LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24116- **D. Surya Kiran**

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.U4CSE24116 – D. Surya Kiran** 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 / /2025

Internal Examiner 1 Internal Examiner 2

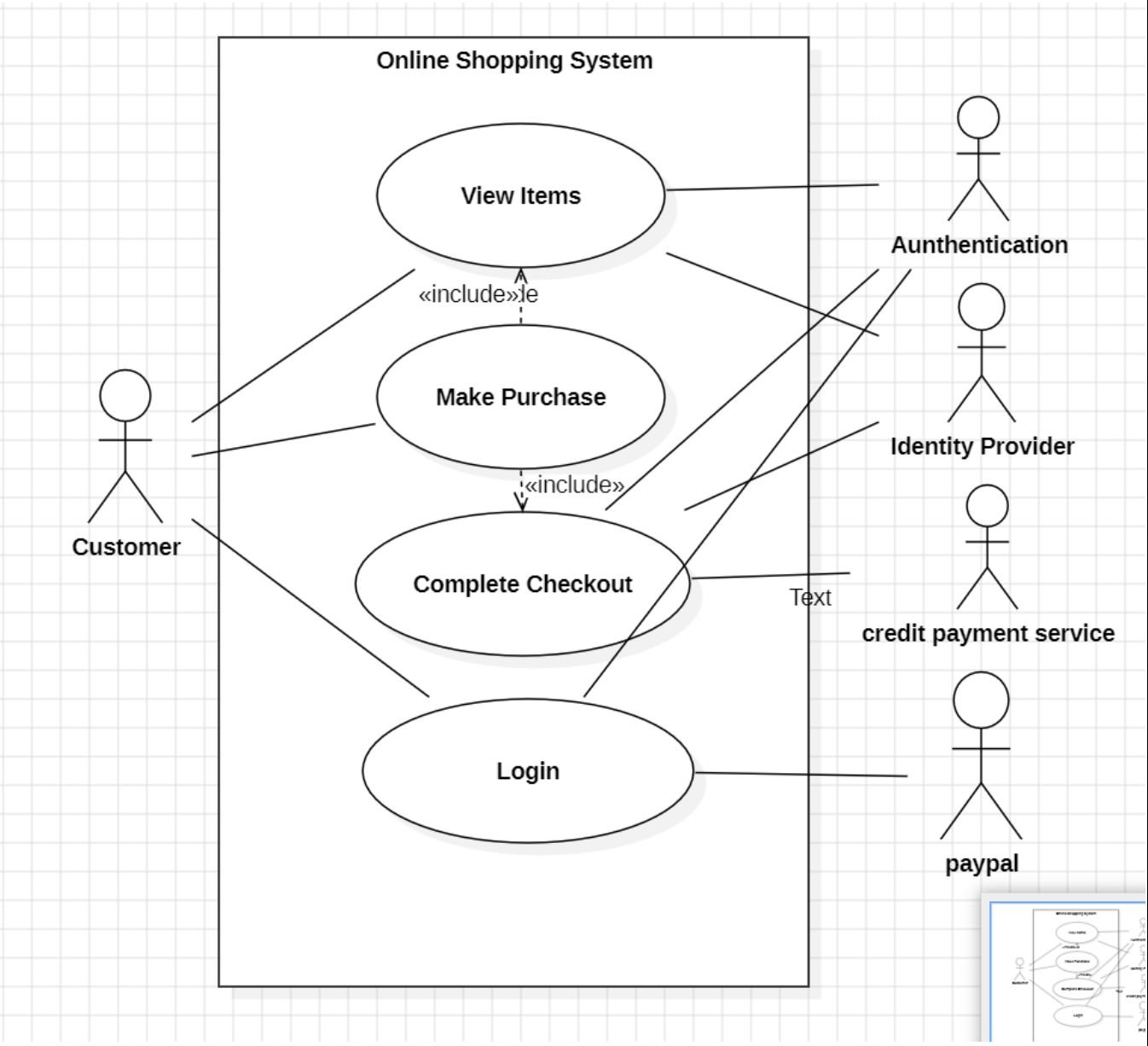
INDEX

5.	**MULTILEVEL INHERITANCE PROGRAMS**	
	5.a) Hospital	
	5.b) CompanyHierarchy	
6.	**HIERARCHICAL INHERITANCE PROGRAMS**	
	6.a) Animal	
	6.b) Vehicle	
7.	**HYBRID INHERITANCE PROGRAMS**	
	7.a) Animal	
	7.b) Vehicle	
	POLYMORPHISM	
8.	**CONSTRUCTOR PROGRAMS**	
	8.a) BookDemo	
9.	**CONSTRUCTOR OVERLOADING PROGRAMS**	
	9.a) MovieTicket	
10.	**METHOD OVERLOADING PROGRAMS**	
	10.a) HotelDemo	
	10.b) ATMDemo	
11.	**METHOD OVERRIDING PROGRAMS**	
	11.a) HospitalSystem	
	11.b) Employee	
	ABSTRACTION	
12.	**INTERFACE PROGRAMS**	
	12.a) FoodDelivery	
	12.b) FlightBooking	
	12.c) SmartDevice	
	12.d) Discount	
13.	**ABSTRACT CLASS PROGRAMS**	
	13.a) Animal	
	13.b) BankAccount	
	13.c) Payment	
	13.d) Charecters	
	ENCAPSULATION	
14.	**ENCAPSULATION PROGRAMS**	
	14.a) AirlineBaggage	
	14.b) ElectricityBilling	
	14.c) InsuranceManagement	
	14.d) TrainReservation	
15.	**PACKAGES PROGRAMS**	
	15.a)User Defined Packages	
	15.b)User Defined Packages	
	15.c)Built – in Package(3 Packages)	
	15.d)Built – in Package(3 Packages)	

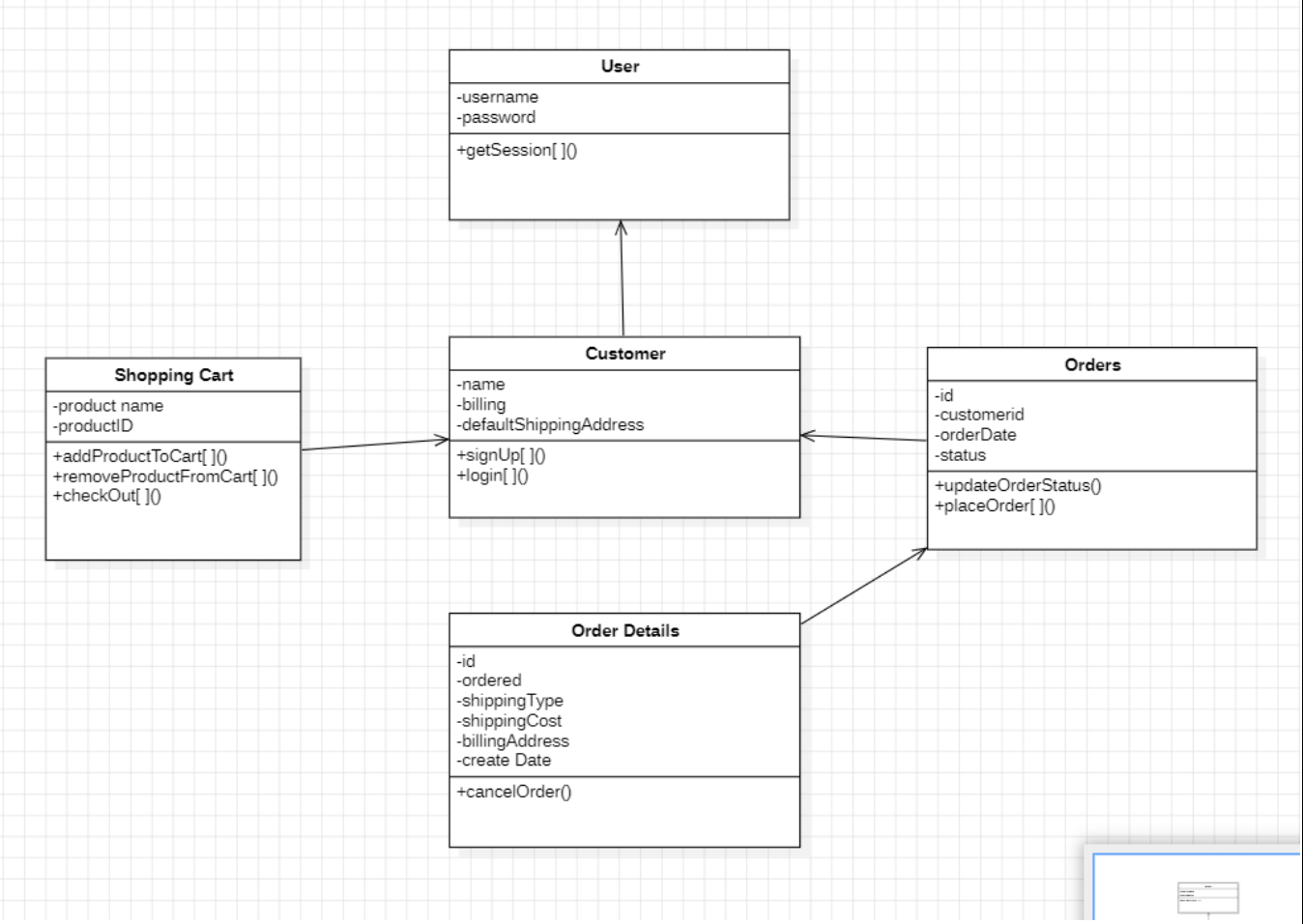
UML DIAGRAMS

1. ONLINE SHOPPING SYSTEM

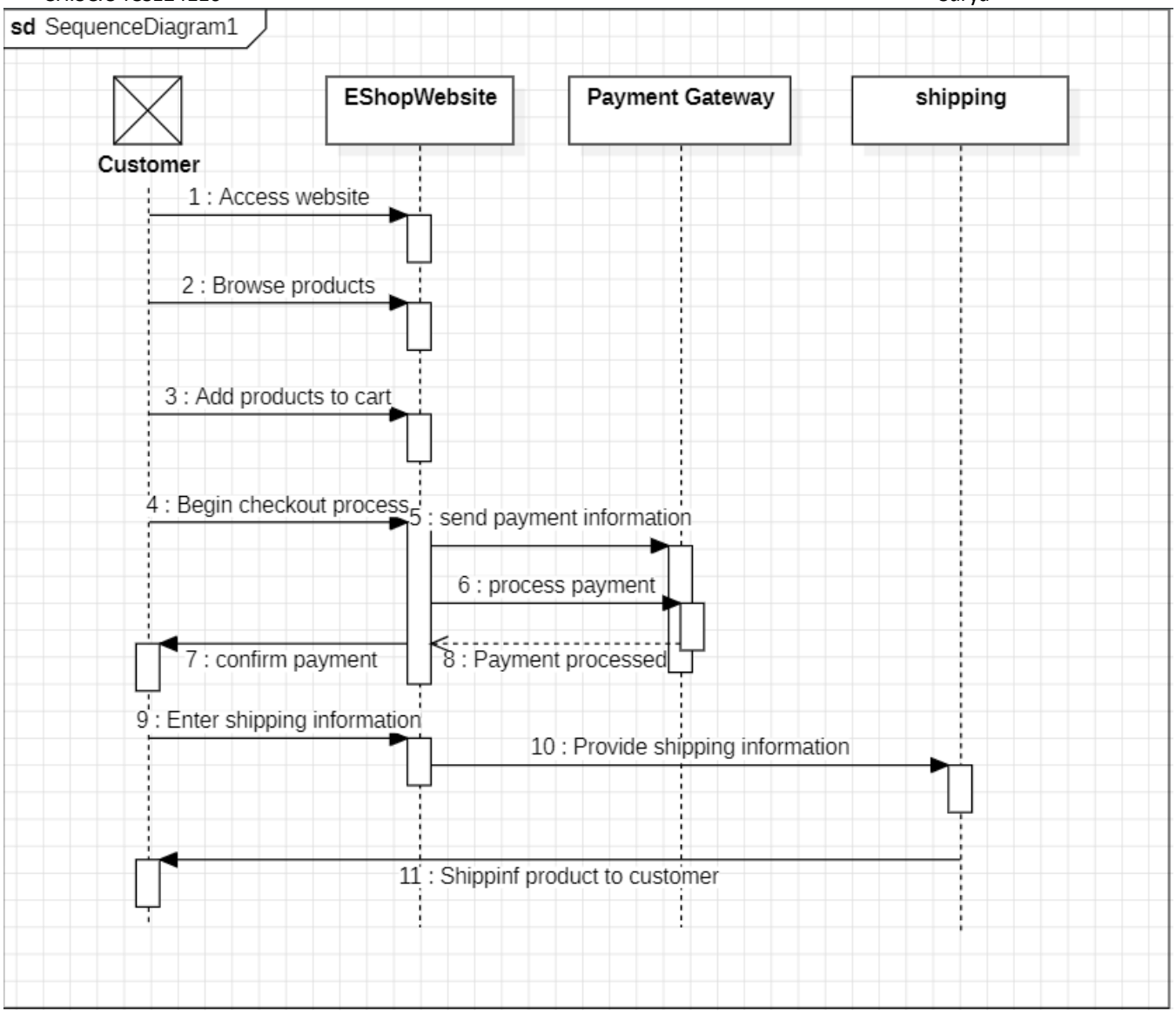
1.a) Use Case Diagram:



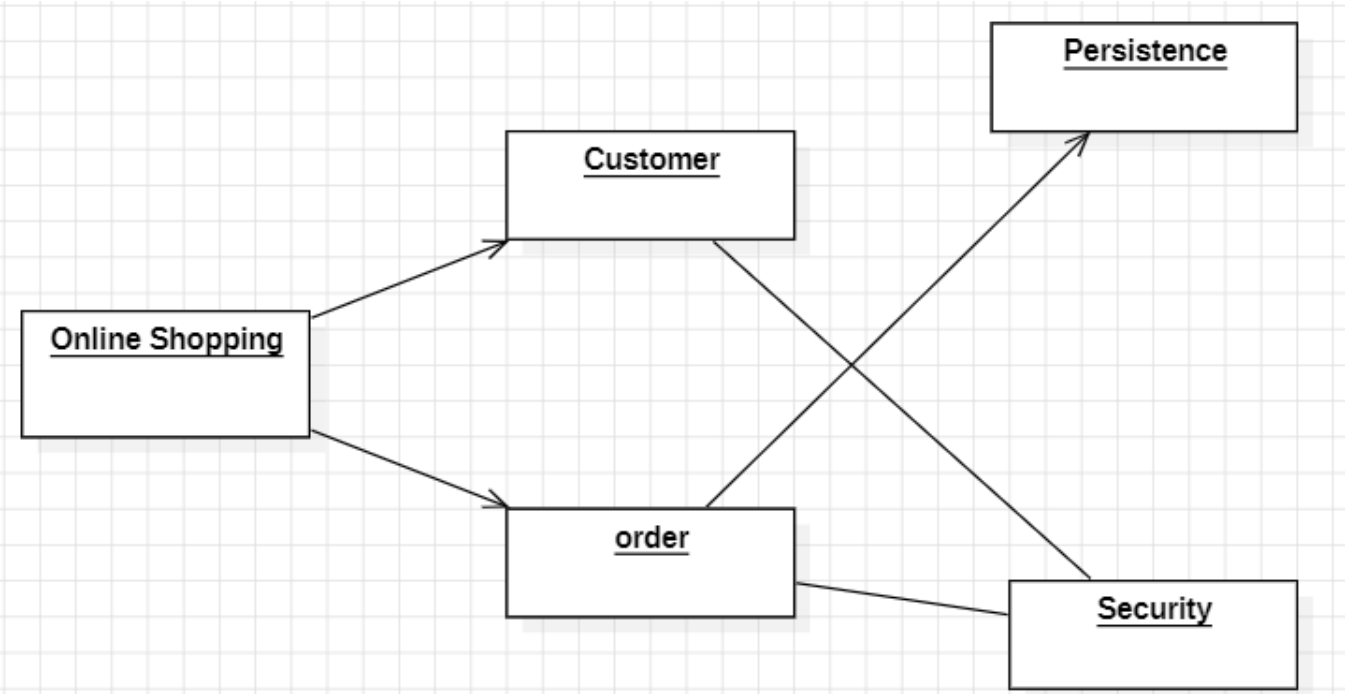
1.b) Class Diagram:



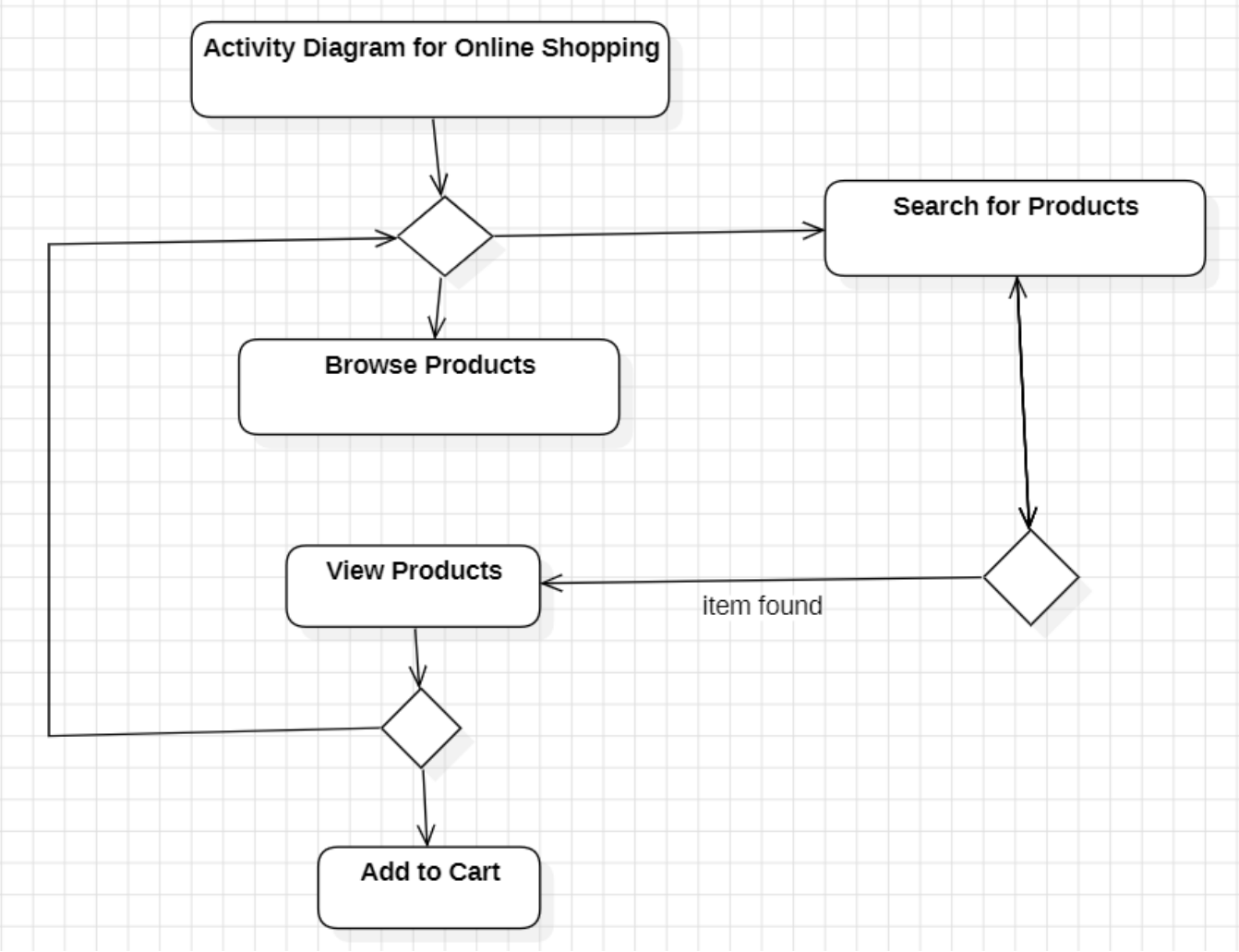
1.c) Sequence Diagram:


sd SequenceDiagram1
Customer
EShopWebsite
Payment Gateway
shipping
1 : Access website
2 : Browse products
3 : Add products to cart
4 : Begin checkout process
5 : send payment information
6 : process payment
7 : confirm payment
8 : Payment processed
9 : Enter shipping information
10 : Provide shipping information
11 : Shippinf product to customer

