LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24111 – Darunkumar. J

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025

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.U4CSE24111 – Darunkumar. J*** 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 13/03/2025

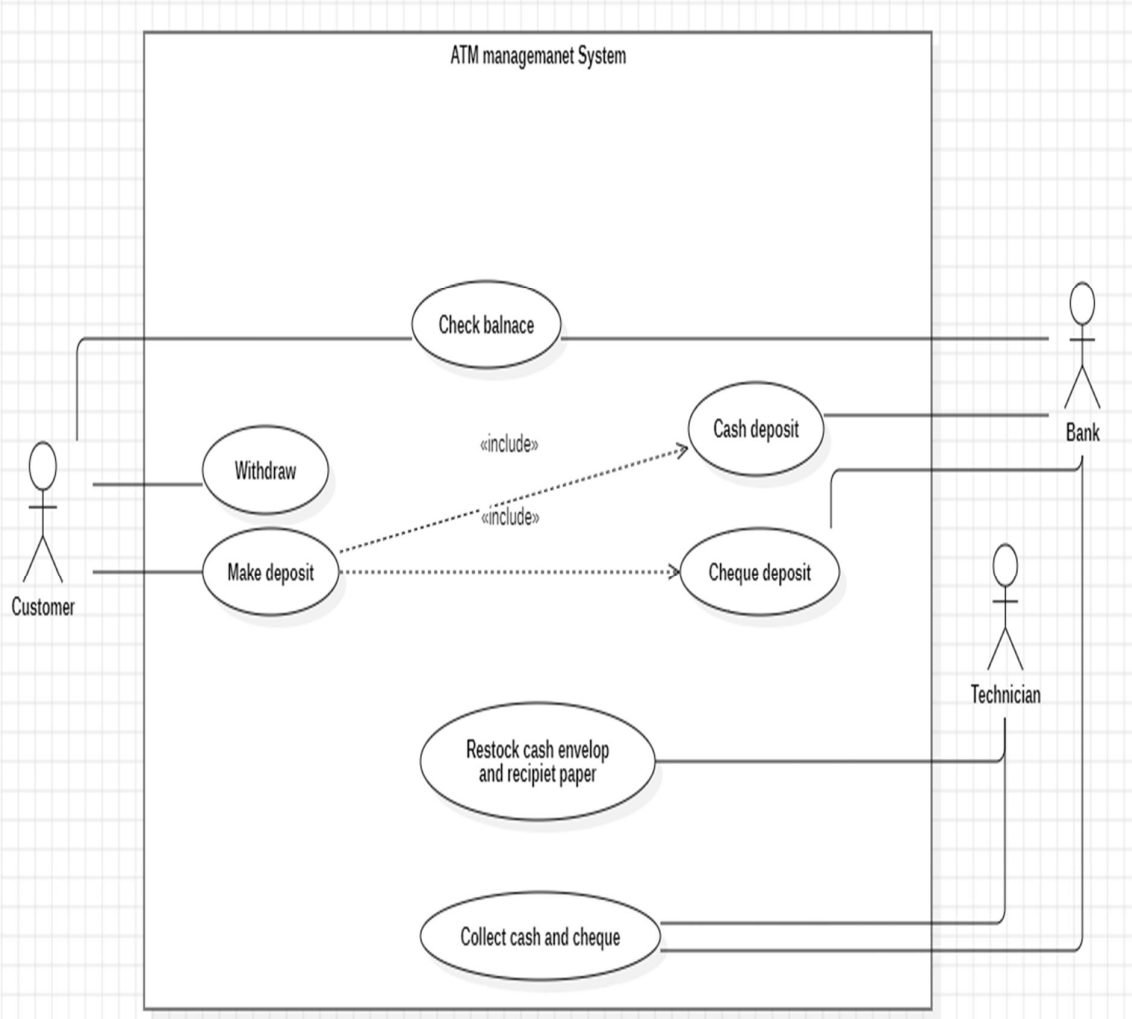
Internal Examiner 1 Internal Examiner 2

INDEX

UML DIAGRAMS

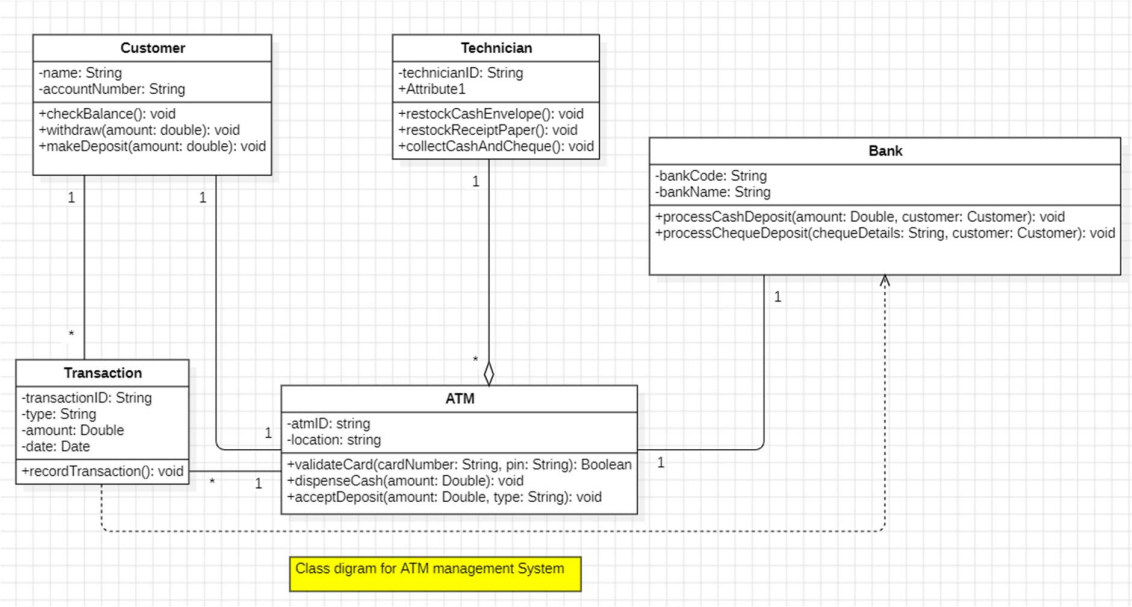
1. ATM MANAGEMENT SYSTEM

1.a) Use Case Diagram:



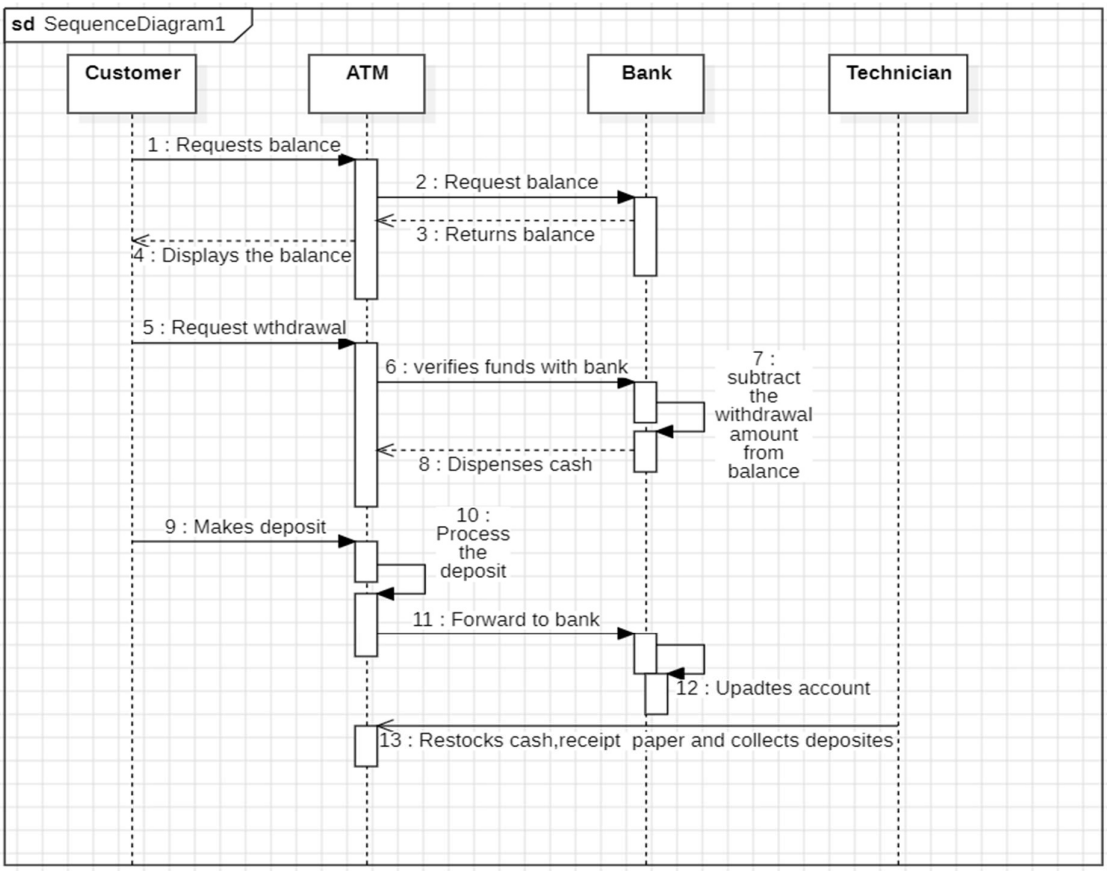
Use-case digram for ATM management System

