Week 10 - S10 - Advanced OOP - UML Diagram - Lab Problem

**Name:** Ramesh Harisabapathi Chettiar

**Date of Submission:** 22/10/25

**QNO1→**

**Problem Statement:**

Draw a UML Class Diagram for a simple Library Management System with classes like Book, Member, and Librarian. Show relationships such as association and aggregation.

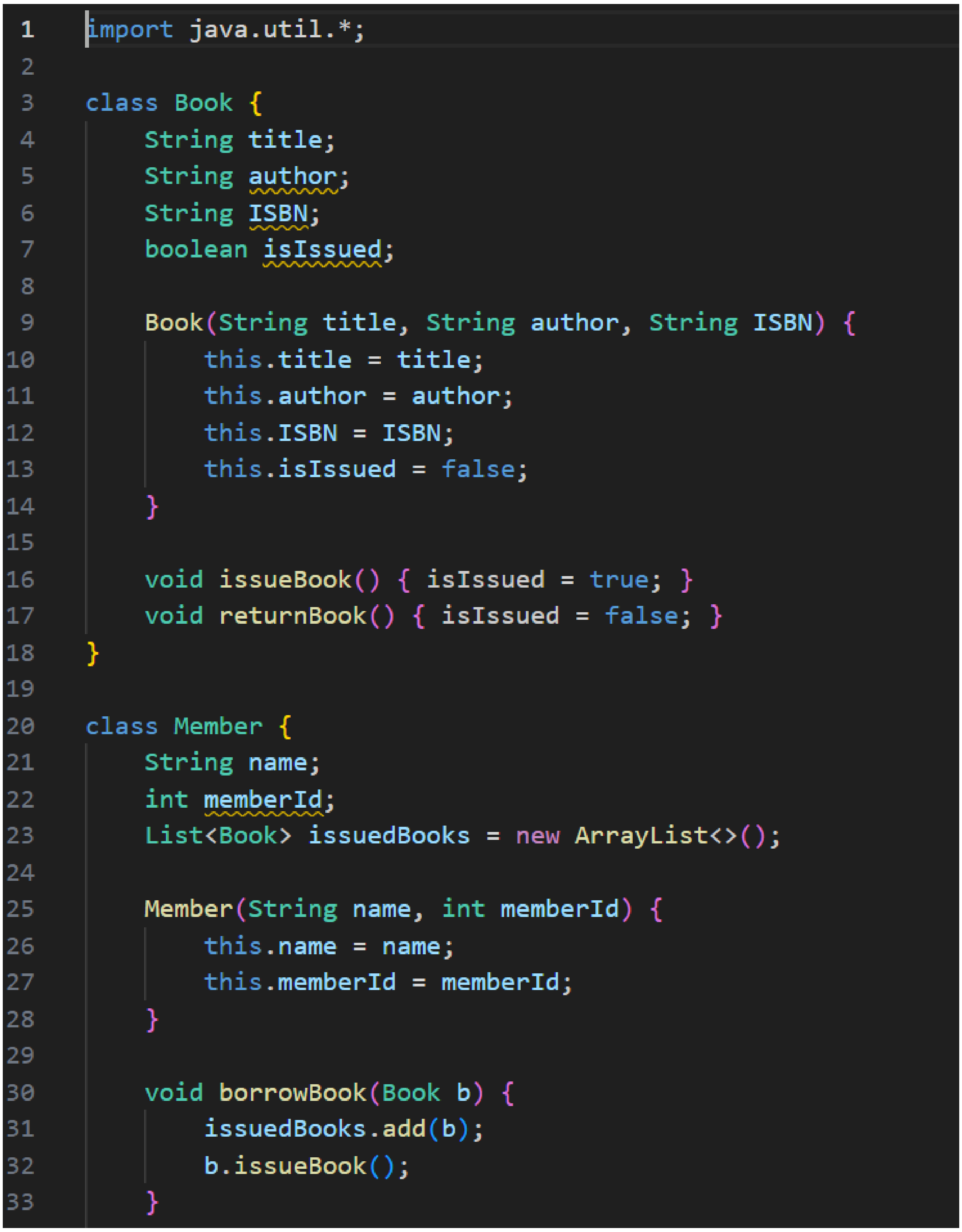
**Hints:**

● Use attributes and methods inside each class.

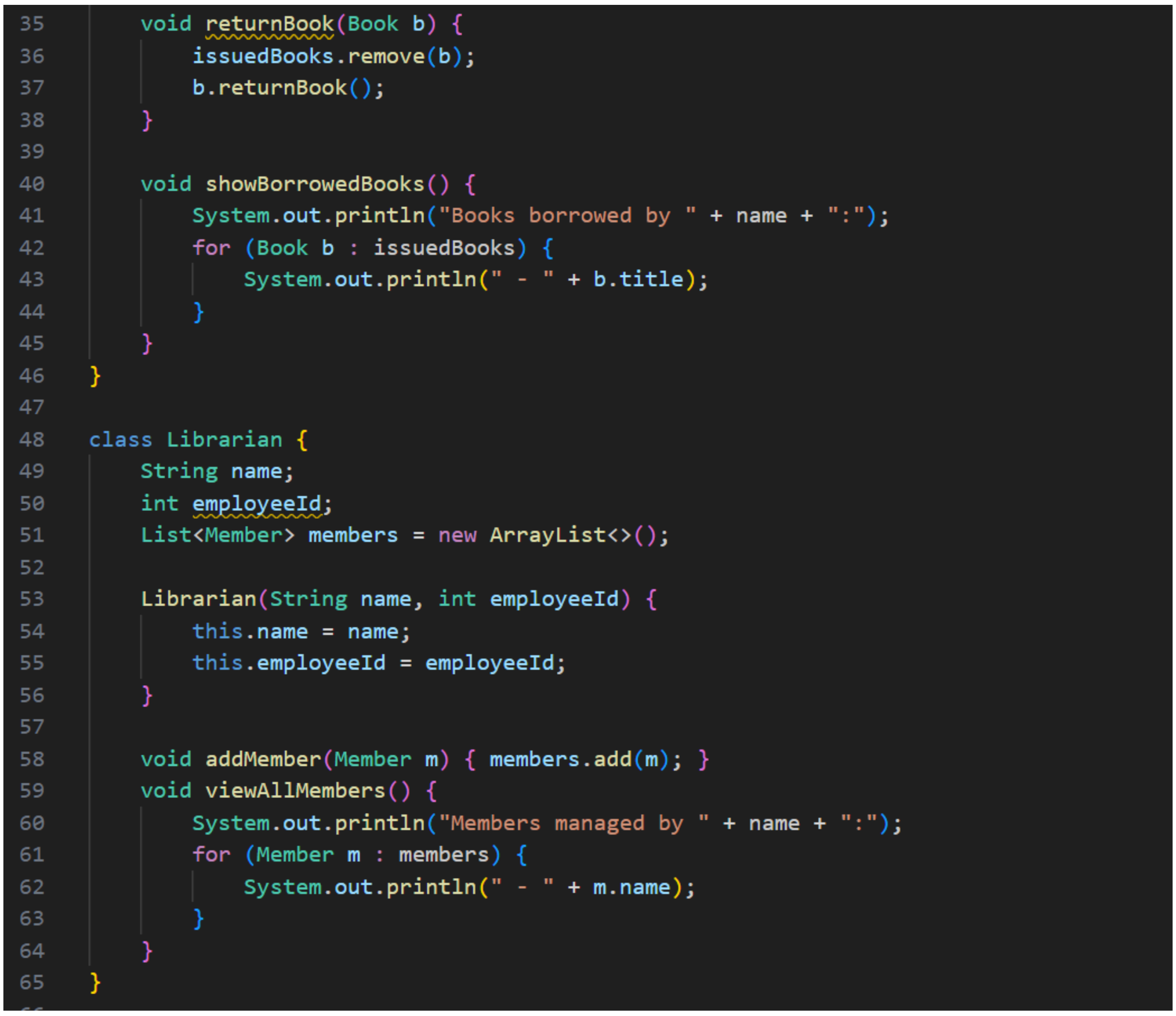
● Show 1-to-many association between Member and Book.

