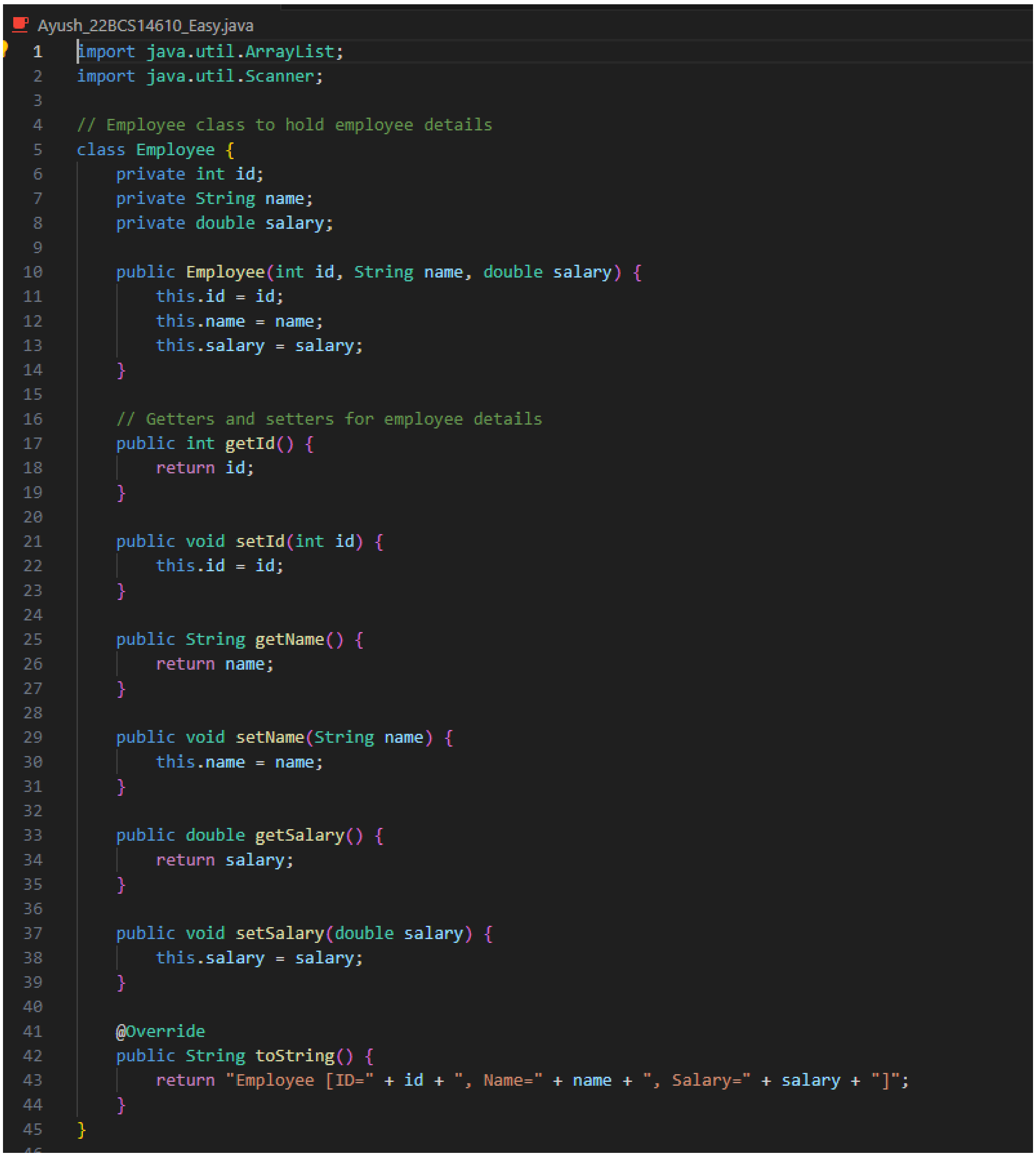
EXP-04

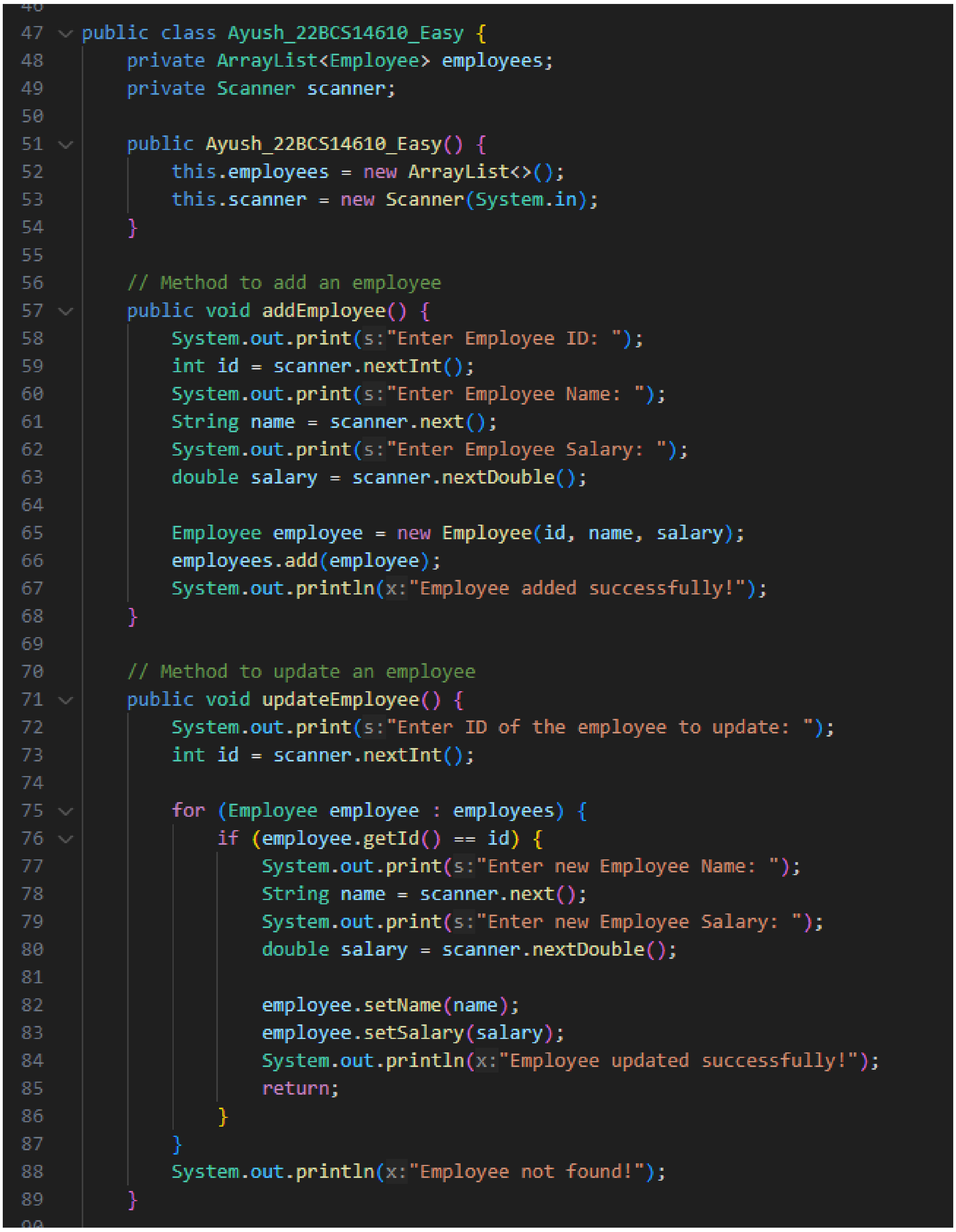
1. Write a Java program to implement an ArrayList that stores employee details (ID, Name, and Salary). Allow users to add, update, remove, and search employees.(Easy)

Code:

