Easy Problem:

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

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;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

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

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

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Salary: " + salary;
    }
}

class EmployeeManagement {
    private static ArrayList<Employee> employees = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int choice;

        do {
            System.out.println("\nEmployee Management System");
            System.out.println("1. Add Employee");
            System.out.println("2. Update Employee");
            System.out.println("3. Remove Employee");
            System.out.println("4. Search Employee");
            System.out.println("5. Display All Employees");
            System.out.println("0. Exit");
            System.out.print("Choose an option: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline
```

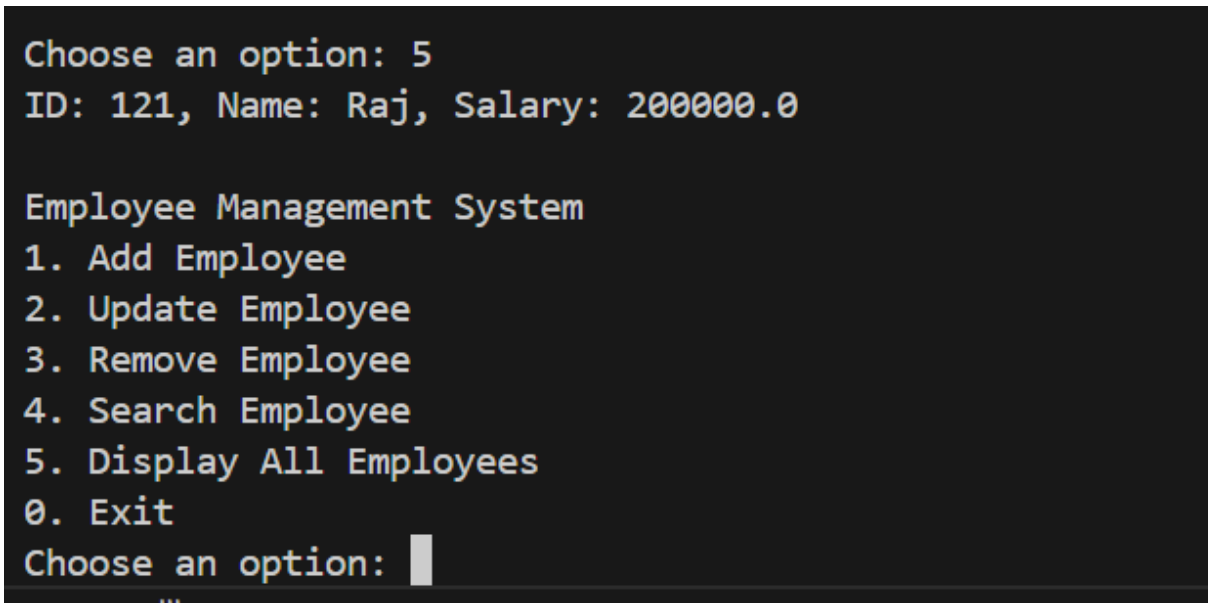
```
        switch (choice) {
            case 1:
                addEmployee();
                break;
            case 2:
                updateEmployee();
                break;
            case 3:
                removeEmployee();
                break;
            case 4:
                searchEmployee();
                break;
            case 5:
                displayEmployees();
                break;
            case 0:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid option! Please try again.");
        }
    } while (choice != 0);

    scanner.close();
}

private static void addEmployee() {
    System.out.print("Enter ID: ");
    int id = scanner.nextInt();
```

```
    scanner.nextLine(); // Consume newline
    System.out.print("Enter Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter Salary: ");
    double salary = scanner.nextDouble();

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

private static void updateEmployee() {
    System.out.print("Enter ID of the employee to update: ");
    int id = scanner.nextInt();

    for (Employee emp : employees) {
        if (emp.getId() == id) {
            System.out.print("Enter new Name: ");
            String newName = scanner.next();
            emp.setName(newName);

            System.out.print("Enter new Salary: ");
            double newSalary = scanner.nextDouble();
            emp.setSalary(newSalary);

            System.out.println("Employee updated successfully!");
            return;
        }
    }

    System.out.println("Employee not found!");
```

```
}

private static void removeEmployee() {
    System.out.print("Enter ID of the employee to remove: ");
    int id = scanner.nextInt();

    for (int i = 0; i < employees.size(); i++) {
        if (employees.get(i).getId() == id) {
            employees.remove(i);
            System.out.println("Employee removed successfully!");
            return;
        }
    }

    System.out.println("Employee not found!");
}

private static void searchEmployee() {
    System.out.print("Enter ID of the employee to search: ");
    int id = scanner.nextInt();

    for (Employee emp : employees) {
        if (emp.getId() == id) {
            System.out.println(emp);
            return;
        }
    }

    System.out.println("Employee not found!");
}
```

```
    private static void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees to display.");
            return;
        }