1. b) Class Diagram:



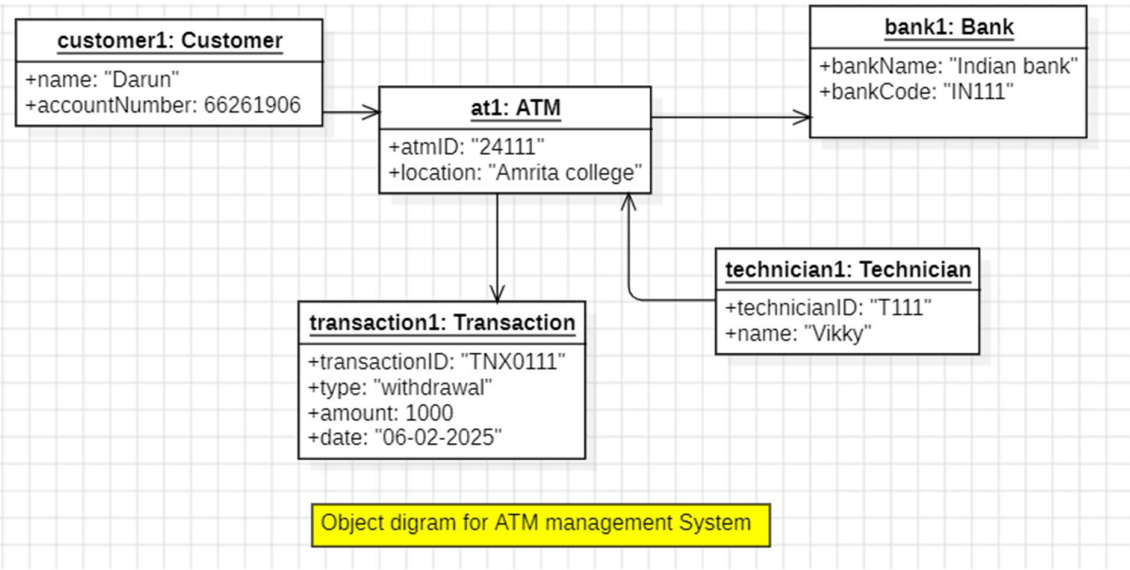
Class digram for ATM management System

1.c) Sequence Diagram:

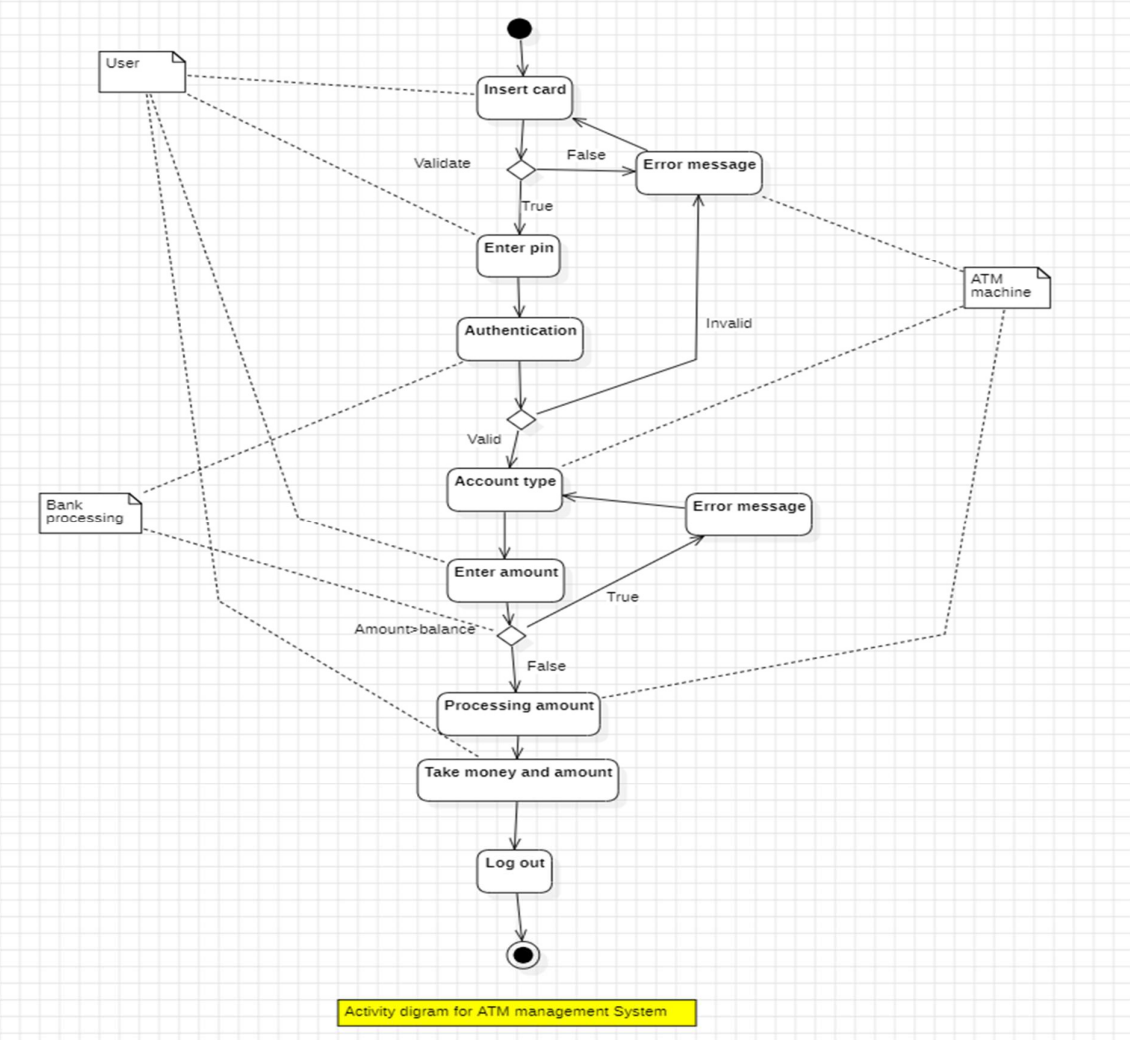


Sequence digram for ATM management System

1.d) Object Diagram:

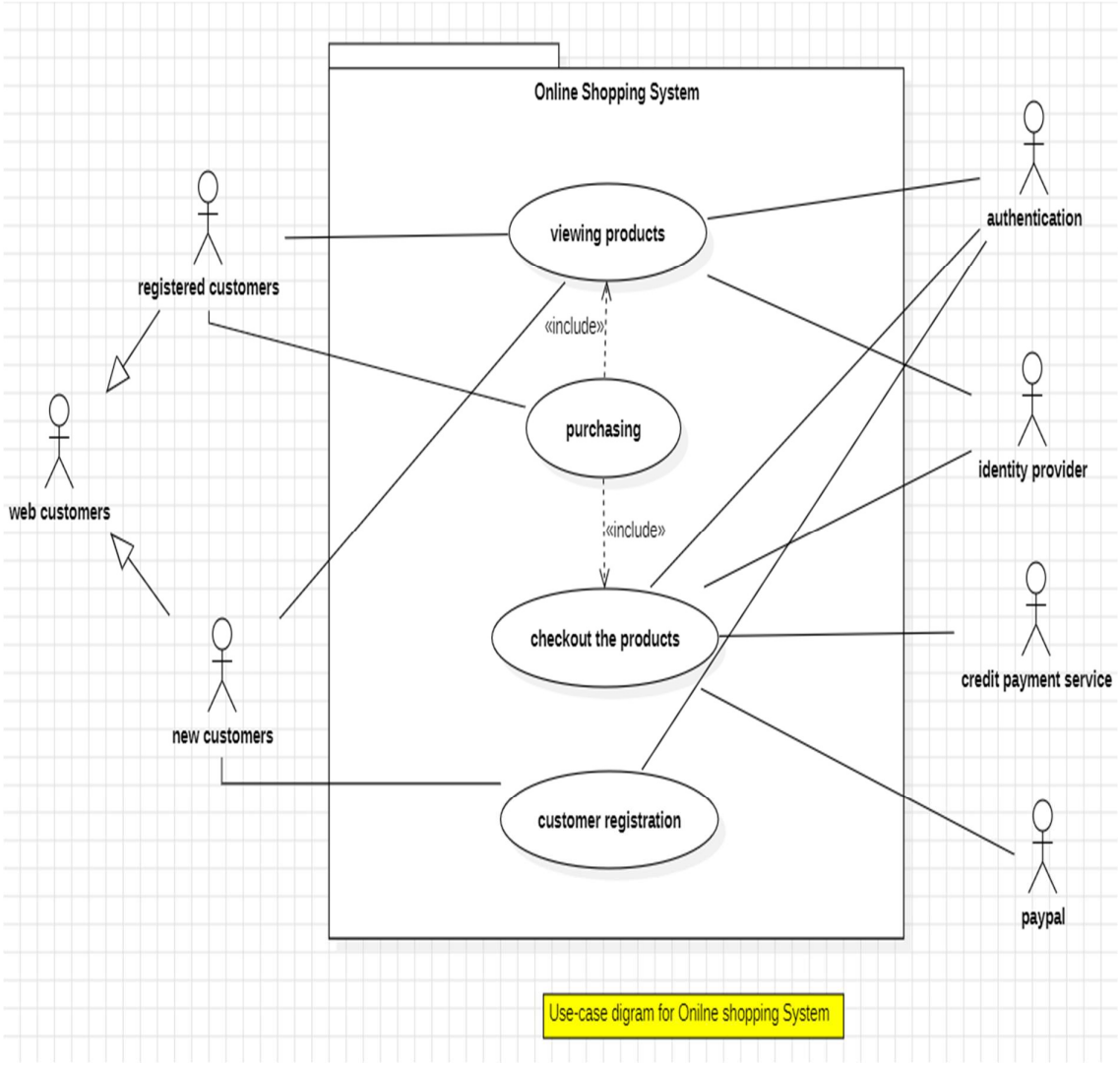


1.e) Activity Diagram:



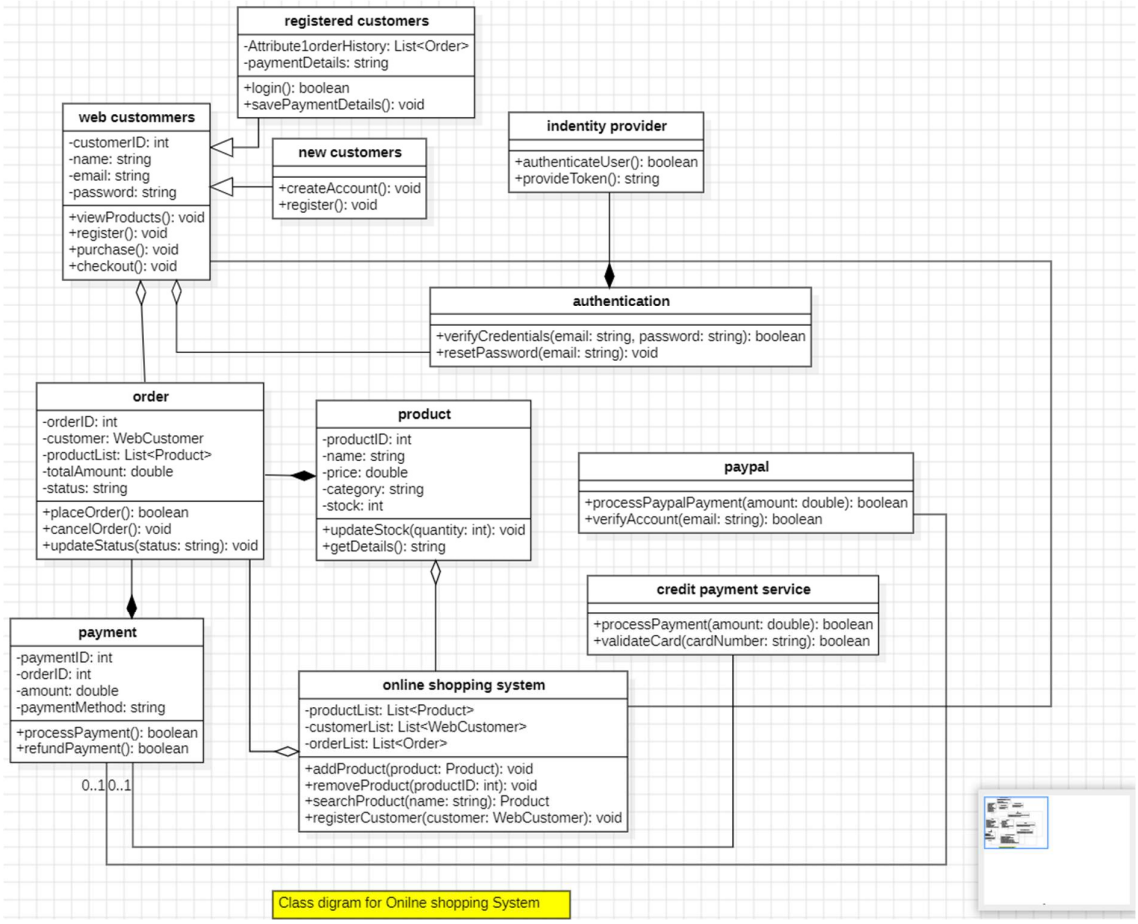
2. ONLINE SHOPPING SYSTEM

2. a) Use Case Diagram:

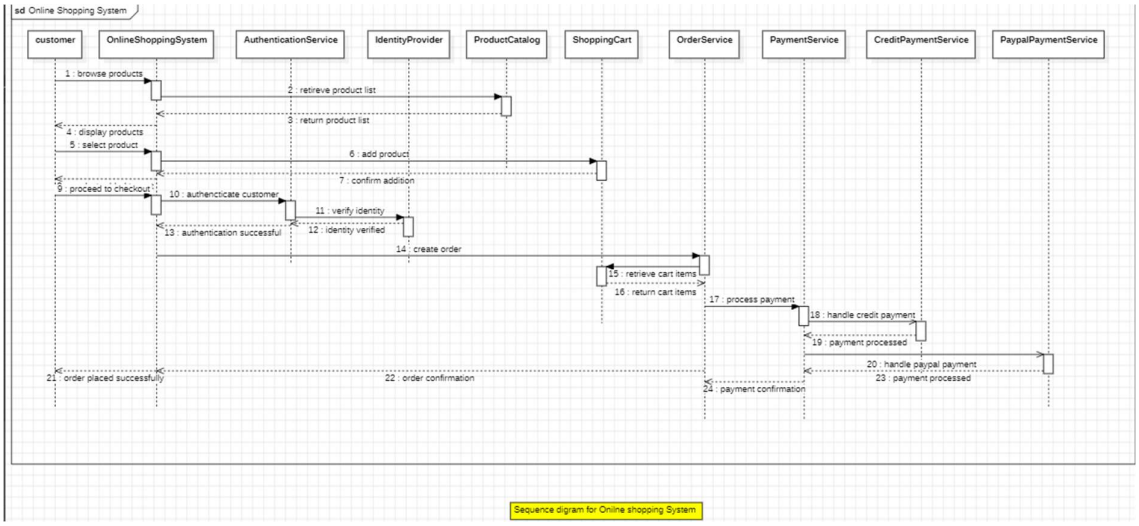


Use-case digram for Onilne shopping System

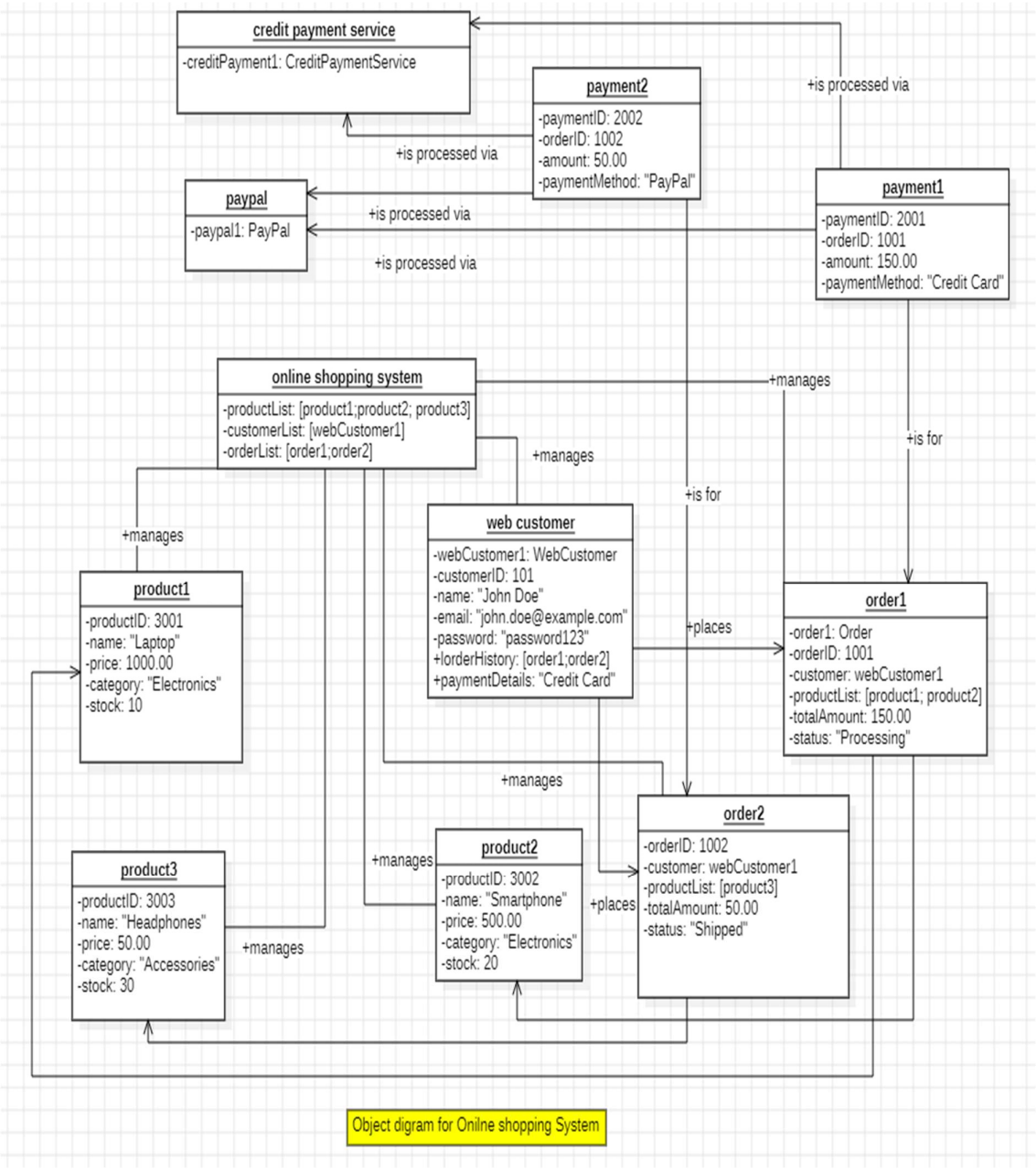
2. b) Class Diagram:



2. c) Sequence Diagram:

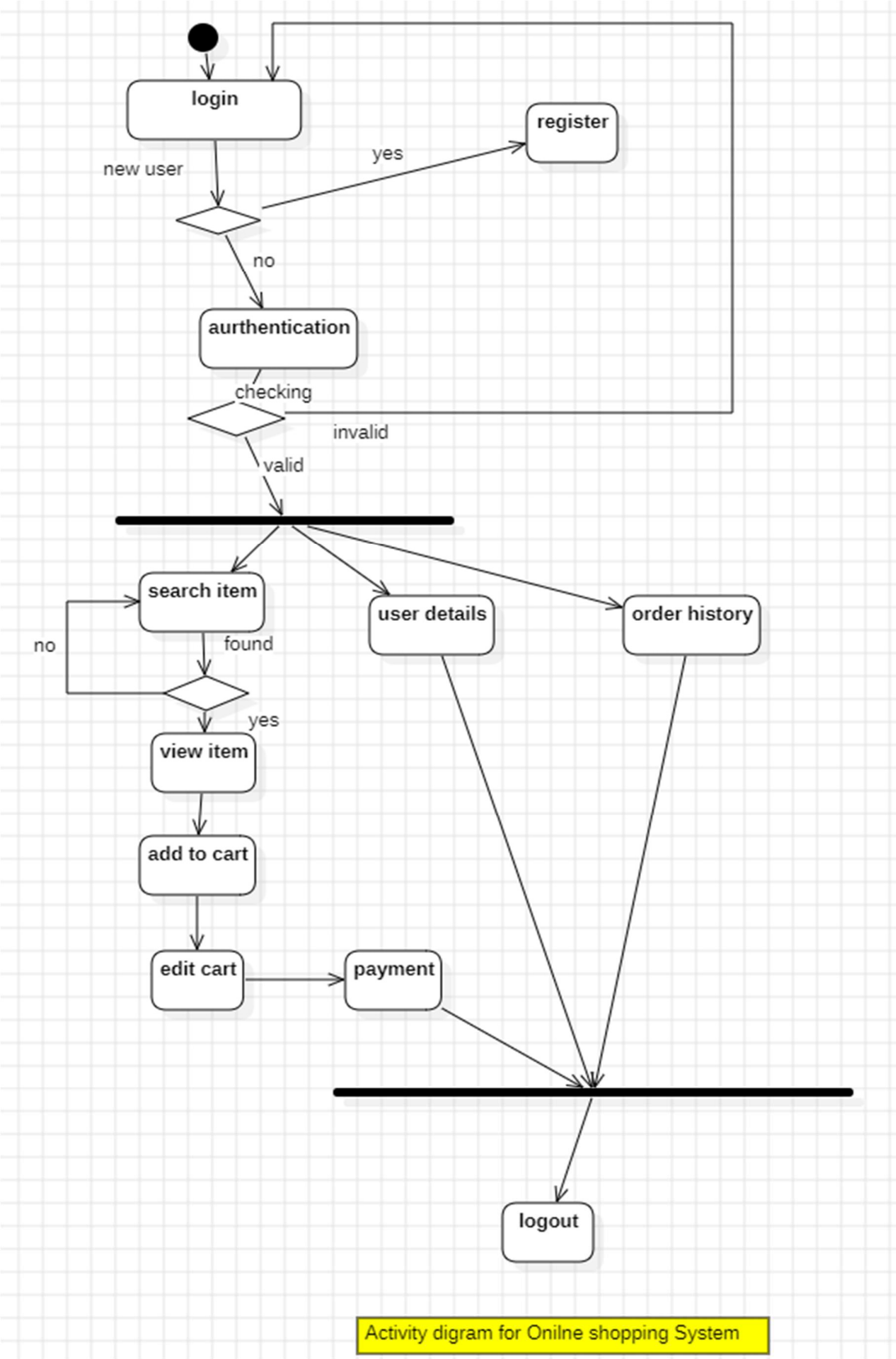


2.d) Object Diagram:



Object digram for Onilne shopping System

2.e) Activity Diagram:



Activity digram for Onilne shopping System

3.Basic Java Programs

3.a) Even or odd:

Code:

```java
import java.util.Scanner;

public class EvenOdd {
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter a number:");
        int n=input.nextInt();
        if (n%2==0){
            System.out.println(n+" is even");
        }

        else{
            System.out.println(n+" is odd");
        }
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac EvenOdd.java

C:\Sem-2\OOP\Basic java programs>java EvenOdd.java
Enter a number:10
10 is even
```

3.b) Factorial:

Code:

```java
import java.util.Scanner;

public class Factorial {
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter a number:");
        int n=input.nextInt();
        long f=1;
        for (int i = 1; i <= n; i++) {
            f*=i;
        }
        System.out.println("Factorial:" + f);
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac Factorial.java

C:\Sem-2\OOP\Basic java programs>java Factorial.java
Enter a number:10
Factorial:3628800
```

3.c) Grade:

Code:

```java
import java.util.Scanner;

public class Grade {
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);

        System.out.print(s:"Enter student's marks (0-100): ");
        int marks=input.nextInt();

        char grade;
        if (marks >= 90 && marks <= 100) {
            grade = 'A';
        }
        else if (marks >= 80) {
            grade = 'B';
        }
        else if (marks >= 70) {
            grade = 'C';
        }
        else if (marks >= 60) {
            grade = 'D';
        }
        else {
            grade = 'F';
        }

        System.out.println("Student's grade is " + grade);
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac Grade.java

C:\Sem-2\OOP\Basic java programs>java Grade.java
Enter student's marks (0-100): 99
Student's grade is A
```

3.d) Maximum of three numbers:

Code:

```java
import java.util.Scanner;

public class Max{
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter the first number:");
        int a=input.nextInt();
        System.out.print(s:"Enter the second number:");
        int b=input.nextInt();
        System.out.print(s:"Enter the third number:");
        int c=input.nextInt();
        if(a>b && a>c){
            System.out.println("The largest number is "+a);
        }

        else if(b>a && b>c){
            System.out.println("The largest number is "+b);
        }

        else{
            System.out.println("The largest number is "+c);
        }
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac Max.java

C:\Sem-2\OOP\Basic java programs>java Max.java
Enter the first number:3
Enter the second number:7
Enter the third number:26
The largest number is 26
```

3.e) Reverse a string:

Code:

```java
import java.util.Scanner;

public class Reverse{
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter a string:");
        String s=input.nextLine();
        String reverse=new StringBuilder(s).reverse().toString();
        System.out.println("The reversed String is " + reverse);
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac Reverse.java

C:\Sem-2\OOP\Basic java programs>java Reverse.java
Enter a string:Darun
The reversed String is nuraD
```

3.f) Simple Interest:

Code:

```java
import java.util.Scanner;

public class SimpleInterest {
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter Principal: ");
        double principal=input.nextDouble();
        System.out.print(s:"Enter Rate of Interest: ");
        double rate=input.nextDouble();
        System.out.print(s:"Enter Time (in years): ");
        double time=input.nextDouble();

        double interest=(principal * rate * time)/100;
        System.out.println("The simple interest is "+interest);
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac SimpleInterest.java

C:\Sem-2\OOP\Basic java programs>java SimpleInterest.java
Enter Principal: 1000
Enter Rate of Interest: 10
Enter Time (in years): 1
The simple interest is 100.0
```

3.g) Sum of two numbers:

Code:

```java
import java.util.Scanner;

public class Sum {
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter the first number:");
        int a=input.nextInt();
        System.out.print(s:"Enter the second number:");
        int b=input.nextInt();
        System.out.println("The sum of the two numbers is "+(a+b));
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac Sum.java

C:\Sem-2\OOP\Basic java programs>java Sum.java
Enter the first number:26
Enter the second number:28
The sum of the two numbers is 54
```

3.h) Sum of digits:

Code:

```java
import java.util.Scanner;

public class SumOfDigits{
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter a number: ");
        int num=input.nextInt();
        int sum=0;

        while(num!=0){
            sum +=num % 10;
            num /=10;
        }

        System.out.println("Sum of Digits:" + sum);
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac SumOfDigits.java

C:\Sem-2\OOP\Basic java programs>java SumOfDigits.java
Enter a number: 24157
Sum of Digits:19
```

3.i) Swapping two numbers:

Code:

```java
import java.util.Scanner;

public class Swap{
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter the first number:");
        int a=input.nextInt();
        System.out.print(s:"Enter the second number:");
        int b=input.nextInt();
        int temp=a;
        a=b;
        b=temp;
        System.out.println("The first number is:"+a);
        System.out.println("The second number is:"+b);
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac Swap.java

C:\Sem-2\OOP\Basic java programs>java Swap.java
Enter the first number:19
Enter the second number:06
The first number is:6
The second number is:19
```

3.j) Temperature converter:

Code:

```java
import java.util.Scanner;

public class TemperatureConverter{
    Run | Debug
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print(s:"Enter temperature in Celsius:");
        double c=input.nextDouble();
        double f=(c*9/5)+32;
        System.out.println("Temperature in Fahrenheit: "+f);
    }
}
```

Output:

```
C:\Sem-2\OOP\Basic java programs>javac TemperatureConverter.java

C:\Sem-2\OOP\Basic java programs>java TemperatureConverter.java
Enter temperature in Celsius:36
Temperature in Fahrenheit: 96.8
```

INHERITANCE

SINGLE INHERITANCE

4.a) single_inheritance1:

CODE:

```
   class Add{
    void add(int a,int b){
        System.out.println("The sum of two of numbers is "+(a+b));
    }
}

class calc extends Add{
    void subt(int a,int b){
        System.out.println("The difference of two of numbers is "+(a-
b));
    }
}

public class single_inheritance2{
    public static void main(String[] args) {
        calc obj = new calc();
        obj.add(10, 20);
        obj.subt(10, 20);
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac single_inheritance1.java

C:\Sem-2\OOP\Inheritance>java single_inheritance1.java
This is a BMW
```

4.b) single_inheritance2:

CODE:

```
class Add{
    void add(int a,int b){
        System.out.println("The sum of two of numbers is "+(a+b));
    }
}

class calc extends Add{
    void subt(int a,int b){
        System.out.println("The difference of two of numbers is "+(a-
b));
    }
}

public class single_inheritance2{
    public static void main(String[] args) {
```

```
        calc obj = new calc();
        obj.add(10, 20);
        obj.subt(10, 20);
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac single_inheritance2.java

C:\Sem-2\OOP\Inheritance>java single_inheritance2.java
The sum of two of numbers is 30
The difference of two of numbers is -10
```

MULTILEVEL INHERITANCE

5.a) multilevel_inheritance1:

CODE:

```
class Add{
    void add(int a,int b){
        System.out.println("The sum of two of numbers is "+(a+b));
    }
}

class basic extends Add{
    void subt(int a,int b){
```

```
        System.out.println("The difference of two of numbers is "+(a-
b));
    }
}

class calc extends basic{
    void mul(int a,int b){
        System.out.println("The product of two of numbers is
"+(a*b));
    }

    void div(int a,int b){
        if(b==0){
            System.out.println("Error! Division by zero is not
allowed");
        }
        else{
            System.out.println("The quotient of two of numbers is
"+(a/b));
        }
    }
}

public class multilevel_inheritance1{
    public static void main(String[] args) {
        calc obj = new calc();
        obj.add(10, 20);
        obj.subt(10, 20);
```

```
        obj.mul(10, 20);
        obj.div(10, 20);
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac multilevel_inheritance1.java

C:\Sem-2\OOP\Inheritance>java multilevel_inheritance1.java
The sum of two of numbers is 30
The difference of two of numbers is -10
The product of two of numbers is 200
The quotient of two of numbers is 0
```

5.b) multilevel_inheritance2:

CODE:

```
class Bank {
    void bankDetails() {
        System.out.println("This is a National Bank.");
    }
}

class Branch extends Bank {
    void branchDetails() {
        System.out.println("Branch: New York Main Branch");
    }
}
```

```
class Customer extends Branch {
    void customerDetails() {
        System.out.println("Customer Name: Eren jeager");
    }
}

public class multilevel_inheritance2 {
    public static void main(String[] args) {
        Customer c = new Customer();

        c.bankDetails();
        c.branchDetails();
        c.customerDetails();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac multilevel_inheritance2.java

C:\Sem-2\OOP\Inheritance>java multilevel_inheritance2.java
This is a National Bank.
Branch: New York Main Branch
Customer Name: Eren jeager
```

HIERARCHICAL INHERITANCE:

6.a) hierarchical_inheritance1:

CODE:

```
class Vehicle {
    void fuelType() {
        System.out.println("Most vehicles run on fuel.");
    }
}

class Car extends Vehicle {
    void carType() {
        System.out.println("Car Type: Sedan");
    }
}

class Bike extends Vehicle {
    void bikeType() {
        System.out.println("Bike Type: Sports Bike");
    }
}

public class hierarchical_inheritance1{
    public static void main(String[] args) {
        Car myCar = new Car();
```

```
        Bike myBike = new Bike();

        System.out.println("Car:");
        myCar.fuelType();
        myCar.carType();

        System.out.println("\nBike:");
        myBike.fuelType();
        myBike.bikeType();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac hierarchical_inheritance1.java

C:\Sem-2\OOP\Inheritance>java hierarchical_inheritance1.java
Car:
Most vehicles run on fuel.
Car Type: Sedan

Bike:
Most vehicles run on fuel.
Bike Type: Sports Bike
```

6.b) hierarchical_inheritance2:

CODE:

```
class BankAccount {
    void accountDetails() {
```

```
        System.out.println("This is a bank account.");
    }
}

class SavingsAccount extends BankAccount {
    void savingsFeatures() {
        System.out.println("Savings Account: Earns interest.");
    }
}

class CurrentAccount extends BankAccount {
    void currentFeatures() {
        System.out.println("Current Account: Suitable for
businesses with no interest.");
    }
}

public class hierarchical_inheritance2{
    public static void main(String[] args) {
        SavingsAccount savings = new SavingsAccount();
        CurrentAccount current = new CurrentAccount();

        System.out.println("Savings Account:");
        savings.accountDetails();
        savings.savingsFeatures();

        System.out.println("\nCurrent Account:");
```

```
        current.accountDetails();
        current.currentFeatures();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac hierarchical_inheritance2.java

C:\Sem-2\OOP\Inheritance>java hierarchical_inheritance2.java
Savings Account:
This is a bank account.
Savings Account: Earns interest.

Current Account:
This is a bank account.
Current Account: Suitable for businesses with no interest.
```

HYBRID INHERITANCE

7.a) hybrid_inheritance1:

CODE:

```
class CelestialBody {
    void showType() {
        System.out.println("This is a celestial body in space.");
    }
}
```

```
class Planet extends CelestialBody {
    void showPlanet() {
        System.out.println("This is a planet orbiting a star.");
    }
}

class Star extends CelestialBody {
    void showStar() {
        System.out.println("This is a star shining in the galaxy.");
    }
}

class Moon extends Planet {
    void showMoon() {
        System.out.println("This is a moon orbiting a planet.");
    }
}

class Galaxy extends Star {
    void showGalaxy() {
        System.out.println("This is a vast galaxy containing stars,
planets, and moons.");
    }
}

public class hybrid_inheritance1 {
    public static void main(String[] args) {
```

```
        Moon earthMoon = new Moon();
        earthMoon.showType();
        earthMoon.showPlanet();
        earthMoon.showMoon();

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

        Galaxy milkyWay = new Galaxy();
        milkyWay.showType();
        milkyWay.showStar();
        milkyWay.showGalaxy();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac hybrid_inheritance1.java

C:\Sem-2\OOP\Inheritance>java hybrid_inheritance1.java
This is a celestial body in space.
This is a planet orbiting a star.
This is a moon orbiting a planet.

----------------

This is a celestial body in space.
This is a star shining in the galaxy.
This is a vast galaxy containing stars, planets, and moons.
```

7.b) hybrid_inheritance2:

CODE:

```
class Character {
    void showCharacter() {
        System.out.println("This is a character from an anime.");
    }
}

class Hero extends Character {
    void showHero() {
        System.out.println("This is a hero with special powers.");
    }
}

class Villain extends Character {
    void showVillain() {
        System.out.println("This is a villain with dark ambitions.");
    }
}

class Sidekick extends Hero {
    void showSidekick() {
        System.out.println("This is a loyal sidekick who supports
the hero.");
    }
```

```
}

class Organization extends Villain {
    void showOrganization() {
        System.out.println("This is a secret organization of powerful villains.");
    }
}

public class hybrid_inheritance2{
    public static void main(String[] args) {
        Sidekick support = new Sidekick();
        support.showCharacter();
        support.showHero();
        support.showSidekick();

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

        Organization darkGroup = new Organization();
        darkGroup.showCharacter();
        darkGroup.showVillain();
        darkGroup.showOrganization();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Inheritance>javac hybrid_inheritance2.java

C:\Sem-2\OOP\Inheritance>java hybrid_inheritance2.java
This is a character from an anime.
This is a hero with special powers.
This is a loyal sidekick who supports the hero.

-----------------

This is a character from an anime.
This is a villain with dark ambitions.
This is a secret organization of powerful villains.
```

POLYMORPHISM

CONSTRUCTOR

8.a) contructor:

CODE:

```
class Person {
    String name;
    int age;
    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name + " , Age: " + age);
    }}

public class constructor{

    public static void main(String[] args) {
        Person p3 = new Person("Kevin", 18);

        p3.display();
```

```
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Polymorphism>javac constructor.java

C:\Sem-2\OOP\Polymorphism>java constructor.java
Name: Kevin , Age: 18
```

CONSTRUCTOR OVERLOADING PROGRAMS

9.a) constructor_overloading1:

CODE:

```
class AnimeCharacter {
    String name;
    String power;
    int age;

    AnimeCharacter() {
        name = "Unknown";
        power = "None";
        age = 0;
    }

    AnimeCharacter(String n) {
```

```
        name = n;
        power = "Unknown";
        age = 0;
    }

    AnimeCharacter(String n, String p) {
        name = n;
        power = p;
        age = 0;
    }

    AnimeCharacter(String n, String p, int a) {
        name = n;
        power = p;
        age = a;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Power: " + power);
        System.out.println("Age: " + age);
        System.out.println();
    }
}

public class constructor_overloading1 {
    public static void main(String[] args) {
```

```
        AnimeCharacter c1 = new AnimeCharacter();
        AnimeCharacter c2 = new AnimeCharacter("Naruto");
        AnimeCharacter c3 = new AnimeCharacter("Sasuke",
"Sharingan");
        AnimeCharacter c4 = new AnimeCharacter("Goku", "Ultra
Instinct", 35);


        c1.display();
        c2.display();
        c3.display();
        c4.display();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Polymorphism>javac constructor_overloading1.java

C:\Sem-2\OOP\Polymorphism>java constructor_overloading1.java
Name: Unknown
Power: None
Age: 0

Name: Naruto
Power: Unknown
Age: 0

Name: Sasuke
Power: Sharingan
Age: 0

Name: Goku
Power: Ultra Instinct
Age: 35
```

9.b) constructor_overloading2:

CODE:

```
class SpaceMission {
    String missionName;
    int crewCount;
    double duration;

    SpaceMission() {
        missionName = "Unknown";
        crewCount = 0;
        duration = 0.0;
    }

    SpaceMission(String name) {
        missionName = name;
        crewCount = 0;
        duration = 0.0;
    }

    SpaceMission(String name, int crew) {
        missionName = name;
        crewCount = crew;
        duration = 0.0;
    }
```

```
    SpaceMission(String name, int crew, double time) {
        missionName = name;
        crewCount = crew;
        duration = time;
    }

    void display() {
        System.out.println("Mission: " + missionName);
        System.out.println("Crew Count: " + crewCount);
        System.out.println("Duration: " + duration + " days");
        System.out.println();
    }
}

public class constructor_overloading2 {
    public static void main(String[] args) {
        SpaceMission m1 = new SpaceMission();
        SpaceMission m2 = new SpaceMission("Apollo 11");
        SpaceMission m3 = new SpaceMission("ISS Expedition", 6);
        SpaceMission m4 = new SpaceMission("Mars One", 4,
180.5);

        m1.display();
        m2.display();
        m3.display();
        m4.display();
    }
```

}

OUTPUT:

```
C:\Sem-2\OOP\Polymorphism>javac constructor_overloading2.java

C:\Sem-2\OOP\Polymorphism>java constructor_overloading2.java
Mission: Unknown
Crew Count: 0
Duration: 0.0 days

Mission: Apollo 11
Crew Count: 0
Duration: 0.0 days

Mission: ISS Expedition
Crew Count: 6
Duration: 0.0 days

Mission: Mars One
Crew Count: 4
Duration: 180.5 days
```

METHOD OVERLOADING PROGRAMS

10.a) method_overloading1:

CODE:

```
class Superhero {
    void power() {
        System.out.println("Superhero uses a mysterious power!");
    }

    void power(String type) {
```

```
        System.out.println("Superhero uses " + type + " power!");
    }

    void power(String type, int level) {
        System.out.println("Superhero uses " + type + " power at
level " + level + "!");
    }
}

public class method_overloading1 {
    public static void main(String[] args) {
        Superhero hero = new Superhero();

        hero.power();
        hero.power("Lightning");
        hero.power("Fire", 9000);
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Polymorphism>javac method_overloading1.java

C:\Sem-2\OOP\Polymorphism>java method_overloading1.java
Superhero uses a mysterious power!
Superhero uses Lightning power!
Superhero uses Fire power at level 9000!
```

10.b) method_overloading2:

CODE:

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

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

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

public class method_overloading2 {
    public static void main(String[] args) {
        Calculator calc = new Calculator();

        System.out.println("Sum (int, int): " + calc.add(5, 10));
        System.out.println("Sum (double, double): " + calc.add(5.5,
3.2));
        System.out.println("Sum (int, int, int): " + calc.add(1, 2, 3));
    }
```

```
}
```

OUTPUT:

```
C:\Sem-2\OOP\Polymorphism>javac method_overloading2.java

C:\Sem-2\OOP\Polymorphism>java method_overloading2.java
Sum (int, int): 15
Sum (double, double): 8.7
Sum (int, int, int): 6
```

METHOD OVERRIDING PROGRAMS

11.a) method_overriding1:

CODE:

```
class Spell {
    void cast() {
        System.out.println("Casting a generic magical spell...");
    }
}

class FireSpell extends Spell {
    @Override
    void cast() {
        System.out.println("Casting Fireball! 🔥");
    }
}
```

```
class IceSpell extends Spell {
    @Override
    void cast() {
        System.out.println("Casting Ice Storm! ❄");
    }
}

public class method_overriding1 {
    public static void main(String[] args) {
        Spell baseSpell = new Spell();
        Spell fire = new FireSpell();
        Spell ice = new IceSpell();

        baseSpell.cast();
        fire.cast();
        ice.cast();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Polymorphism>javac method_overriding1.java

C:\Sem-2\OOP\Polymorphism>java method_overriding1.java
Casting a generic magical spell...
Casting Fireball! ?
Casting Ice Storm! ??
```

11.b) method_overriding2:

CODE:

```
class Delivery {
    void deliver() {
        System.out.println("Delivering a package...");
    }
}

class FoodDelivery extends Delivery {
    @Override
    void deliver() {
        System.out.println("Delivering hot food to your doorstep! ");
    }
}

class GroceryDelivery extends Delivery {
    @Override
    void deliver() {
        System.out.println("Delivering fresh groceries to your home!
");
    }
}

public class method_overrinding2 {
```

```
    public static void main(String[] args) {
        Delivery d1 = new Delivery();
        Delivery d2 = new FoodDelivery();
        Delivery d3 = new GroceryDelivery();

        d1.deliver();
        d2.deliver();
        d3.deliver();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Polymorphism>javac method_overriding2.java

C:\Sem-2\OOP\Polymorphism>java method_overriding2.java
Delivering a package...
Delivering hot food to your doorstep! ?
Delivering fresh groceries to your home! ?
```

ABSTRACTION

INTERFACE PROGRAMS

12.a) interface1:

CODE:

```
interface Music {
    void play();
}

class Guitar implements Music {
    public void play() {
        System.out.println("Playing guitar melody...");
    }
}

class Piano implements Music {
    public void play() {
        System.out.println("Playing piano harmony...");
    }
}

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

```
        Music m1 = new Guitar();
        Music m2 = new Piano();
        m1.play();
        m2.play();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Abstraction>javac interface1.java

C:\Sem-2\OOP\Abstraction>java interface1.java
Playing guitar melody...
Playing piano harmony...
```

12.b) interface2:

CODE:

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

class Dog implements Animal {
    public void sound() {
        System.out.println("Dog says: Woof!");
    }
}
```

```
class Cat implements Animal {
    public void sound() {
        System.out.println("Cat says: Meow!");
    }
}

public class interface2 {
    public static void main(String[] args) {
        Animal d = new Dog();
        Animal c = new Cat();
        d.sound();
        c.sound();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Abstraction>javac interface2.java

C:\Sem-2\OOP\Abstraction>java interface2.java
Dog says: Woof!
Cat says: Meow!
```

12.c) interface3:

CODE:

```
interface Payment {
    void makePayment();
}

class CreditCard implements Payment {
    public void makePayment() {
        System.out.println("Payment made using Credit Card.");
    }
}

class UPI implements Payment {
    public void makePayment() {
        System.out.println("Payment made using UPI.");
    }
}

public class interface3 {
    public static void main(String[] args) {
        Payment p1 = new CreditCard();
        Payment p2 = new UPI();
        p1.makePayment();
        p2.makePayment();
    }
```

```
}
```

OUTPUT:

```
C:\Sem-2\OOP\Abstraction>javac interface3.java

C:\Sem-2\OOP\Abstraction>java interface3.java
Payment made using Credit Card.
Payment made using UPI.
```

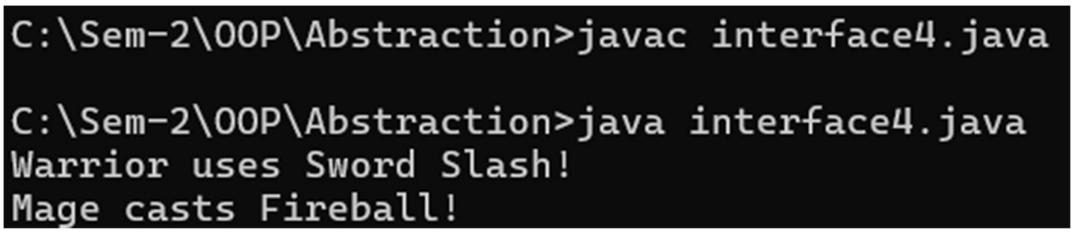
12.d) interface4:

CODE:

```
interface Ability {
    void useAbility();
}

class Warrior implements Ability {
    public void useAbility() {
        System.out.println("Warrior uses Sword Slash!");
    }
}

class Mage implements Ability {
    public void useAbility() {
        System.out.println("Mage casts Fireball!");
    }
```

```
}

public class interface4 {
    public static void main(String[] args) {
        Ability a1 = new Warrior();
        Ability a2 = new Mage();
        a1.useAbility();
        a2.useAbility();
    }
}
```

OUTPUT:



ABSTRACT CLASS PROGRAMS

13.a) abstraction1:

CODE:

```
abstract class Course {
    abstract void enroll();
    abstract void complete();
}

class JavaCourse extends Course {
    void enroll() {
        System.out.println("Enrolled in Java course.");
    }
    void complete() {
        System.out.println("Completed Java course!");
    }
}

class PythonCourse extends Course {
    void enroll() {
        System.out.println("Enrolled in Python course.");
    }
    void complete() {
        System.out.println("Completed Python course!");
    }
```

```
}

public class abstraction1 {
    public static void main(String[] args) {
        Course c1 = new JavaCourse();
        Course c2 = new PythonCourse();
        c1.enroll();
        c1.complete();
        c2.enroll();
        c2.complete();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Abstraction>javac abstraction1.java

C:\Sem-2\OOP\Abstraction>java abstraction1.java
Enrolled in Java course.
Completed Java course!
Enrolled in Python course.
Completed Python course!
```

13.b) abstraction2:

CODE:

```
abstract class SmartDevice {
    abstract void turnOn();
    abstract void turnOff();
```

```
}

class SmartTV extends SmartDevice {
    void turnOn() {
        System.out.println("Smart TV is now ON.");
    }
    void turnOff() {
        System.out.println("Smart TV is now OFF.");
    }
}

class SmartSpeaker extends SmartDevice {
    void turnOn() {
        System.out.println("Smart Speaker is now ON.");
    }
    void turnOff() {
        System.out.println("Smart Speaker is now OFF.");
    }
}

public class abstraction2 {
    public static void main(String[] args) {
        SmartDevice d1 = new SmartTV();
        SmartDevice d2 = new SmartSpeaker();
        d1.turnOn();
        d1.turnOff();
        d2.turnOn();
```

```
        d2.turnOff();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Abstraction>javac abstraction2.java

C:\Sem-2\OOP\Abstraction>java abstraction2.java
Smart TV is now ON.
Smart TV is now OFF.
Smart Speaker is now ON.
Smart Speaker is now OFF.
```

13.c) abstraction3:

CODE:

```
abstract class DeliveryApp {
    abstract void orderFood();
    abstract void trackOrder();
}

class Swiggy extends DeliveryApp {
    void orderFood() {
        System.out.println("Ordering food through Swiggy.");
    }
    void trackOrder() {
        System.out.println("Tracking order on Swiggy.");
```

```
    }
}

class Zomato extends DeliveryApp {
    void orderFood() {
        System.out.println("Ordering food through Zomato.");
    }
    void trackOrder() {
        System.out.println("Tracking order on Zomato.");
    }
}

public class abstraction3 {
    public static void main(String[] args) {
        DeliveryApp app1 = new Swiggy();
        DeliveryApp app2 = new Zomato();
        app1.orderFood();
        app1.trackOrder();
        app2.orderFood();
        app2.trackOrder();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Abstraction>javac abstraction3.java

C:\Sem-2\OOP\Abstraction>java abstraction3.java
Ordering food through Swiggy.
Tracking order on Swiggy.
Ordering food through Zomato.
Tracking order on Zomato.
```

13.d) abstraction4:

CODE:

```
abstract class Restaurant {
    abstract void orderFood();
    abstract void serveFood();
}

class ItalianRestaurant extends Restaurant {
    void orderFood() {
        System.out.println("Ordering pasta from Italian
Restaurant.");
    }

    void serveFood() {
        System.out.println("Serving spaghetti and garlic bread.");
    }
}
```

```
class IndianRestaurant extends Restaurant {
    void orderFood() {
        System.out.println("Ordering biryani from Indian
Restaurant.");
    }

    void serveFood() {
        System.out.println("Serving biryani with raita.");
    }
}

public class abstraction4 {
    public static void main(String[] args) {
        Restaurant r1 = new ItalianRestaurant();
        Restaurant r2 = new IndianRestaurant();

        r1.orderFood();
        r1.serveFood();

        r2.orderFood();
        r2.serveFood();
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Abstraction>javac abstraction4.java

C:\Sem-2\OOP\Abstraction>java abstraction4.java
Ordering pasta from Italian Restaurant.
Serving spaghetti and garlic bread.
Ordering biryani from Indian Restaurant.
Serving biryani with raita.
```

ENCAPSULATION

ENCAPSULATION PROGRAMS

14.a) encapsulation1

CODE:

```
import java.util.Scanner;

class Student {
    private String name;
    private int age;

    public void setName(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public void setAge(int a) {
        if (a > 0) {
            age = a;
        }
```

```
    }

    public int getAge() {
        return age;
    }
}

public class encapsulation1{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student s = new Student();

        System.out.print("Enter name: ");
        s.setName(sc.nextLine());

        System.out.print("Enter age: ");
        s.setAge(sc.nextInt());

        System.out.println("Student Name: " + s.getName());
        System.out.println("Student Age: " + s.getAge());
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac encapsulation1.java

C:\Sem-2\OOP\Encapsulation>java encapsulation1.java
Enter name: darun
Enter age: 18
Student Name: darun
Student Age: 18
```

14.b) encapsulation2

CODE:

```
import java.util.Scanner;

class BankAccount {
    private double balance;

    public void deposit(double amount) {
        if (amount > 0)
            balance += amount;
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance)
            balance -= amount;
    }
```

```
    public double getBalance() {
        return balance;
    }
}

public class encapsulation2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        BankAccount account = new BankAccount();

        System.out.print("Enter deposit amount: ");
        account.deposit(sc.nextDouble());

        System.out.print("Enter withdrawal amount: ");
        account.withdraw(sc.nextDouble());

        System.out.println("Current Balance: " +
account.getBalance());
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac encapsulation2.java

C:\Sem-2\OOP\Encapsulation>java encapsulation2.java
Enter deposit amount: 1906
Enter withdrawal amount: 500
Current Balance: 1406.0
```

14.c) encapsulation3

CODE:

```
import java.util.Scanner;

class Employee {
    private String name;
    private double salary;

    public void setName(String n) {
        name = n;
    }

    public void setSalary(double s) {
        if (s > 0)
            salary = s;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }
```

```
}

public class encapsulation3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Employee emp = new Employee();

        System.out.print("Enter employee name: ");
        emp.setName(sc.nextLine());

        System.out.print("Enter salary: ");
        emp.setSalary(sc.nextDouble());

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

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac encapsulation3.java

C:\Sem-2\OOP\Encapsulation>java encapsulation3.java
Enter employee name: haleetha
Enter salary: 19110603
Employee: haleetha
Salary: $1.9110603E7
```

14.d) encapsulation4

CODE:

```
import java.util.Scanner;

class Product {
    private String productName;
    private double price;

    public void setProductName(String name) {
        productName = name;
    }

    public void setPrice(double p) {
        if (p > 0)
            price = p;
    }

    public String getProductName() {
        return productName;
    }

    public double getPrice() {
        return price;
    }
}
```

```
public class encapsulation4{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Product p = new Product();

        System.out.print("Enter product name: ");
        p.setProductName(sc.nextLine());

        System.out.print("Enter price: ");
        p.setPrice(sc.nextDouble());

        System.out.println("Product: " + p.getProductName());
        System.out.println("Price: $" + p.getPrice());
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac encapsulation4.java

C:\Sem-2\OOP\Encapsulation>java encapsulation4.java
Enter product name: loq laptop
Enter price: 99999
Product: loq laptop
Price: $99999.0
```

PACKAGES PROGRAMS

15.a) package1:

CODE:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.Date;
import java.text.SimpleDateFormat;

public class package1 {
    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            Scanner scanner = new Scanner(file);
            System.out.println("File content:");

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                System.out.println(line);
            }

            // Show formatted current date
            Date now = new Date();
```

```
            SimpleDateFormat formatter = new
SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
            System.out.println("\nRead on: " +
formatter.format(now));


            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac package1.java

C:\Sem-2\OOP\Encapsulation>java package1.java
File not found.

C:\Sem-2\OOP\Encapsulation>javac package1.java

C:\Sem-2\OOP\Encapsulation>java package1.java
File content:
one piece is greatest of all time. I'm not saying it's the best anime ever, but it's definitely on the list. The story i
s engaging, the characters are well-developed, and the animation is top 3.

Read on: 05-04-2025 15:38:28
```

15.b) package2:

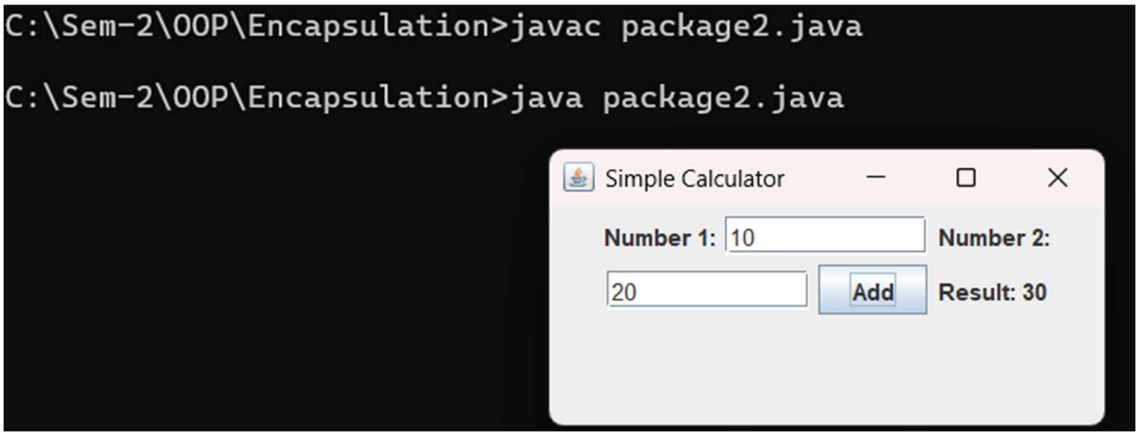
CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.math.BigDecimal;
```

```
public class package2 {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Simple Calculator");
        JTextField field1 = new JTextField(10);
        JTextField field2 = new JTextField(10);
        JButton addButton = new JButton("Add");
        JLabel resultLabel = new JLabel("Result: ");

        JPanel panel = new JPanel();
        panel.add(new JLabel("Number 1:"));
        panel.add(field1);
        panel.add(new JLabel("Number 2:"));
        panel.add(field2);
        panel.add(addButton);
        panel.add(resultLabel);

        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                BigDecimal num1 = new BigDecimal(field1.getText());
                BigDecimal num2 = new BigDecimal(field2.getText());
                BigDecimal result = num1.add(num2);
                resultLabel.setText("Result: " + result.toString());
            }
        });

        frame.add(panel);
```

```
        frame.setSize(300, 150);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

OUTPUT:



15.c) math:

CODE:

mainmath.java:

```
package math;
import math.mypack.calc;

public class mainmath {
```

```
    public static void main(String[] args) {
        calc calc = new calc();

        System.out.println("Add: 5 + 3 = " + calc.add(5, 3));
        System.out.println("Factorial of 5: " + calc.factorial(5));
        System.out.println("2^4 = " + calc.power(2, 4));
    }
}
```

Calc.java:

```
package math.mypack;

public class calc {
    public int add(int a, int b) {
        return a + b;
    }

    public int factorial(int n) {
        if (n <= 1) return 1;
        return n * factorial(n - 1);
    }

    public double power(double base, int exponent) {
        return Math.pow(base, exponent);
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation\math>cd C:\Sem-2\OOP\Encapsulation

C:\Sem-2\OOP\Encapsulation>javac math\mypack\calc.java

C:\Sem-2\OOP\Encapsulation>javac math\mainmath.java

C:\Sem-2\OOP\Encapsulation>java math\mainmath.java
Add: 5 + 3 = 8
Factorial of 5: 120
2^4 = 16.0
```

15.d) science:

CODE:

Sci.java:

```
package science;
import science.phy.phy;

public class sci {
    public static void main(String[] args) {
        phy physics = new phy();

        System.out.println("Force (F = m * a): " +
physics.calculateForce(10, 9.8));
        System.out.println("Kinetic Energy (KE = 0.5 * m * v^2): " +
physics.calculateKineticEnergy(2, 3));
```

```
        System.out.println("Potential Energy (PE = m * g * h): " +
physics.calculateGravitationalPotentialEnergy(2, 10, 9.8));
    }
}
```

Phy.java:

```
package science.phy;


public class phy {
    public double calculateForce(double mass, double
acceleration) {
        return mass * acceleration;
    }


    public double calculateKineticEnergy(double mass, double
velocity) {
        return 0.5 * mass * velocity * velocity;
    }


    public double calculateGravitationalPotentialEnergy(double
mass, double height, double g) {
        return mass * g * height;
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac science\sci.java

C:\Sem-2\OOP\Encapsulation>java science\sci.java
Force (F = m * a): 98.0
Kinetic Energy (KE = 0.5 * m * v^2): 9.0
Potential Energy (PE = m * g * h): 196.0
```

EXCEPTION HANDLING PROGRAMS

16.a) exception_handling1

CODE:

```
public class exception_handling1 {
    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: Division by zero is not
allowed!");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac exception_handling1.java

C:\Sem-2\OOP\Encapsulation>java exception_handling1.java
Error: Division by zero is not allowed!
```

16.b) exception_handling2

CODE:

```
public class exception_handling2 {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3};
            System.out.println(numbers[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Array index is out of bounds!");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac exception_handling2.java

C:\Sem-2\OOP\Encapsulation>java exception_handling2.java
Error: Array index is out of bounds!
```

16.c) exception_handling3

CODE:

```
public class exception_handling3 {
    public static void main(String[] args) {
        try {
            String name = null;
            System.out.println(name.length());
        } catch (NullPointerException e) {
            System.out.println("Error: Attempted to access a null
object!");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac exception_handling3.java

C:\Sem-2\OOP\Encapsulation>java exception_handling3.java
Error: Attempted to access a null object!
```

16.d) exception_handling4

CODE:

```
    public class exception_handling4 {
        public static void main(String[] args) {
```

```
        try {
            String input = "abc";
            int num = Integer.parseInt(input);
            System.out.println("Number: " + num);
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format!");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac exception_handling4.java

C:\Sem-2\OOP\Encapsulation>java exception_handling4.java
Error: Invalid number format!
```

FILE HANDLING PROGRAMS

17.a) file_handling1:

CODE:

```
import java.io.File;

public class file_handling1 {
    public static void main(String[] args) {
        File file = new File("output.txt");
```

```
        if (file.delete()) {
            System.out.println("File deleted successfully.");
        } else {
            System.out.println("File could not be deleted.");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac file_handling1.java

C:\Sem-2\OOP\Encapsulation>java file_handling1.java
File deleted successfully.
```

17.b) file_handling2:

CODE:

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

public class file_handling2 {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("output.txt", true); //
true enables appending
            writer.write("\nAdding more data about BMW cars.");
```

```
            writer.close();
            System.out.println("Data appended successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred while appending to
the file.");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac file_handling2.java

C:\Sem-2\OOP\Encapsulation>java file_handling2.java
Data appended successfully.
```

17.c) file_handling3:

CODE:

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

public class file_handling3 {
    public static void main(String[] args) {
        try {
            File file = new File("output.txt");
```

```
            Scanner reader = new Scanner(file);
            while (reader.hasNextLine()) {
                String data = reader.nextLine();
                System.out.println("File content: " + data);
            }
            reader.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac file_handling3.java

C:\Sem-2\OOP\Encapsulation>java file_handling3.java
File content:
File content: Adding more data about BMW cars.
```

17.d) file_handling4:

CODE:

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

public class file_handling4 {
```

```
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("output.txt");
            writer.write("Hello, this is a BMW file handling example.");
            writer.close();
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the
file.");
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
C:\Sem-2\OOP\Encapsulation>javac file_handling4.java

C:\Sem-2\OOP\Encapsulation>java file_handling4.java
Successfully wrote to the file.
```