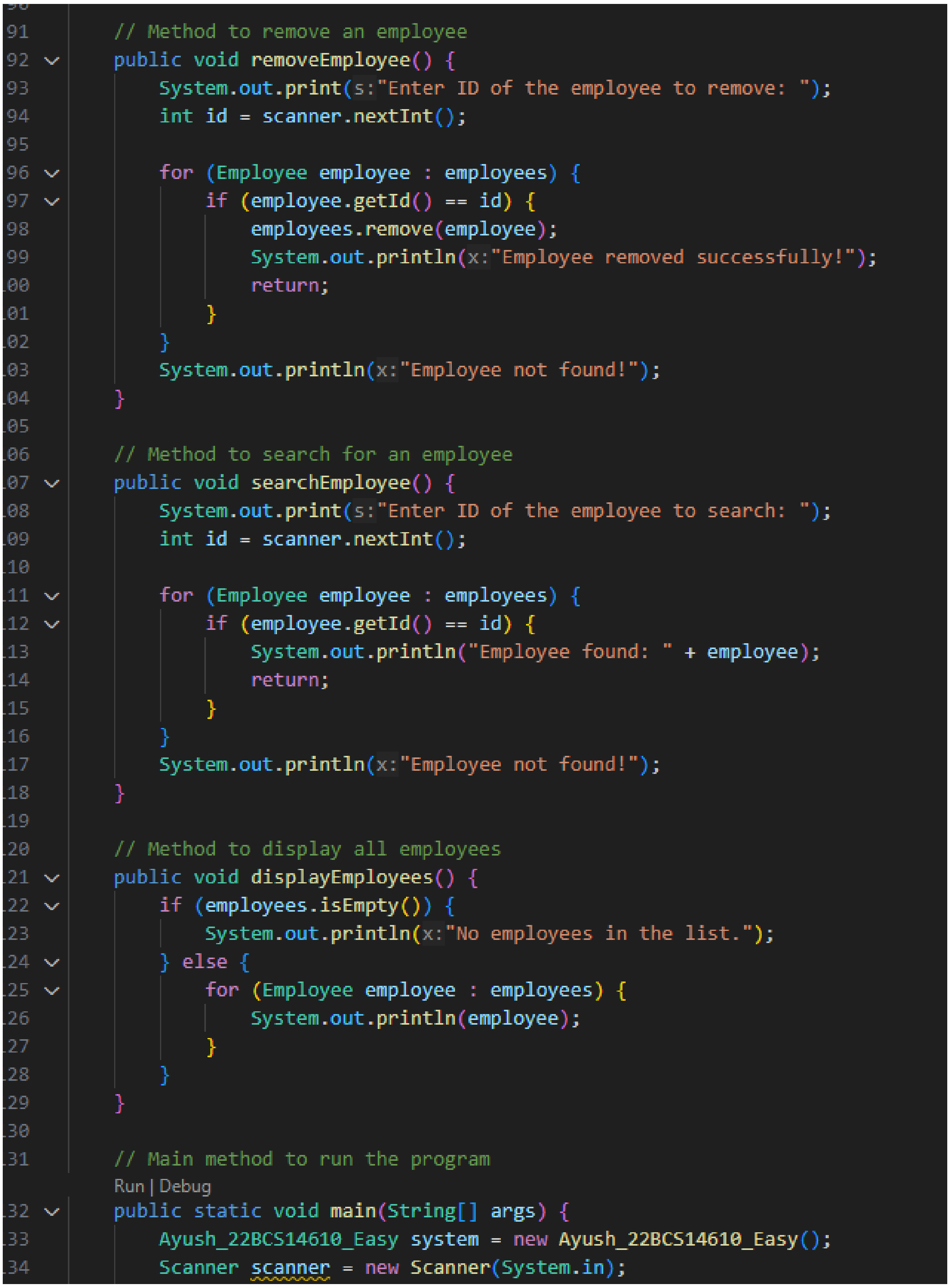


Ayush_22BCS14610_Easy.java

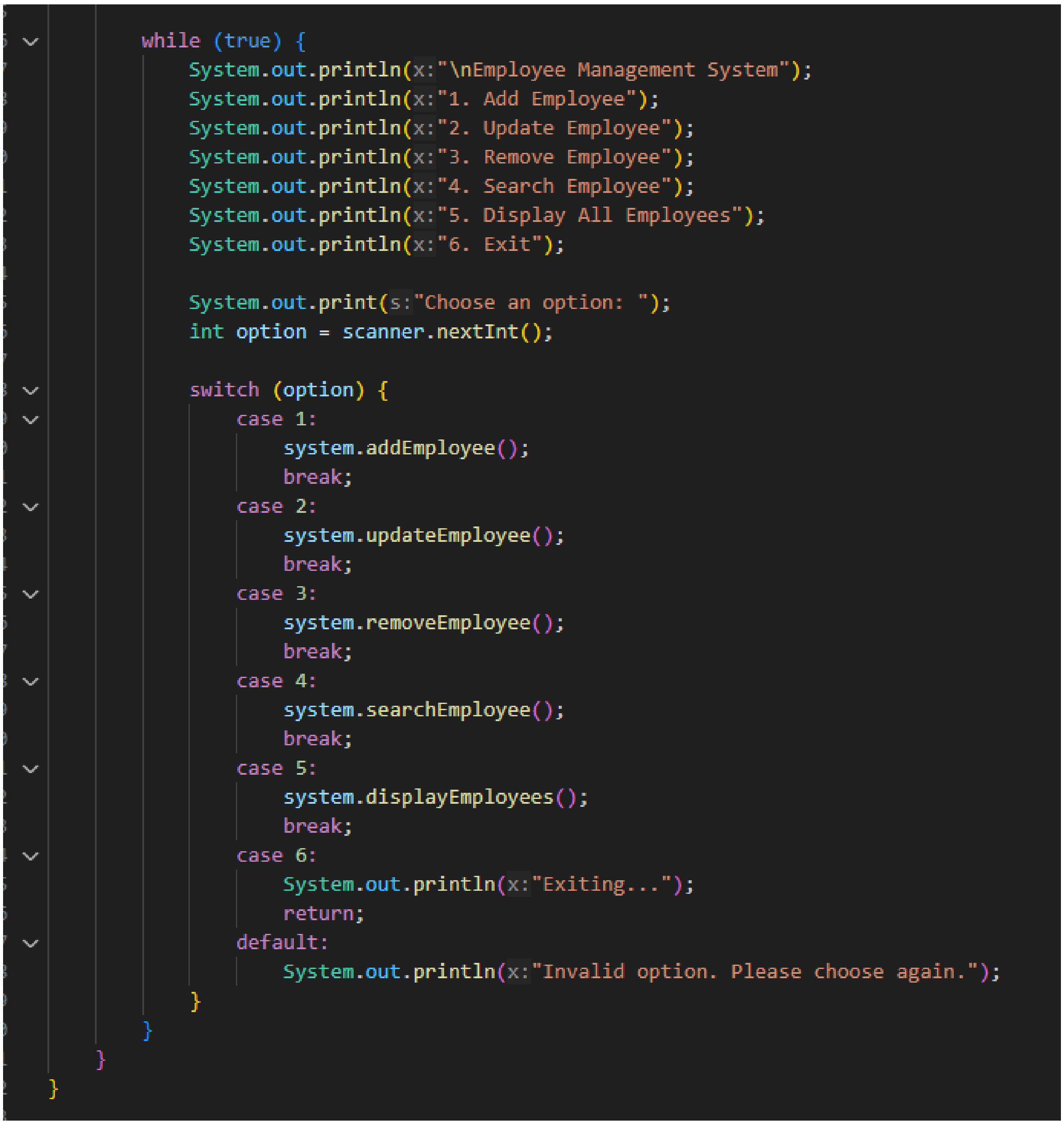
```java
import java.util.ArrayList;
import java.util.Scanner;

// Employee class to hold employee details
class Employee {
    private int id;
    private String name;
    private double salary;

    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    // Getters and setters for employee details
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee [ID=" + id + ", Name=" + name + ", Salary=" + salary + "]";
    }
}
```



```java
public class Ayush_22BCS14610_Easy {
    private ArrayList<Employee> employees;
    private Scanner scanner;

    public Ayush_22BCS14610_Easy() {
        this.employees = new ArrayList<>();
        this.scanner = new Scanner(System.in);
    }

    // Method to add an employee
    public void addEmployee() {
        System.out.print(s:"Enter Employee ID: ");
        int id = scanner.nextInt();
        System.out.print(s:"Enter Employee Name: ");
        String name = scanner.next();
        System.out.print(s:"Enter Employee Salary: ");
        double salary = scanner.nextDouble();

        Employee employee = new Employee(id, name, salary);
        employees.add(employee);
        System.out.println(x:"Employee added successfully!");
    }

    // Method to update an employee
    public void updateEmployee() {
        System.out.print(s:"Enter ID of the employee to update: ");
        int id = scanner.nextInt();

        for (Employee employee : employees) {
            if (employee.getId() == id) {
                System.out.print(s:"Enter new Employee Name: ");
                String name = scanner.next();
                System.out.print(s:"Enter new Employee Salary: ");
                double salary = scanner.nextDouble();

                employee.setName(name);
                employee.setSalary(salary);
                System.out.println(x:"Employee updated successfully!");
                return;
            }
        }
        System.out.println(x:"Employee not found!");
    }
```

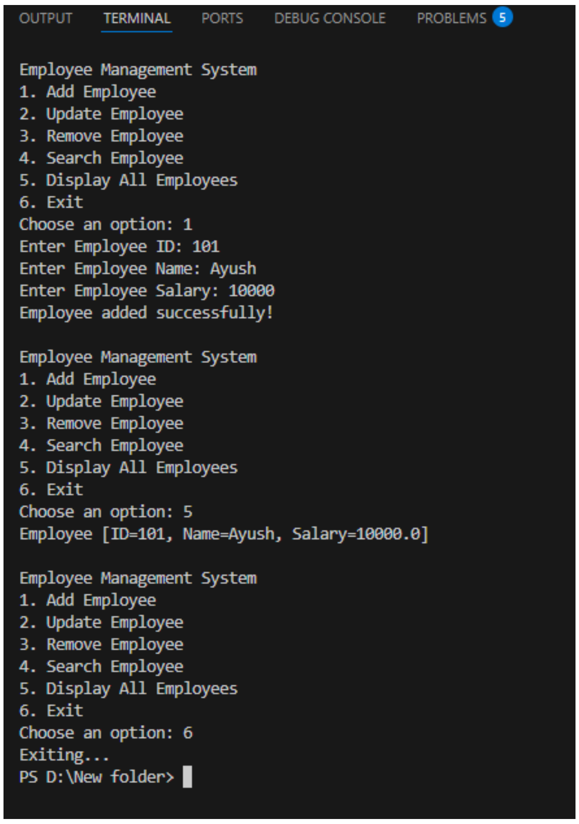


```java
    // Method to remove an employee
    public void removeEmployee() {
        System.out.print(s:"Enter ID of the employee to remove: ");
        int id = scanner.nextInt();

        for (Employee employee : employees) {
            if (employee.getId() == id) {
                employees.remove(employee);
                System.out.println(x:"Employee removed successfully!");
                return;
            }
        }
        System.out.println(x:"Employee not found!");
    }

    // Method to search for an employee
    public void searchEmployee() {
        System.out.print(s:"Enter ID of the employee to search: ");
        int id = scanner.nextInt();

        for (Employee employee : employees) {
            if (employee.getId() == id) {
                System.out.println("Employee found: " + employee);
                return;
            }
        }
        System.out.println(x:"Employee not found!");
    }

    // Method to display all employees
    public void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println(x:"No employees in the list.");
        } else {
            for (Employee employee : employees) {
                System.out.println(employee);
            }
        }
    }

    // Main method to run the program
    Run | Debug
    public static void main(String[] args) {
        Ayush_22BCS14610_Easy system = new Ayush_22BCS14610_Easy();
        Scanner scanner = new Scanner(System.in);
```



```
        while (true) {
            System.out.println(x:"\nEmployee Management System");
            System.out.println(x:"1. Add Employee");
            System.out.println(x:"2. Update Employee");
            System.out.println(x:"3. Remove Employee");
            System.out.println(x:"4. Search Employee");
            System.out.println(x:"5. Display All Employees");
            System.out.println(x:"6. Exit");

            System.out.print(s:"Choose an option: ");
            int option = scanner.nextInt();

            switch (option) {
                case 1:
                    system.addEmployee();
                    break;
                case 2:
                    system.updateEmployee();
                    break;
                case 3:
                    system.removeEmployee();
                    break;
                case 4:
                    system.searchEmployee();
                    break;
                case 5:
                    system.displayEmployees();
                    break;
                case 6:
                    System.out.println(x:"Exiting...");
                    return;
                default:
                    System.out.println(x:"Invalid option. Please choose again.");
            }
        }
    }
}
```

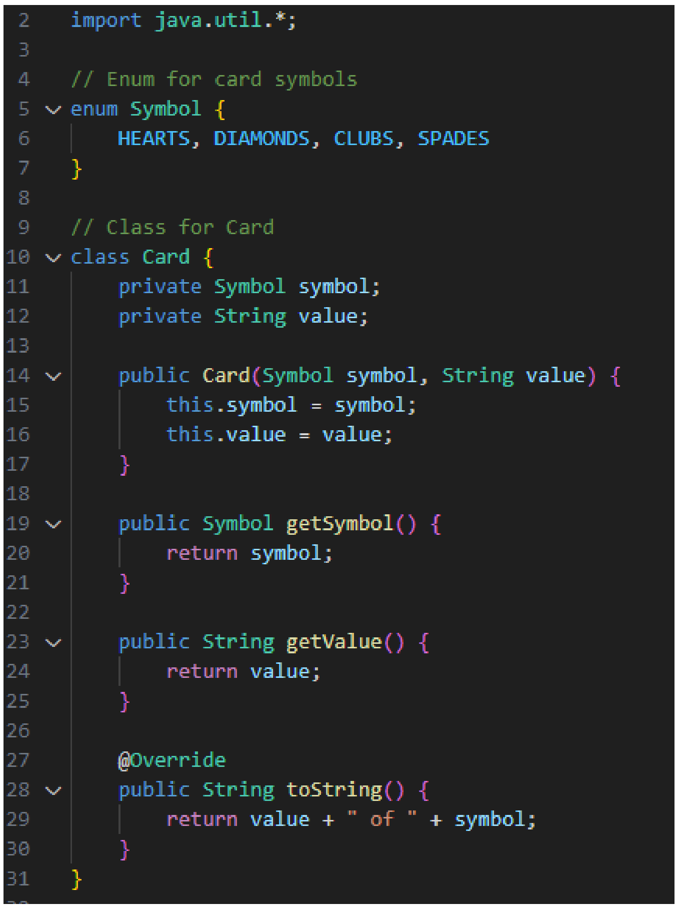
Output:



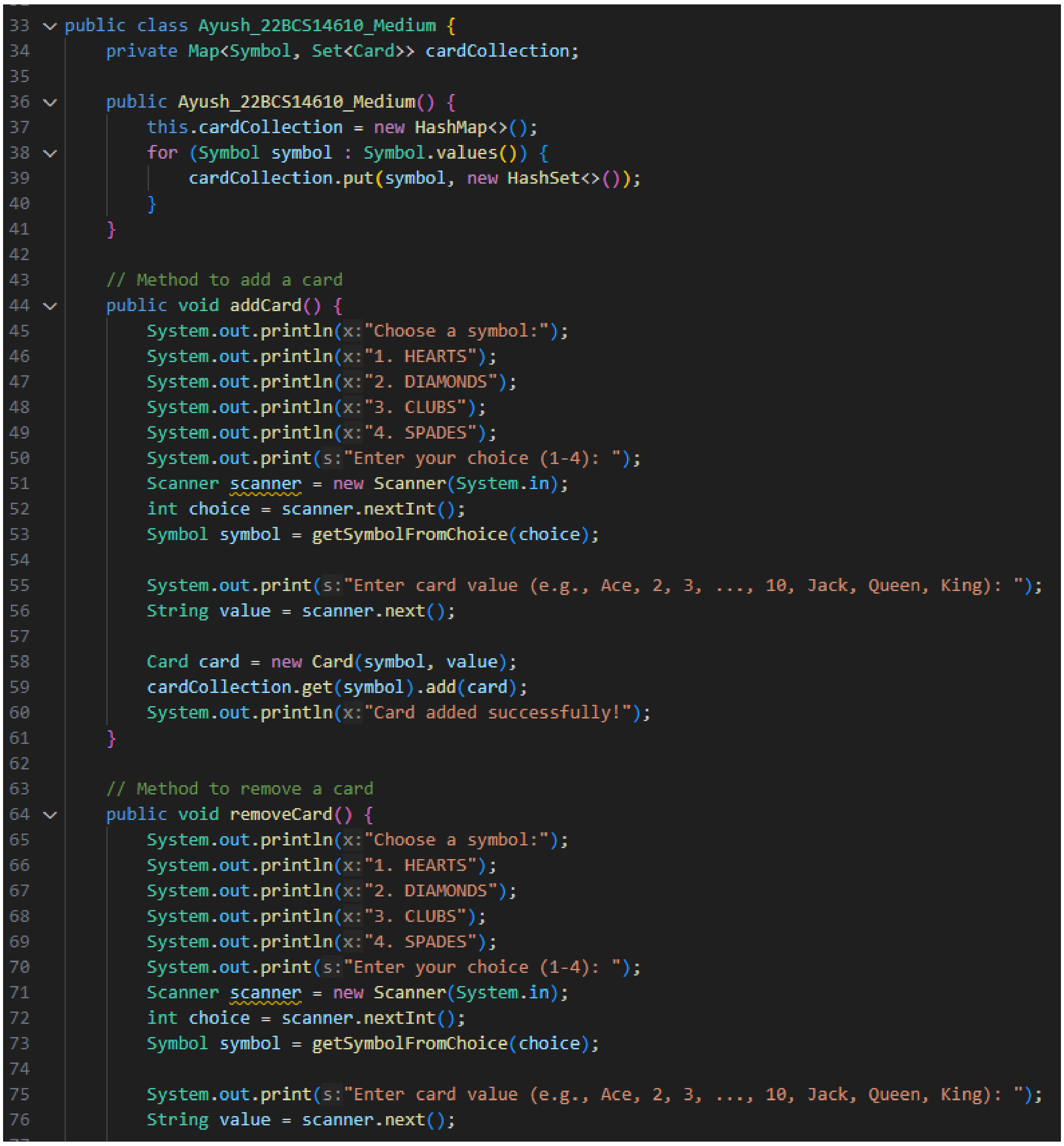
```
Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Choose an option: 1
Enter Employee ID: 101
Enter Employee Name: Ayush
Enter Employee Salary: 10000
Employee added successfully!

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Choose an option: 5
Employee [ID=101, Name=Ayush, Salary=10000.0]

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Choose an option: 6
Exiting...
PS D:\New folder>
```

2. Create a program to collect and store all the cards to assist the users in finding all the cards in a given symbol using Collection interface.(Medium)

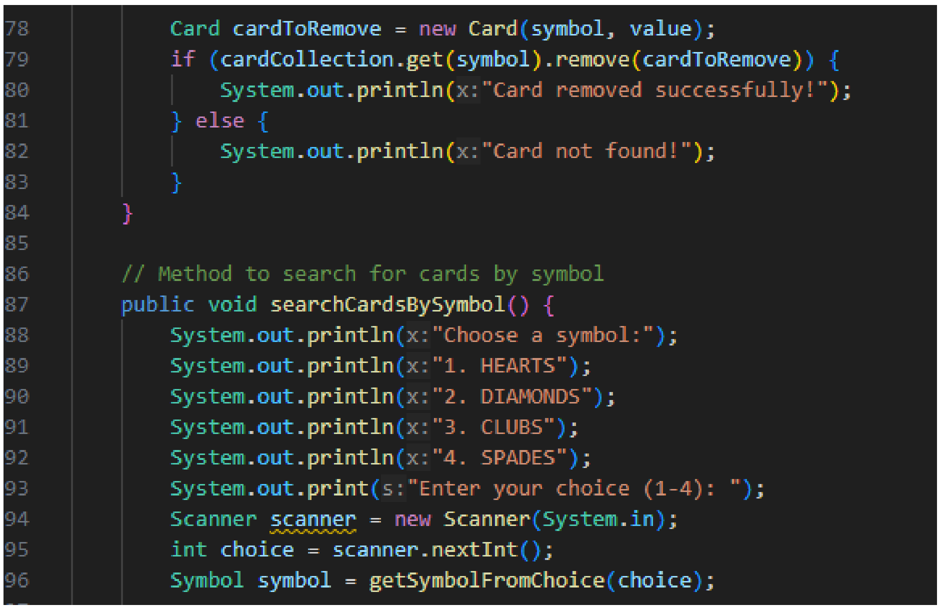
Code:



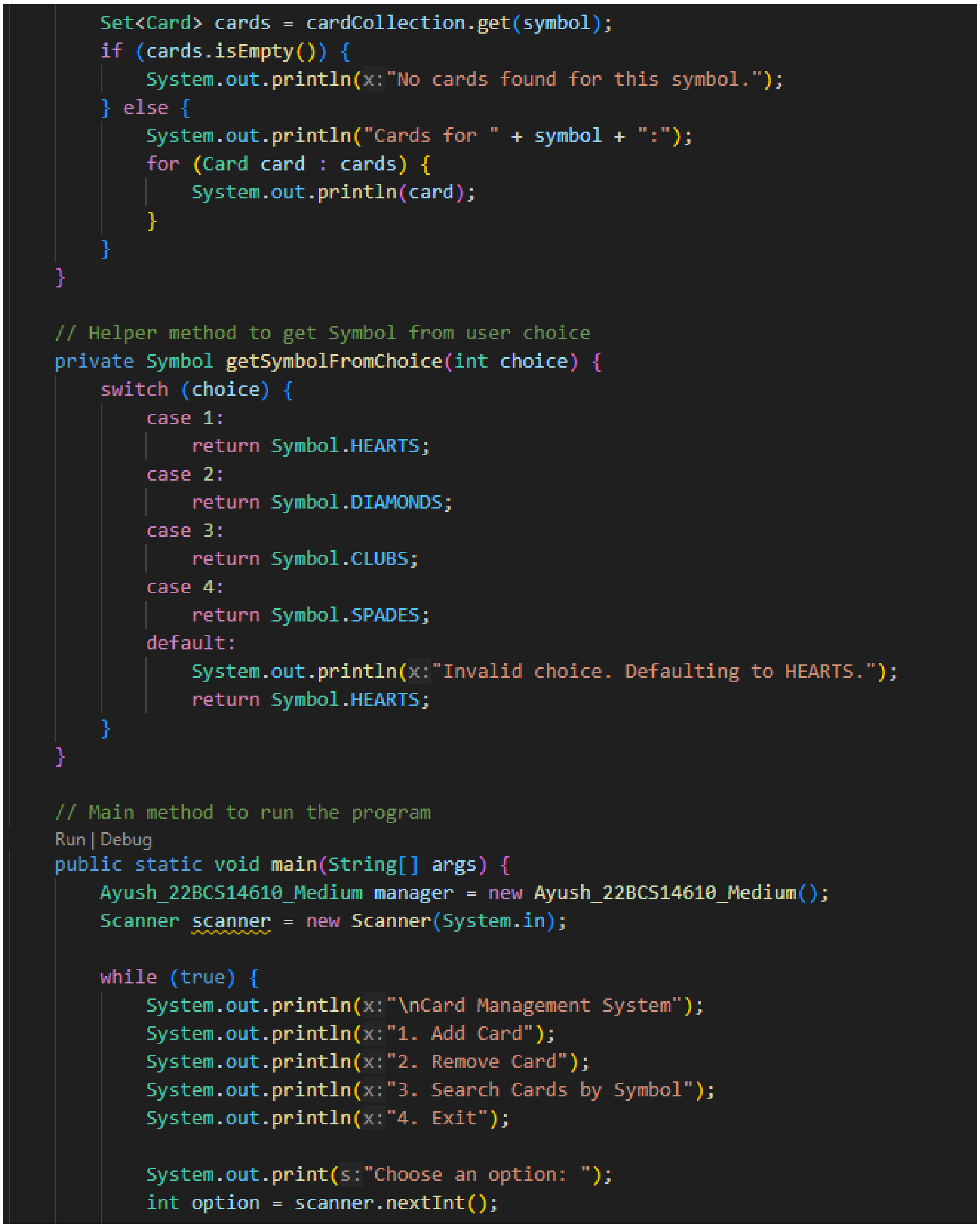
```java
import java.util.*;

// Enum for card symbols
enum Symbol {
    HEARTS, DIAMONDS, CLUBS, SPADES
}

// Class for Card
class Card {
    private Symbol symbol;
    private String value;

    public Card(Symbol symbol, String value) {
        this.symbol = symbol;
        this.value = value;
    }

    public Symbol getSymbol() {
        return symbol;
    }

    public String getValue() {
        return value;
    }

    @Override
    public String toString() {
        return value + " of " + symbol;
    }
}
```



