



M Cynthia Shree

CH.SC.U4CSE24110

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.U4CSE24112 – M Cynthia Shree 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 DIAGRAMS		
1.	LIBRARY MANAGEMENT SYSTEM	
	A) USE CASE DIAGRAM	6
	B) CLASS DIAGRAM	7
	C) SEQUENCE DIAGRAM	8
	D) STATE ACTIVITY DIAGRAM	9
	E) OBJECT DIAGRAM	10
2.	ONLINE FOOD ORDERING SYSTEM	
	A) USE CASE DIAGRAM	11
	B) CLASS DIAGRAM	12
	C) SEQUENCE DIAGRAM	13
	D) STATE ACTIVITY DIAGRAM	14
	E) OBJECT DIAGRAM	15
3.	BASIC JAVA PROGRAMS	
	A) BASIC CALCULATOR	16
	B) PRIME NUMBER CHECK	17
	C) FIBONACCI SERIES	18
	D) FACTORIAL OF A NUMBER	19
	E) PALINDROME	20
	F) EVEN OR ODD NUMBER	21
	G) REVERSE OF A NUMBER	21
	H) SUM OF DIGITS	22
	I) PRINT MULTIPLICATION TABLE	23
	J) MAXIMUM OF TWO NUMBERS	24
INHERITANCE		
4.	SINGLE INHERITANCE	
	A) Bank Interest	25
	B) Student Details	27
5.	MULTILEVEL INHERITANCE	
	A) Employee Details	29

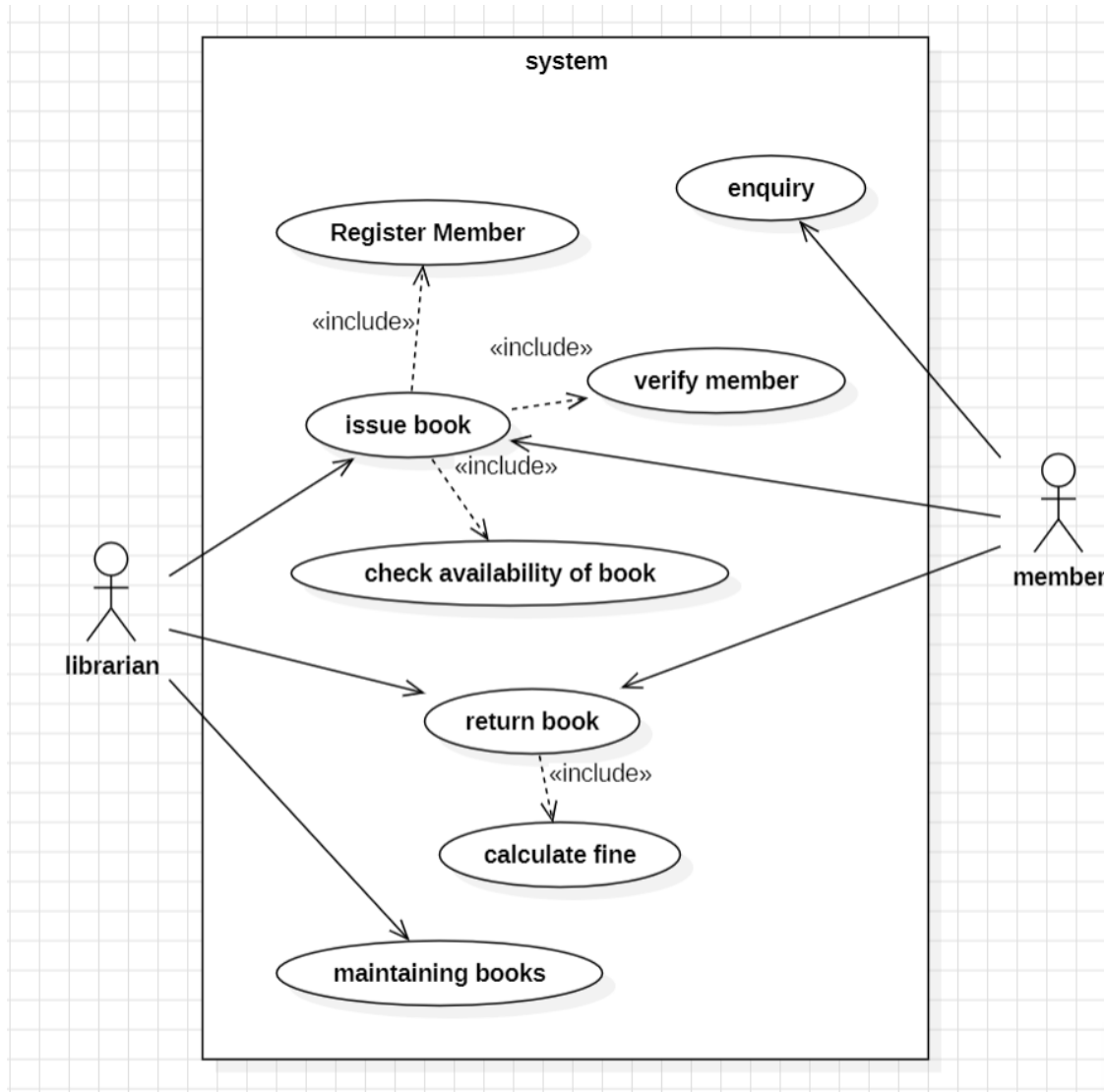
	B) Student Details	31
6.	HIERARCHICAL INHERITANCE	
	A) Workplace	33
	B) Shape Area	34
7.	HYBRID INHERITANCE	
	A) Vehicles	36
	B) Animals Food Preference	37
POLYMORPHISM		
8.	CONSTRUCTOR	
	A) Parent Child Method	39
9.	CONSTRUCTOR OVERLOADING	
	A) Student Details	40
10.	METHOD OVERLOADING	
	A) Addition Method	41
	B) Display Different Datatypes	42
11.	METHOD OVERRIDING	
	A) Displaying Parent Child	44
	B) Animal Sound	45
ABSTRACTION		
12.	INTERFACE	
	A) Animal Noises	46
	B) Vehicle	47
	C) Print and Show Method	49
	D) Method a and b	50
13.	ABSTRACT CLASS	
	A) Bank Type Details	51
	B) Shape Type	52
	C) Vehicle Types	53
	D) Animal Behaviour	55
ENCAPSULATION		
14.	ENCAPSULATION	
	A) Student Name	56
	B) Employee Id and Name	57

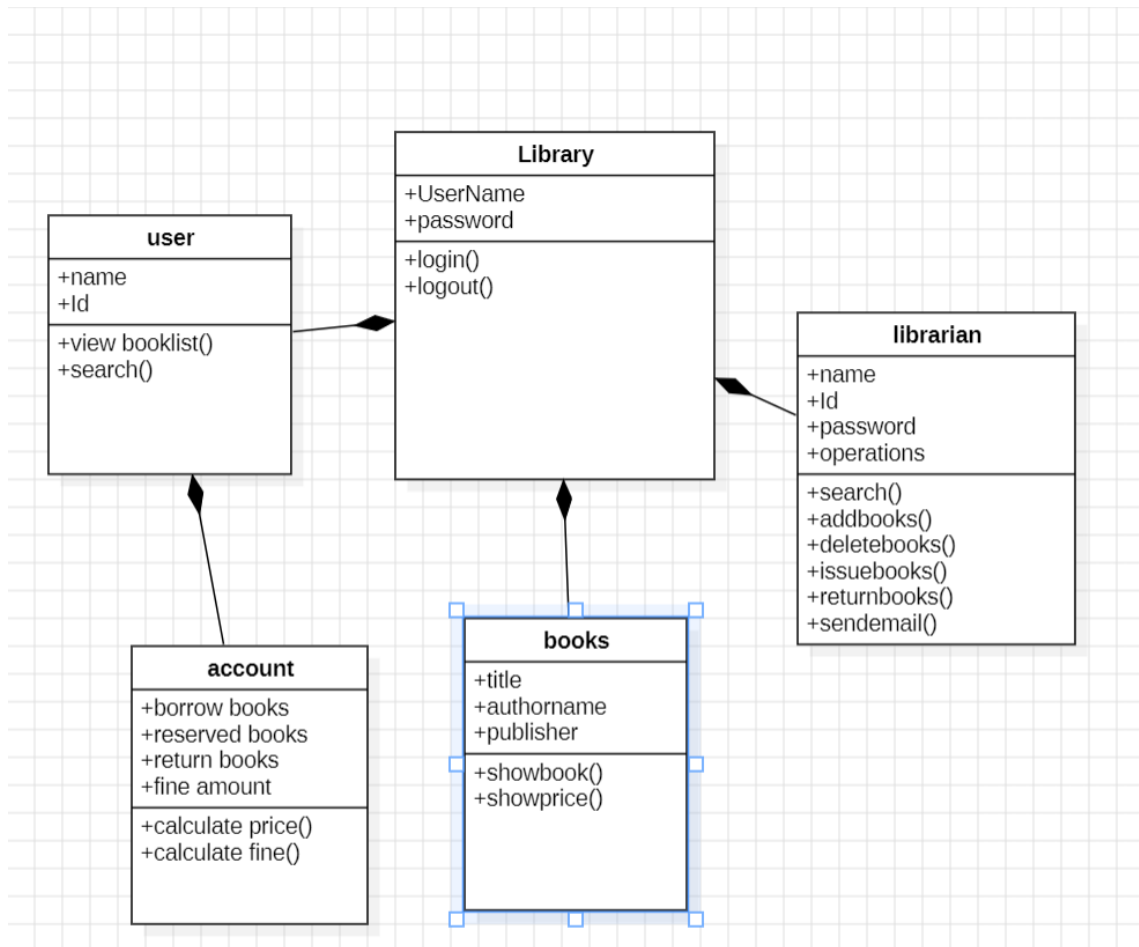
	C) Car Model	58
	D) Bank Balance	60
15.	USER DEFINED PACKAGES	
	A) Display Package	61
	B) Bank Details Package	62
16.	BUILT-IN PACKAGES	
	A) Existence of Files	64
	B) Array and Math Package	65
17.	EXCEPTION HANDLING	
	A) Arithmetic Error	66
	B) Index Error	66
	C) Length Error	67
	D) Age Error	68
18.	FILE HANDLING	
	A) Read File	68
	B) Create File	70
	C) Write File	71
	D) Append File	71