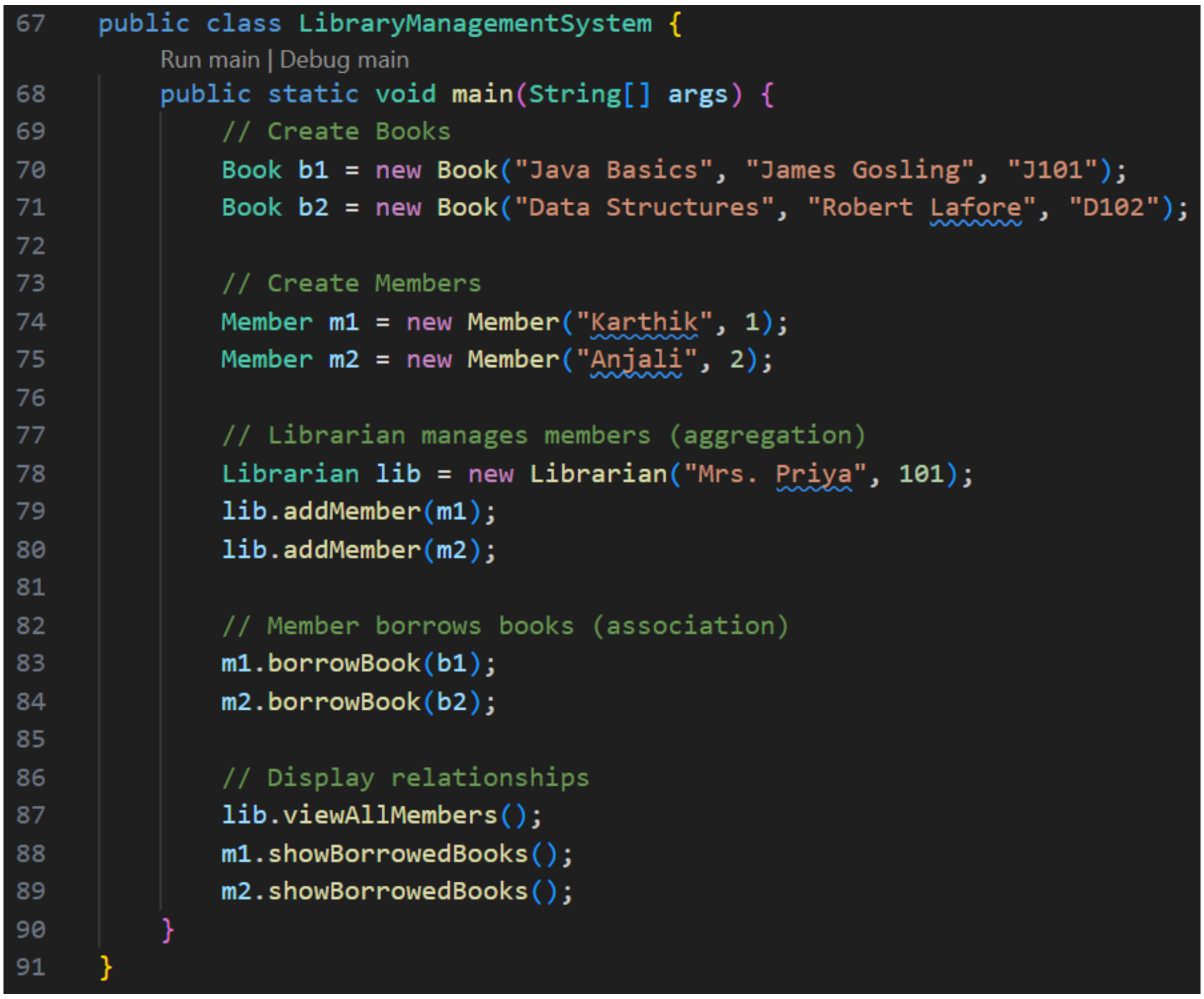
● Mark the relationship direction using arrows.

LibraryManagementSystem.java



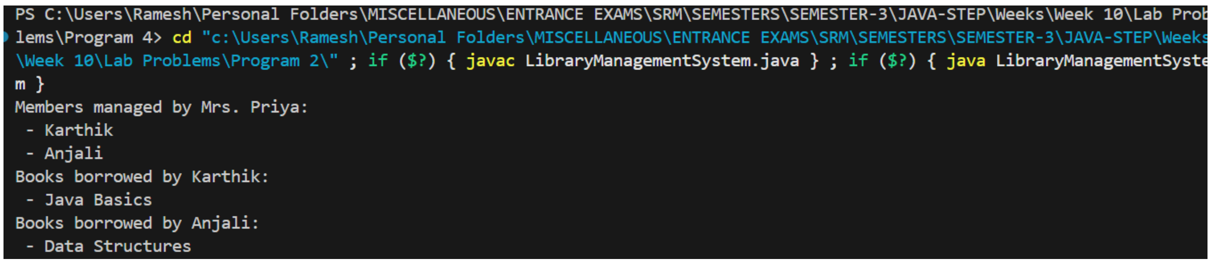
```java
import java.util.*;

class Book {
    String title;
    String author;
    String ISBN;
    boolean isIssued;

    Book(String title, String author, String ISBN) {
        this.title = title;
        this.author = author;
        this.ISBN = ISBN;
        this.isIssued = false;
    }

    void issueBook() { isIssued = true; }
    void returnBook() { isIssued = false; }
}

class Member {
    String name;
    int memberId;
    List<Book> issuedBooks = new ArrayList<>();

    Member(String name, int memberId) {
        this.name = name;
        this.memberId = memberId;
    }

    void borrowBook(Book b) {
        issuedBooks.add(b);
        b.issueBook();
    }
```


```java
    void returnBook(Book b) {
        issuedBooks.remove(b);
        b.returnBook();
    }

    void showBorrowedBooks() {
        System.out.println("Books borrowed by " + name + ":");
        for (Book b : issuedBooks) {
            System.out.println(" - " + b.title);
        }
    }
}

class Librarian {
    String name;
    int employeeId;
    List<Member> members = new ArrayList<>();

    Librarian(String name, int employeeId) {
        this.name = name;
        this.employeeId = employeeId;
    }

    void addMember(Member m) { members.add(m); }
    void viewAllMembers() {
        System.out.println("Members managed by " + name + ":");
        for (Member m : members) {
            System.out.println(" - " + m.name);
        }
    }
}
```



```java
public class LibraryManagementSystem {
    public static void main(String[] args) {
        // Create Books
        Book b1 = new Book("Java Basics", "James Gosling", "J101");
        Book b2 = new Book("Data Structures", "Robert Lafore", "D102");

        // Create Members
        Member m1 = new Member("Karthik", 1);
        Member m2 = new Member("Anjali", 2);

        // Librarian manages members (aggregation)
        Librarian lib = new Librarian("Mrs. Priya", 101);
        lib.addMember(m1);
        lib.addMember(m2);

        // Member borrows books (association)
        m1.borrowBook(b1);
        m2.borrowBook(b2);

        // Display relationships
        lib.viewAllMembers();
        m1.showBorrowedBooks();
        m2.showBorrowedBooks();
    }
}
```

**OUTPUT→**



```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 10\Lab Prob
lems\Program 4> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks
\Week 10\Lab Problems\Program 2\" ; if ($?) { javac LibraryManagementSystem.java } ; if ($?) { java LibraryManagementSyste
m }
Members managed by Mrs. Priya:
 - Karthik
 - Anjali
Books borrowed by Karthik:
 - Java Basics
Books borrowed by Anjali:
 - Data Structures
```

**QNO2→**

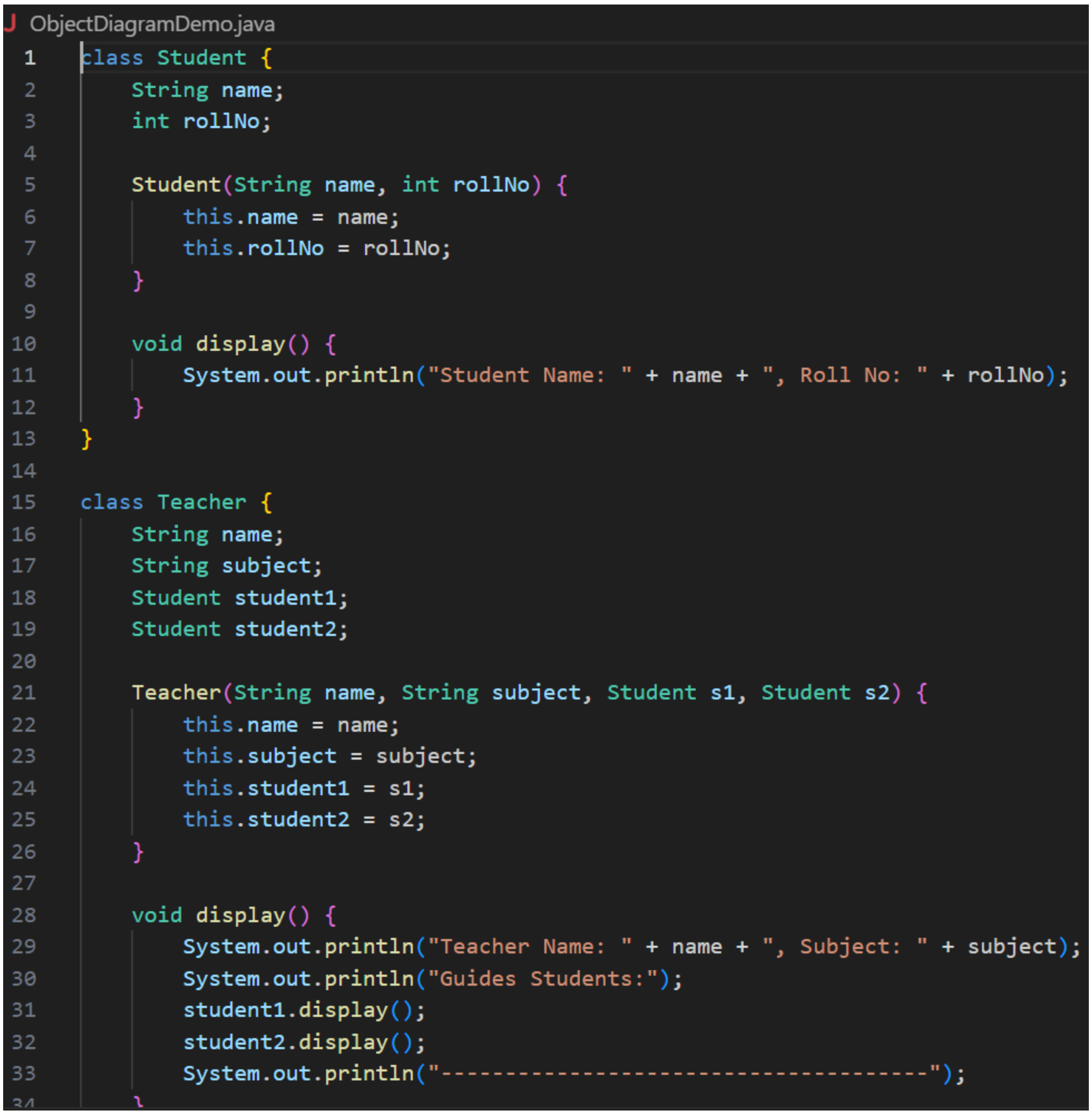
**Problem Statement:**

Draw an Object Diagram representing real instances of classes Student and Teacher where each teacher guides two students.

**Hints:**

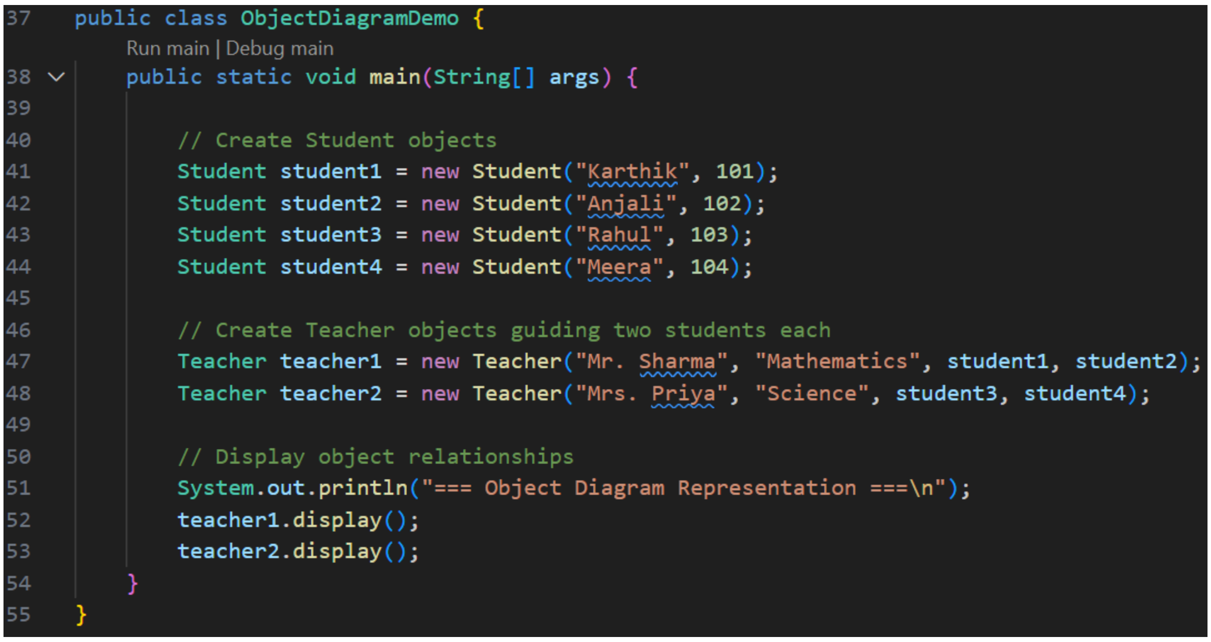
- Show object names (e.g., teacher1:Teacher, student1:Student).
- Indicate object links (runtime relationships).
- Keep attribute values simple (e.g., name = "Karthik").

## ObjectDiagramDemo.java



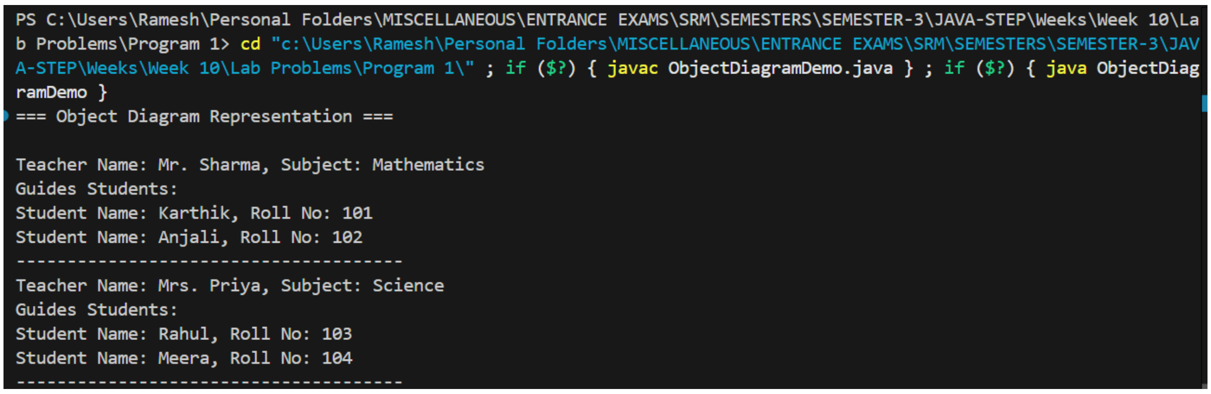
ObjectDiagramDemo.java

```java
class Student {
    String name;
    int rollNo;

    Student(String name, int rollNo) {
        this.name = name;
        this.rollNo = rollNo;
    }

    void display() {
        System.out.println("Student Name: " + name + ", Roll No: " + rollNo);
    }
}

class Teacher {
    String name;
    String subject;
    Student student1;
    Student student2;

    Teacher(String name, String subject, Student s1, Student s2) {
        this.name = name;
        this.subject = subject;
        this.student1 = s1;
        this.student2 = s2;
    }

    void display() {
        System.out.println("Teacher Name: " + name + ", Subject: " + subject);
        System.out.println("Guides Students:");
        student1.display();
        student2.display();
        System.out.println("-------------------------------------");
```



```java
public class ObjectDiagramDemo {
    Run main | Debug main
    public static void main(String[] args) {

        // Create Student objects
        Student student1 = new Student("Karthik", 101);
        Student student2 = new Student("Anjali", 102);
        Student student3 = new Student("Rahul", 103);
        Student student4 = new Student("Meera", 104);

        // Create Teacher objects guiding two students each
        Teacher teacher1 = new Teacher("Mr. Sharma", "Mathematics", student1, student2);
        Teacher teacher2 = new Teacher("Mrs. Priya", "Science", student3, student4);

        // Display object relationships
        System.out.println("=== Object Diagram Representation ===\n");
        teacher1.display();
        teacher2.display();
    }
}
```

