LAB RECORD

23CSE111 – Object Oriented Programming

Submitted by

CH.SC.U4CSE24133 – N.ASHWATH

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

April - 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.U4CSE24133 – N.ASHWATH*** 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

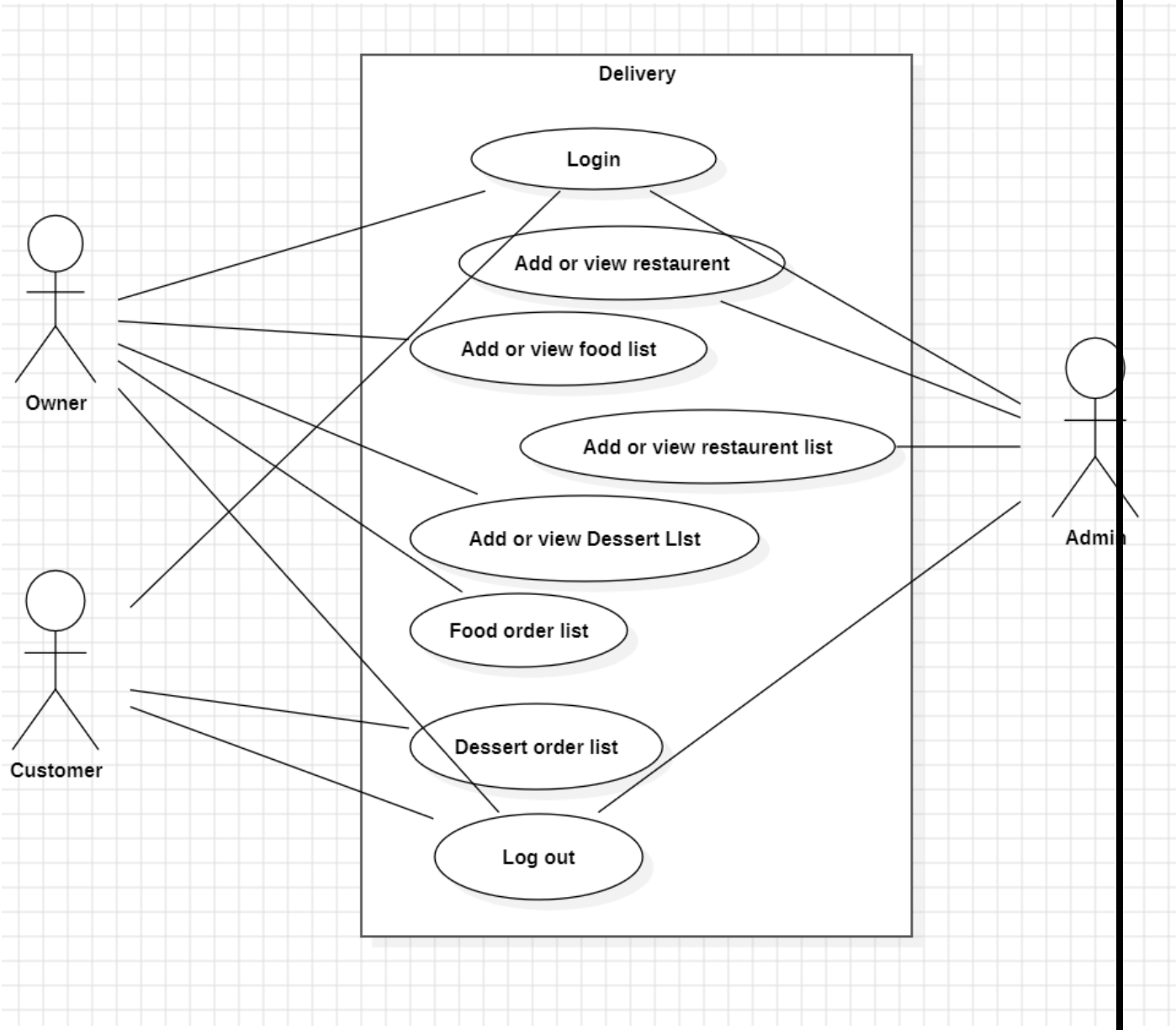
Internal Examiner 1 Internal Examiner 2

Index

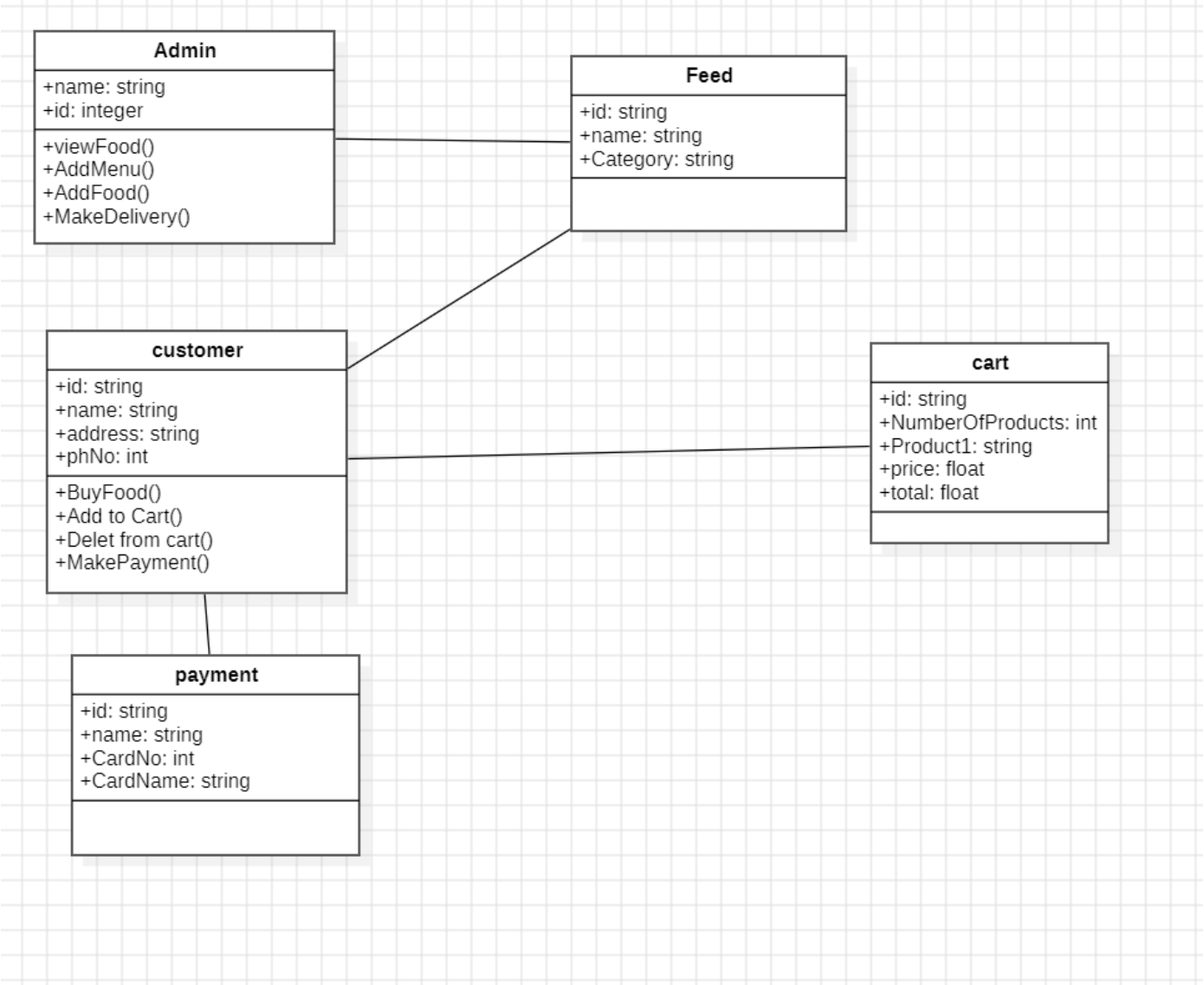
EXPERIMENT-1

DELIVERY

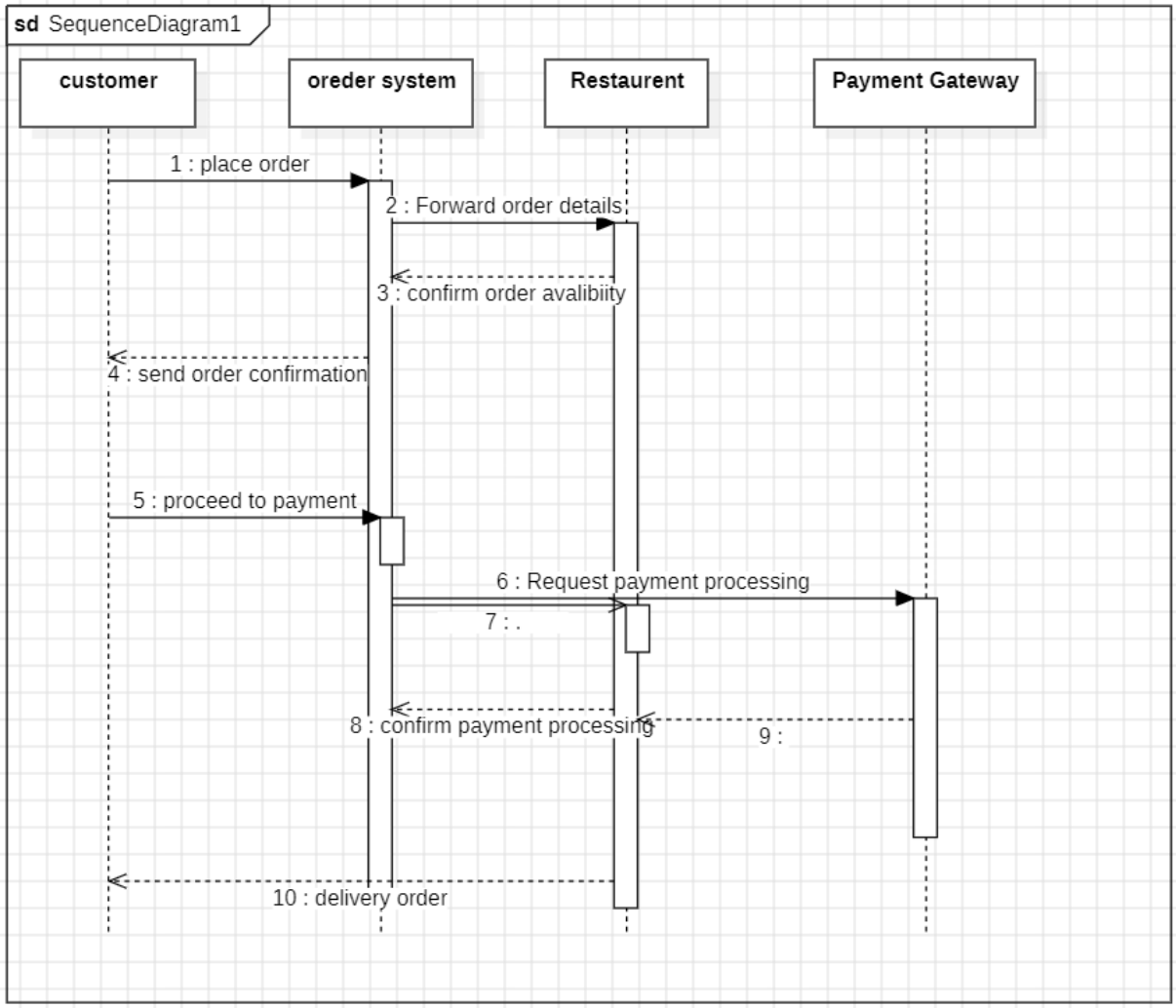
1) Use case



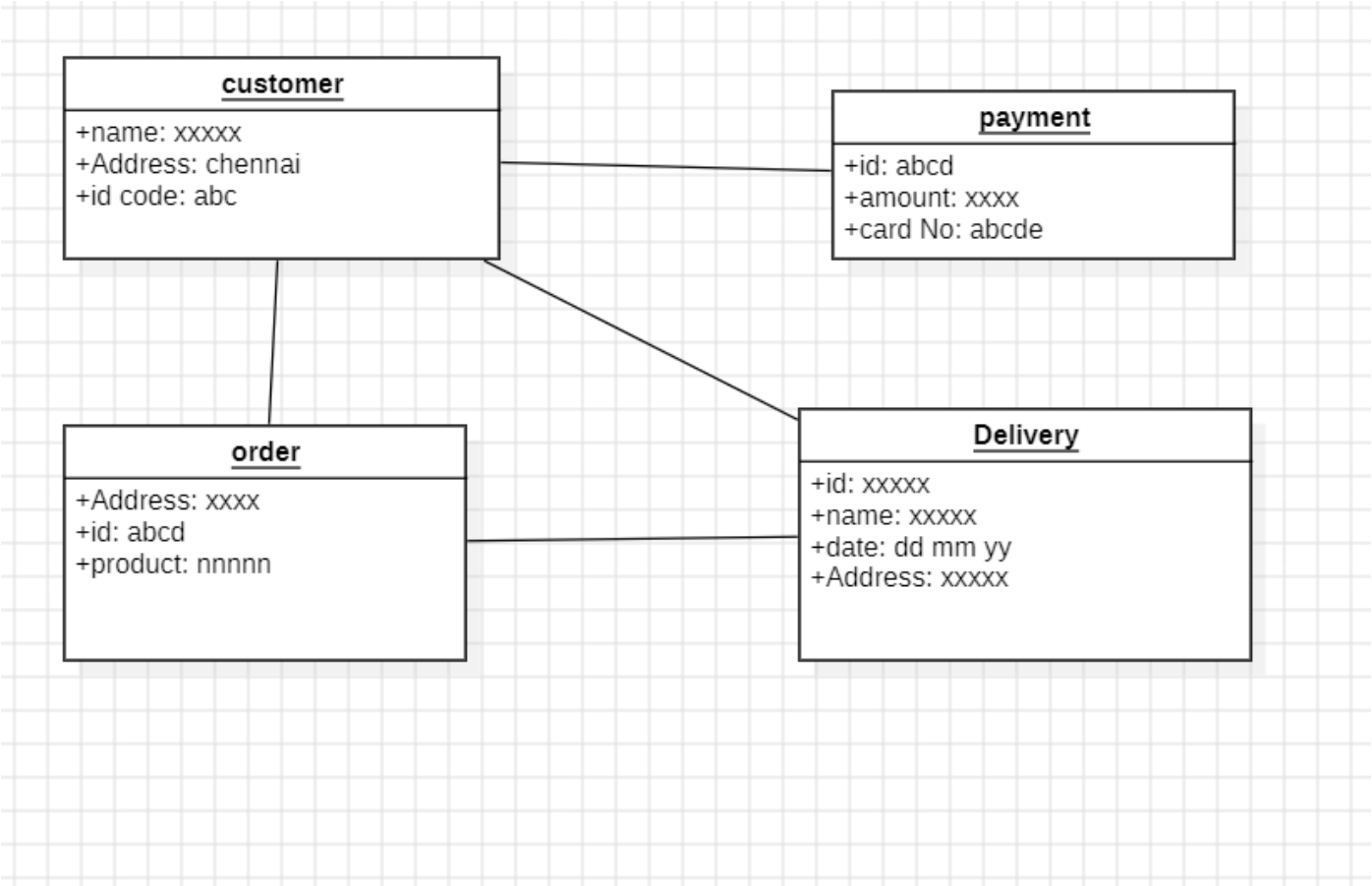
2) class



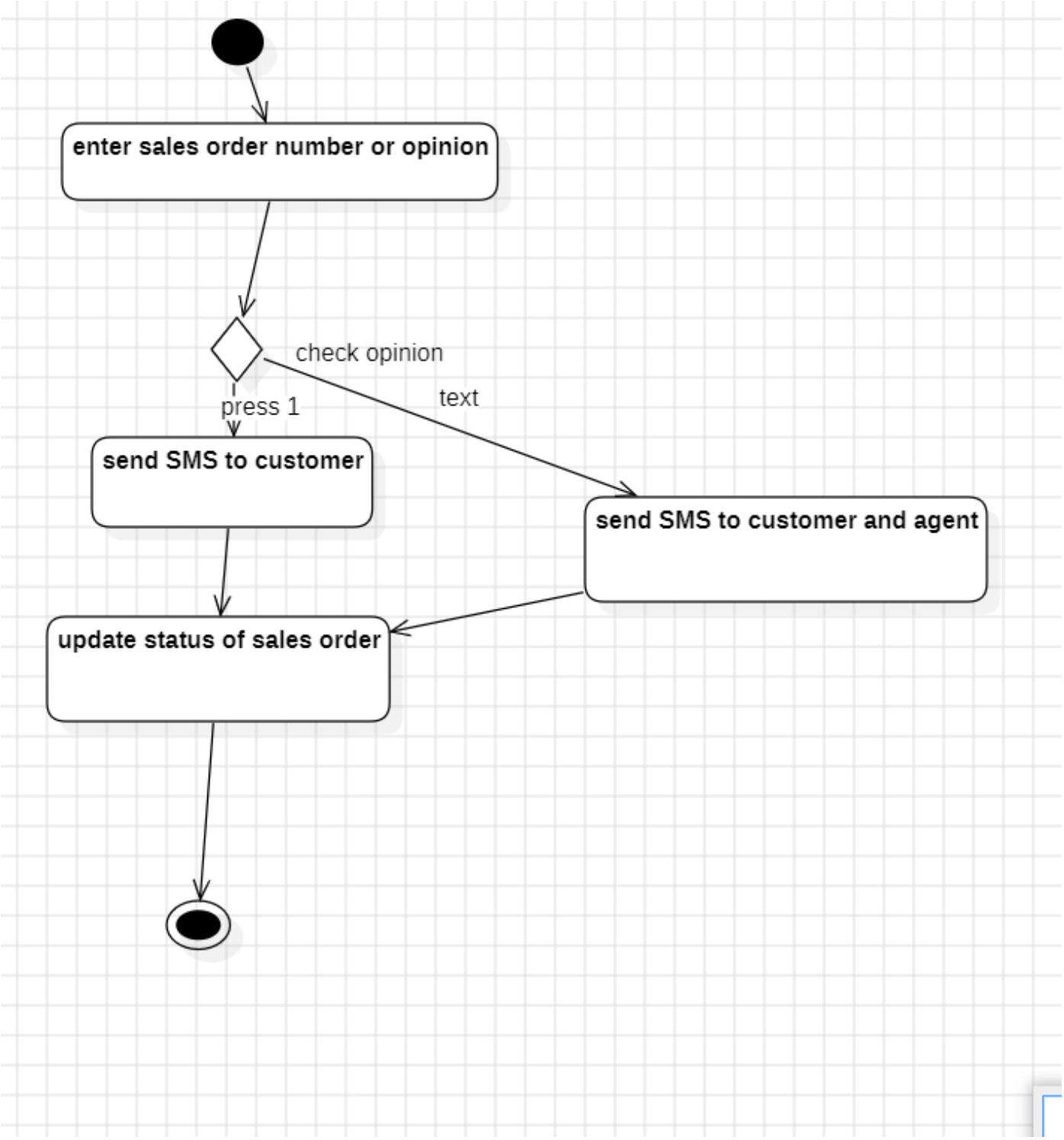
3) sequence



4) object diagram



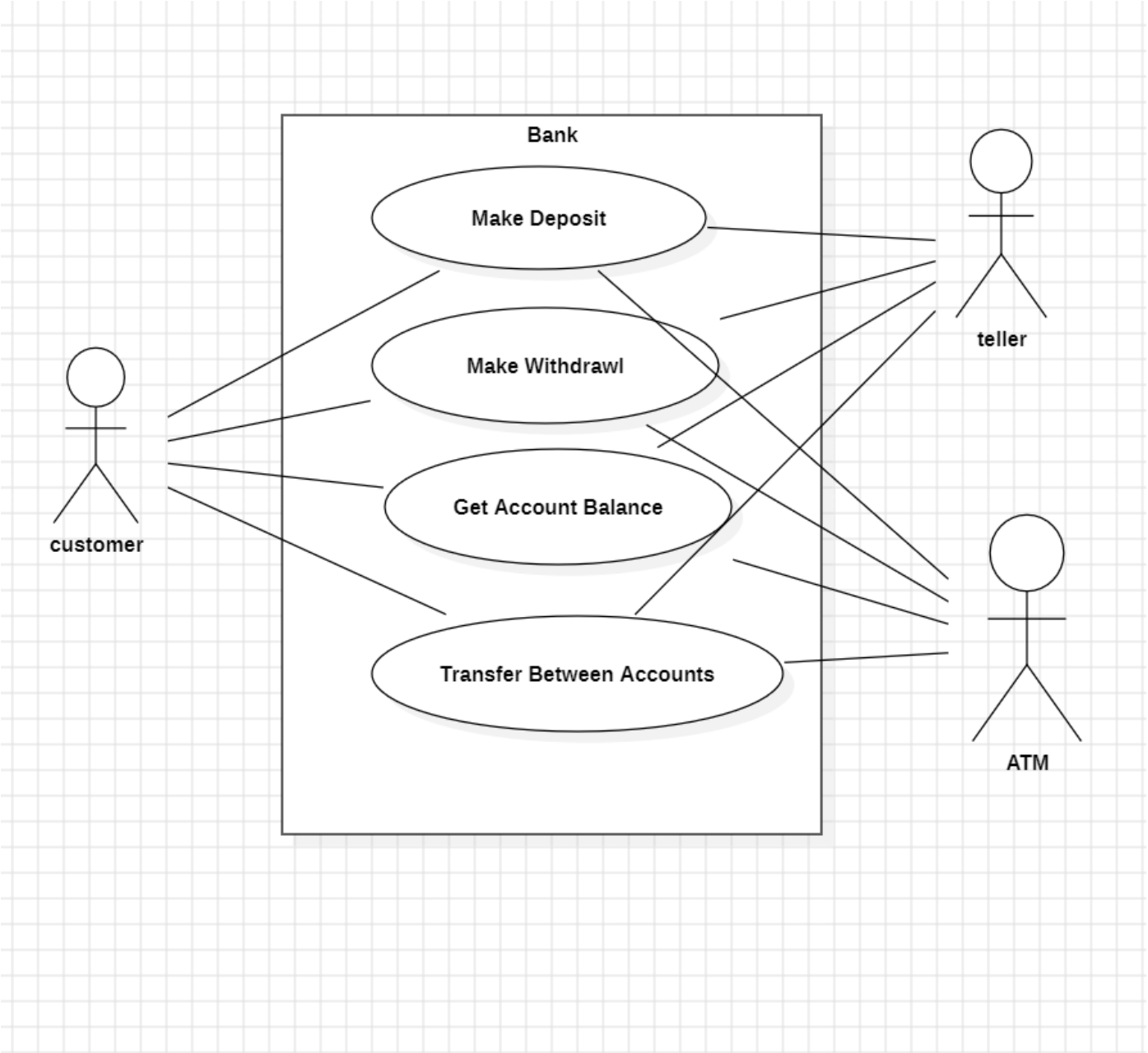
5) Activity diagram


enter sales order number or opinion
check opinion
press 1
text
send SMS to customer
send SMS to customer and agent
update status of sales order

