# Experiment 4

**Student Name: Agrim Justa** **UID: 22BCS15530**
**Branch: CSE**
**Section/Group:22BCS_KPIT 901/A**
**Semester: 6** **Date of Performance: 15/01/25**
**Subject Name: Project Based Learning in Java** **Subject Code: 22CSH-359**

**1. Aim:** Develop Java programs using core concepts such as data structures, collections, and multithreading to manage and manipulate data.

a.) 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.

b.) 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.

c.) 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.

## 2. Objective:

The objective of this Java program is to develop applications utilizing core Java concepts such as data structures, collections, and multithreading to efficiently manage and manipulate data.

## 3. Implementation/Code:

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

class Employee {
    int id;
    String name;
    double salary;

    Employee(int id, String name, double salary) {
```

```
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

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

public class EmployeeManagement {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        Scanner sc = new Scanner(System.in);
        int choice;

        System.out.println("Agrim Justa - UID: 22BCS15530");

        do {
            System.out.println("\n1. Add Employee\n2. Update Employee\n3. Remove Employee\n4. Search
Employee\n5. Display All\n6. Exit");
            System.out.print("Enter choice: ");
            choice = sc.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter ID: ");
                    int id = sc.nextInt();
                    sc.nextLine();
                    System.out.print("Enter Name: ");
                    String name = sc.nextLine();
                    System.out.print("Enter Salary: ");
                    double salary = sc.nextDouble();
                    employees.add(new Employee(id, name, salary));
                    System.out.println("Employee added successfully.");
                    break;

                case 2:
                    System.out.print("Enter ID to update: ");
```

```
                id = sc.nextInt();
                for (Employee emp : employees) {
                    if (emp.id == id) {
                        sc.nextLine();
                        System.out.print("Enter New Name: ");
                        emp.name = sc.nextLine();
                        System.out.print("Enter New Salary: ");
                        emp.salary = sc.nextDouble();
                        System.out.println("Employee updated successfully.");
                        break;
                    }
                }
                break;

            case 3:
                System.out.print("Enter ID to remove: ");
                id = sc.nextInt();
                employees.removeIf(emp -> emp.id == id);
                System.out.println("Employee removed successfully.");
                break;

            case 4:
                System.out.print("Enter ID to search: ");
                id = sc.nextInt();
                for (Employee emp : employees) {
                    if (emp.id == id) {
                        System.out.println(emp);
                        break;
                    }
                }
                break;

            case 5:
                System.out.println("Employee List:");
                for (Employee emp : employees) {
                    System.out.println(emp);
                }
                break;
        }
```

```
            } while (choice != 6);
            sc.close();
        }
}
```

**Output**

```
Agrim Justa - UID: 22BCS15530

1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All
6. Exit
Enter choice: 1
Enter ID: 101
Enter Name: Rajesh Sharma
Enter Salary: 60000
Employee added successfully.

Enter choice: 1
Enter ID: 102
Enter Name: Priya Verma
Enter Salary: 75000
Employee added successfully.

Enter choice: 5
Employee List:
ID: 101, Name: Rajesh Sharma, Salary: 60000.0
ID: 102, Name: Priya Verma, Salary: 75000.0

Enter choice: 6
```

**B**)

```
import java.util.*;
```

```
class CardCollection {
    public static void main(String[] args) {
        HashMap<String, List<String>> cards = new HashMap<>();
        Scanner sc = new Scanner(System.in);

        System.out.println("Agrim Justa - UID: 22BCS15530");

        while (true) {
            System.out.println("\n1. Add Card\n2. Search by Symbol\n3. Display All Cards\n4. Exit");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter Symbol: ");
                    String symbol = sc.nextLine();
                    System.out.print("Enter Card Name: ");
                    String cardName = sc.nextLine();
                    cards.putIfAbsent(symbol, new ArrayList<>());
                    cards.get(symbol).add(cardName);
                    System.out.println("Card added successfully.");
                    break;

                case 2:
                    System.out.print("Enter Symbol to search: ");
                    symbol = sc.nextLine();
                    System.out.println("Cards under " + symbol + ": " + cards.getOrDefault(symbol,
Collections.emptyList()));
                    break;

                case 3:
                    System.out.println("All Cards:");
                    for (var entry : cards.entrySet()) {
                        System.out.println(entry.getKey() + " -> " + entry.getValue());
                    }
                    break;

                case 4:
```

```
                    sc.close();
                    return;
                }
            }
        }
    }
```

## Output

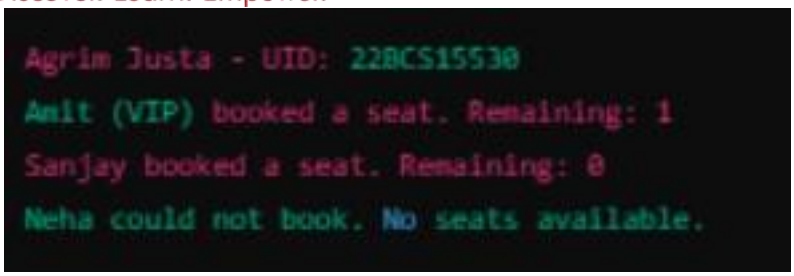
```
Agrim Justa - UID: 22BCS15530

1. Add Card
2. Search by Symbol
3. Display All Cards
4. Exit
Enter choice: 1
Enter Symbol: Spades
Enter Card Name: King of Spades
Card added successfully.

Enter choice: 1
Enter Symbol: Hearts
Enter Card Name: Queen of Hearts
Card added successfully.

Enter choice: 2
Enter Symbol to search: Hearts
Cards under Hearts: [Queen of Hearts]

Enter choice: 3
All Cards:
Spades -> [King of Spades]
Hearts -> [Queen of Hearts]

Enter choice: 4
```

C)class TicketBookingSystem {

```
    private int availableSeats = 2;

    public synchronized void bookSeat(String name) {
        if (availableSeats > 0) {
            System.out.println(name + " booked a seat. Remaining: " + (--availableSeats));
```

```
            } else {
                System.out.println(name + " could not book. No seats available.");
            }
        }
}

class BookingThread extends Thread {
        TicketBookingSystem system;
        String name;

        BookingThread(TicketBookingSystem system, String name, int priority) {
            this.system = system;
            this.name = name;
            setPriority(priority);
        }

        public void run() {
            system.bookSeat(name);
        }
}

public class TicketBooking {
        public static void main(String[] args) {
            TicketBookingSystem system = new TicketBookingSystem();

            System.out.println("Agrim Justa - UID: 22BCS15530");

            BookingThread user1 = new BookingThread(system, "Amit (VIP)", Thread.MAX_PRIORITY);
            BookingThread user2 = new BookingThread(system, "Sanjay", Thread.NORM_PRIORITY);
            BookingThread user3 = new BookingThread(system, "Neha", Thread.MIN_PRIORITY);

            user1.start();
            user2.start();
            user3.start();
        }
}
```

**Output**




## 4. Learning Outcome

- Learnt to use Java arrays for storing and managing data such as videos and customers.
- Learnt to handle user input using the Scanner class and process console-based interactions.
- Learnt to implement basic CRUD operations (Create, Read, Update) in a video rental system.
- Learnt to design and implement menu-driven console applications using loops and switch-case statements.
- Learnt to search for data within arrays using loops and conditional statements.