**OUTPUT→**



```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 10\La
b Problems\Program 1> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAV
A-STEP\Weeks\Week 10\Lab Problems\Program 1\" ; if ($?) { javac ObjectDiagramDemo.java } ; if ($?) { java ObjectDiag
ramDemo }
=== Object Diagram Representation ===

Teacher Name: Mr. Sharma, Subject: Mathematics
Guides Students:
Student Name: Karthik, Roll No: 101
Student Name: Anjali, Roll No: 102
--------------------------------------
Teacher Name: Mrs. Priya, Subject: Science
Guides Students:
Student Name: Rahul, Roll No: 103
Student Name: Meera, Roll No: 104
--------------------------------------
```

**QNO3→**

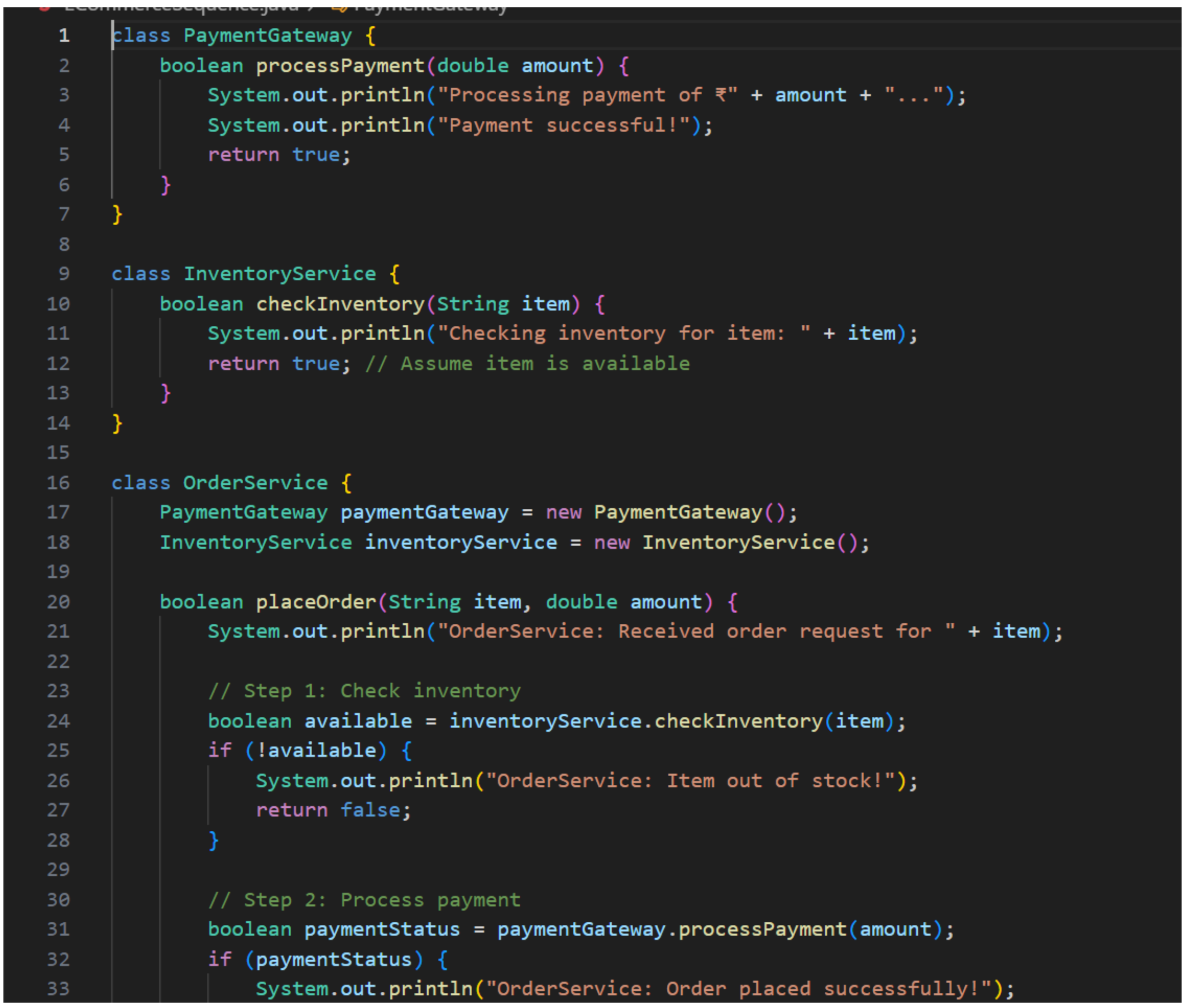
**Problem Statement:**

Draw a Sequence Diagram showing the process of placing an order on an e-commerce website. Include Customer, OrderService, PaymentGateway, and InventoryService.

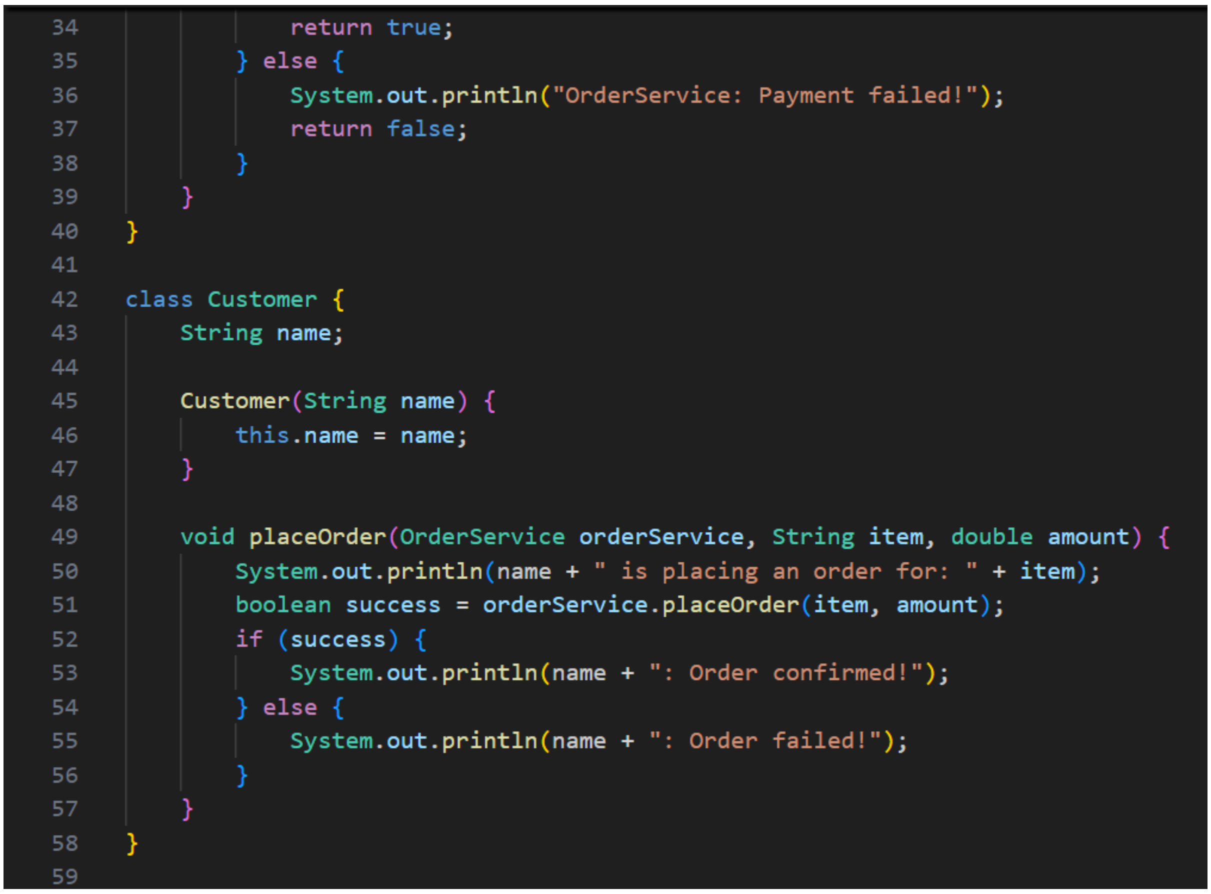
**Hints:**

- Show the flow of method calls from customer to services.
- Include return arrows to indicate responses.
- Use activation boxes for ongoing operations.

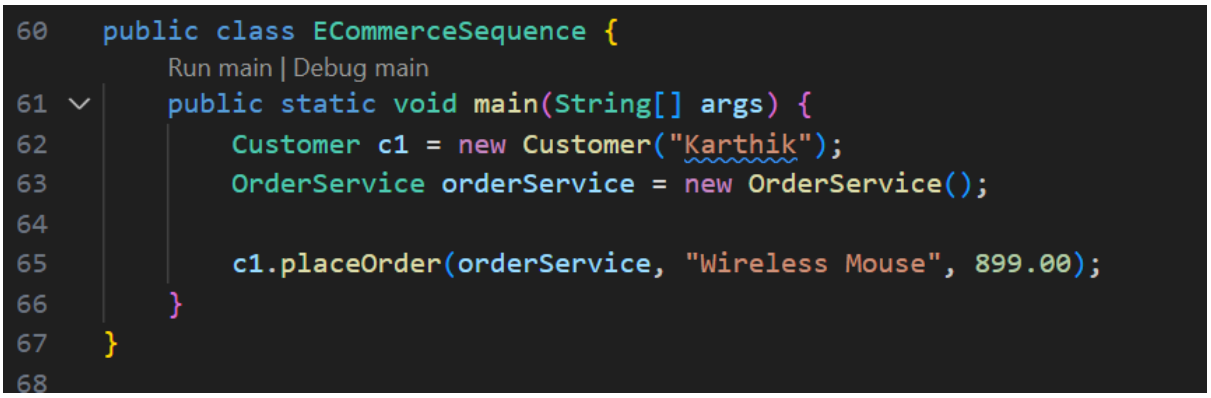
ECommerceSequence.java



```java
class PaymentGateway {
    boolean processPayment(double amount) {
        System.out.println("Processing payment of ₹" + amount + "...");
        System.out.println("Payment successful!");
        return true;
    }
}

class InventoryService {
    boolean checkInventory(String item) {
        System.out.println("Checking inventory for item: " + item);
        return true; // Assume item is available
    }
}

class OrderService {
    PaymentGateway paymentGateway = new PaymentGateway();
    InventoryService inventoryService = new InventoryService();

    boolean placeOrder(String item, double amount) {
        System.out.println("OrderService: Received order request for " + item);

        // Step 1: Check inventory
        boolean available = inventoryService.checkInventory(item);
        if (!available) {
            System.out.println("OrderService: Item out of stock!");
            return false;
        }

        // Step 2: Process payment
        boolean paymentStatus = paymentGateway.processPayment(amount);
        if (paymentStatus) {
            System.out.println("OrderService: Order placed successfully!");
```

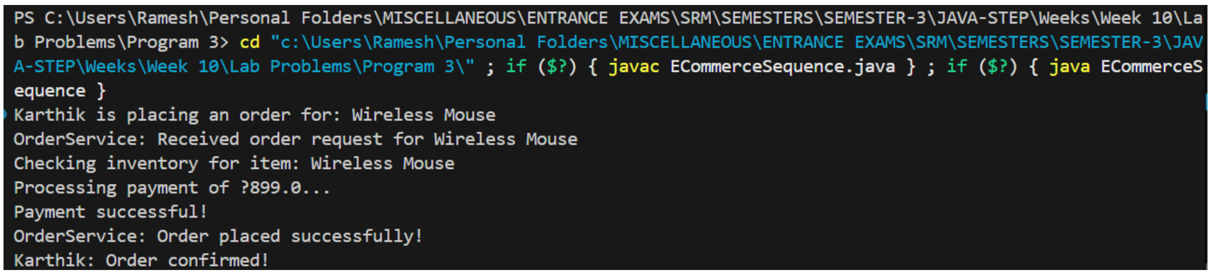


```java
            return true;
        } else {
            System.out.println("OrderService: Payment failed!");
            return false;
        }
    }
}

class Customer {
    String name;

    Customer(String name) {
        this.name = name;
    }

    void placeOrder(OrderService orderService, String item, double amount) {
        System.out.println(name + " is placing an order for: " + item);
        boolean success = orderService.placeOrder(item, amount);
        if (success) {
            System.out.println(name + ": Order confirmed!");
        } else {
            System.out.println(name + ": Order failed!");
        }
    }
}