1.d) **Object Diagram:**

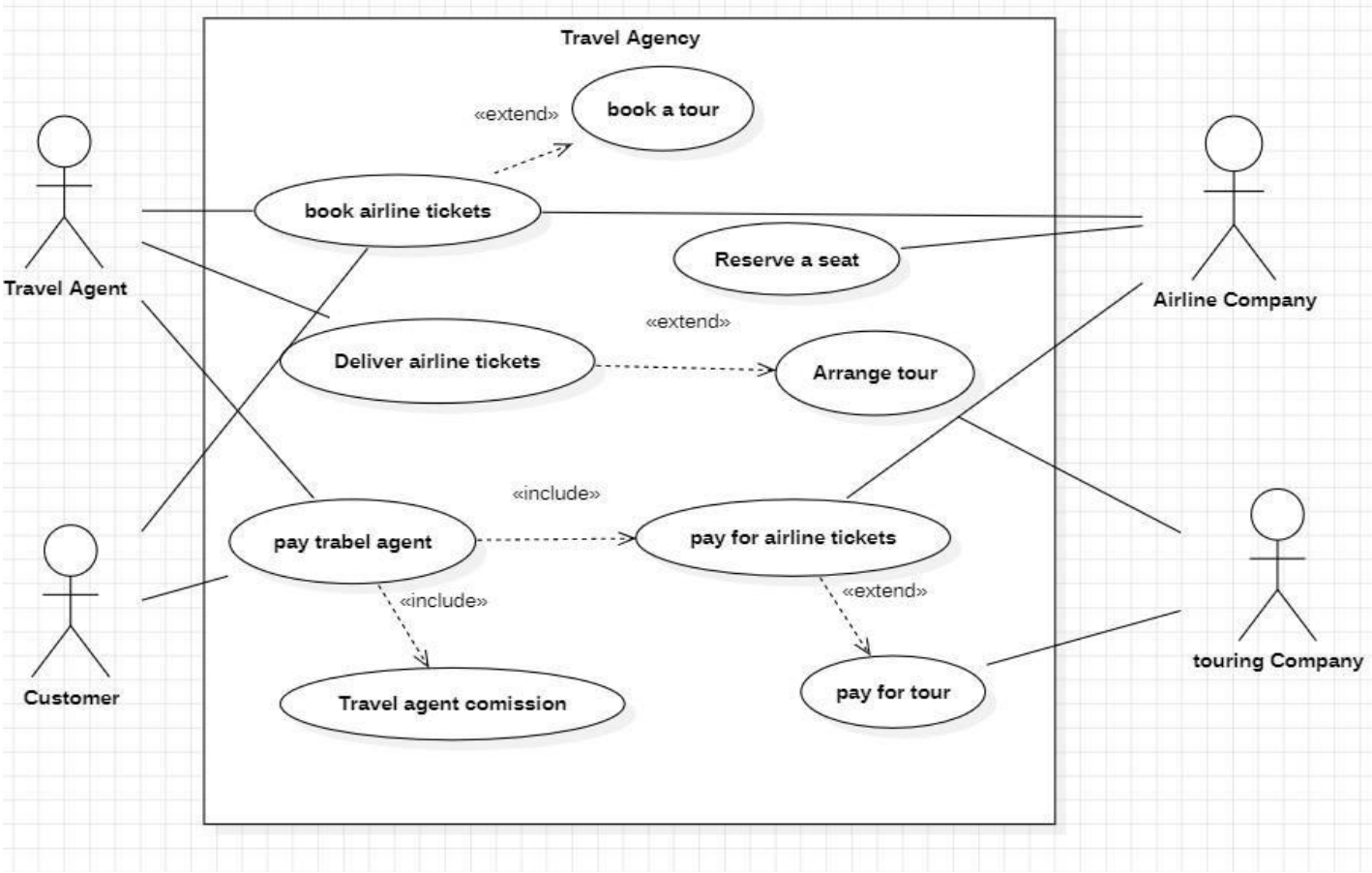


1.e) **State-Activity Diagram:**

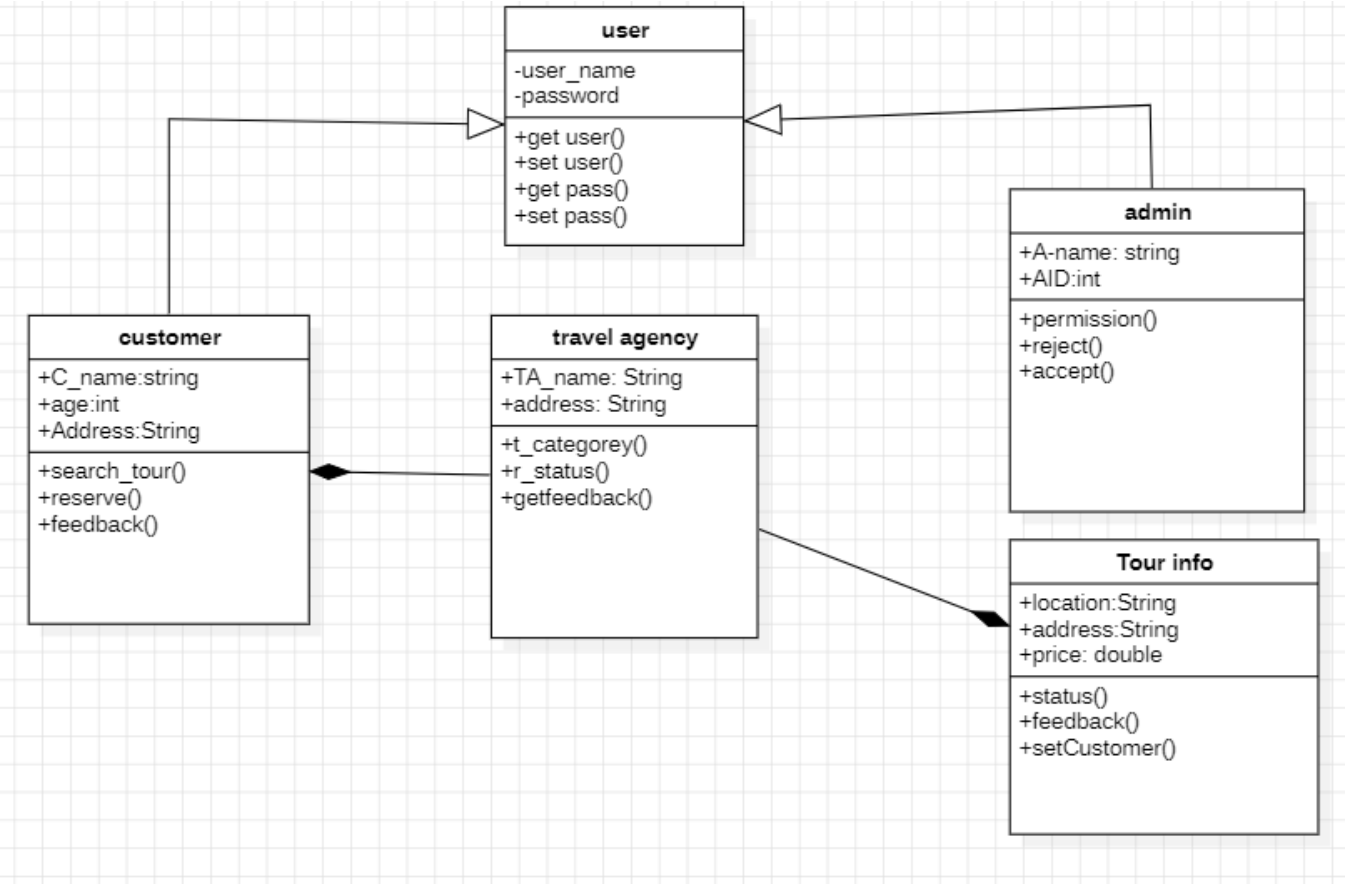

Activity Diagram for Online Shopping
Search for Products
Browse Products
View Products
item found
Add to Cart