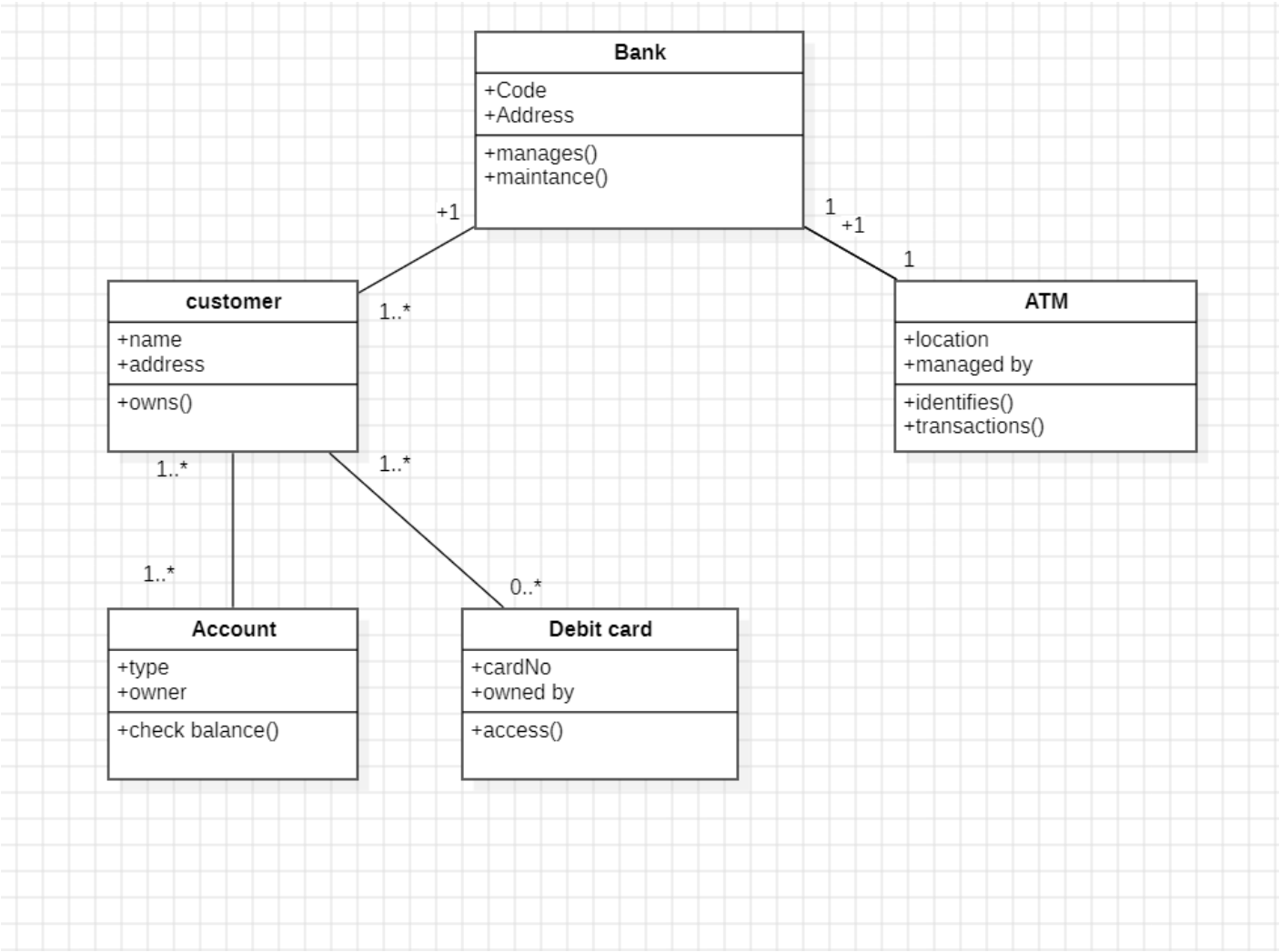
EXPERIMENT-2

BANK

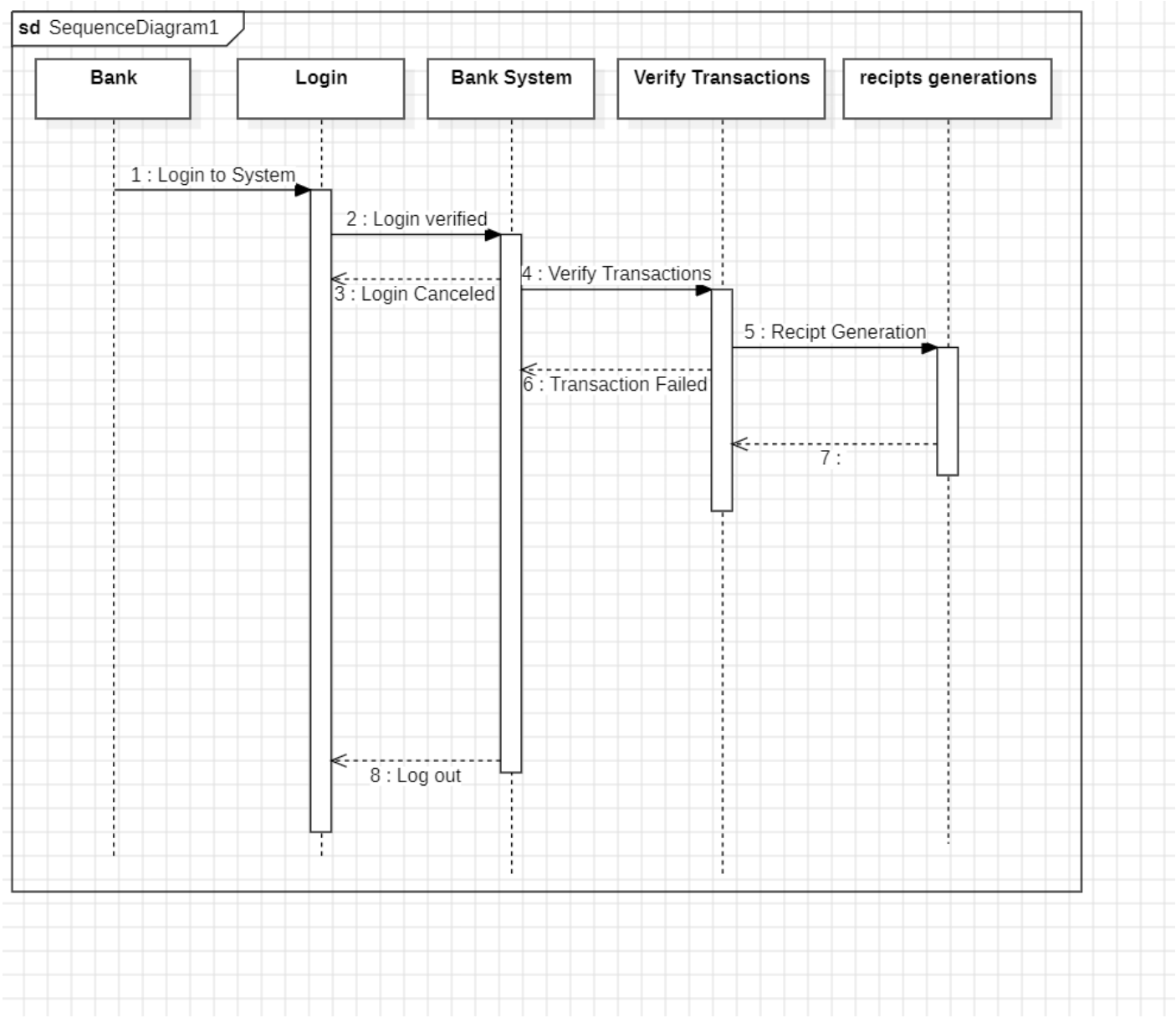
1) use case



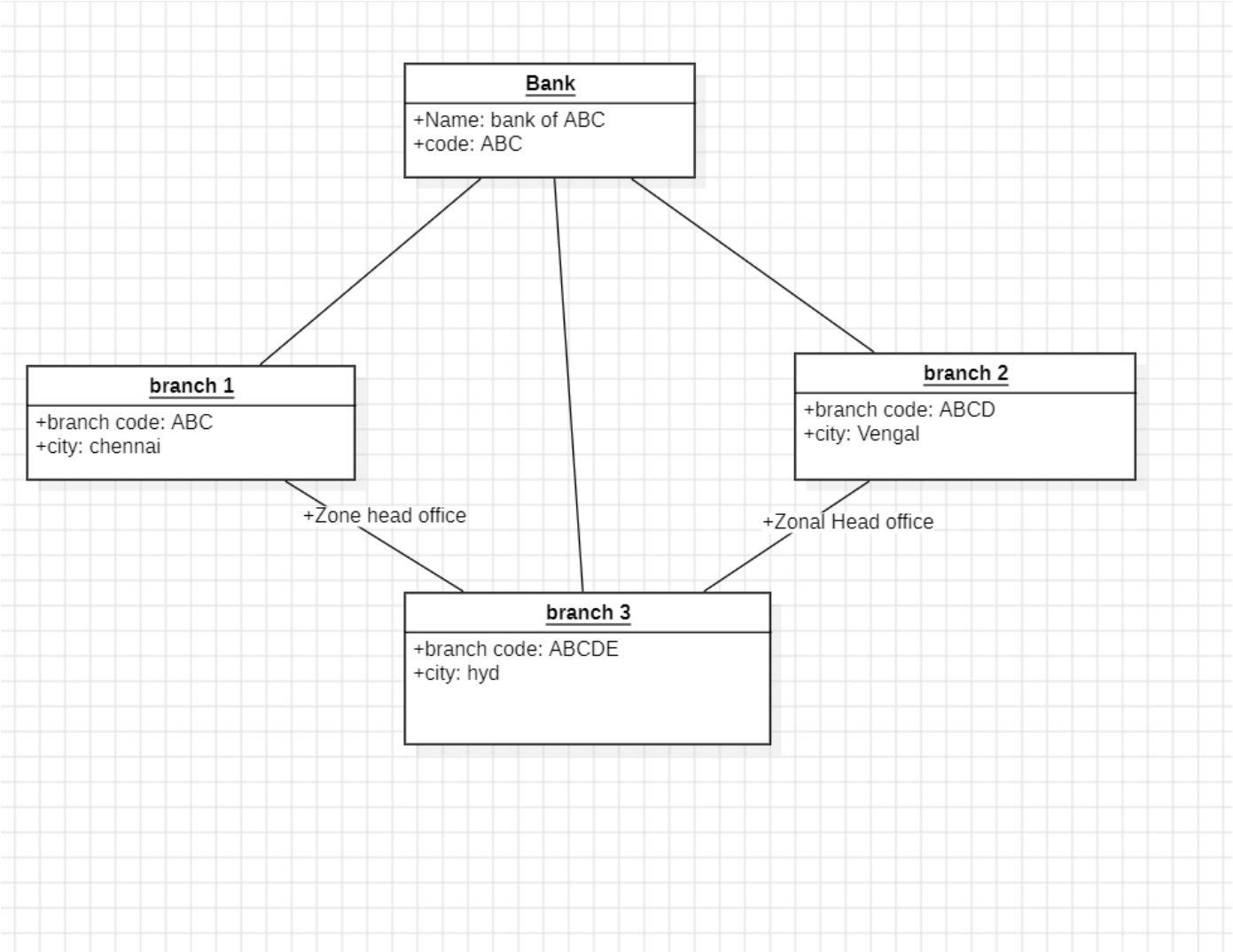
2) class



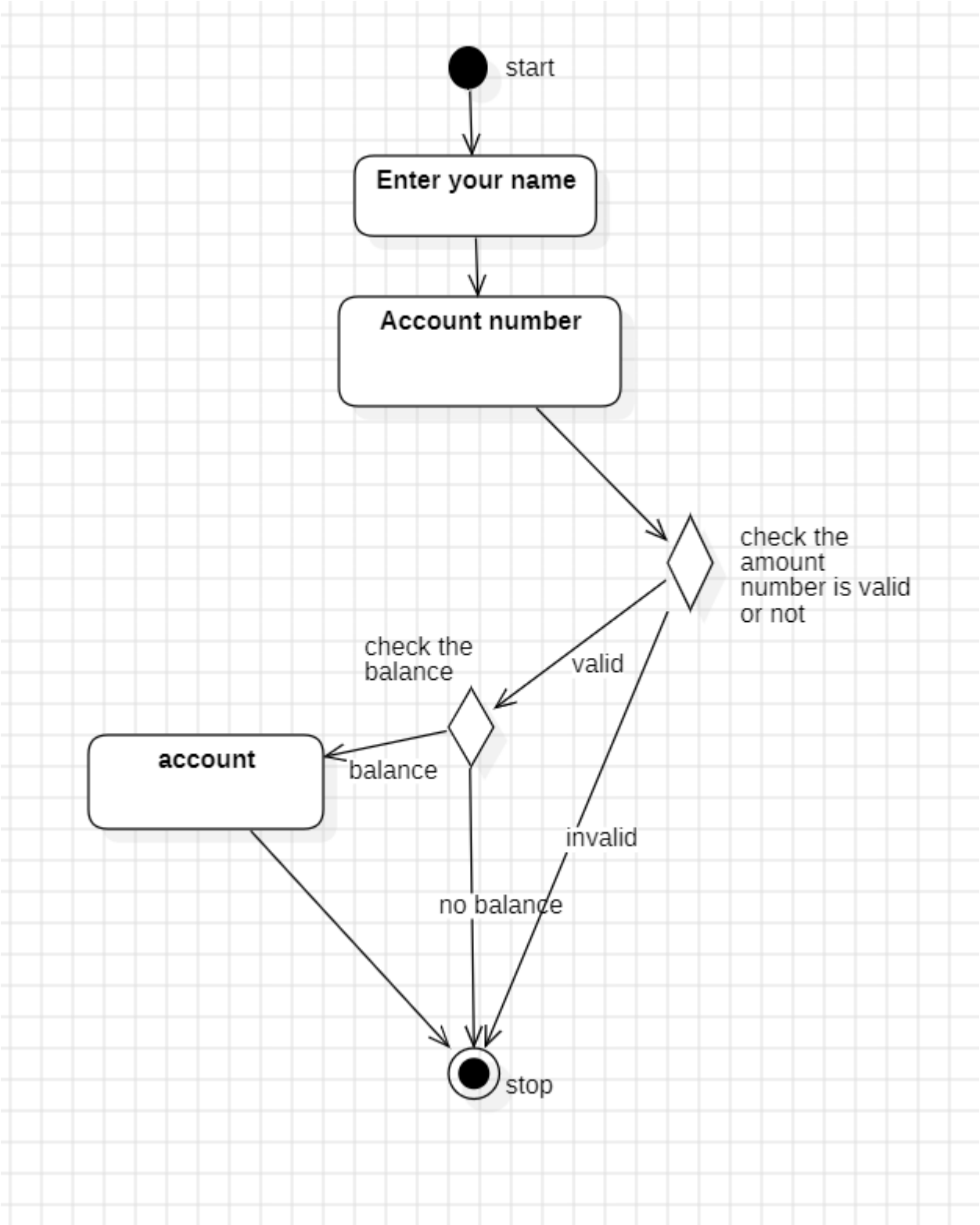
3) sequence



4) object diagram



5) activity diagram


start
Enter your name
Account number
check the
amount
number is valid
or not
valid
check the
balance
account
balance
invalid
no balance
stop

EXPERIMENT-3

1. For Loop Example

   Java Code:

```
public class ForLoopExample {
    public static void main(String[] args) {
        System.out.println("For Loop:");
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);