```



```java
public class ECommerceSequence {
    public static void main(String[] args) {
        Customer c1 = new Customer("Karthik");
        OrderService orderService = new OrderService();

        c1.placeOrder(orderService, "Wireless Mouse", 899.00);
    }
}

```

**OUTPUT→**



```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 10\La
b Problems\Program 3> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAV
A-STEP\Weeks\Week 10\Lab Problems\Program 3\" ; if ($?) { javac ECommerceSequence.java } ; if ($?) { java ECommerceS
equence }
Karthik is placing an order for: Wireless Mouse
OrderService: Received order request for Wireless Mouse
Checking inventory for item: Wireless Mouse
Processing payment of ?899.0...
Payment successful!
OrderService: Order placed successfully!
Karthik: Order confirmed!
```

**QNO4→**

**Problem Statement:**

Draw a Use Case Diagram showing the main user actions in an ATM system such as Withdraw Money, Check Balance, and Deposit Money.

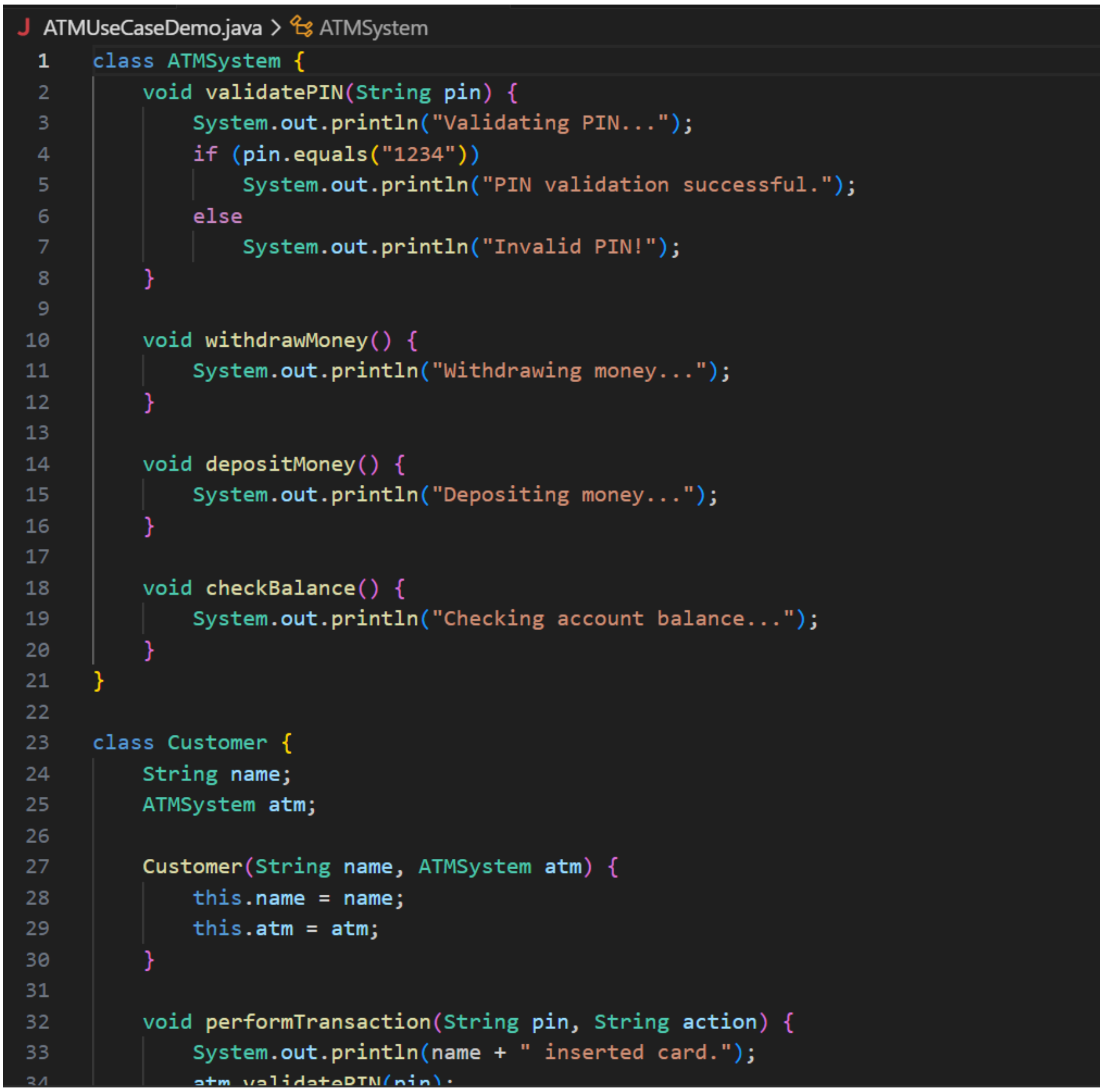
**Hints:**

● Use actor symbols for users.