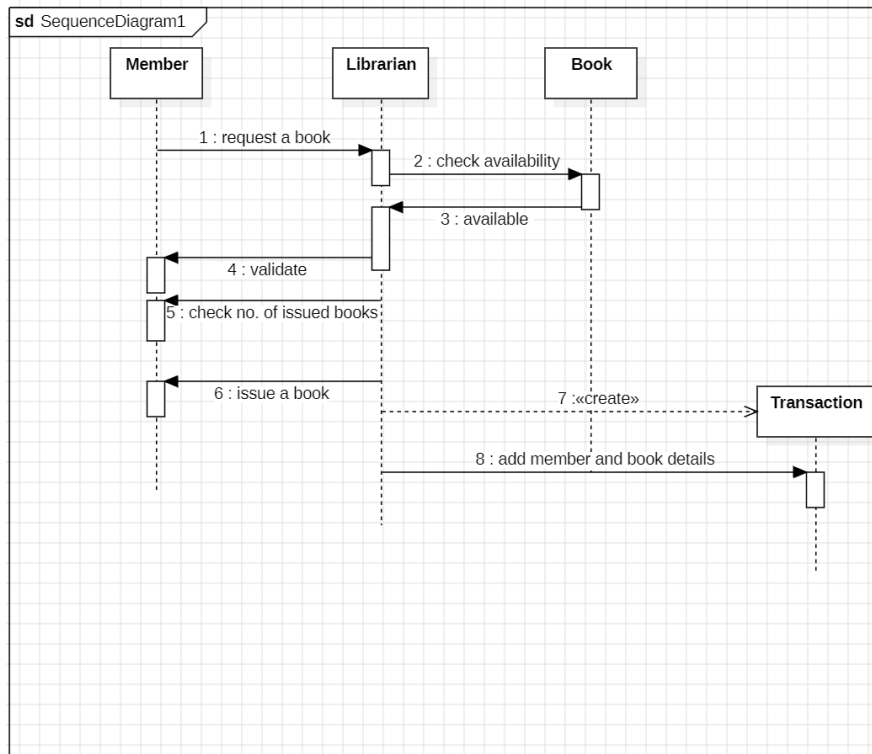
UML DIAGRAMS

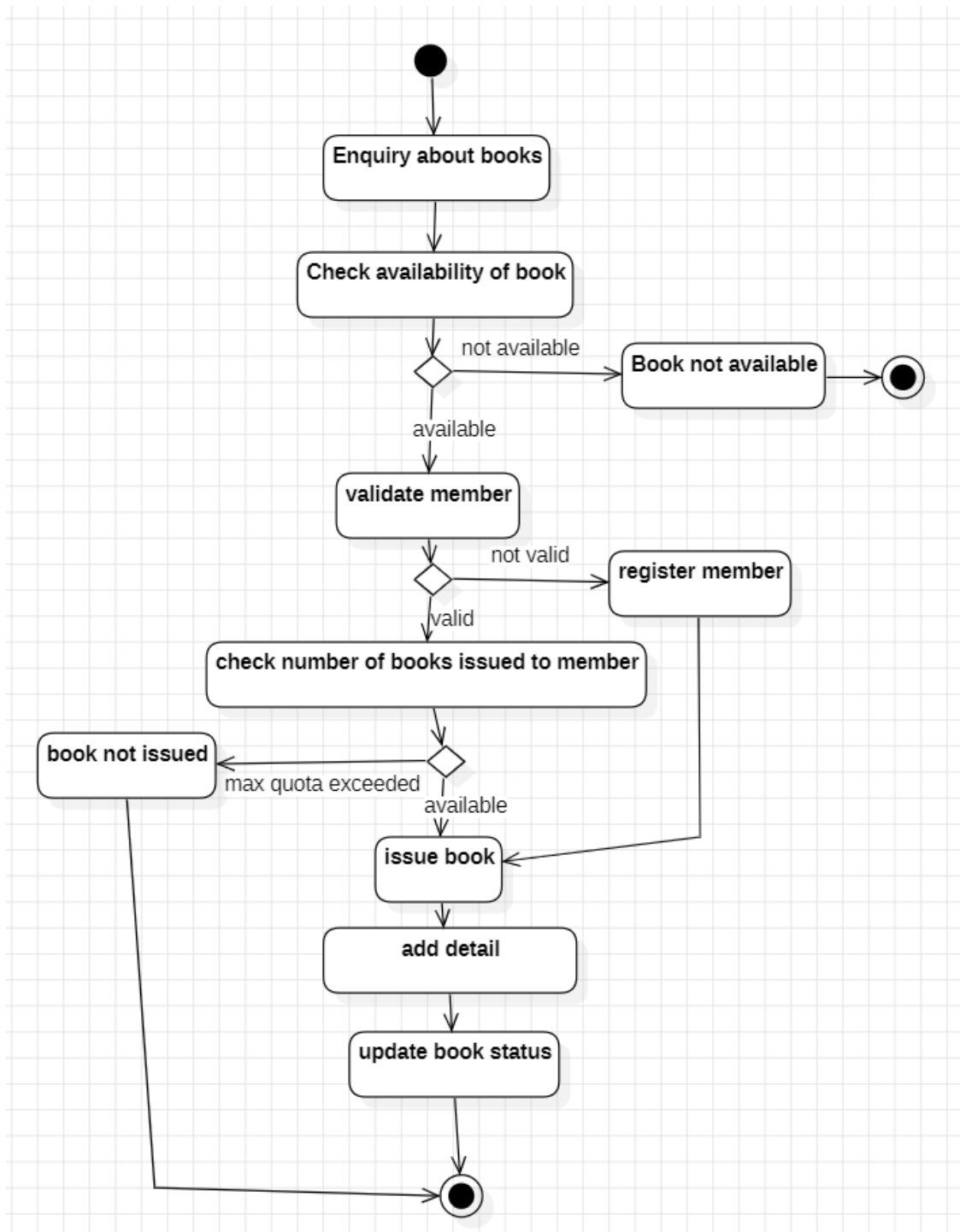
1. LIBRARY MANAGEMENT SYSTEM

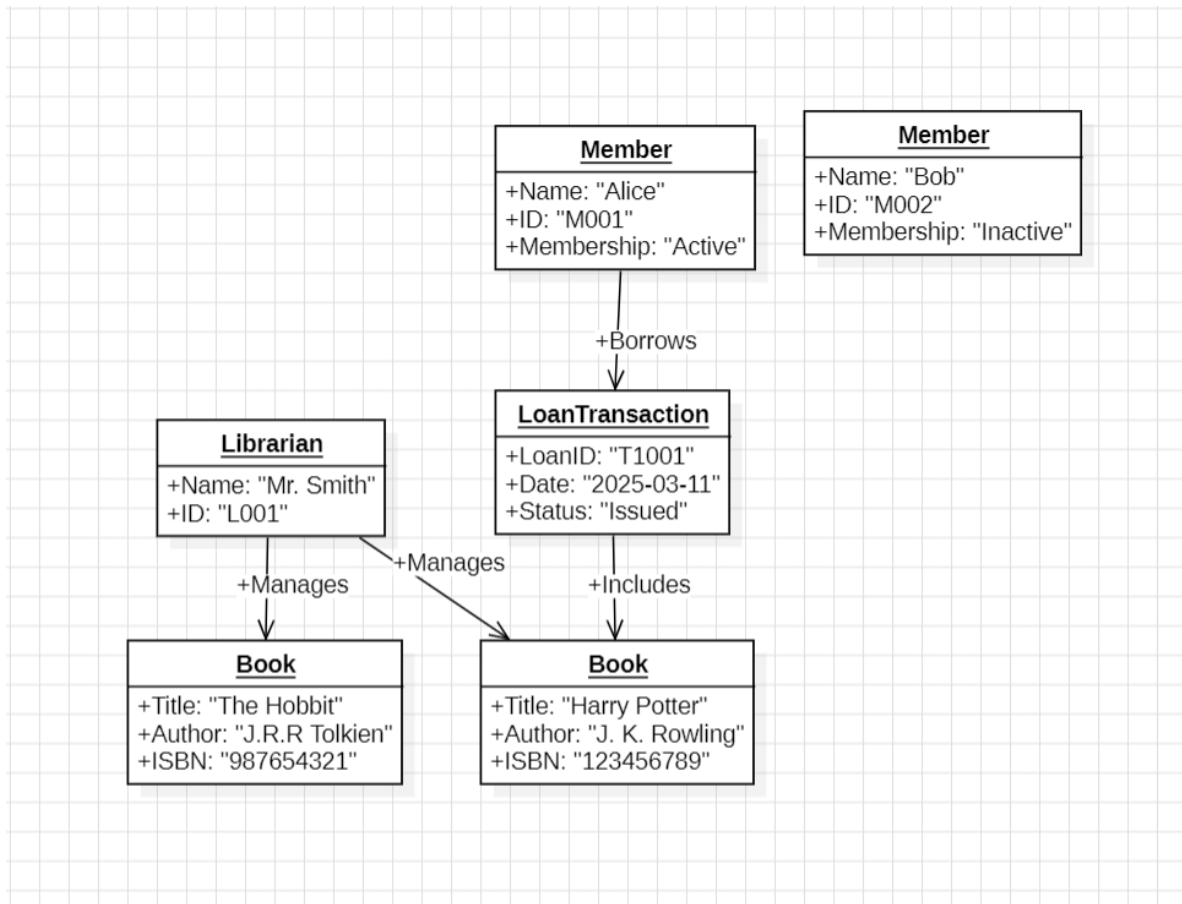
A) Use Case Diagram:



B) Class Diagram:

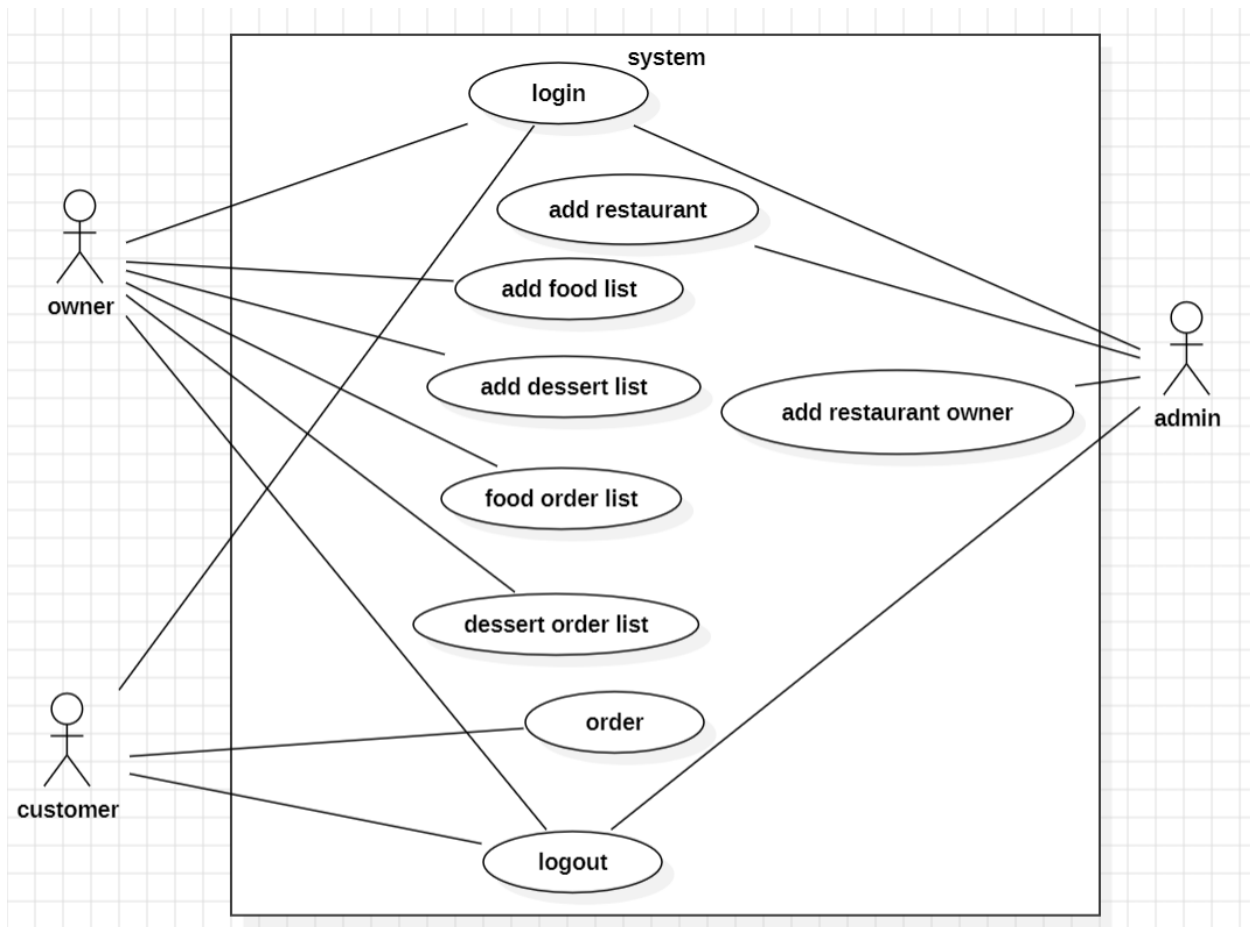
C) Sequence Diagram:

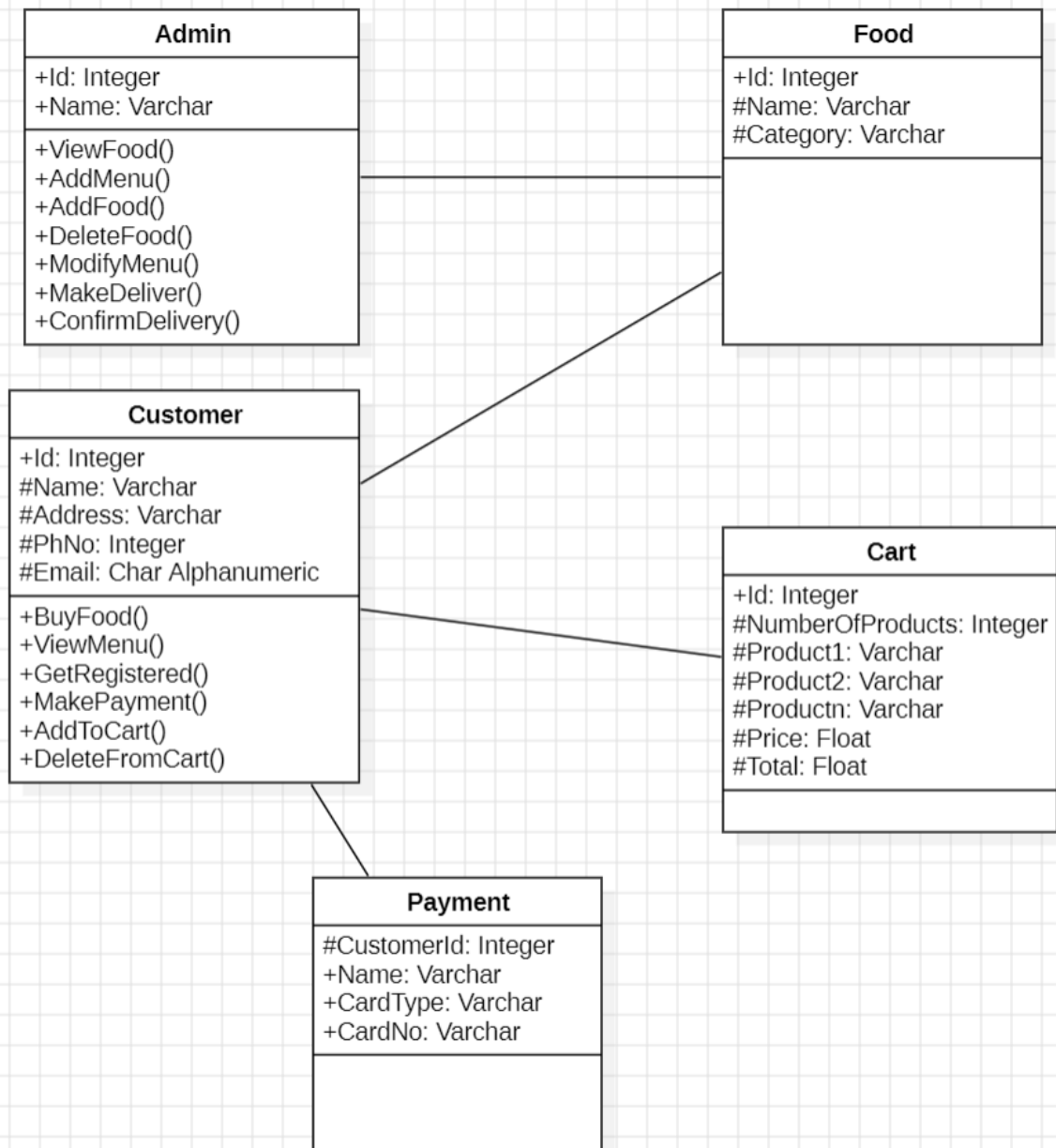
D) State Activity Diagram:

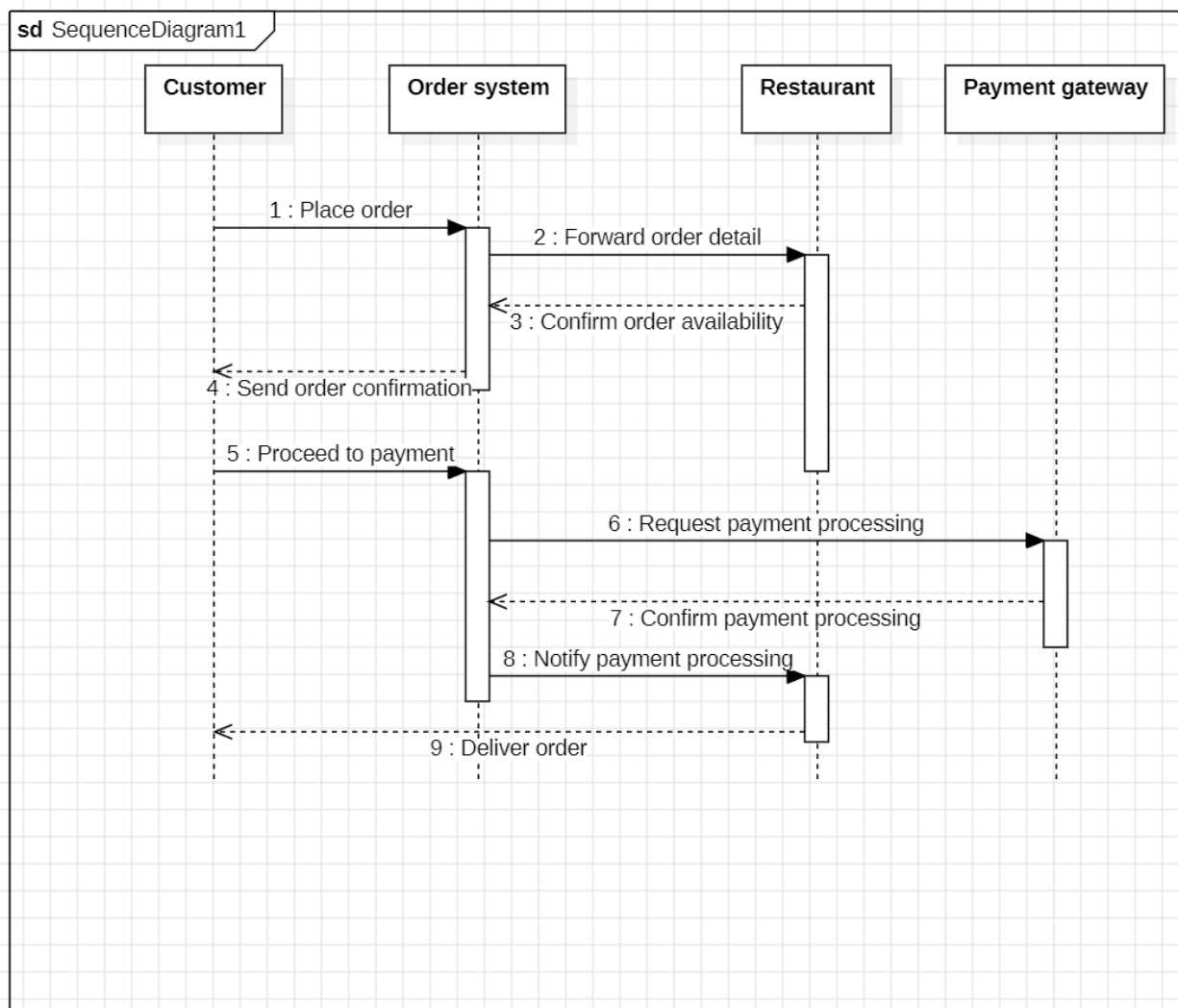
E) Object Diagram:

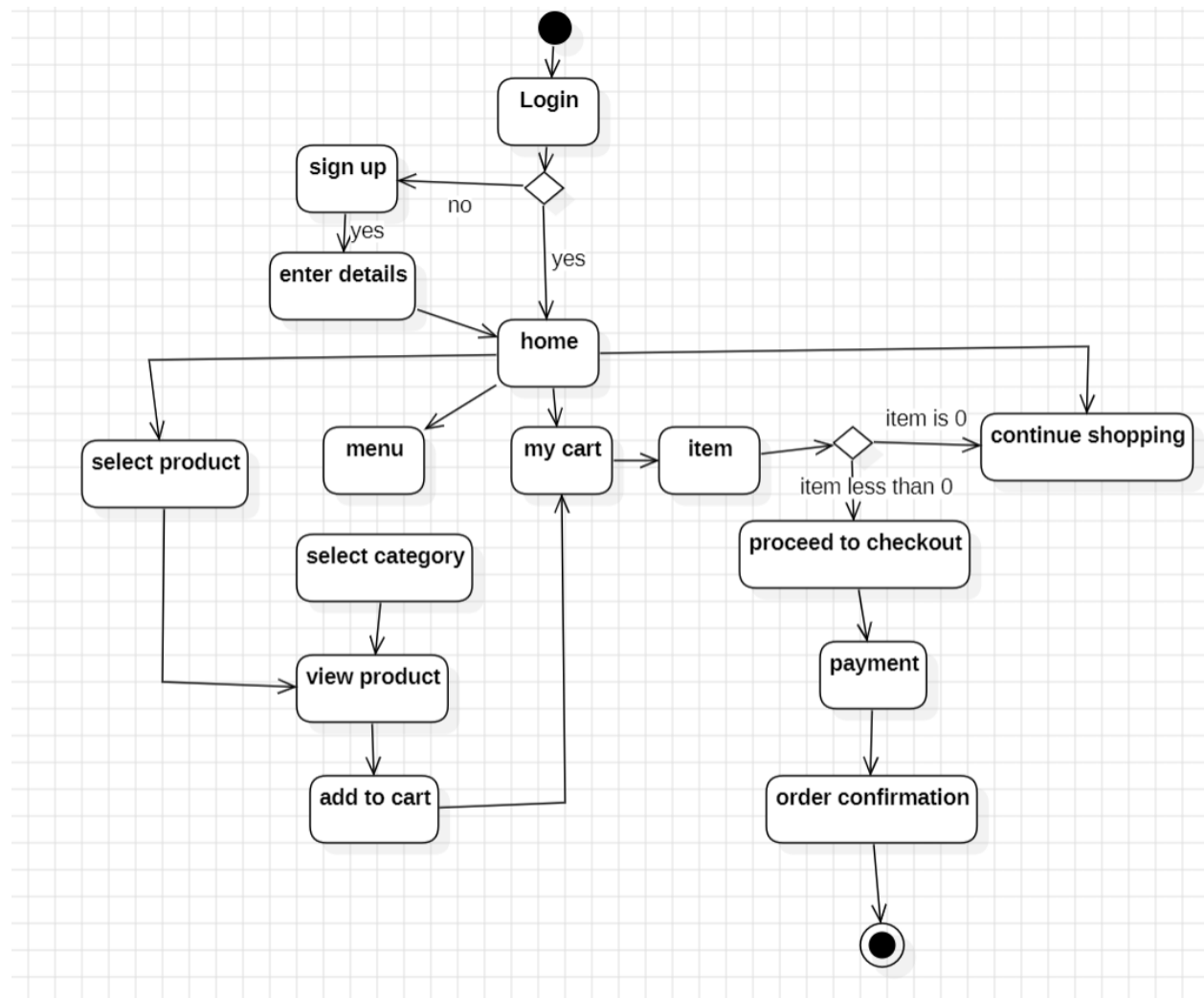
ONLINE FOOD ORDERING SYSTEM

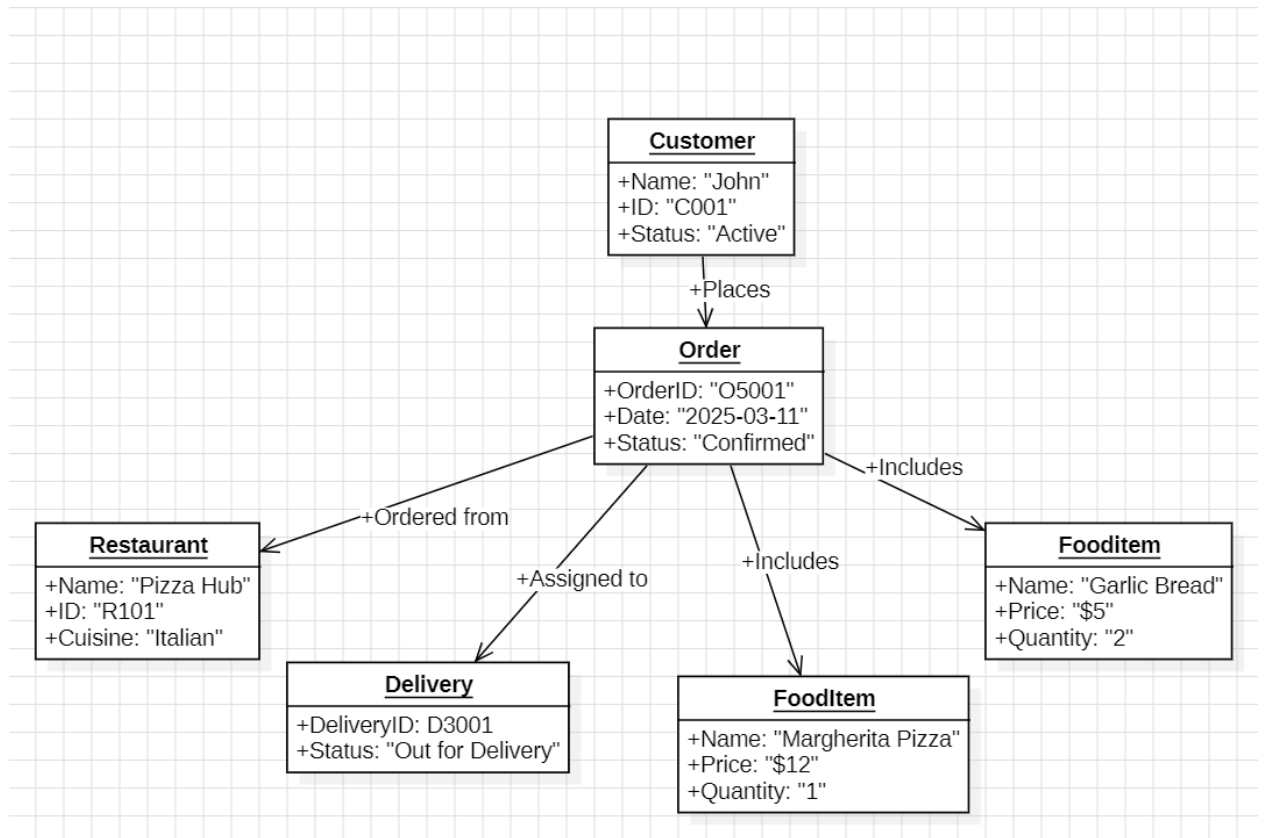
A) Use Case Diagram:



B) Class Diagram:

C) Sequence Diagram:

D) State Activity Diagram:

E) Object Diagram:

Basic Java Programs

A) Basic Calculator

CODE:

```
import java.util.Scanner;

public class Calculator {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        double num1 = sc.nextDouble();

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

        double num2 = sc.nextDouble();

        System.out.println("Choose an operation (+, -, *, /): ");

        char operator = sc.next().charAt(0);

        double result = 0;

        switch (operator) {

            case '+':

                result = num1 + num2;

                break;

            case '-':

                result = num1 - num2;

                break;

            case '*':

                result = num1 * num2;

                break;

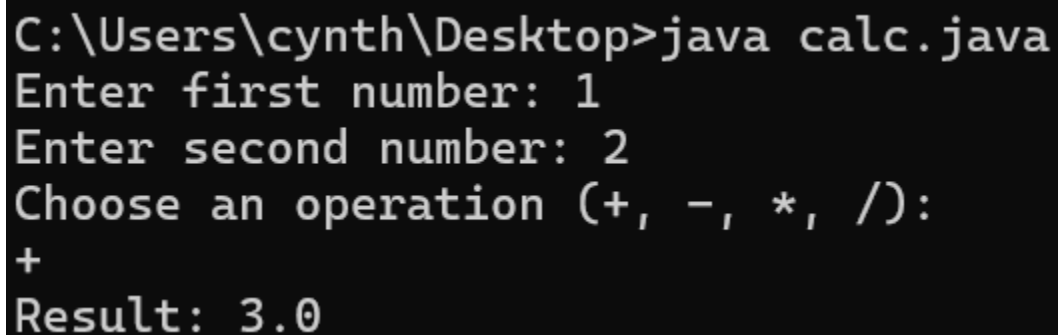
            case '/':

                result = num1 / num2;
```



```
        break;
    default:
        System.out.println("Invalid operator!");
        return;
    }
    System.out.println("Result: " + result);
}
}
```

OUTPUT



```
C:\Users\cynth\Desktop>java calc.java
Enter first number: 1
Enter second number: 2
Choose an operation (+, -, *, /):
+
Result: 3.0
```

B) Prime Number Check

CODE

```
import java.util.Scanner;

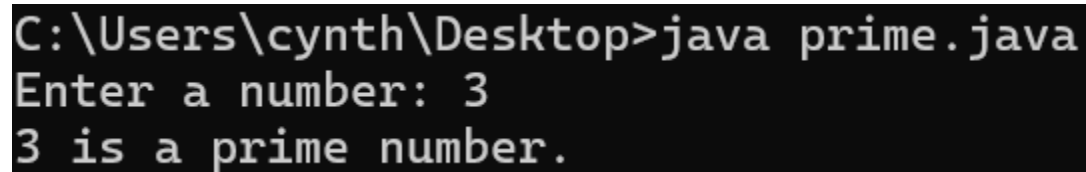
public class PrimeNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        boolean isPrime = true;

        for (int i = 2; i <= num / 2; ++i) {
            if (num % i == 0) {
```

```
        isPrime = false;
        break;
    }
}

if (isPrime && num > 1) {
    System.out.println(num + " is a prime number.");
} else {
    System.out.println(num + " is not a prime number.");
}
}
```

OUTPUT



```
C:\Users\cynth\Desktop>java prime.java
Enter a number: 3
3 is a prime number.
```

C) Fibonacci Series

CODE

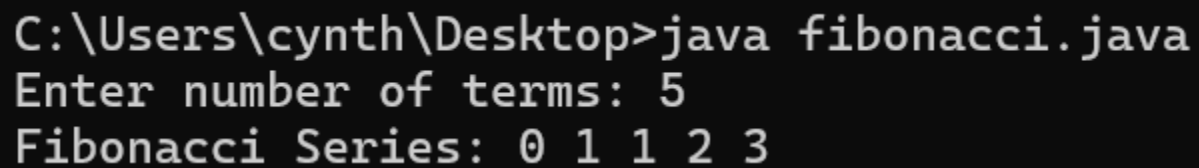
```
import java.util.Scanner;

public class Fibonacci {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of terms: ");
        int n = sc.nextInt();
        int a = 0, b = 1, c;

        System.out.print("Fibonacci Series: " + a + " " + b);
```

```
        for (int i = 2; i < n; i++) {
            c = a + b;
            System.out.print(" " + c);
            a = b;
            b = c;
        }
    }
}
```

OUTPUT



```
C:\Users\cynth\Desktop>java fibonacci.java
Enter number of terms: 5
Fibonacci Series: 0 1 1 2 3
```

D) Factorial of a Number

CODE

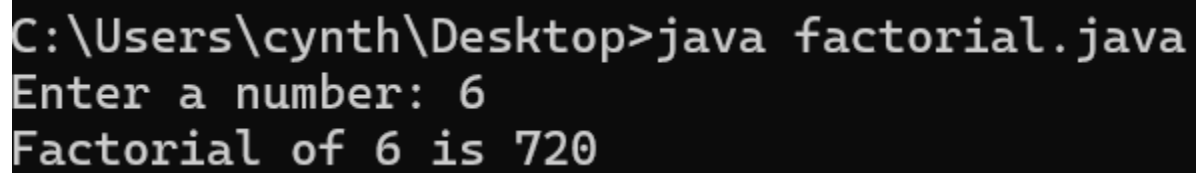
```
import java.util.Scanner;

public class Factorial {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        long factorial = 1;

        for (int i = 1; i <= num; i++) {
            factorial *= i;
        }
    }
}
```

```
        System.out.println("Factorial of " + num + " is " + factorial);
    }
}
```

OUTPUT



```
C:\Users\cynth\Desktop>java factorial.java
Enter a number: 6
Factorial of 6 is 720
```

E) PALINDROME

CODE

```
import java.util.Scanner;

public class Palindrome {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = sc.nextLine();
        String reversed = new StringBuilder(str).reverse().toString();

        if (str.equals(reversed)) {
            System.out.println(str + " is a palindrome.");
        } else {
            System.out.println(str + " is not a palindrome.");
        }
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop>java palindrome.java
Enter a string: mom
mom is a palindrome.
```

F) EVEN OR ODD NUMBER

CODE

```
import java.util.Scanner;

public class EvenOdd {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();

        if (num % 2 == 0) {
            System.out.println(num + " is an even number.");
        } else {
            System.out.println(num + " is an odd number.");
        }
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop>java evenodd.java
Enter a number: 5
5 is an odd number.
```

G) REVERSE OF A NUMBER

CODE

```
import java.util.Scanner;

public class ReverseNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int num = sc.nextInt();

        int reversed = 0;

        while (num != 0) {

            int digit = num % 10;

            reversed = reversed * 10 + digit;

            num /= 10;

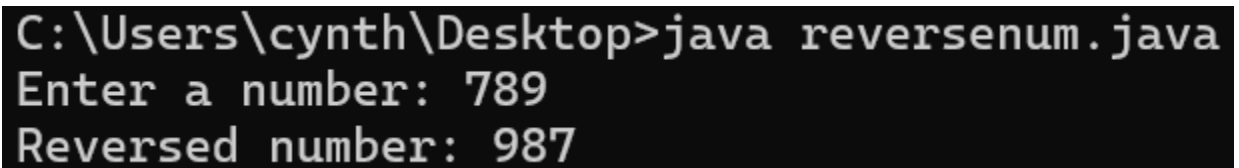
        }

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

    }

}
```

OUTPUT

A screenshot of a terminal window with a black background and white text. The prompt 'C:\Users\cynth\Desktop>' is followed by the command 'java reversenum.java'. The program then prompts 'Enter a number: ' and the user has entered '789'. The program outputs 'Reversed number: 987'.

H) SUM OF DIGITS

CODE

```
import java.util.Scanner;
```

```
public class SumOfDigits {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Enter a number: ");  
        int num = sc.nextInt();  
        int sum = 0;  
  
        while (num != 0) {  
            sum += num % 10;  
            num /= 10;  
        }  
  
        System.out.println("Sum of digits: " + sum);  
    }  
}
```

OUTPUT



```
C:\Users\cynth\Desktop>java newsumofdigits.java  
Enter a number: 974  
Sum of digits: 20
```

I) PRINT MULTIPLICATION TABLE

CODE

```
import java.util.Scanner;  
  
public class MultiplicationTable {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);
```

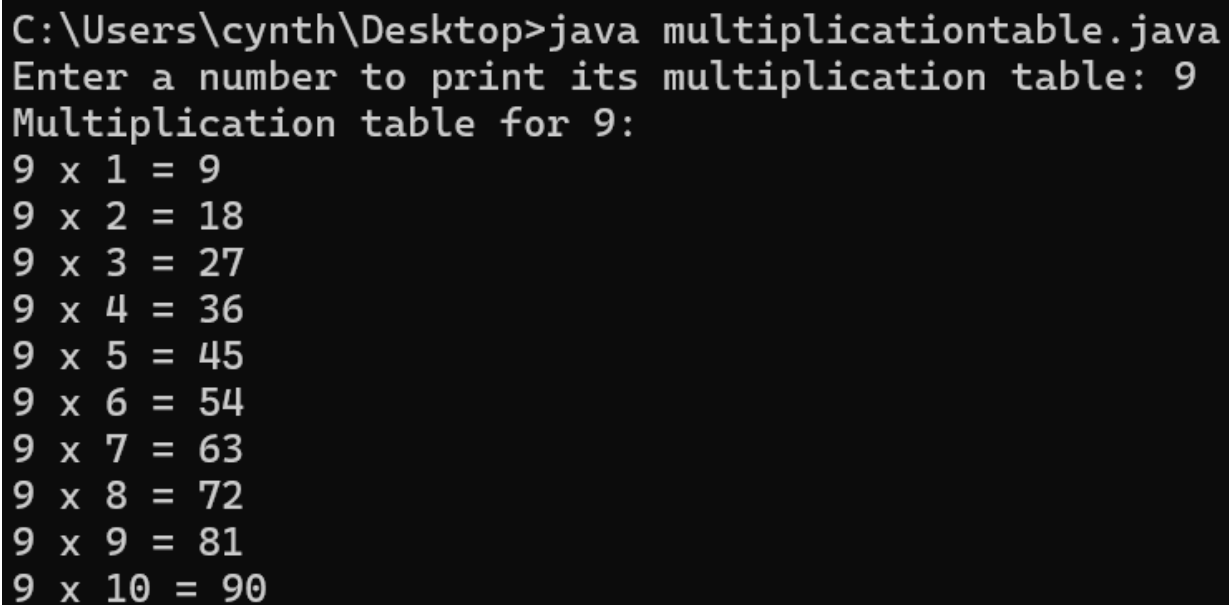
```
System.out.print("Enter a number to print its multiplication table: ");

int num = sc.nextInt();

System.out.println("Multiplication table for " + num + ":");

for (int i = 1; i <= 10; i++) {
    System.out.println(num + " x " + i + " = " + (num * i));
}
}
```

OUTPUT



```
C:\Users\cynth\Desktop>java multiplicationtable.java
Enter a number to print its multiplication table: 9
Multiplication table for 9:
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
```

J) MAXIMUM OF TWO NUMBERS

CODE

```
import java.util.Scanner;

public class MaxOfTwo {
    public static void main(String[] args) {
```



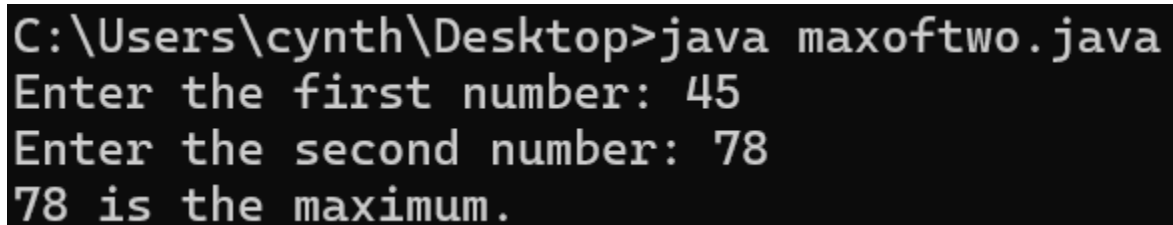
```
Scanner sc = new Scanner(System.in);

System.out.print("Enter the first number: ");
int num1 = sc.nextInt();

System.out.print("Enter the second number: ");
int num2 = sc.nextInt();

if (num1 > num2) {
    System.out.println(num1 + " is the maximum.");
} else if (num2 > num1) {
    System.out.println(num2 + " is the maximum.");
} else {
    System.out.println("Both numbers are equal.");
}
}
```

OUTPUT



```
C:\Users\cynth\Desktop>java maxoftwo.java
Enter the first number: 45
Enter the second number: 78
78 is the maximum.
```

Inheritance Programs

Single Inheritance Programs

A) Bank Interest

CODE:

```
class BankAccount
{
float balance;
void deposit(float amount)
{
balance+=amount;
System.out.println("balance:"+balance);
}

void withdraw(float amount)
{
balance-=amount;
System.out.println("balance:"+balance);
}
}

class Interest extends BankAccount
{
float interest;

void interest()
{
interest=balance/100;
System.out.println("interest:"+interest);
}
}

class singleinheritance1{
public static void main(String args[])
{
```

```
Interest obj= new Interest();  
obj.deposit(1000);  
obj.withdraw(100);  
obj.interest();  
}  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java singleinheritancel.java  
balance:1000.0  
balance:900.0  
interest:9.0
```

B) Student details

CODE:

```
class student {  
    int rollno;  
    String name;  
  
    student(String name, int rollno) {  
        this.name=name;  
        this.rollno=rollno;  
    }  
  
    void displayStudentDetails() {  
  
        System.out.println("name:"+name);  
        System.out.println("rollno:"+rollno);  
    }  
}  
  
class exam extends student {
```

```
int marks;

exam(String name, int rollno, int marks){
    super(name,rollno);
    this.marks=marks;
}

void displayExamDetails() {
    displayStudentDetails();
    System.out.println("marks:"+marks);

}

}

class singleinheritance2 {
    public static void main(String[] args) {

        exam student1 = new exam("su", 101, 23);
        student1.displayExamDetails();

        exam student2 = new exam("ru", 102, 24);
        student2.displayExamDetails();

    }

}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java singleinheritance2.java
name:su
rollno:101
marks:23
name:ru
rollno:102
marks:24
```

Multilevel Inheritance

A) Employee details

CODE:

```
class person {  
  
    String name;  
  
    void setName(String name)  
    {  
        this.name=name;  
    }  
  
    void display()  
    {  
        System.out.println("name:"+name);  
    }  
}  
  
class employee extends person {  
  
    String job;  
  
    void setEmp(String job)  
    {  
        this.job=job;  
    }  
  
    void displayEmp()  
    {
```

```
System.out.println("job:"+job);
}

}

class manager extends employee {

String department;
void setMan(String department)
{
this.department=department;
}

void displayMan()
{
display();
displayEmp();
System.out.println("department:"+department);
}

}

class multilevel1
{
public static void main(String[] args) {

manager obj1= new manager();
obj1.setName("su");
obj1.setEmp("data analyst");
obj1.setMan("cse");
```

```
obj1.displayMan();  
}  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java multilevel1.java  
name:su  
job:data analyst  
department:cse
```

B) Student details

CODE:

```
class name {  
    String name;  
  
    name(String name) {  
        this.name=name;  
    }  
  
    void displayname() {  
        System.out.println("name:"+name);  
    }  
}  
  
class school extends name {  
    String school;  
  
    school(String name, String school) {  
        super(name);  
        this.school=school;  
    }  
}
```

```
}
```

```
void displayschool() {  
    displayname();  
    System.out.println("school:"+school);  
}  
}
```

```
class college extends school {  
    String college;
```

```
    college(String name, String school, String college) {  
        super(name, school);  
        this.college=college;  
    }
```

```
    void displaycollege() {  
        displayschool();  
        System.out.println("college:"+college);  
    }  
}
```

```
class multilevel2 {  
    public static void main(String args[])  
    {  
        college student1= new college("su","ABC highschool","RI college");  
        student1.displaycollege();  
    }  
}
```


OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java multilevel2.java
name:su
school:ABC highschool
college:RI college
```

Hierarchical Inheritance

A) Workplace

CODE:

```
class employee {
    void work()
    {
        System.out.println("employee works");
    }
}

class manager extends employee {
    void manages()
    {
        System.out.println("manager manages");
    }
}

class analyst extends employee {
    void analyses()
    {
        System.out.println("analyst analyses");
    }
}
```

```
class Hierarchical1 {  
    public static void main(String args[])  
    {  
        manager man1= new manager();  
        man1.work();  
        man1.manages();  
  
        analyst ana2= new analyst();  
        ana2.work();  
        ana2.analyses();  
    }  
  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java Hierarchical1.java  
employee works  
manager manages  
employee works  
analyst analyses
```

B) Shape area

CODE:

```
class shape {  
  
    void display() {  
  
        System.out.println("this is a shape");  
    }  
  
}  
  
class circle extends shape{
```

```
int r;

void areac(int r) {

System.out.println("area of circle:"+r*3.14r);

}

}

class rect extends shape {

int l;

int b;

void arear(int l, int b) {

System.out.println("area of rect:"+l*b);

}

}

class Hierarchical2 {

public static void main(String args[]) {

circle c1=new circle();

c1.display();

c1.areac(2);

rect r1=new rect();

r1.display();

r1.arear(4,2);

}

}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java Hierarchical2.java
this is a shape
area of circle:12.56
this is a shape
area of rect:8
```

Hybrid Inheritance

A) Vehicles

CODE:

```
interface Vehicle {
    void start();
}

class Car implements Vehicle {
    public void start() {
        System.out.println("Car is starting...");
    }

    void drive() {
        System.out.println("Car is driving");
    }
}

class Bike implements Vehicle {
    public void start() {
        System.out.println("Bike is starting...");
    }

    void ride() {
        System.out.println("Bike is riding");
    }
}
```

```
class HybridCar extends Car{
    void hybridFeature() {
        System.out.println("HybridCar runs on fuel and electricity");
    }
}

class hybrid1 {
    public static void main(String[] args) {
        HybridCar myHybrid = new HybridCar();
        myHybrid.start();
        myHybrid.drive();
        myHybrid.hybridFeature();

        Bike myBike = new Bike();
        myBike.start();
        myBike.ride();
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java hybrid1.java
Car is starting...
Car is driving
HybridCar runs on fuel and electricity
Bike is starting...
Bike is riding
```

B) Animals Food Preference

CODE:

```
class animal {
    void eat(){
        System.out.println("animal eats");
    }
}
```

```
}  
}
```

```
interface carnivore {  
    void eatsmeat();  
}
```

```
interface herbivore{  
    void eatsgrass();  
}
```

```
class omnivore extends animal implements carnivore, herbivore {  
    public void eatsmeat() {
```

```
        System.out.println("omnivore eats meat");
```

```
    }
```

```
    public void eatsgrass() {  
        System.out.println("omnivore eats grass");
```

```
    }
```

```
}
```

```
class hybrid2 {  
    public static void main (String[] args) {  
        omnivore obj1=new omnivore();  
        obj1.eat();  
        obj1.eatsmeat();
```

```
obj1.eatsgrass();  
}
```

```
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java hybrid2.java  
animal eats  
omnivore eats meat  
omnivore eats grass
```

POLYMORPHISM

Constructor Programs

A) Parent Child method

CODE:

```
class parent {  
  
    parent() {  
        System.out.println("dis is parent");  
    }  
}  
  
class child extends parent{  
    child() {  
        super();  
        System.out.println("dis is child");  
    }  
}
```

```
class poly1 {  
    public static void main(String[] args) {  
        child obj1=new child();  
  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java poly1.java  
dis is parent  
dis is child
```

Constructor Overloading Programs

A) Student Details

CODE:

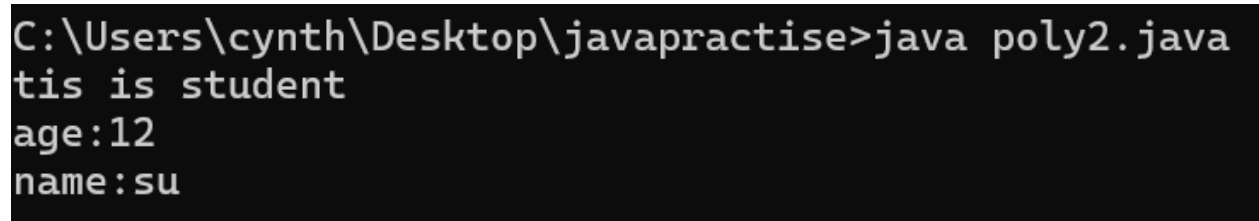
```
class student {  
    int age;  
    String name;  
  
    student() {  
        System.out.println("tis is student");  
    }  
  
    student(int age) {  
        this.age=age;  
        System.out.println("age:"+age);  
    }  
}
```



```
student(int age, String name) {  
    this.age=age;  
    this.name=name;  
  
    System.out.println("name:"+name);  
  
}  
}
```

```
class poly2 {  
    public static void main (String[] args) {  
        student obj1= new student();  
        student obj2= new student(12);  
        student obj3= new student(12,"su");  
    }  
}
```

OUTPUT



```
C:\Users\cynth\Desktop\javapractise>java poly2.java  
tis is student  
age:12  
name:su
```

Method Overloading Programs

A) Addition method

CODE:

```
class matho {  
    int add(int a,int b) {  
        return a+b;
```

```
}

int add(int a,int b,int c) {
return a+b+c;
}

double add(double a,double b) {
return a+b;
}

}

class polymovl1 {
public static void main(String[] args) {
matho obj1=new matho();
System.out.println(obj1.add(1, 2));
System.out.println(obj1.add(1, 2, 2));
System.out.println(obj1.add(1.2, 2.4));
}
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java polymovl1.java
3
5
3.5999999999999996
```

B) Display Different Datatypes

CODE:

```
class Display {
```

```
void show(int x) {
    System.out.println("x: " + x);
}

void show(String x) {
    System.out.println("x: " + x);
}

void show(double x) {
    System.out.println("x: " + x);
}

}

class polymovl2 {
    public static void main(String[] args) {
        Display obj = new Display();
        obj.show(100);
        obj.show("Hello");
        obj.show(25.75);
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java polymovl2.java
x: 100
x: Hello
x: 25.75
```

Method Overriding Programs

A) Displaying Parent Child

CODE:

```
class Parent {
    void show() {
        System.out.println("This is the Parent class");
    }
}

class Child extends Parent {
    void show() {
        System.out.println("This is the Child class");
    }
}

class polymovr1 {
    public static void main(String[] args) {
        Parent obj1 = new Parent();
        obj1.show();
        Child obj2 = new Child();
        obj2.show();
        Parent obj3 = new Child();
        obj3.show();
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java polymovr1.java
This is the Parent class
This is the Child class
This is the Child class
```

B) Animal Sound

CODE:

```
class animal {
void sound() {
System.out.println("animal makes sound");
}
}
```

```
class cat extends animal {
void sound() {
super.sound();
System.out.println("cat meows");
}
}
```

```
class polymovr2 {
public static void main(String[] args) {
cat obj=new cat();
obj.sound();
}
```

```
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java polymovr2.java  
animal makes sound  
cat meows
```

ABSTRACTION

Interface Programs

A) Animal Noises

CODE:

```
interface Animal {  
    void sound();  
}
```

```
class Dog implements Animal {  
    public void sound() {  
        System.out.println("Dog barks");  
    }  
}
```

```
class Cat implements Animal {  
    public void sound() {  
        System.out.println("Cat meows");  
    }  
}
```

```
class interface1 {  
    public static void main(String[] args) {
```

```
Dog d = new Dog();  
d.sound();  
  
Cat c = new Cat();  
c.sound();  
}  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java interface1.java  
Dog barks  
Cat meows
```

B) Vehicle

CODE:

```
interface Vehicle {  
    void start();  
    void stop();  
}  
  
class Car implements Vehicle {  
    public void start() {  
        System.out.println("Car starts with key");  
    }  
  
    public void stop() {  
        System.out.println("Car stops with brake");  
    }  
}
```

```
class Bike implements Vehicle {  
    public void start() {  
        System.out.println("Bike starts with kick");  
    }  
  
    public void stop() {  
        System.out.println("Bike stops with disc brake");  
    }  
}  
  
class interface2 {  
    public static void main(String[] args) {  
        Vehicle v1 = new Car();  
        v1.start();  
        v1.stop();  
  
        Vehicle v2 = new Bike();  
        v2.start();  
        v2.stop();  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java interface2.java  
Car starts with key  
Car stops with brake  
Bike starts with kick  
Bike stops with disc brake
```


C) Print and Show Method

CODE:

```
interface Printable {  
    void print();  
}  
  
interface Showable {  
    void show();  
}  
  
class Display implements Printable, Showable {  
    public void print() {  
        System.out.println("Printing...");  
    }  
  
    public void show() {  
        System.out.println("Showing...");  
    }  
}  
  
class interface3 {  
    public static void main(String[] args) {  
        Display obj = new Display();  
        obj.print();  
        obj.show();  
    }  
}
```

OUTPUT

```
C:\Users\cyntn\Desktop\javapractise>java interface3.java
Printing...
Showing...
```

D) Method a and b

CODE:

```
interface a {
    void methoda();
}

interface b extends a {
    void methodb();
}

class c implements b {
    public void methoda() {
        System.out.println("methoda");
    }

    public void methodb() {
        System.out.println("methodb");
    }
}

class interface4 {
    public static void main(String[] args) {
```

```
c obj= new c();  
obj.methoda();  
obj.methodb();  
}  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java interface4.java  
methoda  
methodb
```

Abstract Class Programs

A) Bank type details

CODE:

```
abstract class bank {  
    abstract void intr();  
  
    void type() {  
        System.out.println("dis a bank");  
    }  
}  
  
class sbc extends bank{  
    public void intr() {  
        System.out.println("int rate:5%");  
    }  
}  
  
class hdfc extends bank {
```

```
public void intr() {  
    System.out.println("int rate: 7%");  
}  
}
```

```
class abstract1 {  
    public static void main (String[] args) {  
        sbc obj1=new sbc();  
        obj1.type();  
        obj1.intr();  
  
        hdfc obj2=new hdfc();  
        obj2.type();  
        obj2.intr();  
  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java abstract1.java  
dis a bank  
int rate:5%  
dis a bank  
int rate: 7%
```

B) Shape Type

CODE:

```
abstract class Shape {  
    Shape() {  
        System.out.println("Shape constructor called");  
    }  
}
```

```
        abstract void draw();
    }

    class Circle extends Shape {
        public void draw() {
            System.out.println("Drawing a circle");
        }
    }

    class abstract2 {
        public static void main(String[] args) {
            Circle obj = new Circle();
            obj.draw();
        }
    }
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java abstract2.java
Shape constructor called
Drawing a circle
```

C) Vehicle Types

CODE:

```
abstract class Vehicle {
    abstract void start();

    void fuelType() {
        System.out.println("This vehicle uses fuel");
    }
}
```

```
    }  
}  
  
class Car extends Vehicle {  
    public void start() {  
        System.out.println("Car starts with a key");  
    }  
}  
  
class Bike extends Vehicle {  
    public void start() {  
        System.out.println("Bike starts with a kick");  
    }  
}  
  
class abstract3 {  
    public static void main(String[] args) {  
        Vehicle v1 = new Car();  
        v1.start();  
        v1.fuelType();  
  
        Vehicle v2 = new Bike();  
        v2.start();  
        v2.fuelType();  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java abstract3.java
Car starts with a key
This vehicle uses fuel
Bike starts with a kick
This vehicle uses fuel
```

D) Animal behaviour

CODE:

```
abstract class Animal {
    abstract void sound();

    void sleep() {
        System.out.println("Sleeping...");
    }
}

class Dog extends Animal {
    public void sound() {
        System.out.println("Dog barks");
    }
}

class abstract4 {
    public static void main(String[] args) {
        Dog obj = new Dog();
        obj.sound();
        obj.sleep();
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java abstract4.java  
Dog barks  
Sleeping...
```

ENCAPSULATION

Encapsulation Programs

A) Student name

CODE:

```
class Student {  
    private String name;  
  
    public void setName(String newName) {  
        name = newName;  
    }  
  
    public String getName() {  
        return name;  
    }  
}  
  
class encap1 {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.setName("John");  
        System.out.println("Student Name: " + s.getName());  
    }  
}
```


OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java encap1.java  
Student Name: John
```

B) Employee id and name

CODE:

```
class Employee {  
    private int id;  
    private String name;  
    private double salary;  
  
    public void setId(int newId) {  
        id = newId;  
    }  
  
    public void setName(String newName) {  
        name = newName;  
    }  
  
    public void setSalary(double newSalary) {  
        salary = newSalary;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public String getName() {
```

```
        return name;
    }

    public double getSalary() {
        return salary;
    }
}

class encap2{
    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.setId(101);
        emp.setName("Alice");
        emp.setSalary(50000);

        System.out.println("Employee ID: " + emp.getId());
        System.out.println("Employee Name: " + emp.getName());
        System.out.println("Employee Salary: " + emp.getSalary());
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java encap2.java
Employee ID: 101
Employee Name: Alice
Employee Salary: 50000.0
```

C) Car Model

CODE:

```
class Car {  
    private String model;  
    private int year;  
  
    public Car(String model, int year) {  
        this.model = model;  
        this.year = year;  
    }  
  
    public String getModel() {  
        return model;  
    }  
  
    public int getYear() {  
        return year;  
    }  
}  
  
class encap3 {  
    public static void main(String[] args) {  
        Car myCar = new Car("Tesla", 2024);  
        System.out.println("Car Model: " + myCar.getModel());  
        System.out.println("Car Year: " + myCar.getYear());  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java encap3.java
Car Model: Tesla
Car Year: 2024
```

D) Bank Balance

CODE:

```
class account {
    private double balance;

    public account(double balance) {
        this.balance=balance;
    }

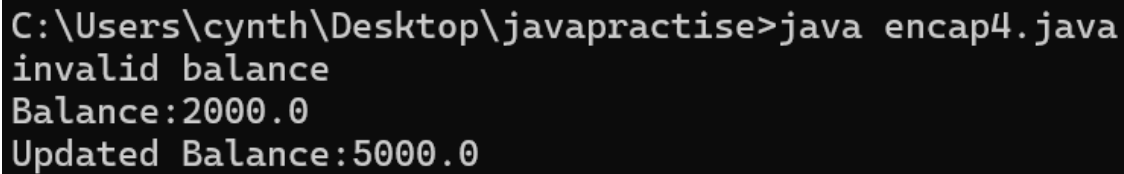
    public void setbalance(double newbalance) {
        if (newbalance>0) {
            balance=newbalance;
        }
        else {
            System.out.println("invalid balance");
        }
    }

    double getbalance() {
        return balance;
    }
}

class encap4 {
    public static void main(String[] args) {
        account obj=new account(2000);
```

```
obj.setbalance(-500);  
System.out.println("Balance:"+obj.getbalance());  
obj.setbalance(5000);  
System.out.println("Updated Balance:"+obj.getbalance());  
  
}  
}
```

OUTPUT



```
C:\Users\cynth\Desktop\javapractise>java encap4.java  
invalid balance  
Balance:2000.0  
Updated Balance:5000.0
```

User Defined Packages

A) Display Package

CODE:

Package File:

```
package mypackage;
```

```
public class pack1 {  
    public void display() {  
        System.out.println("Hello from pack1 in mypackage!");  
    }  
}
```

Main File:

```
import mypackage.pack1;
```

```
class userdefpack1 {  
    public static void main(String[] args) {  
        pack1 obj = new pack1();  
        obj.display();  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>javac -d . pack1.java  
C:\Users\cynth\Desktop\javapractise>javac userdefpack1.java  
C:\Users\cynth\Desktop\javapractise>java userdefpack1.java  
Hello from pack1 in mypackage!
```

B) Bank Details Package

CODE:

Package 1 File:

```
package bank;
```

```
public class pack21 {  
    private double balance;
```

```
    public pack21(double balance) {  
        this.balance=balance;  
    }  
    public void getbalance() {  
        System.out.println("balance:"+balance);  
    }  
}
```

```
}
```

Package 2 File:

```
package bank;
```

```
public class pack22 {
```

```
private String name;
```

```
public pack22(String name) {
```

```
this.name=name;
```

```
}
```

```
public void getname() {
```

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

```
}
```

```
}
```

Main File:

```
import bank.pack21;
```

```
import bank.pack22;
```

```
class userdefpack2 {
```

```
public static void main(String[] args) {
```

```
pack21 obj1= new pack21(4000.5);
```

```
obj1.getbalance();
```

```
pack22 obj2= new pack22("su");
```

```
obj2.getname();
```

```
}
```

```
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>javac -d . pack21.java
C:\Users\cynth\Desktop\javapractise>javac -d . pack22.java
C:\Users\cynth\Desktop\javapractise>javac userdefpack2.java
C:\Users\cynth\Desktop\javapractise>java userdefpack2.java
balance:4000.5
name:su
```

Built-in Packages

A) Existence of File

CODE:

```
import java.util.Scanner;
import java.io.File;
import java.time.LocalDate;

class builtinpack1 {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter file name: ");
        String fileName = sc.nextLine();

        File file = new File(fileName);
        if (file.exists()) {
            System.out.println("File exists: " + file.getName());
        } else {
            System.out.println("File does not exist.");
        }
    }
}
```



```
        LocalDate today = LocalDate.now();

        System.out.println("Today's Date: " + today);

        sc.close();
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java builtinpack1.java
Enter file name: cynthia
File does not exist.
Today's Date: 2025-03-31
```

B) Array and Math Package

CODE:

```
import java.util.ArrayList;

import java.lang.Math;

import java.math.BigDecimal;

class builtinpack2 {

    public static void main(String[] args) {

        ArrayList<String> obj1= new ArrayList<>();

        obj1.add("su");

        obj1.add("ru");

        System.out.println("Array:"+obj1);

        System.out.println("sqrt of16:"+Math.sqrt(16));

        BigDecimal obj2=new BigDecimal(283.349);
```

```
System.out.println("Big Decimal:"+obj2);
```

```
}  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java builtinpack2.java  
Array:[su, ru]  
sqrt of16:4.0  
Big Decimal:283.34899999999998954081092961132526397705078125
```

Exception Handling Programs

A) Arithmetic Error

CODE:

```
class exphan1 {  
    public static void main(String[] args) {  
        try {  
            int a = 10, b = 0;  
            int result = a / b;  
            System.out.println("Result: " + result);  
        } catch (ArithmeticException e) {  
            System.out.println("Error: Cannot divide by zero!");  
        }  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java exphan1.java  
Error: Cannot divide by zero!
```

B) Index Error

CODE:

```
class exphan2 {  
    public static void main(String[] args) {  
        try {  
            int[] arr = {1, 2, 3};  
            System.out.println(arr[5]);  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("Error: Array index out of bounds!");  
        }  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java exphan2.java  
Error: Array index out of bounds!
```

C) Length Error**CODE:**

```
class exphan3 {  
    public static void main(String[] args) {  
        try {  
            String str = null;  
            System.out.println(str.length());  
        } catch (NullPointerException e) {  
            System.out.println("Error: Null reference encountered!");  
        }  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java exphan3.java
Error: Null reference encountered!
```

D) Age Error

CODE:

```
class exphan4 {
    static void intage(int age) {
        if (age<0) {
            throw new ArithmeticException("invalid age");
        }
        else {
            System.out.println("age:"+age);
        }
    }

    public static void main(String[] args) {
        exphan4 obj= new exphan4();
        obj.intage(-9);
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java exphan4.java
Exception in thread "main" java.lang.ArithmeticException: invalid age
    at exphan4.intage(exphan4.java:4)
    at exphan4.main(exphan4.java:13)
```

File Handling Programs

A) Read File

CODE:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

class filehan1 {
    public static void main(String[] args) {
        try {
            File file= new File("filehan1.txt");
            Scanner sc= new Scanner(file);
            while(sc.hasNextLine()) {

                String data=sc.nextLine();
                System.out.println(data);
            }
            sc.close();

        }
        catch (FileNotFoundException e){
            System.out.println("file not found");

        }
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java filehan1.java
etrcyvybiunioomp
wrextcvbunioomp
ewrxtcvubinm
wzextrtvubinom
fcgvhbkjn
```

B) Create File

CODE:

```
import java.io.File;
import java.io.IOException;

class filehan2 {
    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            if (file.createNewFile()) {
                System.out.println("File created: " + file.getName());
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java filehan2.java
File created: example.txt
```

C) Write File

CODE:

```
import java.io.FileWriter;
import java.io.IOException;

class filehan3 {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("example.txt");
            writer.write("Hello, this is a test file.");
            writer.close();
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java filehan3.java
Successfully wrote to the file.
```

D) Append File

CODE:

```
import java.io.FileWriter;
import java.io.IOException;
```

```
class filehan4 {  
    public static void main(String[] args) {  
        try {  
            FileWriter writer = new FileWriter("example.txt", true);  
            writer.write("\nAppending new line.");  
            writer.close();  
            System.out.println("Successfully appended to the file.");  
        } catch (IOException e) {  
            System.out.println("Error: " + e.getMessage());  
        }  
    }  
}
```

OUTPUT

```
C:\Users\cynth\Desktop\javapractise>java filehan4.java  
Successfully appended to the file.
```