2. TRAVEL AGENCY

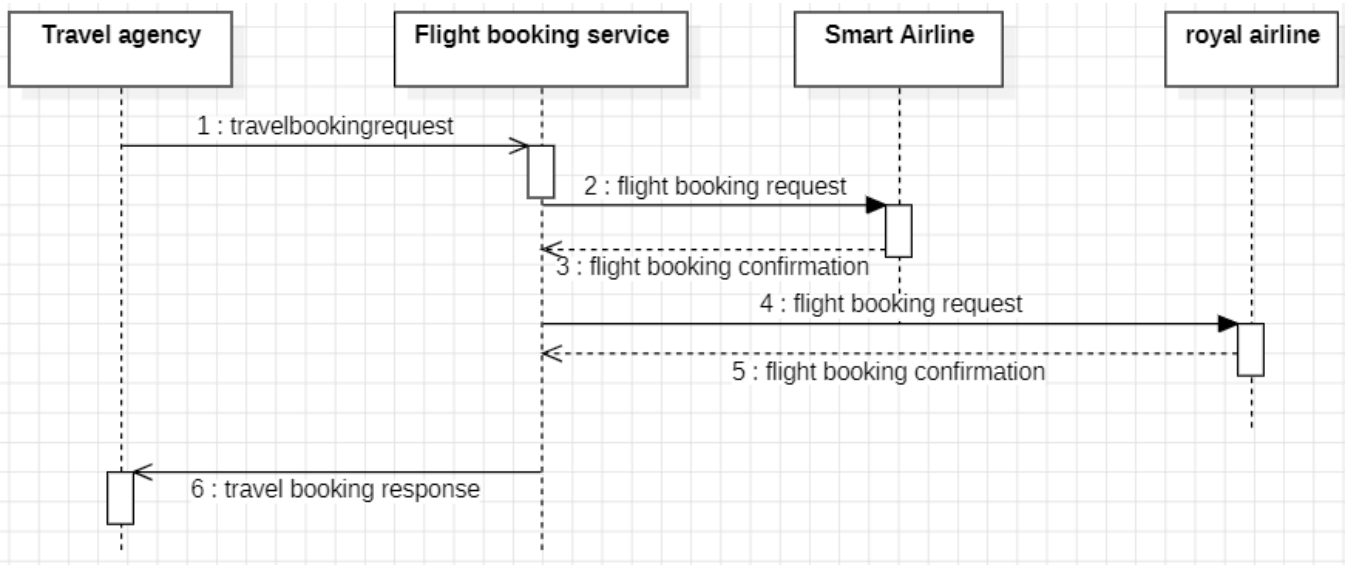
2.a) Use Case Diagram:



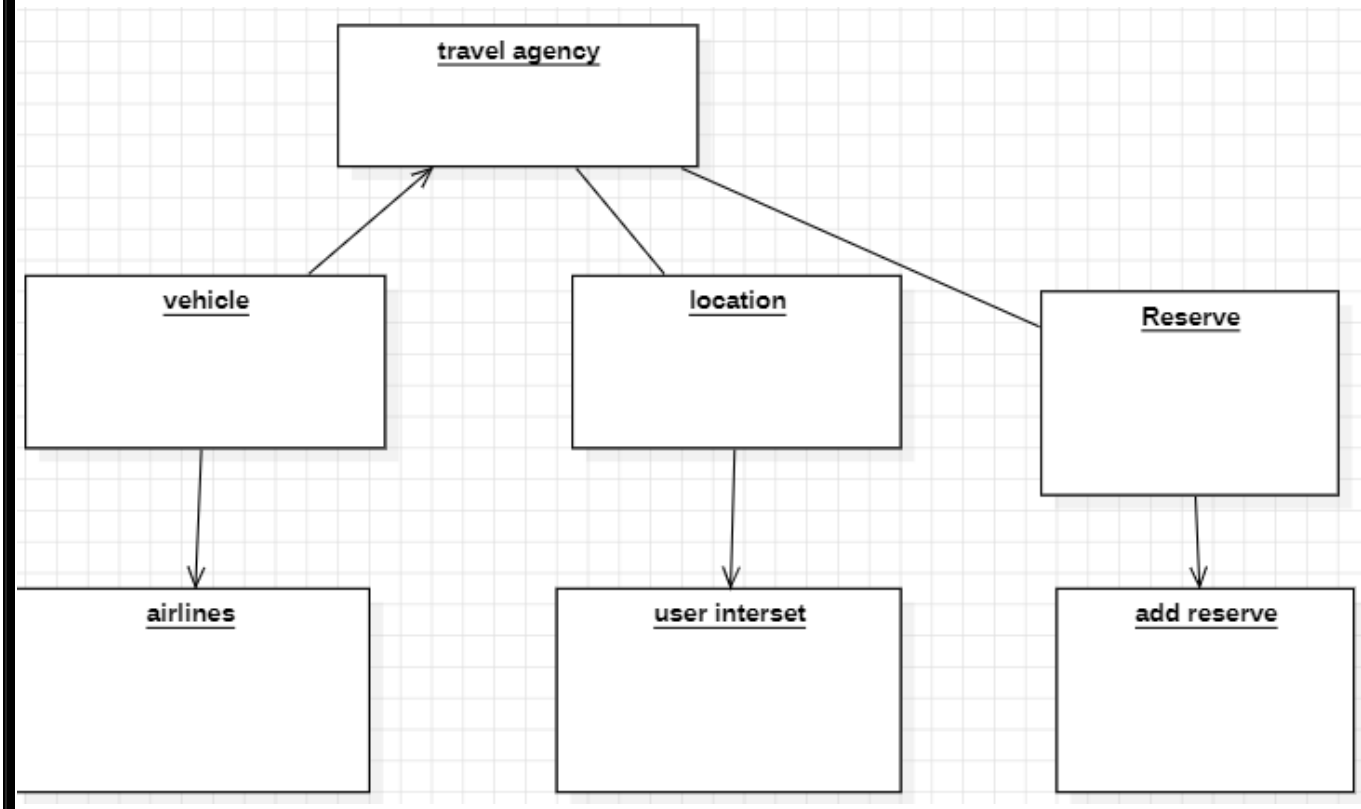
2.b) Class Diagram:



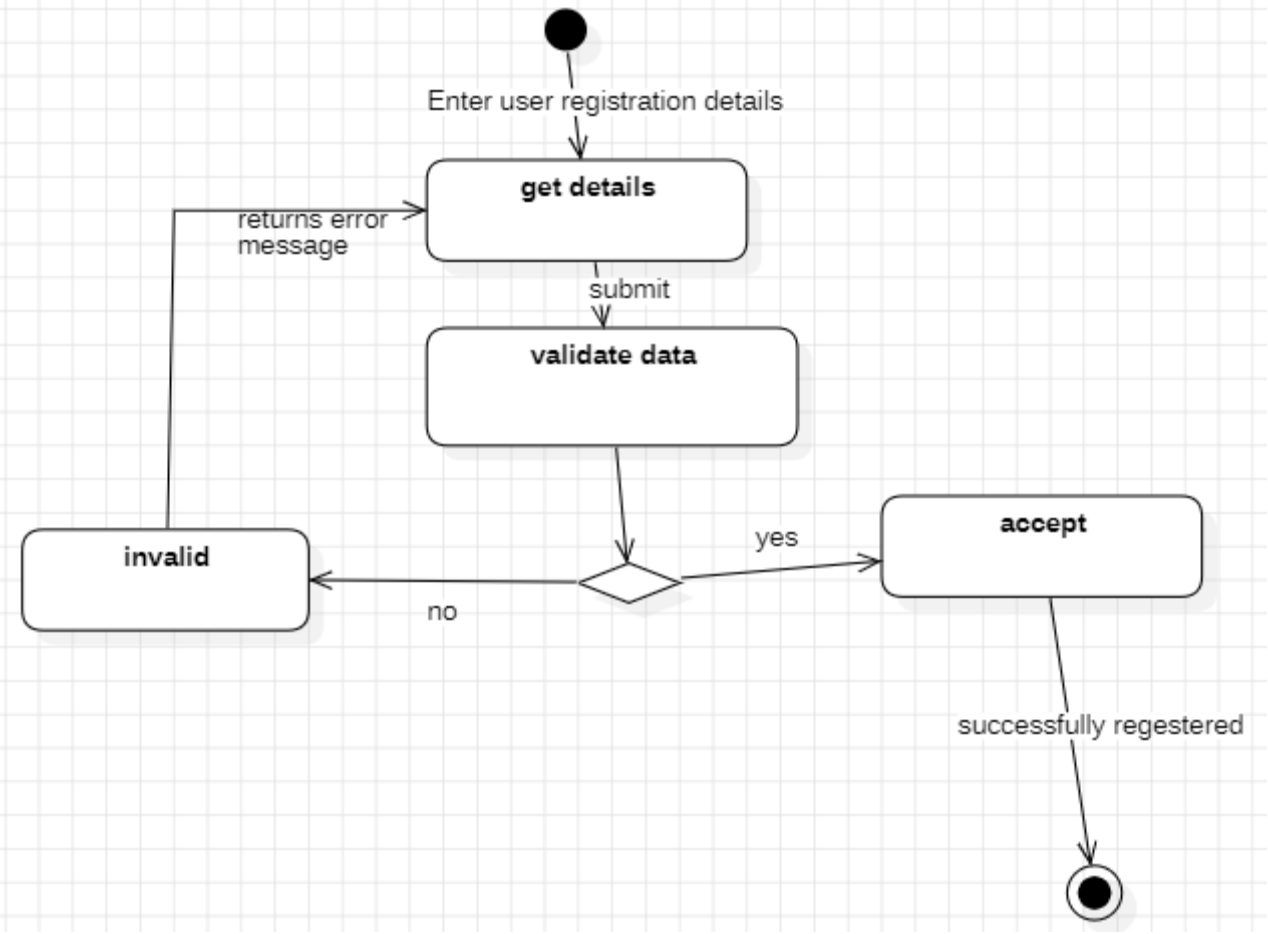
2.c) Sequence Diagram:



2.d) Object Diagram:



2.e) State-Activity Diagram:


Enter user registration details
get details
returns error message
submit
validate data
invalid
yes
accept
no
successfully regestered

3. Basic Java Programs

3.a) PascalTriangle :

Code:

```
public class PascalTriangle {
public static void main(String[] args) {
int rows = 5;
for (int i = 0; i < rows; i++) {
int number = 1;
for (int j = 0; j < rows - i; j++) Sys-
tem.out.print(" ");
for (int j = 0; j <= i; j++) { Sys-
tem.out.print(number + " "); number
= number * (i - j) / (j + 1);
}
System.out.println();
}
}
}
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac PascalTriangle.java

C:\Users\DELL\Downloads\java programs>java PascalTriangle
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1

C:\Users\DELL\Downloads\java programs>
```

3.b) Factorial :

Code:

```
public class Factorial {
    public static void main(String[] args) {
        int num = 5, fact = 1;
        for (int i = 1; i <= num; i++) {
            fact *= i; // Multiplying i with fact
        }
        System.out.println("Factorial: " + fact);
    }
}
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac Factorial.java

C:\Users\DELL\Downloads\java programs>java Factorial
Factorial: 120

C:\Users\DELL\Downloads\java programs>
```

3.c) Sum Natural Numbers :

Code:

```
    public class SumNaturalNumbers {
    public static void main(String[] args) {
    int n = 10, sum = 0, i = 1;

    while (i <= n) {
      sum += i;
      i++; // Increments i
   }

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

Output:

```
C:\Users\DELL\Downloads\java programs>javac SumNaturalNumbers.java

C:\Users\DELL\Downloads\java programs>java SumNaturalNumbers
Sum: 55

C:\Users\DELL\Downloads\java programs>
```

4.

3.d) **Reverse Numbers :**

Code:

```
public class ReverseNumbers {
    public static void main(String[] args) {
        for (int i = 10; i >= 1; i--) {
            System.out.print(i + " ");
        }
    }
}
```

Output;

```
C:\Users\DELL\Downloads\java programs>javac ReverseNumbers.java

C:\Users\DELL\Downloads\java programs>java ReverseNumbers
10 9 8 7 6 5 4 3 2 1
C:\Users\DELL\Downloads\java programs>
```

3.e) Fibonacci :

Code:

```
public class Fibonacci {
    public static void main(String[] args) {
      int n = 10, a = 0, b = 1, c;

     for (int i = 1; i <= n; i++) { Sys-
     tem.out.print(a + " ");
    c = a + b;
    a = b;
   b = c;
  }
 }
}
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac Fibonacci.java

C:\Users\DELL\Downloads\java programs>java Fibonacci
0 1 1 2 3 5 8 13 21 34
C:\Users\DELL\Downloads\java programs>
```

3.f) BMI Calculator :

Code:

```
import java.util.Scanner; pub-
lic class BMICalculator {
public static void main(String[] args) { Scan-
ner scanner = new Scanner(System.in); Sys-
tem.out.print("Enter weight in kilograms: ");
double weight = scanner.nextDouble(); Sys-
tem.out.print("Enter height in meters: "); dou-
ble height = scanner.nextDouble();
double bmi = weight / (height * height); Sys-
tem.out.printf("Your BMI is: %.2f\n", bmi); if
(bmi < 18.5) { System.out.println("Category:
Underweight");
} else if (bmi < 24.9) { Sys-
tem.out.println("Category: Normal weight");
} else if (bmi < 29.9) { Sys-
tem.out.println("Category: Overweight");
} else { System.out.println("Cate-
gory: Obese");
}
scanner.close();
}
}
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac BMICalculator.java

C:\Users\DELL\Downloads\java programs>java BMICalculator
Enter weight in kilograms: 50
Enter height in meters: 6
Your BMI is: 1.39
Category: Underweight

C:\Users\DELL\Downloads\java programs>
```

3.g) Sum Of Digits :

Code:

```
import java.util.Scanner; pub-
lic class SumOfDigits {
public static void main(String[] args) {  Scan-
ner scanner = new Scanner(System.in); Sys-
tem.out.print("Enter a four-digit number: ");
int number = scanner.nextInt();
if (number < 1000 || number > 9999) { Sys-
tem.out.println("Please enter a valid four-digit number.");
} else {
int sum = 0, temp = number;
while (temp > 0) {
sum += temp % 10;
temp /= 10;
}
System.out.println("Sum of digits: " + sum);
}
scanner.close();
}
}
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac SumOfDigits.java

C:\Users\DELL\Downloads\java programs>java SumOfDigits
Enter a four-digit number: 7831
Sum of digits: 19

C:\Users\DELL\Downloads\java programs>
```

3.h) **positive Negative :**

Code:

```
import java.util.Scanner; pub-
lic class Positive_Negative {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a number: ");
int n = scanner.nextInt();
if (n > 0) {
System.out.println(n + " is Positive");
} else if (n < 0) { Sys-
tem.out.println(n + " is Negative");
} else {
System.out.println(n + " is Zero");
}
}
}
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac Positive_Negative.java

C:\Users\DELL\Downloads\java programs>java Positive_Negative
Enter a number: 8
8 is Positive

C:\Users\DELL\Downloads\java programs>
```

3.i) Largest Number :

Code:

```
import java.util.Scanner;
public class Largest_Number {
public static void main(String[] args) { Scan-
ner scanner = new Scanner(System.in); Sys-
tem.out.print("Enter number of elements: ");
int n = scanner.nextInt();
int a[] = new int[n]; Sys-
tem.out.println("Enter elements: ");
for (int i = 0; i < n; i++) {
a[i] = scanner.nextInt();
}
int max = a[0];
for (int i = 0; i < n; i++) {
if (max < a[i]) {
max = a[i];
}
}
System.out.println("Maximum value: " + max);
}
}
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac Largest_Number.java

C:\Users\DELL\Downloads\java programs>java Largest_Number
Enter number of elements: 10
Enter elements:
5
5
5
5
```

3.j) Biggest Number :

Code:

```
import java.util.Scanner; pub-

lic class BiggestNumber {
     public static void main(String[] args) { Scan-
           ner scanner = new Scanner(System.in);

              System.out.print("Enter three numbers: ");
              int x = scanner.nextInt();
              int y = scanner.nextInt();
              int z = scanner.nextInt();


          int largest;
          if (x > y && x > z) {
             largest = x;
          } else if (y > z) {
             largest = y;
          } else {
             largest = z;
          }

              System.out.println("Largest number is: " + largest);

              scanner.close();
        }
    }
```

Output:

```
C:\Users\DELL\Downloads\java programs>javac BiggestNumber.java

C:\Users\DELL\Downloads\java programs>java BiggestNumber
Enter three numbers: 7 8 9
Largest number is: 9

C:\Users\DELL\Downloads\java programs>
```

INHERITANCE

4)SINGLE INHERITANCE PROGRAMS

4a) Animal Sounds

Code:

```java
class Room {
    double length, width;

    public Room(double length, double width) {
        this.length = length;
        this.width = width;
    }

    public double calculateArea() {
        return length * width;
    }
}

class VolumeRoom extends Room {
    double height;

    public VolumeRoom(double length, double width, double height) {
        super(length, width);
        this.height = height;
    }

    public double calculateVolume() {
        return length * width * height;
    }
}

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

        double length = 10.5;
        double width = 8.2;
        double height = 12.3;

        VolumeRoom room = new VolumeRoom(length, width, height);

        System.out.println("Room Area: " + room.calculateArea() + " square units");
        System.out.println("Room Volume: " + room.calculateVolume() + " cubic units");
    }
}
```

Output:

```
D:\oops>javac RoomDemo.java

D:\oops>java RoomDemo
Room Area: 86.1 square units
Room Volume: 1059.03 cubic units
```

4b) Animalsounds

Code:

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

class Dog extends Animal{
public void bark(){
System.out.println("Dog barks");
}
}
public class SingleInheritance{
public static void main(String[] args){
Dog myobj=new Dog();
myobj.bark();

Animal obj=new Animal();
obj.sound();
}
}
```

Output:

```
D:\oops>javac SingleInheritance.java

D:\oops>java SingleInheritance
Dog barks
Animal makes sound
```

5) MULTILEVEL INHERITANCE PROGRAMS

5a)Hospital

Code:

```java
class Person {
    void walk() {
        System.out.println("A person can walk");
    }
}
class Doctor extends Person {
    void treatPatients() {
        System.out.println("Doctor is treating patients");
    }
}
class Surgeon extends Doctor {
    void performSurgery() {
        System.out.println("Surgeon is performing surgery");
    }
}
public class Hospital {
    public static void main(String[] args) {
        Surgeon surgeon = new Surgeon();
        surgeon.walk();
        surgeon.treatPatients();
        surgeon.performSurgery();
    }
}
```

Output:

```
D:\oops>javac Hospital.java

D:\oops>java Hospital
A person can walk
Doctor is treating patients
Surgeon is performing surgery
```

5b) CompanyHierarchy

Code:

```java
class Employee {
    void work() {
        System.out.println("Employee is working");
    }
}
class Manager extends Employee {
    void manageTeam() {
        System.out.println("Manager is managing the team");
    }
}
class Director extends Manager {
    void makeDecisions() {
        System.out.println("Director is making high-level decisions");
    }
}
public class CompanyHierarchy {
    public static void main(String[] args) {
        Director director = new Director();
        director.work();
        director.manageTeam();
        director.makeDecisions();
    }
}
```

Output:

```
D:\oops>javac CompanyHierarchy.java

D:\oops>java CompanyHierarchy
Employee is working
Manager is managing the team
Director is making high-level decisions
```

6) HIERARCHICAL INHERITANCE PROGRAMS

6a)Animal

Code:

```java
class Animal {
    void eat() {
        System.out.println("Animals eat food.");
    }
}
class Dog extends Animal {
    void bark() {
        System.out.println("Dog barks.");
    }
}
class Cat extends Animal {
    void meow() {
        System.out.println("Cat meows.");
    }
}
public class HierarchicalInheritanceExample1 {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();
        dog.bark();

        System.out.println();

        Cat cat = new Cat();
        cat.eat();
        cat.meow();
    }
}
```

Output:

```
D:\oops>javac HierarchicalInheritanceExample1.java

D:\oops>java HierarchicalInheritanceExample1
Animals eat food.
Dog barks.

Animals eat food.
Cat meows.
```

6b) Vehicle

Code:

```java
class Vehicle {
    void start() {
        System.out.println("Vehicle is starting...");
    }
}
class Car extends Vehicle {
    void drive() {
        System.out.println("Car is driving.");
    }
}
class Bike extends Vehicle {
    void ride() {
        System.out.println("Bike is being ridden.");
    }
}
public class HierarchicalInheritanceExample2 {
    public static void main(String[] args) {
        Car car = new Car();
        car.start();
        car.drive();

        System.out.println();

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

Output:

```
D:\oops>javac HierarchicalInheritanceExample2.java

D:\oops>java HierarchicalInheritanceExample2
Vehicle is starting...
Car is driving.