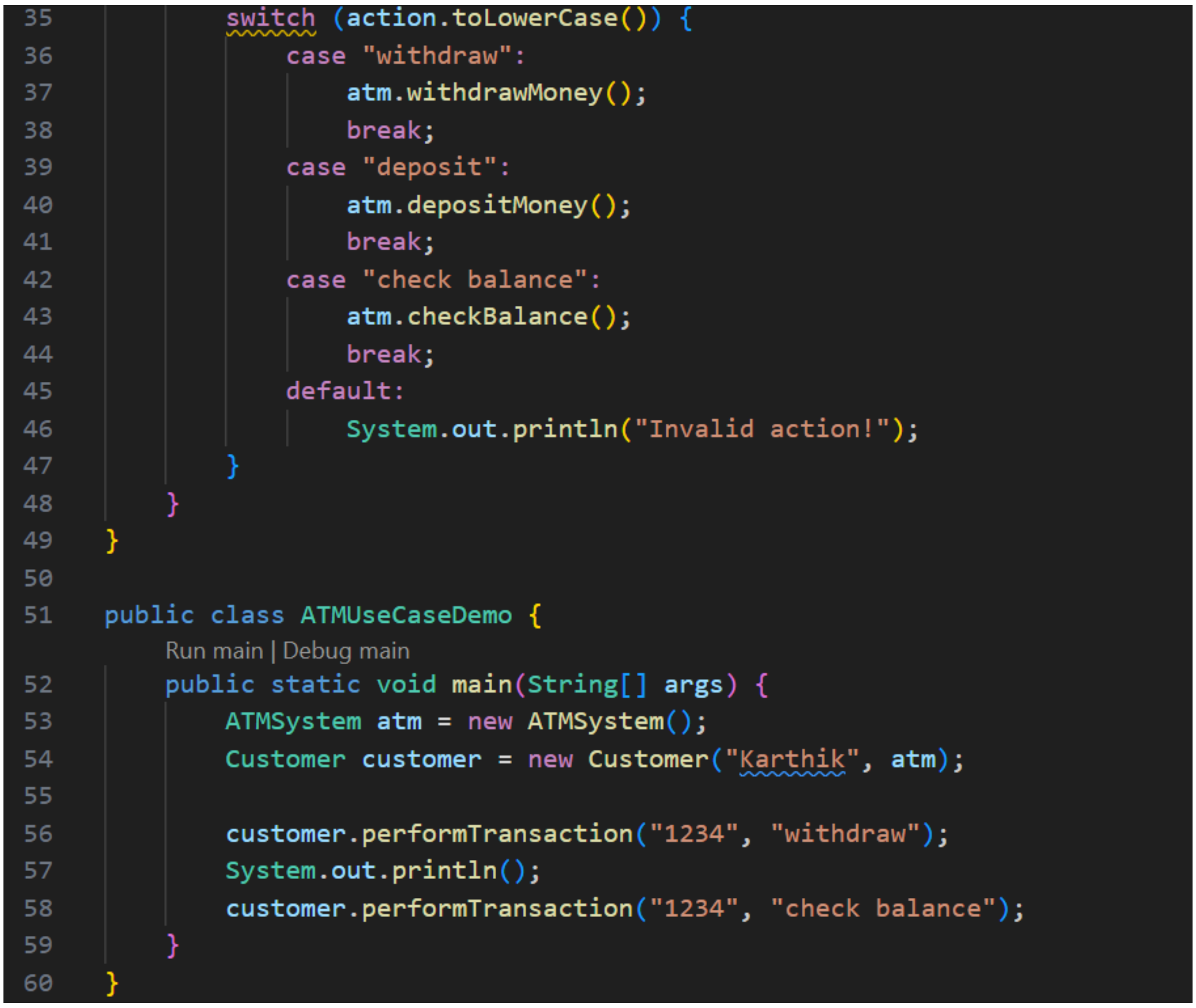
● Connect actors to use cases with lines.

● Optionally use include or extend relationships.

ATMUseCaseDemo.java

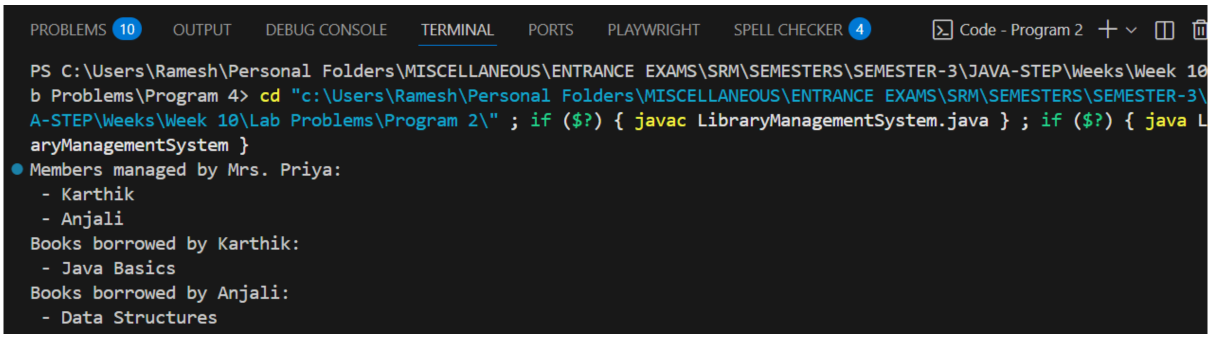


```java
class ATMSystem {
    void validatePIN(String pin) {
        System.out.println("Validating PIN...");
        if (pin.equals("1234"))
            System.out.println("PIN validation successful.");
        else
            System.out.println("Invalid PIN!");
    }

    void withdrawMoney() {
        System.out.println("Withdrawing money...");
    }

    void depositMoney() {
        System.out.println("Depositing money...");
    }

    void checkBalance() {
        System.out.println("Checking account balance...");
    }
}

class Customer {
    String name;
    ATMSystem atm;

    Customer(String name, ATMSystem atm) {
        this.name = name;
        this.atm = atm;
    }

    void performTransaction(String pin, String action) {
        System.out.println(name + " inserted card.");
        atm.validatePIN(pin);
```



```java
        switch (action.toLowerCase()) {
            case "withdraw":
                atm.withdrawMoney();
                break;
            case "deposit":
                atm.depositMoney();
                break;
            case "check balance":
                atm.checkBalance();
                break;
            default:
                System.out.println("Invalid action!");
        }
    }
}

public class ATMUseCaseDemo {
    Run main | Debug main
    public static void main(String[] args) {
        ATMSystem atm = new ATMSystem();
        Customer customer = new Customer("Karthik", atm);

        customer.performTransaction("1234", "withdraw");
        System.out.println();
        customer.performTransaction("1234", "check balance");
    }
}
```

**OUTPUT→**



```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 10
b Problems\Program 4> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\
A-STEP\Weeks\Week 10\Lab Problems\Program 2\" ; if ($?) { javac LibraryManagementSystem.java } ; if ($?) { java L
aryManagementSystem }
Members managed by Mrs. Priya:
 - Karthik
 - Anjali
Books borrowed by Karthik:
 - Java Basics
Books borrowed by Anjali:
 - Data Structures
```