        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
```

Output:



Medium Problem:

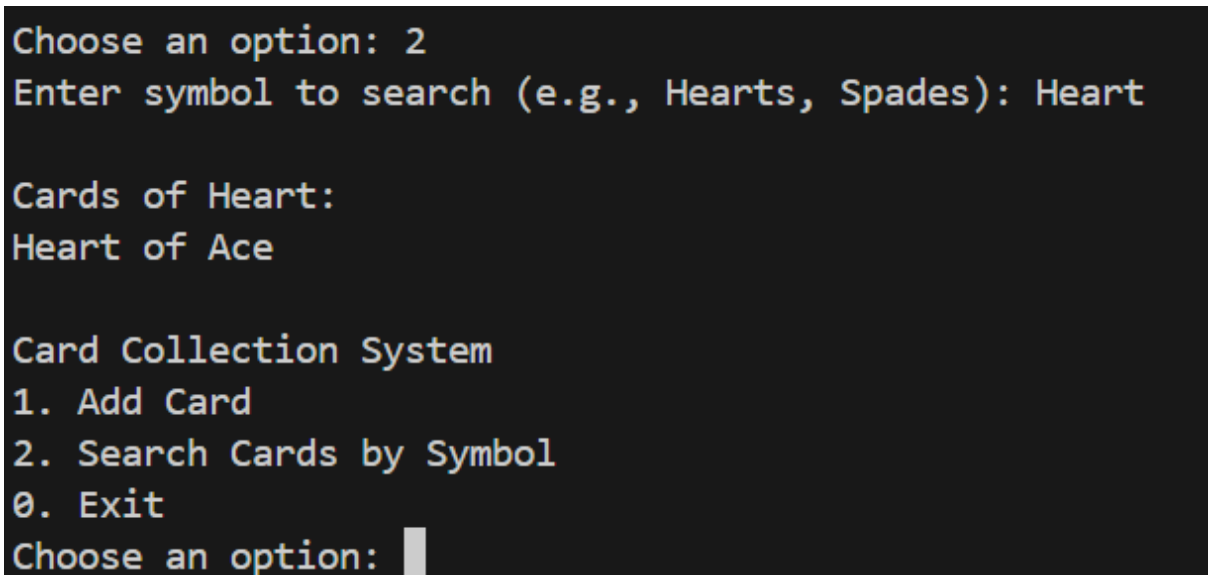
```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Scanner;
```

```
class Card {
    private String symbol;
    private String value;

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

    public String getSymbol() {
        return symbol;
    }

    public String getValue() {
        return value;
    }

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

class cardCollection {
    private static HashMap<String, List<Card>> cardMap = new HashMap<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
```

```
        System.out.println("\nCard Collection System");
        System.out.println("1. Add Card");
        System.out.println("2. Search Cards by Symbol");
        System.out.println("0. Exit");
        System.out.print("Choose an option: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1:
                addCard();
                break;
            case 2:
                searchCards();
                break;
            case 0:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid option! Please try again.");
        }
    }
}

private static void addCard() {
    System.out.print("Enter card symbol (e.g., Hearts, Diamonds): ");
    String symbol = scanner.nextLine().trim();
    System.out.print("Enter card value (e.g., Ace, 2-10, Jack, Queen, King): ");
    String value = scanner.nextLine().trim();
```

```
        // Initialize list if symbol doesn't exist
        if (!cardMap.containsKey(symbol)) {
            cardMap.put(symbol, new ArrayList<>());
        }

        cardMap.get(symbol).add(new Card(symbol, value));
        System.out.println("Card added successfully!");
    }

    private static void searchCards() {
        System.out.print("Enter symbol to search (e.g., Hearts, Spades): ");
        String symbol = scanner.nextLine().trim();

        if (cardMap.containsKey(symbol)) {
            System.out.println("\nCards of " + symbol + ":");
            for (Card card : cardMap.get(symbol)) {
                System.out.println(card);
            }
        } else {
            System.out.println("No cards found for this symbol!");
        }
    }
}
```

Output:



Hard Problem:

```
import java.util.concurrent.locks.ReentrantLock;


class TicketBookingSystem {
    private boolean[] seats;
    private final ReentrantLock lock = new ReentrantLock();
    private final int totalSeats;

    public TicketBookingSystem(int seats) {
        this.totalSeats = seats;
        this.seats = new boolean[seats];
    }

    public void bookSeat(int userId, boolean isVIP) {
        lock.lock();
        try {
            for (int i = 0; i < seats.length; i++) {
                if (!seats[i]) {
```

```
                seats[i] = true;
                System.out.println("User " + userId + (isVIP ? " (VIP)" : "")
                        + " booked seat " + (i + 1));
                return;
            }
        }
        System.out.println("No seats available for User " + userId
                + (isVIP ? " (VIP)" : ""));
    } finally {
        lock.unlock();
    }
  }
}

class BookingThread implements Runnable {
  private final TicketBookingSystem system;
  private final int userId;
  private final boolean isVIP;

  public BookingThread(TicketBookingSystem system, int userId, boolean isVIP) {
    this.system = system;
    this.userId = userId;
    this.isVIP = isVIP;
  }

  @Override
  public void run() {
    system.bookSeat(userId, isVIP);
  }
}
```

```
public class TicketBookingSystemMain {
    public static void main(String[] args) {
        final TicketBookingSystem system = new TicketBookingSystem(10);

        // Create VIP threads with higher priority
        for (int i = 0; i < 5; i++) {
            Thread thread = new Thread(new BookingThread(system, i + 1, true));
            thread.setPriority(Thread.MAX_PRIORITY);
            thread.start();
        }

        // Create regular threads with normal priority
        for (int i = 0; i < 5; i++) {
            Thread thread = new Thread(new BookingThread(system, i + 6, false));
            thread.setPriority(Thread.NORM_PRIORITY);
            thread.start();
        }
    }
}
```

Output: