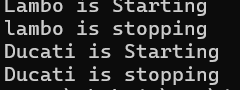
 v2.start("Ducati");

 v2.stop("Ducati");

}

}

**Output:**

****

**iii)**

**Code:**

abstract class Shape {

// Abstract method to calculate the area

abstract double calculateArea();

}

class Square extends Shape {

private double side;

public Square(double side) {

this.side = side;

}

@Override

double calculateArea() {

return side \* side; // Area of square: side²

}

}

public class Main3 {

public static void main(String[] args) {

Shape square = new Square(4.0);

System.out.println("Area of the square: " + square.calculateArea());

}

}

**Output:**

****

**iv)**

**Code:**

abstract class Shape2D{

abstract void draw();

abstract void resize();

}

class Rectangle extends Shape2D{

void draw(){

System.out.println("you are Drawing Rectangle");

}

void resize(){

System.out.println("you can resize the lenght and breadth of rectangle");

}

}

class Circle extends Shape2D{

void draw(){

System.out.println("you are Drawing Circle");

}

void resize(){

System.out.println("you can resize the Radius of circle ");

}

}

public class Main4{

public static void main(String[] args){

Shape2D s1=new Rectangle();

s1.draw();

s1.resize();

Shape2D s2=new Circle();

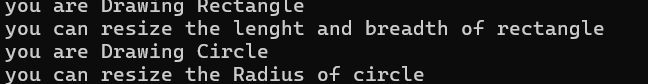
s2.draw();

s2.resize();

}

}

**Output:**

****

**ENCAPSULATION PROGRAMS**

**14**

**i)**

**Code:**

class Patient {

private String name;

private double weight; // in kg

private double height; // in meters

public void setName(String name) {

this.name = name;

}

public void setWeight(double weight) {

this.weight = weight;

}

public void setHeight(double height) {

this.height = height;

}

public String getName() {

return name;

}

public double calculateBMI() {

return weight / (height \* height);

}

public String getHealthStatus() {

double bmi = calculateBMI();

if (bmi < 18.5) return "Underweight";

else if (bmi < 24.9) return "Normal weight";

else if (bmi < 29.9) return "Overweight";

else return "Obese";

}

}

public class PatientTest {

public static void main(String[] args) {

Patient p = new Patient();

p.setName("John Doe");

p.setWeight(70);

p.setHeight(1.75);

System.out.println("Patient: " + p.getName());

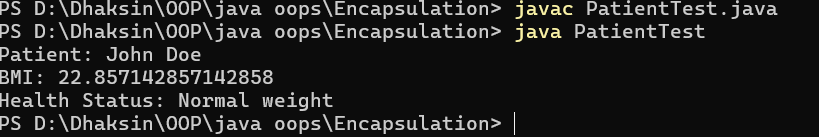
System.out.println("BMI: " + p.calculateBMI());

System.out.println("Health Status: " + p.getHealthStatus());

}

}

**OUTPUT:**

****

**ii)**

**Code:**

class Product {

private String productName;

private double price;

private int stock;

public void setProductName(String productName) {

this.productName = productName;

}

public void setPrice(double price) {

this.price = price > 0 ? price : 0;

}

public void setStock(int stock) {

this.stock = stock >= 0 ? stock : 0;

}

public String getProductName() {

return productName;

}

public double getPrice() {

return price;

}

public boolean isAvailable() {

return stock > 0;

}

}

public class ProductTest {

public static void main(String[] args) {

Product product = new Product();

product.setProductName("Laptop");

product.setPrice(1200);

product.setStock(5);

System.out.println("Product: " + product.getProductName());

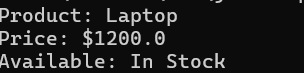
System.out.println("Price: $" + product.getPrice());

System.out.println("Available: " + (product.isAvailable() ? "In Stock" : "Out of Stock"));

}

}

**OUTPUT:**

****

**iii)**

**Code:**

class House {

private String location;

private double area; // in square meters

private double pricePerSquareMeter;

public void setLocation(String location) {

this.location = location;

}

public void setArea(double area) {

this.area = area > 0 ? area : 0;

}

public void setPricePerSquareMeter(double price) {

this.pricePerSquareMeter = price > 0 ? price : 0;

}

public String getLocation() {

return location;

}

public double getTotalPrice() {

return area \* pricePerSquareMeter;

}

}

public class HouseTest {

public static void main(String[] args) {

House house = new House();

house.setLocation("Downtown");

house.setArea(200);

house.setPricePerSquareMeter(1500);

System.out.println("House Location: " + house.getLocation());

System.out.println("Total Price: $" + house.getTotalPrice());

}

}

**OUTPUT:**

****

**iv)**

**Code:**

class GameCharacter {

private String name;

private int level;

private int strength;

private int intelligence;

public void setName(String name) {

this.name = name;

}

public void setLevel(int level) {

this.level = Math.max(level, 1);

}

public void setStrength(int strength) {

this.strength = Math.max(strength, 1);

}

public void setIntelligence(int intelligence) {

this.intelligence = Math.max(intelligence, 1);

}

public String getName() {

return name;

}

public int getPowerLevel() {

return (strength \* 2 + intelligence \* 3) \* level;

}

}

public class GameTest {

public static void main(String[] args) {

GameCharacter hero = new GameCharacter();

hero.setName("Ketta Paiyyan");

hero.setLevel(10);

hero.setStrength(20);

hero.setIntelligence(15);

System.out.println("Hero: " + hero.getName());

System.out.println("Power Level: " + hero.getPowerLevel());

}

}

**OUTPUT:**

****

**PACKAGES PROGRAMS**

**15**

**i)**

**code:**

import java.awt.\*;

import java.awt.event.\*;

 public class SimpleAWTApp {

    SimpleAWTApp() {

        Frame frame = new Frame("AWT Example");

        Button button = new Button("Click Me!");

        button.setBounds(50, 100, 80, 30);

        frame.add(button);

        frame.setSize(300, 200);

        frame.setLayout(null);

        frame.setVisible(true);

        frame.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {

                frame.dispose();

            }

        });

    }

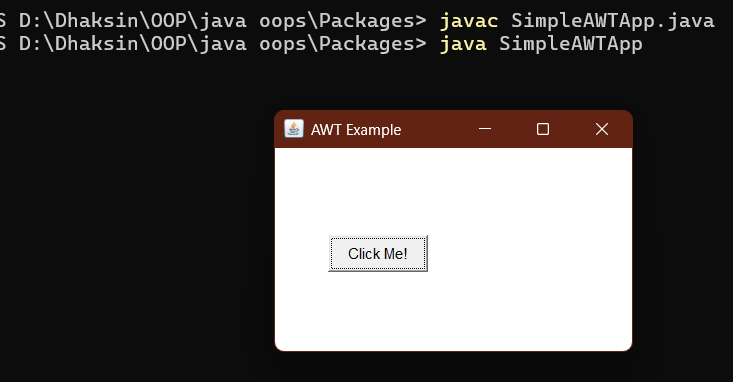
    public static void main(String[] args) {

        new SimpleAWTApp();

    }

}

**output:**

****

**ii)**

**code:**

#package

package mypackage;

public class basiccalc {

    public int add(int a, int b) {

        return a + b;

    }

    public int sub(int a, int b) {

        return a - b;

    }

    public int mul(int a, int b) {

        return a \* b;

    }

    public int div(int a, int b) {

        if (b == 0) {

            System.out.println("Denominator with 0 is not defined or infinite");

            return 0;

        }

        else {

            return a / b;

        }

    }

}

import mypackage.basiccalc;

public class calc {

    public static void main(String[] args) {

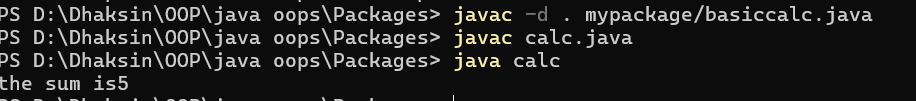
        basiccalc c1=new basiccalc();

        System.out.println("the sum is"+c1.add(2,3));

    }

}

**output:**

****

**iii)**

**code:**

#package

package mypackage;

public class geometery {

    public double areaRectangle(double lenght,double breath){

        return lenght\*breath;

    }

    public double areaCircle(double radius){

        return 3.14\*(radius\*radius);

    }

    public double areaTriangle(double lenght,double height){

        return 0.5\*(lenght+height);

    }

}

import java.lang.\*;

import java.util.Scanner;

import mypackage.geometery;

public class geometerycal {

    public static void main(String[] args) {

        Scanner input=new Scanner(System.in);

        geometery a1=new geometery();

        System.out.println("Enter the lenght of Rectangle");

        double lenght=input.nextDouble();

        System.out.println("Enter the Breath of Rectangle");

        double breath=input.nextDouble();

        System.out.println("Area of Rectangle");

       System.out.println(a1.areaRectangle(lenght, breath));

        System.out.println("Enter the radius of Circle");

        double radius=input.nextDouble();

        System.out.println("Area of Circle");

        System.out.println(a1.areaCircle(radius));

        System.out.println("Enter lenght of triangle");

        double tlength=input.nextDouble();

        System.out.println("Enter height of triangle");

        double height=input.nextDouble();

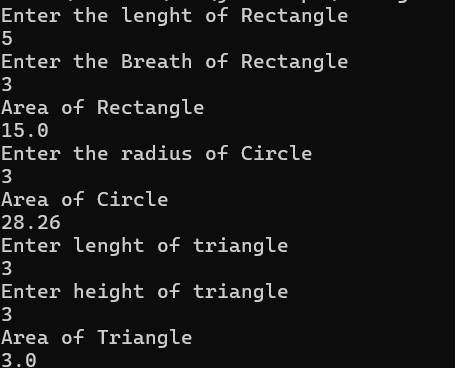
        System.out.println("Area of Triangle");

        System.out.println(a1.areaTriangle(tlength, height));

    }

}

**output:**

****

**iv)**

**code:**

import java.util.\*;

import java.time.\*;

import java.io.\*;

public class EmployeePayroll {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        try {

            System.out.print("Enter number of employees: ");

            int numEmployees = scanner.nextInt();

            scanner.nextLine();

            ArrayList<Employee> employees = new ArrayList<>();

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

                System.out.println("\nEmployee " + (i + 1) + ":");

                System.out.print("Enter name: ");

                String name = scanner.nextLine();

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

                double salary = scanner.nextDouble();

                System.out.print("Enter joining year: ");

                int joiningYear = scanner.nextInt();

                scanner.nextLine();

                employees.add(new Employee(name, salary, joiningYear));

            }

            FileWriter writer = new FileWriter("EmployeePayroll.txt");

            System.out.println("\n--- Employee Payroll ---");

            writer.write("--- Employee Payroll ---\n");

            for (Employee emp : employees) {

                emp.calculateBonus();

                emp.displayDetails();

                writer.write(emp.getDetailsForFile());

            }

            writer.close();

            System.out.println("\nPayroll saved to 'EmployeePayroll.txt'");

        } catch (IOException e) {

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

        } catch (InputMismatchException e) {

            System.out.println("Invalid input. Please enter numbers correctly.");

        }

        scanner.close();

    }

}

class Employee {

    private String name;

    private double salary;

    private int joiningYear;

    private double bonus;

    public Employee(String name, double salary, int joiningYear) {

        this.name = name;

        this.salary = salary;

        this.joiningYear = joiningYear;

    }

    public void calculateBonus() {

        int currentYear = LocalDate.now().getYear();

        int yearsWorked = currentYear - joiningYear;

        if (yearsWorked >= 5) {

            bonus = salary \* 0.1;

        } else {

            bonus = salary \* 0.05;

        }

    }

    public void displayDetails() {

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

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

        System.out.println("Joining Year: " + joiningYear);

        System.out.println("Bonus: ₹" + bonus);

        System.out.println("Total Salary: ₹" + (salary + bonus));

        System.out.println();

    }

    public String getDetailsForFile() {

        return "Employee: " + name + "\n" +

               "Salary: ₹" + salary + "\n" +

               "Joining Year: " + joiningYear + "\n" +

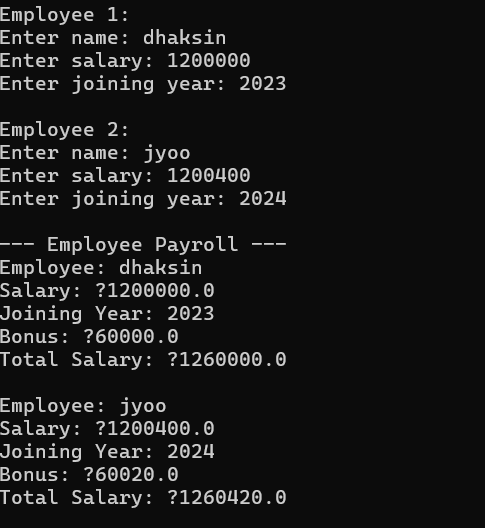
               "Bonus: ₹" + bonus + "\n" +

               "Total Salary: ₹" + (salary + bonus) + "\n\n";

    }

}

**output:**

****

**EXCEPTION HANDLING PROGRAMS**

**16.**

**i)Division my 0**

**Code:**

public class main{

public static void main(String[] args){

try{

int result=10/0;

}

catch(ArithmeticException e){

System.out.println("Caught exception "+e);

}

finally {

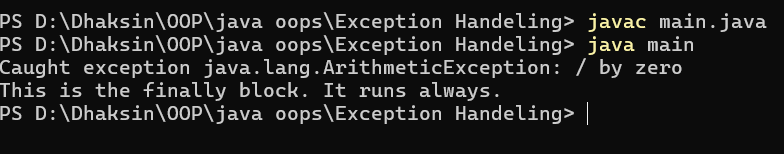
System.out.println("This is the finally block. It runs always.");

}

}

}

**Output:**

****

**ii)Age Exception with throws**

**Code:**

class ageLimitException extends Exception{

ageLimitException(String message){

super(message);

}}

public class main2{

public static void main(String[] args){

try{

ageLimit(17);

}

catch(ageLimitException e){

System.out.println("Caught error "+e);

}

}

public static void ageLimit(int age) throws ageLimitException{

if (age<18){

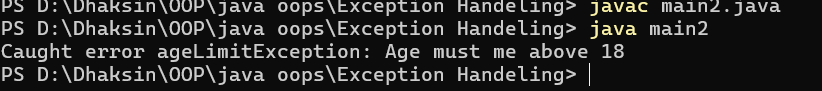
throw new ageLimitException("Age must me above 18 ");

}

}

}

**Output:**

****

**iii)Age Exception**

**Code:**

public class main3{

public static void main(String[] args){

try{

ageLimit(11);

}

catch(IllegalArgumentException e){

System.out.println("Caught Error "+e);

}

}

public static void ageLimit(int age){

if (age<18){

throw new IllegalArgumentException("Not elligible to vote");

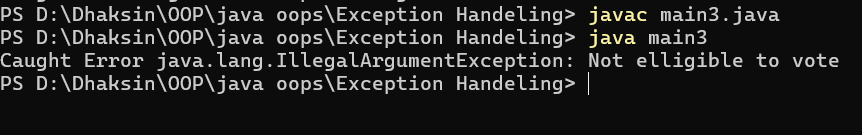
}

System.out.println("Successfully accessed");

}

}

**Output:**

****

**iv)Filemissing error**

**Code**

import java.io.\*;

class Filemissing extends Exception{

Filemissing(String message){

super(message);

}

}

public class main4{

public static void main(String[] args) throws Filemissing, IOException {

String filename="example.txt";

try{

readfile("example.txt");

}

catch(FileNotFoundException e){

throw new Filemissing("File not found: " + filename);

}

}

public static void readfile(String filename) throws Filemissing,IOException {

FileReader file = new FileReader(filename);

BufferedReader reader= new BufferedReader(file);

String line=reader.readLine();

while(line!=null){

System.out.println(line);

line=reader.readLine();

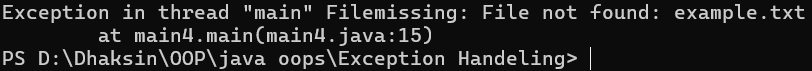
}

reader.close();

}

}

**Output:**



**FILE HANDLING PROGRAMS**

**17.**

**i)Reading a File**

**code:**

import java.io.BufferedReader;

import java.io.FileReader;

public class readfile {

    public static void main(String[] args) {

        try {

            FileReader r=new FileReader("output.txt");

            BufferedReader v=new BufferedReader(r);

            String e=v.readLine();

            while(e!=null){

                System.out.println(e);

                e=v.readLine();

            }

            v.close();

        } catch (Exception e) {

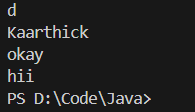
            System.out.println("error");

        }

    }

  }

**Screen Shot:**



**ii)Writing to a file:**

**code:**

import java.io.BufferedWriter;

import java.io.FileWriter;

public class filehandeling {

    public static void main(String[] args) {

        try{

            FileWriter file=new FileWriter("output.txt",true);

            BufferedWriter b=new BufferedWriter(file);

            b.write("dhaksin");

            b.newLine();

            b.write("hello");

            b.close();

        }

        catch(Exception e){

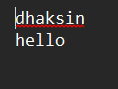
            System.out.println("error");

    }

    }

}

**Screen Shot:**

****

**iii)Reading and Writing to a file with word count:**

**code:**

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class writeNread {

    public static void main(String[] args) {

        try (FileReader file = new FileReader("D:/Code/Java/input.txt");

             BufferedReader v = new BufferedReader(file);

             FileWriter A = new FileWriter("D://Code//Java//output.txt/");

             BufferedWriter f = new BufferedWriter(A)) {

            String line=v.readLine();

            int count=0;

            while (line!= null) {

                System.out.println(line);

                count+=line.length();

                f.write(line);

                f.newLine();

                line=v.readLine();

            }

            System.out.println(count);

        } catch (IOException e) {

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

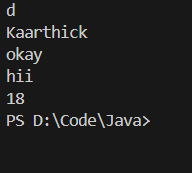
        }

    }

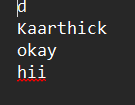
}

**Screen Shot:**

**#Reading**

****

**#writing**

****

**iv)writing to file multiple try catch:**

**code:**

import java.io.\*;

public class FileHandlingExample {

    public static void main(String[] args) {

             String fileName = "example.txt";

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

            writer.write("Hello, this is a test file!\n");

            writer.write("This is line 2.");

            System.out.println("Successfully wrote to the file.");

        } catch (IOException e) {

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

        }

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

            System.out.println("\nFile contents:");

            String line;

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

                System.out.println(line);

            }

        } catch (IOException e) {

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

        }

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

            writer.write("\nThis is an appended line.");

            System.out.println("\nSuccessfully appended to the file.");

        } catch (IOException e) {

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

        }

        try (FileInputStream fis = new FileInputStream(fileName)) {

            System.out.println("\nReading using FileInputStream:");

            int content;

            while ((content = fis.read()) != -1) {

                System.out.print((char) content);

            }

        } catch (IOException e) {

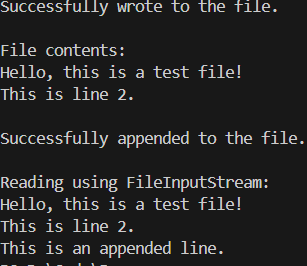
            System.out.println("Error reading file: " + e.getMessage());

        }

    }

}

**Screen Shot:**

****