Vehicle is starting...
Bike is being ridden.
```

7) HYBRID INHERITANCE PROGRAMS

7a) Animal

Code:

```
interface Animal {
    void eat();
}
class Mammal implements Animal {
    public void eat() {
        System.out.println("Mammals eat food.");
    }

    void breathe() {
        System.out.println("Mammals breathe oxygen.");
    }
}
interface Canine {
    void bark();
}
class Dog extends Mammal implements Canine {
    public void bark() {
        System.out.println("Dog barks.");
    }
}
public class HybridInheritanceExample1 {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();
dog.breathe();
dog.bark();
}
```

Output:

```
D:\oops>javac HybridInheritanceExample1.java

D:\oops>java HybridInheritanceExample1
Mammals eat food.
Mammals breathe oxygen.
Dog barks.
```

7b) Vehicle

Code:

```java
interface Vehicle {
    void start();
}
class FourWheeler implements Vehicle {
    public void start() {
        System.out.println("Four-wheeler vehicle is starting.");
    }

    void wheels() {
        System.out.println("This vehicle has four wheels.");
    }
}
interface Electric {
    void batteryType();
}
class ElectricCar extends FourWheeler implements Electric {
    public void batteryType() {
        System.out.println("Electric car uses a lithium-ion battery.");
    }
}
public class HybridInheritanceExample2 {
    public static void main(String[] args) {
        ElectricCar tesla = new ElectricCar();
        tesla.start();
        tesla.wheels();
        tesla.batteryType();
    }
}
```

Output:

```
D:\oops>javac HybridInheritanceExample2.java

D:\oops>java HybridInheritanceExample2
Four-wheeler vehicle is starting.
This vehicle has four wheels.
Electric car uses a lithium-ion battery.
```

POLYMORPHISM

8) CONSTRUCTOR PROGRAMS

a) BookDemo

Code:

```java
class Book {
    String title;
    String author;
    int pages;

    Book() {
        title = "The Alchemist";
        author = "Paulo Coelho";
        pages = 200;
    }

    void displayInfo() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Pages: " + pages);
    }
}

public class BookDemo {
    public static void main(String[] args) {
        Book b1 = new Book();
        b1.displayInfo();
    }
}
```

Output:

```
D:\oops>javac BookDemo.java

D:\oops>java BookDemo
Title: The Alchemist
Author: Paulo Coelho
Pages: 200
```

9)CONSTRUCTOR OVERLOADING PROGRAMS

9.a)MovieTicket

Code:

```
class MovieTicket {
    String movieName;
    String seatType;
    double price;

    MovieTicket() {
        this.movieName = "Not Selected";
        this.seatType = "Regular";
        this.price = 10.0;
    }

    MovieTicket(String movieName) {
        this.movieName = movieName;
        this.seatType = "Regular";
        this.price = 10.0;
    }

    MovieTicket(String movieName, String seatType, double price) {
        this.movieName = movieName;
        this.seatType = seatType;
        this.price = price;
    }

    void display() {
        System.out.println("Movie: " + movieName);
        System.out.println("Seat Type: " + seatType);
        System.out.println("Price: $" + price);
    }

    public static void main(String[] args) {
        MovieTicket t1 = new MovieTicket();
        MovieTicket t2 = new MovieTicket("Inception");
        MovieTicket t3 = new MovieTicket("Avatar", "VIP", 25.0);

        t1.display();
        t2.display();
        t3.display();
    }
}
```

Output:

```
D:\oops>java MovieTicket
Movie: Not Selected
Seat Type: Regular
Price: $10.0
Movie: Inception
Seat Type: Regular
Price: $10.0
Movie: Avatar
Seat Type: VIP
Price: $25.0
```

10)METHOD OVERLOADING PROGRAMS

10.a) HotelDemo

Code:

```java
class Hotel {
    void bookRoom(String name) {
        System.out.println(name + " booked a Standard Room.");
    }
    void bookRoom(String name, int nights) {
        System.out.println(name + " booked a room for " + nights + " nights.");
    }
    void bookRoom(String name, int nights, String roomType) {
        System.out.println(name + " booked a " + roomType + " for " + nights + " nights.");
    }
}

public class HotelDemo {
    public static void main(String[] args) {
        Hotel hotel = new Hotel();

        hotel.bookRoom("Alice");
        hotel.bookRoom("Bob", 3);
        hotel.bookRoom("Charlie", 5, "Deluxe Room");
    }
}
```

Output:

```
D:\oops>javac HotelDemo.java

D:\oops>java HotelDemo
Alice booked a Standard Room.
Bob booked a room for 3 nights.
Charlie booked a Deluxe Room for 5 nights.
```

10.b)ATMDemo

Code:

```java
class ATM {
    void withdraw(double amount) {
        System.out.println("Withdrawing $" + amount);
    }
    void withdraw(double amount, int pin) {
        System.out.println("Withdrawing $" + amount + " using PIN: " + pin);
    }
    void withdraw(double amount, int pin, String accountType) {
        System.out.println("Withdrawing $" + amount + " from " + accountType + " account using PIN: " + pin);
    }
}
public class ATMDemo {
    public static void main(String[] args) {
        ATM atm = new ATM();

        atm.withdraw(500);
        atm.withdraw(1000, 1234);
        atm.withdraw(2000, 5678, "Savings");
    }
}
```

Output:

```
D:\oops>javac ATMDemo.java

D:\oops>java ATMDemo
Withdrawing $500.0
Withdrawing $1000.0 using PIN: 1234
Withdrawing $2000.0 from Savings account using PIN: 5678
```

11) METHOD OVERRIDING PROGRAMS

11.a) HospitalSystem

Code:

Surya

```java
class Doctor {
    void treatPatient() {
        System.out.println("General doctor treating patient.");
    }
}

class Cardiologist extends Doctor {
 void treatPatient() {
        System.out.println("Cardiologist treating heart-related issues.");
    }
}

class Dermatologist extends Doctor {
    void treatPatient() {
        System.out.println("Dermatologist treating skin-related issues.");
    }
}

public class HospitalSystem {
    public static void main(String[] args) {
        Doctor d1 = new Cardiologist();
        Doctor d2 = new Dermatologist();

        d1.treatPatient();
        d2.treatPatient();
    }
}
```

Output:

```
D:\oops>javac HospitalSystem.java

D:\oops>java HospitalSystem
Cardiologist treating heart-related issues.
Dermatologist treating skin-related issues.
```

11.b) EmployeeMain

Code:

Surya

```java
class Employee {
    void calculateSalary() {
        System.out.println("Calculating general employee salary.");
    }
}

class FullTimeEmployee extends Employee {
    void calculateSalary() {
        System.out.println("Calculating salary for Full-Time Employee with benefits.");
    }
}

class PartTimeEmployee extends Employee {
    void calculateSalary() {
        System.out.println("Calculating salary for Part-Time Employee based on hours worked.");
    }
}

public class EmployeeMain {
    public static void main(String[] args) {
        Employee e1 = new FullTimeEmployee();
        Employee e2 = new PartTimeEmployee();

        e1.calculateSalary();
        e2.calculateSalary();
    }
}
```

Output:

```
D:\oops>javac EmployeeMain.java

D:\oops>java EmployeeMain
Calculating salary for Full-Time Employee with benefits.
Calculating salary for Part-Time Employee based on hours worked.
```

12) INTERFACE PROGRAMS

12a) FoodDelivery

Code:

```java
interface FoodDelivery {
    void deliverOrder(String foodItem);
}
class Zomato implements FoodDelivery {
    public void deliverOrder(String foodItem) {
        System.out.println("Zomato is delivering " + foodItem);
    }
}
class Swiggy implements FoodDelivery {
    public void deliverOrder(String foodItem) {
        System.out.println("Swiggy is delivering " + foodItem);
    }
}

public class InterfaceEx1 {
    public static void main(String[] args) {
        FoodDelivery f1 = new Zomato();
        FoodDelivery f2 = new Swiggy();

        f1.deliverOrder("Pizza");
        f2.deliverOrder("Burger");
    }
}
```

Output:

```
D:\oops>javac InterfaceEx1.java

D:\oops>java InterfaceEx1
Zomato is delivering Pizza
Swiggy is delivering Burger
```

12b)FlightBooking

Code:

```java
interface FlightBooking {
    void bookTicket(String destination);
}
class Indigo implements FlightBooking {
    public void bookTicket(String destination) {
        System.out.println("Indigo flight booked to " + destination);
    }
}
class AirIndia implements FlightBooking {
    public void bookTicket(String destination) {
        System.out.println("Air India flight booked to " + destination);
    }
}

public class InterfaceEx2 {
    public static void main(String[] args) {
        FlightBooking f1 = new Indigo();
        FlightBooking f2 = new AirIndia();

        f1.bookTicket("New York");
        f2.bookTicket("London");
    }
}
```

Output:

```
D:\oops>javac InterfaceEx2.java

D:\oops>java InterfaceEx2
Indigo flight booked to New York
Air India flight booked to London
```

12c) SmartDevice

Code:

```java
interface SmartDevice {
    void turnOn();
    void turnOff();
}
class Light implements SmartDevice {
    public void turnOn() {
        System.out.println("Light turned ON.");
    }

    public void turnOff() {
        System.out.println("Light turned OFF.");
    }
}

class Fan implements SmartDevice {

    public void turnOn() {
        System.out.println("Fan turned ON.");
    }

    public void turnOff() {
        System.out.println("Fan turned OFF.");
    }
}

public class InterfaceEx3 {
    public static void main(String[] args) {
        SmartDevice device1 = new Light();
        SmartDevice device2 = new Fan();

        device1.turnOn();
        device2.turnOff();
    }
}
```

Output:

```
D:\oops>javac InterfaceEx3.java

D:\oops>java InterfaceEx3
Light turned ON.
Fan turned OFF.
```

12d) Discount

Code:

```java
interface Discountable {
    void applyDiscount(double price);
}
class Electronics implements Discountable {

    public void applyDiscount(double price) {
        System.out.println("Final price after 10% discount: $" + (price - (price * 0.10)));
    }
}

class Clothing implements Discountable {
    public void applyDiscount(double price) {
        System.out.println("Final price after 20% discount: $" + (price - (price * 0.20)));
    }
}

public class InterfaceEx4 {
    public static void main(String[] args) {
        Discountable d1 = new Electronics();
        Discountable d2 = new Clothing();

        d1.applyDiscount(100);
        d2.applyDiscount(200);
    }
}
```

Output:

```
D:\oops>javac InterfaceEx4.java

D:\oops>java InterfaceEx4
Final price after 10% discount: $90.0
Final price after 20% discount: $160.0
```

13) ABSTRACT CLASS PROGRAMS

13 a) Animal

Code:

```
abstract class Animal {
    String name;

    Animal(String name) {
        this.name = name;
    }
    abstract void makeSound();

    void showName() {
        System.out.println("Animal: " + name);
    }
}
class Dog extends Animal {
    Dog(String name) {
        super(name);
    }
    void makeSound() {
        System.out.println(name + " says: Woof Woof!");
    }
}
class Cat extends Animal {
    Cat(String name) {
        super(name);
    }
    void makeSound() {
        System.out.println(name + " says: Meow Meow!");
    }
}

public class AbstractEx1 {
    public static void main(String[] args) {
        Animal a1 = new Dog("Buddy");
        Animal a2 = new Cat("Whiskers");

        a1.makeSound();
        a2.makeSound();
    }
}
```

Output:

```
D:\oops>javac AbstractEx1.java

D:\oops>java AbstractEx1
Buddy says: Woof Woof!
Whiskers says: Meow Meow!
```

13 b) BankAccount
Code:

```java
abstract class BankAccount {
    double balance;

    BankAccount(double balance) {
        this.balance = balance;
    }

    abstract void calculateInterest();

    void showBalance() {
        System.out.println("Current Balance: $" + balance);
    }
}

class SavingsAccount extends BankAccount {
    SavingsAccount(double balance) {
        super(balance);
    }

    void calculateInterest() {
        System.out.println("Savings Account Interest: " + (balance * 0.04));
    }
}
class CurrentAccount extends BankAccount {
    CurrentAccount(double balance) {
        super(balance);
    }
    void calculateInterest() {
        System.out.println("Current Account has no interest.");
    }
}

public class AbstractEx2{
    public static void main(String[] args) {
        BankAccount acc1 = new SavingsAccount(1000);
        BankAccount acc2 = new CurrentAccount(5000);

        acc1.calculateInterest();
        acc2.calculateInterest();
        acc1.showBalance();
    }
}
```

Output:

```
D:\oops>javac AbstractEx2.java

D:\oops>java AbstractEx2
Savings Account Interest: 40.0
Current Account has no interest.
Current Balance: $1000.0
```

13 c) Payment

Code:

```java
abstract class Payment {
    double amount;

    Payment(double amount) {
        this.amount = amount;
    }
    abstract void processPayment();

    void paymentConfirmation() {
        System.out.println("Payment of $" + amount + " is confirmed.");
    }
}
class CreditCardPayment extends Payment {
    CreditCardPayment(double amount) {
        super(amount);
    }
    void processPayment() {
        System.out.println("Processing Credit Card payment of $" + amount);
    }
}
class PayPalPayment extends Payment {
    PayPalPayment(double amount) {
        super(amount);
    }
    void processPayment() {
        System.out.println("Processing PayPal payment of $" + amount);
    }
}

public class AbstractEx3{
    public static void main(String[] args) {
        Payment p1 = new CreditCardPayment(500);
        Payment p2 = new PayPalPayment(200);

        p1.processPayment();
        p1.paymentConfirmation();

        p2.processPayment();
        p2.paymentConfirmation();
    }
}
```

Output:

```
D:\oops>javac AbstractEx3.java

D:\oops>java AbstractEx3
Processing Credit Card payment of $500.0
Payment of $500.0 is confirmed.
Processing PayPal payment of $200.0
Payment of $200.0 is confirmed.
```

13 d) Characters

Code:

```java
abstract class Character {
    String name;

    Character(String name) {
        this.name = name;
    }
    abstract void attack();

    void showCharacter() {
        System.out.println("Character: " + name);
    }
}
class Warrior extends Character {
    Warrior(String name) {
        super(name);
    }
    void attack() {
        System.out.println(name + " attacks with a sword!");
    }
}
class Mage extends Character {
    Mage(String name) {
        super(name);
    }
    void attack() {
        System.out.println(name + " casts a fireball spell!");
    }
}

public class AbstractEx4{
    public static void main(String[] args) {
        Character c1 = new Warrior("Thor");
        Character c2 = new Mage("Merlin");

        c1.showCharacter();
        c1.attack();

        c2.showCharacter();
        c2.attack();
    }
}
```

Output:

```
D:\oops>javac AbstractEx4.java

D:\oops>java AbstractEx4
Character: Thor
Thor attacks with a sword!
Character: Merlin
Merlin casts a fireball spell!
```

ENCAPSULATION PROGRAMS

14a) AirlineBaggageSystem

Code:

```java
class Baggage {
    private String baggageID;
    private double weight;

    public Baggage(String baggageID, double weight) {
        this.baggageID = baggageID;
        setWeight(weight);
    }

    public String getBaggageID() { return baggageID; }

    public double getWeight() { return weight; }

    public void setWeight(double weight) {
        if (weight >= 0) {
            this.weight = weight;
        } else {
            System.out.println("Invalid weight! Cannot be negative.");
        }
    }

    public void displayBaggageDetails() {
        System.out.println("Baggage ID: " + baggageID);
        System.out.println("Weight: " + weight + " kg");
    }
}

public class AirlineBaggageSystem {
    public static void main(String[] args) {
        Baggage bag1 = new Baggage("BAG123", 25.5);
        bag1.displayBaggageDetails();

        bag1.setWeight(-5);
        bag1.setWeight(30);
        bag1.displayBaggageDetails();
    }
}
```

Output:

```
D:\oops>javac AirlineBaggageSystem.java

D:\oops>java AirlineBaggageSystem
Baggage ID: BAG123
Weight: 25.5 kg
Invalid weight! Cannot be negative.
Baggage ID: BAG123
Weight: 30.0 kg
```

14b) ElectricityBilling System

Code:

```java
class ElectricityBill {
    private int unitsConsumed;
    private static final double RATE_PER_UNIT = 5.0;

    public ElectricityBill(int unitsConsumed) {
        setUnitsConsumed(unitsConsumed);
    }

    public int getUnitsConsumed() { return unitsConsumed; }

    public void setUnitsConsumed(int unitsConsumed) {
        if (unitsConsumed >= 0) {
            this.unitsConsumed = unitsConsumed;
        } else {
            System.out.println("Invalid units! Cannot be negative.");
        }
    }

    public double calculateBill() {
        return unitsConsumed * RATE_PER_UNIT;
    }

    public void displayBill() {
        System.out.println("Electricity Consumption: " + unitsConsumed + " units");
        System.out.println("Total Bill: $" + calculateBill());
    }
}

public class ElectricityBillingSystem {
    public static void main(String[] args) {
        ElectricityBill bill1 = new ElectricityBill(200);
        bill1.displayBill();

        bill1.setUnitsConsumed(-10);
        bill1.setUnitsConsumed(300);
        bill1.displayBill();
    }
}
```

Output:

```
D:\oops>javac ElectricityBillingSystem.java

D:\oops>java ElectricityBillingSystem
Electricity Consumption: 200 units
Total Bill: $1000.0
Invalid units! Cannot be negative.
Electricity Consumption: 300 units
Total Bill: $1500.0
```

14c) Insurance Management System

Code:

```java
class InsurancePolicy {
    private String policyNumber;
    private String policyHolderName;
    private double premiumAmount;

    public InsurancePolicy(String policyNumber, String policyHolderName, double premiumAmount) {
        this.policyNumber = policyNumber;
        this.policyHolderName = policyHolderName;
        setPremiumAmount(premiumAmount);
    }

    public String getPolicyNumber() { return policyNumber; }
    public String getPolicyHolderName() { return policyHolderName; }
    public double getPremiumAmount() { return premiumAmount; }

    public void setPremiumAmount(double premiumAmount) {
        if (premiumAmount > 0) {
            this.premiumAmount = premiumAmount;
        } else {
            System.out.println("Invalid premium amount! Must be positive.");
        }
    }

    public void displayPolicyDetails() {
        System.out.println("Policy Number: " + policyNumber);
        System.out.println("Policy Holder: " + policyHolderName);
        System.out.println("Premium Amount: $" + premiumAmount);
    }
}

public class InsuranceManagementSystem {
    public static void main(String[] args) {
        InsurancePolicy policy1 = new InsurancePolicy("INS78945", "John Doe", 500.0);
        policy1.displayPolicyDetails();

        policy1.setPremiumAmount(-100);
        policy1.setPremiumAmount(750);
        policy1.displayPolicyDetails();
    }
}
```

Output:

```
D:\oops>javac InsuranceManagementSystem.java

D:\oops>java InsuranceManagementSystem
Policy Number: INS78945
Policy Holder: John Doe
Premium Amount: $500.0
Invalid premium amount! Must be positive.
Policy Number: INS78945
Policy Holder: John Doe
Premium Amount: $750.0
```

14d) Train ReservationSystem
Code:

```
class TrainTicket {
    private String passengerName;
    private int trainNumber;
    private int seatNumber;

    public TrainTicket(String passengerName, int trainNumber, int seatNumber) {
        this.passengerName = passengerName;
        this.trainNumber = trainNumber;
        this.seatNumber = seatNumber;
    }

    public String getPassengerName() { return passengerName; }
    public int getTrainNumber() { return trainNumber; }
    public int getSeatNumber() { return seatNumber; }

    public void displayTicketDetails() {
        System.out.println("Passenger: " + passengerName);
        System.out.println("Train Number: " + trainNumber);
        System.out.println("Seat Number: " + seatNumber);
    }
}
public class TrainReservationSystem {
    public static void main(String[] args) {
        TrainTicket ticket1 = new TrainTicket("Hari", 10123, 45);
        ticket1.displayTicketDetails();
    }
}
```

Output:

```
D:\oops>javac TrainReservationSystem.java

D:\oops>java TrainReservationSystem
Passenger: Hari
Train Number: 10123
Seat Number: 45
```

15)PACKAGES PROGRAMS

15.a) User Defined Packages

Code:

```java
package mypackage;

public class Patient {
    private String name;
    private int age;
    private String disease;

    public Patient(String name, int age, String disease) {
        this.name = name;
        this.age = age;
        this.disease = disease;
    }

    public void updateDisease(String newDisease) {
        this.disease = newDisease;
        System.out.println(name + "'s disease updated to: " + disease);
    }

    public void displayDetails() {
        System.out.println("Patient Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Disease: " + disease);
    }

    public static void main(String[] args) {
        Patient patient1 = new Patient(name:"Alice", age:30, disease:"Flu");
        patient1.displayDetails();
        patient1.updateDisease(newDisease:"Cold");
    }
}
```

Output:

```
D:\oops>javac -d . mypackage/Patient.java

D:\oops>java mypackage.Patient
Patient Name: Alice
Age: 30
Disease: Flu
Alice's disease updated to: Cold
```

15.b) User Defined Packages

Code:

```java
package mypackage;

public class Product {
    private String name;
    private double price;
    private int stock;

    public Product(String name, double price, int stock) {
        this.name = name;
        this.price = price;
        this.stock = stock;
    }

    public void sellProduct(int quantity) {
        if (quantity > 0 && quantity <= stock) {
            stock -= quantity;
            System.out.println(quantity + " " + name + "(s) sold.");
        } else {
            System.out.println("Insufficient stock for " + name);
        }
    }

    public void restock(int quantity) {
        if (quantity > 0) {
            stock += quantity;
            System.out.println(name + " restocked: " + quantity + " added.");
        }
    }

    public void displayProduct() {
        System.out.println("Product: " + name + ", Price: $" + price + ", Stock: " + stock);
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Product laptop = new Product(name:"Laptop", price:1200, stock:5);
        laptop.displayProduct();
        laptop.sellProduct(quantity:2);
        laptop.restock(quantity:3);
        laptop.displayProduct();
    }
}
```

Output:

```
D:\oops>javac -d . mypackage/Product.java

D:\oops>java mypackage.Product
Product: Laptop, Price: $1200.0, Stock: 5
2 Laptop(s) sold.
Laptop restocked: 3 added.
Product: Laptop, Price: $1200.0, Stock: 6
```

15c) Built – in Package(3 Packages)

Code:

```java
import java.lang.*;
import java.util.regex.*;
import java.time.*;

public class StringUtility {
    public static boolean isValidUsername(String username) {
        String usernameRegex = "^[a-zA-Z]\\w{4,14}$";
        Pattern pattern = Pattern.compile(usernameRegex);
        Matcher matcher = pattern.matcher(username);
        return matcher.matches();
    }
public static void main(String[] args) {
     String name = "JavaProgramming";
        System.out.println("Original Name: " + name);
        System.out.println("Uppercase: " + name.toUpperCase());
        System.out.println("Substring (5-10): " + name.substring(5, 10));


        String user1 = "John123";
        String user2 = "1InvalidUser";
        System.out.println("User 1 Valid: " + isValidUsername(user1));
        System.out.println("User 2 Valid: " + isValidUsername(user2));
        LocalDate today = LocalDate.now();
        System.out.println("Today's Date: " + today);
    }
}
```

Output:

```
D:\oops>javac StringUtility.java

D:\oops>java StringUtility
Original Name: JavaProgramming
Uppercase: JAVAPROGRAMMING
Substring (5-10): rogra
User 1 Valid: true
User 2 Valid: false
Today's Date: 2025-04-04
```

15d) Built – in Package(3 Packages)

Code:

```java
import java.lang.*;
import java.util.regex.*;
import java.time.*;

public class PasswordValidator {
    public static boolean isValidPassword(String password) {
        String passwordRegex = "^(?=.*[A-Z])(?=.*\\d)(?=.*[@#$%^&+=]).{8,}$";
        Pattern pattern = Pattern.compile(passwordRegex);
        Matcher matcher = pattern.matcher(password);
        return matcher.matches();
    }

    public static void main(String[] args) {
        String pass1 = "Secure@123";
        String pass2 = "weakpass";

        System.out.println("Password 1 Valid: " + isValidPassword(pass1));
        System.out.println("Password 2 Valid: " + isValidPassword(pass2));

        LocalDateTime now = LocalDateTime.now();
        System.out.println("Checked on: " + now);
    }
}
```

Output:

```
D:\oops>javac PasswordValidator.java

D:\oops>java PasswordValidator
Password 1 Valid: true
Password 2 Valid: false
Checked on: 2025-04-04T16:07:57.619313
```

16)EXCEPTION HANDLING PROGRAMS

16a) FileExceptionExample

Code:

```java
import java.io.*;

public class FileExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader file = new FileReader("test.txt");
        } catch (FileNotFoundException e) {
            System.out.println("File not found!");
        }
    }
}
```

Output:

```
D:\oops>javac FileExceptionExample.java

D:\oops>java FileExceptionExample
File not found!
```

16b) Division Example

Code:

```java
public class DivisionExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero!");
        }
    }
}
```

Output:

```
D:\oops>javac DivisionExample.java

D:\oops>java DivisionExample
Cannot divide by zero!
```

16c) StringIndexOutOfBoundsExample

Code:

```java
public class StringIndexOutOfBoundsExample {
    public static void main(String[] args) {
        try {
            String str = "Hello";
            System.out.println(str.charAt(10));
        } catch (StringIndexOutOfBoundsException e) {
            System.out.println("String index is out of bounds!");
        }
    }
}
```

Output:

```
D:\oops>javac StringIndexOutOfBoundsExample.java

D:\oops>java StringIndexOutOfBoundsExample
String index is out of bounds!
```

16d)NumberFormatExample

Code:

```java
public class NumberFormatExample {
    public static void main(String[] args) {
        try {
            int num = Integer.parseInt("ABC");
        } catch (NumberFormatException e) {
            System.out.println("Invalid number format!");
        }
    }
}
```

Output:

```
D:\oops>javac NumberFormatExample.java

D:\oops>java NumberFormatExample
Invalid number format!
```

17)FILE HANDLING PROGRAMS

17a) WriteFileExample

Code:

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

public class WriteFileExample {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("sample.txt");
            writer.write("Hello, this is a sample text file.");
            writer.close();
            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
        }
    }
}
```

Output:

```
D:\oops>javac WriteFileExample.java

D:\oops>java WriteFileExample
Successfully wrote to the file.
```

17b) FileExistsExample

Code:

```java
import java.io.File;

public class FileExistsExample {
    public static void main(String[] args) {
        File file = new File("sample.txt");
        if (file.exists()) {
            System.out.println("File exists.");
        } else {
            System.out.println("File does not exist.");
        }
    }
}
```

Output:

```
D:\oops>javac FileExistsExample.java

D:\oops>java FileExistsExample
File exists.
```

17c) CreateFileExample

Code:

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

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

Output:

```
D:\oops>javac CreateFileExample.java

D:\oops>java CreateFileExample
File already exists.
```

17d) AppendFileExample

Code:

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

public class AppendFileExample {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("system.txt", true);
            writer.write("\nAppending this line to the file.");
            writer.close();
            System.out.println("Successfully appended to the file.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
        }
    }
}
```

Output:

```
D:\oops>javac AppendFileExample.java

D:\oops>java AppendFileExample
Successfully appended to the file.
```