        }
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac DoWhileLoopExample.java

C:\Users\ashwa\OneDrive\experiment-3>java DoWhileLoopExample.java
Do-While Loop:
1
2
3
4
5
6
7
8
9
10

C:\Users\ashwa\OneDrive\experiment-3>
```

2. While Loop Example

Java Code:

```
public class WhileLoopExample {
    public static void main(String[] args) {
        int i = 1;
        System.out.println("While Loop:");
        while (i <= 10) {
            System.out.println(i);
            i++;
        }
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac WhileLoopExample.java

C:\Users\ashwa\OneDrive\experiment-3>java WhileLoopExample.java
While Loop:
1
2
3
4
5
6
7
8
9
10

C:\Users\ashwa\OneDrive\experiment-3>
```

3. Do-While Loop Example

Java Code:

```
public class DoWhileLoopExample {
    public static void main(String[] args) {
        int i = 1;
        System.out.println("Do-While Loop:");
        do {
            System.out.println(i);
            i++;
        } while (i <= 10);
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac DoWhileLoopExample.java

C:\Users\ashwa\OneDrive\experiment-3>java DoWhileLoopExample.java
Do-While Loop:
1
2
3
4
5
6
7
8
9
10

C:\Users\ashwa\OneDrive\experiment-3>
```

4. Sum of First N Numbers (Using For Loop)

Java Code:

```
public class SumUsingForLoop {
    public static void main(String[] args) {
        int n = 5, sum = 0;
        for (int i = 1; i <= n; i++) {
            sum += i;
        }
        System.out.println("Sum of first " + n + " numbers: " + sum);
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac SumUsingForLoop.java

C:\Users\ashwa\OneDrive\experiment-3>java SumUsingForLoop.java
Sum of first 5 numbers: 15
```

5. Multiplication Table (Using While Loop)

Java Code:

```
public class MultiplicationTable {
    public static void main(String[] args) {
        int num = 5, i = 1;
```

```
        System.out.println("Multiplication Table of " +
num + ":");
        while (i <= 10) {
            System.out.println(num + " x " + i + " = " +
(num * i));
            i++;
        }
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac MultiplicationTable.java

C:\Users\ashwa\OneDrive\experiment-3>java MultiplicationTable.java
Multiplication Table of 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

C:\Users\ashwa\OneDrive\experiment-3>
```

6. Reverse Number (Using Do-While Loop)

Java Code:

```
public class ReverseNumber {
    public static void main(String[] args) {
        int num = 1234, reversed = 0;
        do {
            int digit = num % 10;
```

```
            reversed = reversed * 10 + digit;
            num /= 10;
        } while (num != 0);
        System.out.println("Reversed Number: " +
reversed);
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac ReverseNumber.java

C:\Users\ashwa\OneDrive\experiment-3>java ReverseNumber.java
Reversed Number: 4321
```

7. Fibonacci Series (Using For Loop)

Java Code:

```
public class FibonacciForLoop {
    public static void main(String[] args) {
        int n = 10, a = 0, b = 1;
        System.out.print("Fibonacci Series: ");
        for (int i = 1; i <= n; i++) {
            System.out.print(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac FibonacciForLoop.java

C:\Users\ashwa\OneDrive\experiment-3>java FibonacciForLoop.java
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
C:\Users\ashwa\OneDrive\experiment-3>
```

8. Check Prime Number (Using While Loop)

Java Code:

```
public class PrimeNumberCheck {
    public static void main(String[] args) {
        int num = 29, i = 2;
        boolean isPrime = true;

        while (i <= num / 2) {
            if (num % i == 0) {
                isPrime = false;
                break;
            }
            i++;
        }

        if (isPrime)
            System.out.println(num + " is a Prime
Number");
        else
            System.out.println(num + " is Not a Prime
Number");
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac PrimeNumberCheck.java

C:\Users\ashwa\OneDrive\experiment-3>java PrimeNumberCheck.java
29 is a Prime Number
```

9. Factorial Using Do-While Loop

Java Code:

```
public class FactorialDoWhile {
    public static void main(String[] args) {
        int num = 5, fact = 1;
        int i = 1;
        do {
            fact *= i;
            i++;
        } while (i <= num);
        System.out.println("Factorial of " + num + "
is: " + fact);
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac FactorialDoWhile.java

C:\Users\ashwa\OneDrive\experiment-3>java FactorialDoWhile.java
Factorial of 5 is: 120
```

10. Infinite Loop Example (Using While Loop)

Java Code:

```
public class InfiniteLoop {
    public static void main(String[] args) {
        while (true) {
            System.out.println("This is an infinite loop.
Press Ctrl+C to stop.");
        }
    }
}
```

Output:

```
C:\Users\ashwa\OneDrive\experiment-3>javac InfiniteLoop.java

C:\Users\ashwa\OneDrive\experiment-3>java InfiniteLoop.java
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
This is an infinite loop. Press Ctrl+C to stop.
```

EXPERIMENT-4

SINGLE INHERITANCE PROGRAMS

4 (a)

Code:

```
// Parent class
class Animal {
    void eat() {
        System.out.println("This animal eats food.");
    }
}

// Child class (single inheritance)
class Dog extends Animal {
    void bark() {
        System.out.println("The dog barks.");
    }

    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();   // inherited method
        d.bark();  // method in child class
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.6.7-hotspot\bin\java.ex
.ls-java-project\bin' 'Dog'
This animal eats food.
The dog barks.
```

4 (b)

Code:

```
// Parent class
class Shape {
    double length = 5;
    double width = 3;
```

```
    void displayDimensions() {
        System.out.println("Length: " + length + ", Width: " + width);
    }
}

// Child class
class Rectangle extends Shape {
    void calculateArea() {
        double area = length * width;
        System.out.println("Area of rectangle: " + area);
    }

    public static void main(String[] args) {
        Rectangle rect = new Rectangle();
        rect.displayDimensions(); // inherited method
        rect.calculateArea();     // method in child class
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.6.7-hotspot\bin\java
f4920\jdt_ws\jdt.ls-java-project\bin' 'Rectangle'
Length: 5.0, Width: 3.0
Area of rectangle: 15.0
```

EXPERIMENT-5

MULTILEVEL INHERITANCE PROGRAMS

5(a)

Code:

```
class Father:
    def skills(self):
        print("Father: Gardening, Carpentry")

class Mother:
    def skills(self):
        print("Mother: Cooking, Painting")

class Child(Father, Mother):
    def skills(self):
        super().skills()  # Will call Father.skills() due to method resolution order (MRO)
        Mother.skills(self)  # Explicitly calling Mother.skills()
        print("Child: Programming")

# Create object
c = Child()
c.skills()
```

Output:

```
Father: Gardening, Carpentry
Mother: Cooking, Painting
Child: Programming
```

5(b)

```
interface A {
    void methodA();
}

interface B {
    void methodB();
}
```

```
class C implements A, B {
    public void methodA() {
        System.out.println("This is method A");
    }

    public void methodB() {
        System.out.println("This is method B");
    }
}

public class Main {
    public static void main(String[] args) {
        C obj = new C();
        obj.methodA();
        obj.methodB();
    }
}
```

Output:

```
This is method A
This is method B
```

EXPERIMENT-6

6(a)

Code:

```
// Parent class
class Animal {
    void eat() {
        System.out.println("This animal eats food.");
    }
}

// Child class 1
class Dog extends Animal {
    void bark() {
        System.out.println("The dog barks.");
    }
}

// Child class 2
class Cat extends Animal {
    void meow() {
        System.out.println("The cat meows.");
    }
}

// Main class
```

```
public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();  // from Animal
        dog.bark();

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

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.6.7-hotspot\bin\java.exe'
.ls-java-project\bin' 'Main'
This animal eats food.
The dog barks.
This animal eats food.
The cat meows.
PS C:\Users\ashwa>
```