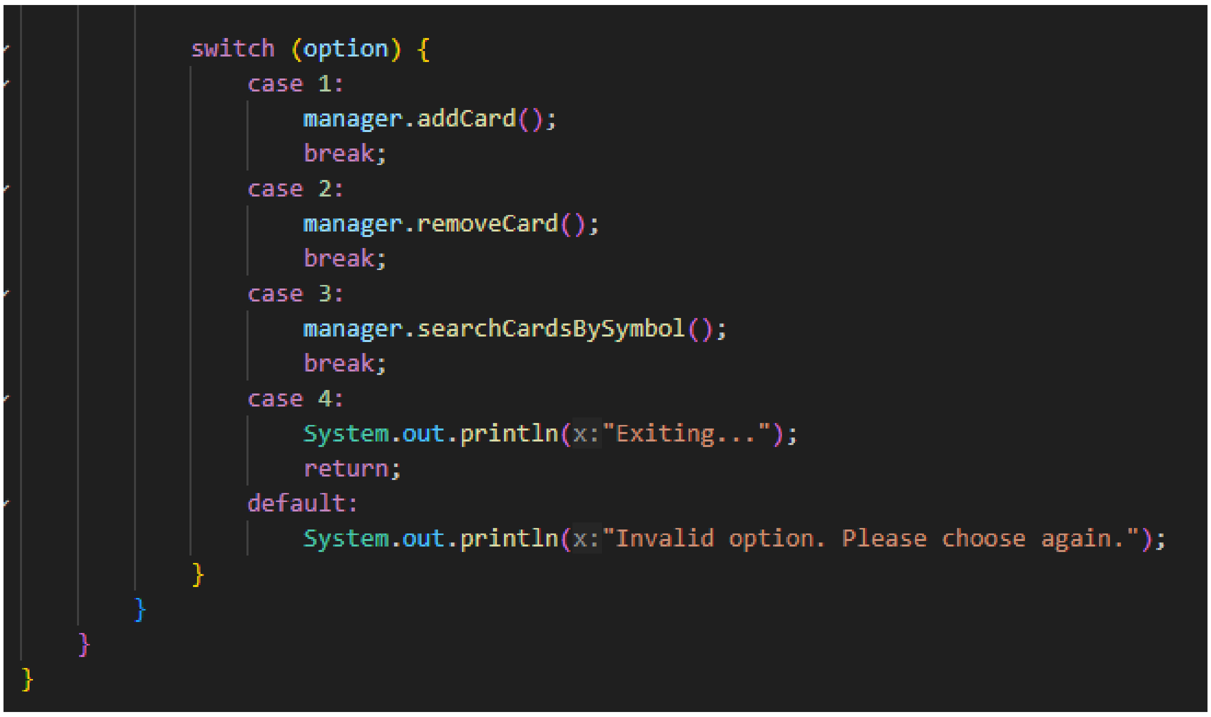
```java
public class Ayush_22BCS14610_Medium {
    private Map<Symbol, Set<Card>> cardCollection;

    public Ayush_22BCS14610_Medium() {
        this.cardCollection = new HashMap<>();
        for (Symbol symbol : Symbol.values()) {
            cardCollection.put(symbol, new HashSet<>());
        }
    }

    // Method to add a card
    public void addCard() {
        System.out.println(x:"Choose a symbol:");
        System.out.println(x:"1. HEARTS");
        System.out.println(x:"2. DIAMONDS");
        System.out.println(x:"3. CLUBS");
        System.out.println(x:"4. SPADES");
        System.out.print(s:"Enter your choice (1-4): ");
        Scanner scanner = new Scanner(System.in);
        int choice = scanner.nextInt();
        Symbol symbol = getSymbolFromChoice(choice);

        System.out.print(s:"Enter card value (e.g., Ace, 2, 3, ..., 10, Jack, Queen, King): ");
        String value = scanner.next();

        Card card = new Card(symbol, value);
        cardCollection.get(symbol).add(card);
        System.out.println(x:"Card added successfully!");
    }

    // Method to remove a card
    public void removeCard() {
        System.out.println(x:"Choose a symbol:");
        System.out.println(x:"1. HEARTS");
        System.out.println(x:"2. DIAMONDS");
        System.out.println(x:"3. CLUBS");
        System.out.println(x:"4. SPADES");
        System.out.print(s:"Enter your choice (1-4): ");
        Scanner scanner = new Scanner(System.in);
        int choice = scanner.nextInt();
        Symbol symbol = getSymbolFromChoice(choice);

        System.out.print(s:"Enter card value (e.g., Ace, 2, 3, ..., 10, Jack, Queen, King): ");
        String value = scanner.next();
```



```java
        Card cardToRemove = new Card(symbol, value);
        if (cardCollection.get(symbol).remove(cardToRemove)) {
            System.out.println(x:"Card removed successfully!");
        } else {
            System.out.println(x:"Card not found!");
        }
    }

    // Method to search for cards by symbol
    public void searchCardsBySymbol() {
        System.out.println(x:"Choose a symbol:");
        System.out.println(x:"1. HEARTS");
        System.out.println(x:"2. DIAMONDS");
        System.out.println(x:"3. CLUBS");
        System.out.println(x:"4. SPADES");
        System.out.print(s:"Enter your choice (1-4): ");
        Scanner scanner = new Scanner(System.in);
        int choice = scanner.nextInt();
        Symbol symbol = getSymbolFromChoice(choice);
```

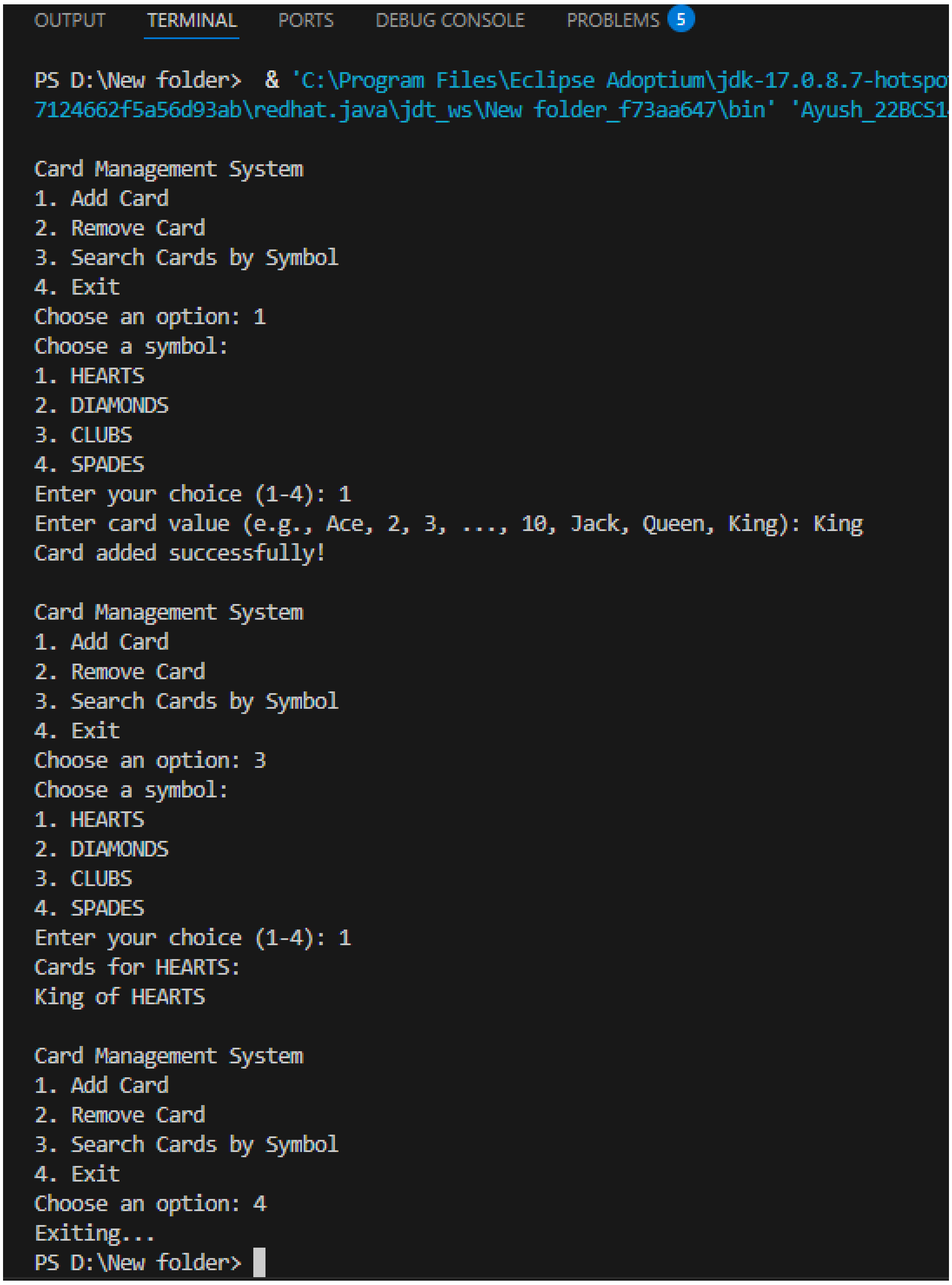


```
        Set<Card> cards = cardCollection.get(symbol);
        if (cards.isEmpty()) {
            System.out.println(x:"No cards found for this symbol.");
        } else {
            System.out.println("Cards for " + symbol + ":");
            for (Card card : cards) {
                System.out.println(card);
            }
        }
    }

    // Helper method to get Symbol from user choice
    private Symbol getSymbolFromChoice(int choice) {
        switch (choice) {
            case 1:
                return Symbol.HEARTS;
            case 2:
                return Symbol.DIAMONDS;
            case 3:
                return Symbol.CLUBS;
            case 4:
                return Symbol.SPADES;
            default:
                System.out.println(x:"Invalid choice. Defaulting to HEARTS.");
                return Symbol.HEARTS;
        }
    }

    // Main method to run the program
    Run | Debug
    public static void main(String[] args) {
        Ayush_22BCS14610_Medium manager = new Ayush_22BCS14610_Medium();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println(x:"\nCard Management System");
            System.out.println(x:"1. Add Card");
            System.out.println(x:"2. Remove Card");
            System.out.println(x:"3. Search Cards by Symbol");
            System.out.println(x:"4. Exit");

            System.out.print(s:"Choose an option: ");
            int option = scanner.nextInt();
```



```java
            switch (option) {
                case 1:
                    manager.addCard();
                    break;
                case 2:
                    manager.removeCard();
                    break;
                case 3:
                    manager.searchCardsBySymbol();
                    break;
                case 4:
                    System.out.println(x:"Exiting...");
                    return;
                default:
                    System.out.println(x:"Invalid option. Please choose again.");
            }
        }
    }
}
```

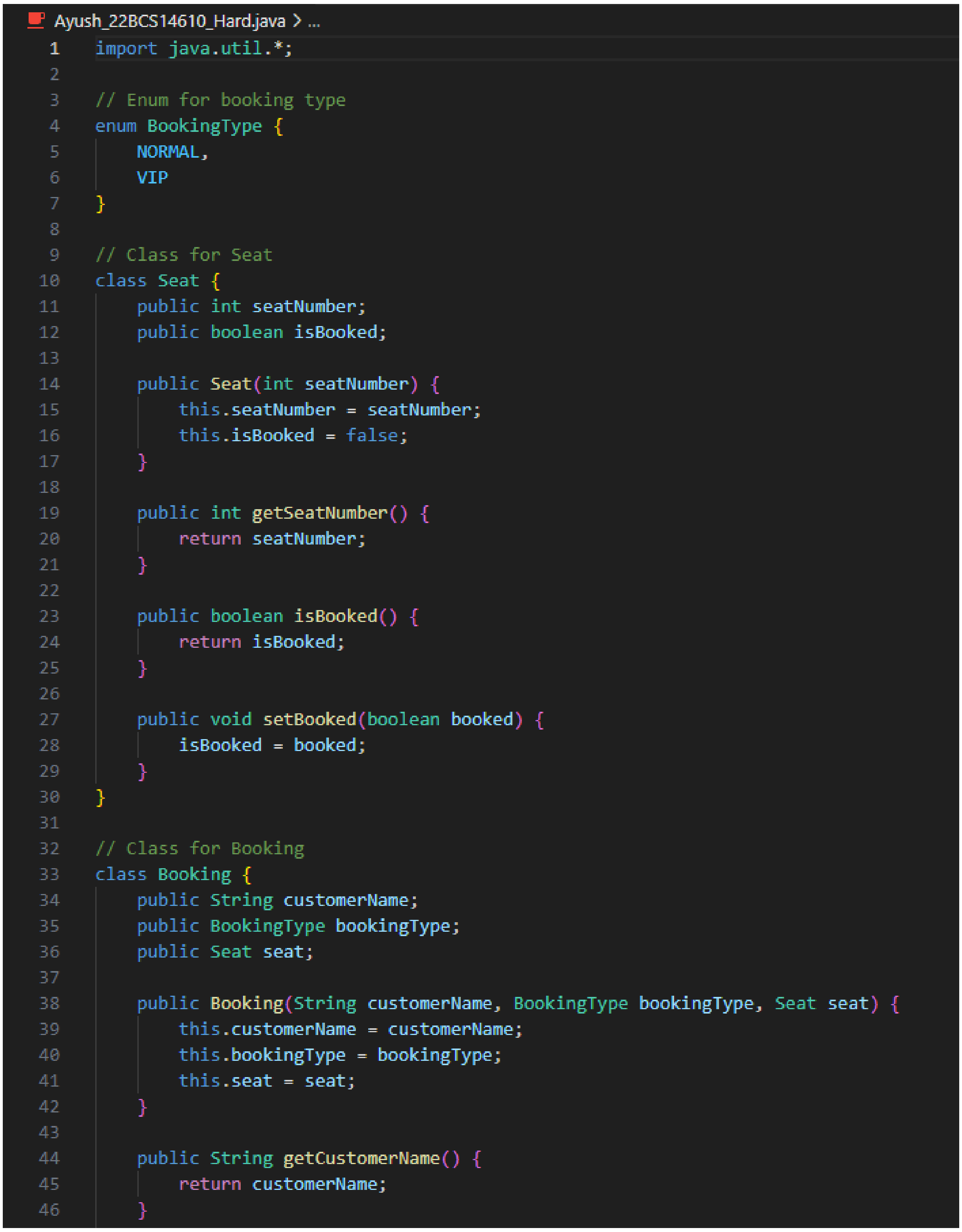
Output:



```
PS D:\New folder>  & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.8.7-hotspo
7124662f5a56d93ab\redhat.java\jdt_ws\New folder_f73aa647\bin' 'Ayush_22BCS1

Card Management System
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Exit
Choose an option: 1
Choose a symbol:
1. HEARTS
2. DIAMONDS
3. CLUBS
4. SPADES
Enter your choice (1-4): 1
Enter card value (e.g., Ace, 2, 3, ..., 10, Jack, Queen, King): King
Card added successfully!

Card Management System
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Exit
Choose an option: 3
Choose a symbol:
1. HEARTS
2. DIAMONDS
3. CLUBS
4. SPADES
Enter your choice (1-4): 1
Cards for HEARTS:
King of HEARTS

Card Management System
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Exit
Choose an option: 4
Exiting...
PS D:\New folder>
```

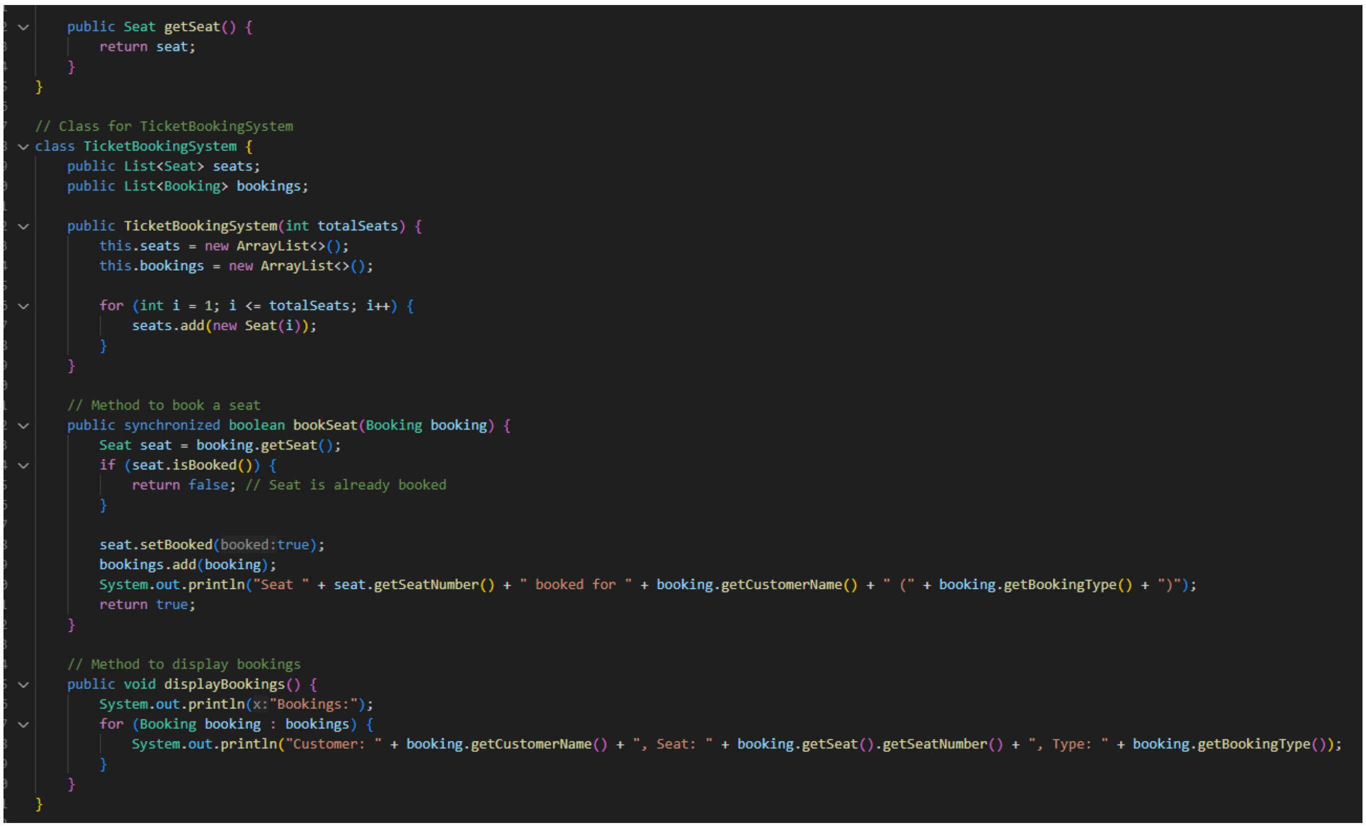
3. Develop a ticket booking system with synchronized threads to ensure no double booking of seats. Use thread priorities to simulate VIP bookings being processed first.(Hard)

Code:

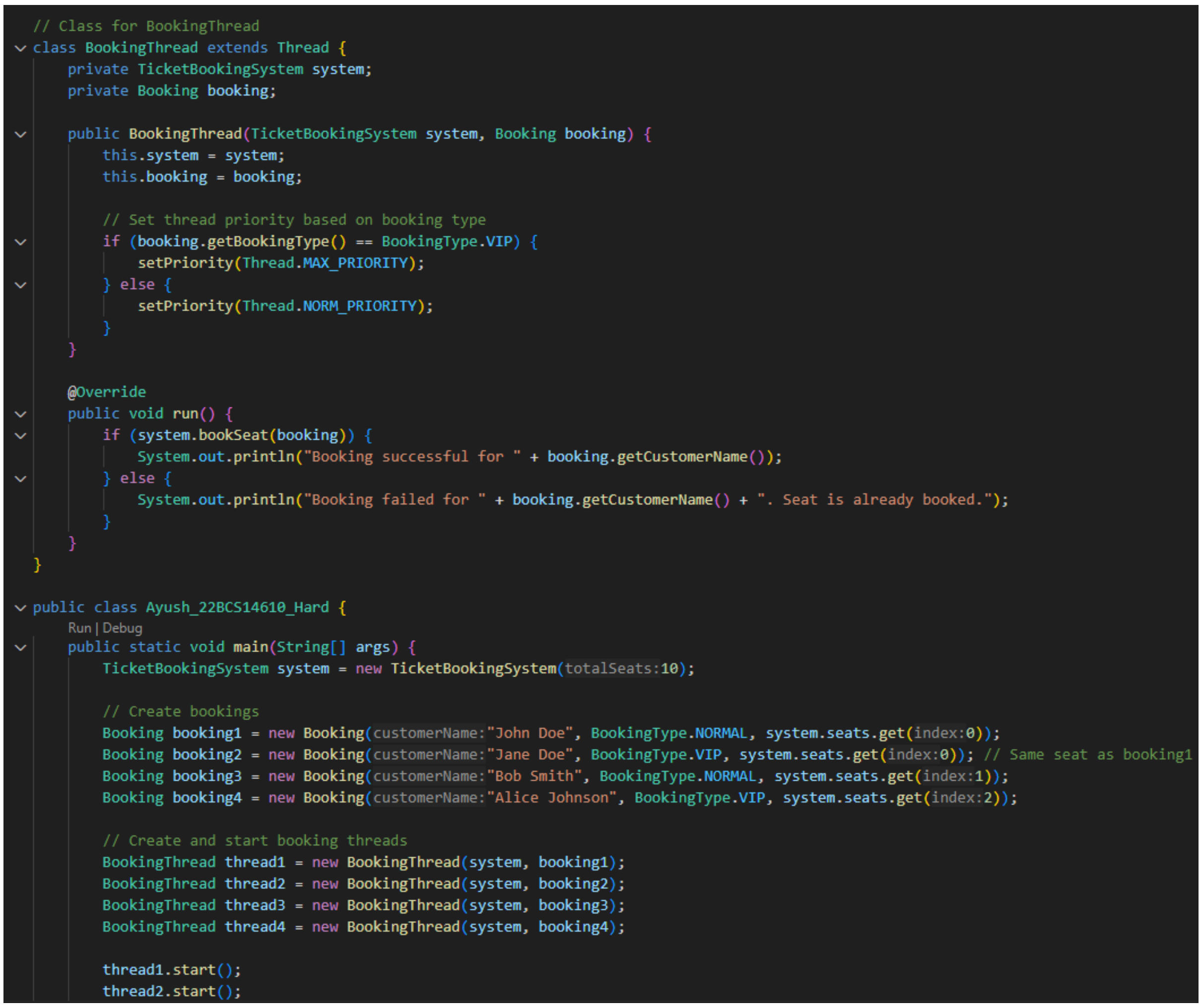


Ayush_22BCS14610_Hard.java > ...

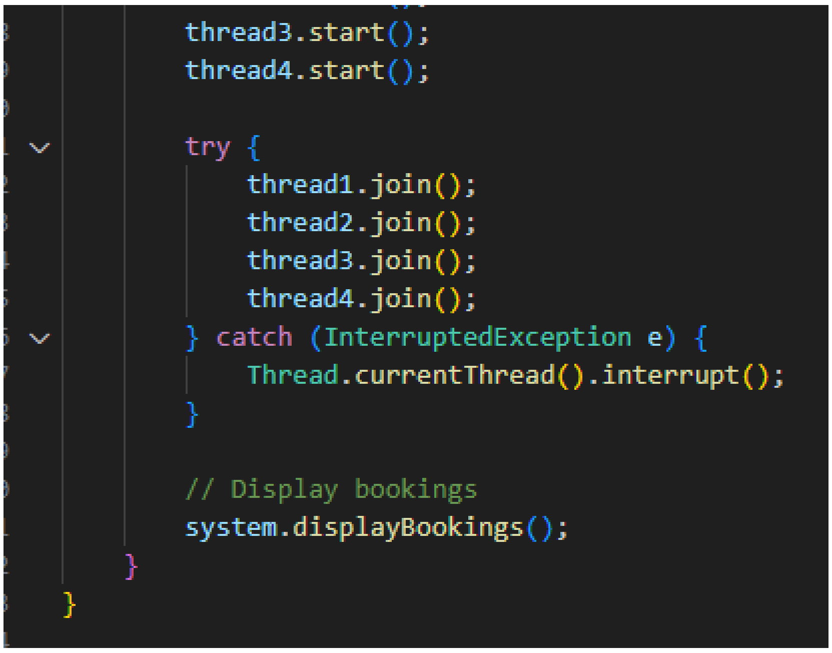
```java
import java.util.*;

// Enum for booking type
enum BookingType {
    NORMAL,
    VIP
}

// Class for Seat
class Seat {
    public int seatNumber;
    public boolean isBooked;

    public Seat(int seatNumber) {
        this.seatNumber = seatNumber;
        this.isBooked = false;
    }

    public int getSeatNumber() {
        return seatNumber;
    }

    public boolean isBooked() {
        return isBooked;
    }

    public void setBooked(boolean booked) {
        isBooked = booked;
    }
}

// Class for Booking
class Booking {
    public String customerName;
    public BookingType bookingType;
    public Seat seat;

    public Booking(String customerName, BookingType bookingType, Seat seat) {
        this.customerName = customerName;
        this.bookingType = bookingType;
        this.seat = seat;
    }

    public String getCustomerName() {
        return customerName;
    }
```



```java
    public Seat getSeat() {
        return seat;
    }
}

// Class for TicketBookingSystem
class TicketBookingSystem {
    public List<Seat> seats;
    public List<Booking> bookings;

    public TicketBookingSystem(int totalSeats) {
        this.seats = new ArrayList<>();
        this.bookings = new ArrayList<>();

        for (int i = 1; i <= totalSeats; i++) {
            seats.add(new Seat(i));
        }
    }

    // Method to book a seat
    public synchronized boolean bookSeat(Booking booking) {
        Seat seat = booking.getSeat();
        if (seat.isBooked()) {
            return false; // Seat is already booked
        }

        seat.setBooked(booked:true);
        bookings.add(booking);
        System.out.println("Seat " + seat.getSeatNumber() + " booked for " + booking.getCustomerName() + " (" + booking.getBookingType() + ")");
        return true;
    }

    // Method to display bookings
    public void displayBookings() {
        System.out.println(x:"Bookings:");
        for (Booking booking : bookings) {
            System.out.println("Customer: " + booking.getCustomerName() + ", Seat: " + booking.getSeat().getSeatNumber() + ", Type: " + booking.getBookingType());
        }
    }
}
```

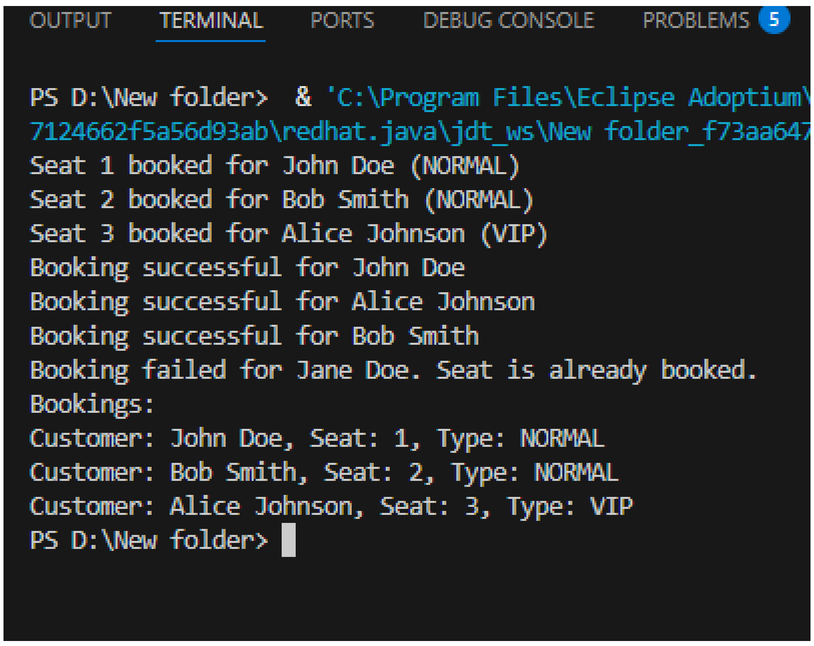


```java
// Class for BookingThread
class BookingThread extends Thread {
    private TicketBookingSystem system;
    private Booking booking;

    public BookingThread(TicketBookingSystem system, Booking booking) {
        this.system = system;
        this.booking = booking;

        // Set thread priority based on booking type
        if (booking.getBookingType() == BookingType.VIP) {
            setPriority(Thread.MAX_PRIORITY);
        } else {
            setPriority(Thread.NORM_PRIORITY);
        }
    }

    @Override
    public void run() {
        if (system.bookSeat(booking)) {
            System.out.println("Booking successful for " + booking.getCustomerName());
        } else {
            System.out.println("Booking failed for " + booking.getCustomerName() + ". Seat is already booked.");
        }
    }
}

public class Ayush_22BCS14610_Hard {
    Run | Debug
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem(totalSeats:10);

        // Create bookings
        Booking booking1 = new Booking(customerName:"John Doe", BookingType.NORMAL, system.seats.get(index:0));
        Booking booking2 = new Booking(customerName:"Jane Doe", BookingType.VIP, system.seats.get(index:0)); // Same seat as booking1
        Booking booking3 = new Booking(customerName:"Bob Smith", BookingType.NORMAL, system.seats.get(index:1));
        Booking booking4 = new Booking(customerName:"Alice Johnson", BookingType.VIP, system.seats.get(index:2));

        // Create and start booking threads
        BookingThread thread1 = new BookingThread(system, booking1);
        BookingThread thread2 = new BookingThread(system, booking2);
        BookingThread thread3 = new BookingThread(system, booking3);
        BookingThread thread4 = new BookingThread(system, booking4);

        thread1.start();
        thread2.start();
```



```java
        thread3.start();
        thread4.start();

        try {
            thread1.join();
            thread2.join();
            thread3.join();
            thread4.join();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        // Display bookings
        system.displayBookings();
    }
}
```

Output:



```
PS D:\New folder>  & 'C:\Program Files\Eclipse Adoptium\
7124662f5a56d93ab\redhat.java\jdt_ws\New folder_f73aa647
Seat 1 booked for John Doe (NORMAL)
Seat 2 booked for Bob Smith (NORMAL)
Seat 3 booked for Alice Johnson (VIP)
Booking successful for John Doe
Booking successful for Alice Johnson
Booking successful for Bob Smith
Booking failed for Jane Doe. Seat is already booked.
Bookings:
Customer: John Doe, Seat: 1, Type: NORMAL
Customer: Bob Smith, Seat: 2, Type: NORMAL
Customer: Alice Johnson, Seat: 3, Type: VIP
PS D:\New folder>
```