6(b)

Code:

```
// Parent class
class Vehicle {
    void start() {
        System.out.println("Vehicle is starting.");
    }
}
```

```
// Child class 1
class Car extends Vehicle {
    void drive() {
        System.out.println("Car is driving.");
    }
}

// Child class 2
class Bike extends Vehicle {
    void ride() {
        System.out.println("Bike is riding.");
    }
}

// Main class
public class Transport {
    public static void main(String[] args) {
        Car car = new Car();
        car.start();  // from Vehicle
        car.drive();

        Bike bike = new Bike();
        bike.start();  // from Vehicle
        bike.ride();
```

```
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.6.7-hotspot\b
.ls-java-project\bin' 'Transport'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:

        at Transport.main(vehicle.java:24)
PS C:\Users\ashwa>
```

EXPERIMENT-7

7(a)

Code:

```
// Interface A
interface A {
    void showA();
}

// Class B implements A
class B implements A {
    public void showA() {
        System.out.println("Showing A from B");
    }

    public void showB() {
```

```
        System.out.println("Showing B");
    }
}

// Class C
class C {
    public void showC() {
        System.out.println("Showing C");
    }
}

// Class D extends C and uses B (Hybrid Inheritance)
class D extends C {
    B b = new B(); // using composition to access B's methods

    public void showAll() {
        b.showA();
        b.showB();
        showC();
    }
}

public class Main {
    public static void main(String[] args) {
        D obj = new D();
```

```
        obj.showAll();
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse
30468\jdt_ws\jdt.ls-java-project\bin' 'Main'
Showing A from B
Showing B
Showing C
PS C:\Users\ashwa>
```

7(b)

Code:

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

interface MusicSystem {
    void playMusic();
}

class Car {
    void drive() {
        System.out.println("Driving car");
    }
}
```

```
// Hybrid: MyCar extends Car and implements multiple interfaces
class MyCar extends Car implements Engine, MusicSystem {
    public void start() {
        System.out.println("Engine started");
    }

    public void playMusic() {
        System.out.println("Playing music");
    }
}

public class Main2 {
    public static void main(String[] args) {
        MyCar car = new MyCar();
        car.start();
        car.playMusic();
        car.drive();
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclips
30468\jdt_ws\jdt.ls-java-project\bin' 'Main2'
Engine started
Playing music
Car is driving.
PS C:\Users\ashwa>
```

EXPERIMENT-8

8(a)

Code:

```
// A simple class with a constructor
public class Car {
    String brand;
    int year;

    // Constructor
    Car(String b, int y) {
        brand = b;
        year = y;
    }

    // Method to display car info
    void display() {
        System.out.println("Brand: " + brand);
        System.out.println("Year: " + year);
    }

    // Main method to run the program
    public static void main(String[] args) {
        // Creating an object of Car using constructor
        Car myCar = new Car("Toyota", 2022);
```

```
        // Displaying car details
        myCar.display();
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.6.7-hotspot\bi
30468\jdt_ws\jdt.ls-java-project\bin' 'Car'
Error: Main method not found in class Car, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
PS C:\Users\ashwa>
```

EXPERIMENT-9

9(a)

Code:

```
class Student {
    String name;
    int age;

    // Constructor 1: No arguments
    Student() {
        name = "Unknown";
        age = 0;
    }
```

```
    // Constructor 2: One argument
    Student(String n) {
        name = n;
        age = 18; // default age
    }

    // Constructor 3: Two arguments
    Student(String n, int a) {
        name = n;
        age = a;
    }

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

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student();                // calls Constructor 1
        Student s2 = new Student("Alice");         // calls Constructor
2
        Student s3 = new Student("Bob", 21);       // calls
Constructor 3
```

```
        s1.display();
        s2.display();
        s3.display();
    }
}
```

Output:

```
30468\jdt_ws\jdt.ls-java-project\bin' 'Main'
Showing A from B
Showing B
Showing C
PS C:\Users\ashwa> ^C
PS C:\Users\ashwa>
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.6.7-
30468\jdt_ws\jdt.ls-java-project\bin' 'Main'
Showing A from B
Showing B
Showing C
```

EXPERIMENT-10

10(a)

Code:

```
public class Calculator {

    // Method to add two integers
    public int add(int a, int b) {
        return a + b;
    }

    // Overloaded method to add three integers
```

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

    // Overloaded method to add two doubles
    public double add(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum of 5 and 10: " + calc.add(5, 10));
        System.out.println("Sum of 1, 2, and 3: " + calc.add(1, 2, 3));
        System.out.println("Sum of 4.5 and 3.2: " + calc.add(4.5, 3.2));
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adopti
30468\jdt_ws\jdt.ls-java-project\bin' 'Calculator'
Sum of 5 and 10: 15
Sum of 1, 2, and 3: 6
Sum of 4.5 and 3.2: 7.7
PS C:\Users\ashwa>
```

10(b)

Code:

```
public class Greeter {

    // Method with no parameters
    public void greet() {
        System.out.println("Hello!");
    }

    // Overloaded method with one parameter
```

```
    public void greet(String name) {
        System.out.println("Hello, " + name + "!");
    }

    // Overloaded method with two parameters
    public void greet(String name, int age) {
        System.out.println("Hello, " + name + "! You are " + age + " years old.");
    }

    public static void main(String[] args) {
        Greeter greeter = new Greeter();
        greeter.greet();
        greeter.greet("Alice");
        greeter.greet("Bob", 25);
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files
30468\jdt_ws\jdt.ls-java-project\bin' '
Hello!
Hello, Alice!
Hello, Bob! You are 25 years old.
PS C:\Users\ashwa>
```

EXPERIMENT-11

11(a)

Code:

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

class Dog extends Animal {
```

```
    // Overriding the sound method
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal myAnimal = new Animal(); // Animal reference and object
        myAnimal.sound(); // Calls Animal's method

        Animal myDog = new Dog(); // Animal reference but Dog object
        myDog.sound(); // Calls Dog's overridden method
    }
}
```

Output:

```
PS C:\Users\ashwa>  & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.6.7-hotspot\
.ls-java-project\bin' 'Main'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:

        at Main.main(Animal.java:16)
PS C:\Users\ashwa>
```

11(b)

Code:

```
class Shape {
    void area() {
        System.out.println("Calculating area of a shape");
    }
}

class Circle extends Shape {
    @Override
```

```
    void area() {
        System.out.println("Area of circle = π * r * r");
    }
}

public class TestOverride {
    public static void main(String[] args) {
        Shape s1 = new Shape();
        s1.area();  // Calls Shape's method

        Shape s2 = new Circle();
        s2.area();  // Calls Circle's overridden method
    }
}
```

Output:

```
Calculating area of a shape
Area of circle = π * r * r
```