



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 2.1

Student Name: Akshat dua

UID: 22bcs10591

Branch: CSE

Section/Group: Kpit-901/A

Semester: 6

Date of Performance: 21/02/25

Subject Name: PBLJ

Subject Code: 22CSH-359

1. Aim:

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

2. Objective:

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.
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.
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.

3. Implementation/Code:

1. Program of Array List

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

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

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

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

public void display() {
    System.out.println("ID: " + id + ", Name: " + name + ", Salary: " +
salary);
}
}

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

        while (true)
        {
            System.out.println("\n1. Add Employee");
            System.out.println("2. Update Employee");
            System.out.println("3. Remove Employee");
            System.out.println("4. Search Employee");
            System.out.println("5. Exit");

            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();
```

Name: Akshat dua
Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
switch (choice)
{
    case 1:
        System.out.print("Enter Id of the Employee: ");
        int id = sc.nextInt();

        sc.nextLine(); // Consume newline

        System.out.print("Enter name of the Employee: ");
        String name = sc.nextLine();

        System.out.print("Enter salary of the Employee: ");
        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: ");
        int updateId = sc.nextInt();
        boolean updated = false;

        for (Employee e : employees)
        {
            if (e.id == updateId)
            {
                sc.nextLine(); // Consume newline
```

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        System.out.print("Enter new Name: ");
        e.name = sc.nextLine();

        System.out.print("Enter new Salary: ");
        e.salary = sc.nextDouble();

        System.out.println("Employee updated successfully!");
        updated = true;

        break;
    }
}

if (!updated)
{
    System.out.println("Employee with ID " + updateId + " not
found.");
}
break;

case 3:
    System.out.print("Enter ID to Remove: ");
    int removeId = sc.nextInt();

    boolean removed = employees.removeIf(e -> e.id == removeId);

    if (removed)
    {
```

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        System.out.println("Employee removed successfully!");
    }
    else
    {
        System.out.println("Employee with ID " + removeId + " not
found.");
    }
    break;
```

case 4:

```
    System.out.print("Enter ID to search: ");
    int searchId = sc.nextInt();
    boolean found = false;

    for (Employee e : employees)
    {
        if (e.id == searchId)
        {
            e.display();
            found = true;
            break;
        }
    }

    if (!found)
    {
        System.out.println("Employee with ID " + searchId + " not
found.");
```

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        }  
        break;  
  
    case 5:  
        System.out.println("Exiting program...");  
        sc.close();  
        return;  
  
    default:  
        System.out.println("Invalid choice! Please enter a number  
between 1 and 5.");  
    }  
}  
}
```

2. Program using collection interface

```
import java.util.*;  
  
class Card  
{  
    private final String value;  
  
    public Card(String value)  
    {  
        this.value = value;  
    }  
  
    @Override  
    public String toString()
```

Name: Akshat dua
Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
{
    return value;
}

public class CardCollection
{
    private static final Map<String, Set<Card>> cardMap = new
    HashMap<>();
    private static final Scanner scanner = new Scanner(System.in);

    private static void addCard()
    {
        System.out.print("Enter Card Symbol: ");
        String symbol = scanner.nextLine().trim();
        System.out.print("Enter Card Value: ");
        String value = scanner.nextLine().trim();

        cardMap.computeIfAbsent(symbol, k -> new HashSet<>()).add(new
        Card(value));
        System.out.println("Card added successfully!");
    }

    private static void findCardsBySymbol()
    {
        System.out.print("Enter Symbol to search: ");
        String symbol = scanner.nextLine().trim();
        if (cardMap.containsKey(symbol))
        {
            System.out.println("Cards under " + symbol + ": " +
            cardMap.get(symbol));
        }
        else
        {

```

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        System.out.println("No cards found for symbol: " + symbol);
    }
}

private static void displayAllCards()
{
    if (cardMap.isEmpty())
    {
        System.out.println("No cards available.");
    }
    else
    {
        cardMap.forEach((symbol, cards) ->
            System.out.println(symbol + ": " + cards));
    }
}

public static void main(String[] args)
{
    while (true)
    {
        System.out.println("\n1. Add Card\n2. Find Cards by Symbol\n3.
Display All Cards\n4. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice)
        {
            case 1 -> addCard();
            case 2 -> findCardsBySymbol();
            case 3 -> displayAllCards();
            case 4 -> {
                System.out.println("Exiting...");
            }
        }
    }
}
```

Name: Akshat dua

Uid: 22bcs10591


```
        scanner.close();
        return;
    }
    default -> System.out.println("Invalid choice! Try again.");
}
}
}
```

3. Program using Thread Priorities

```
import java.util.Scanner;
import java.util.concurrent.locks.*;

class SeatReservationSystem
{
    private int seatsAvailable;
    private final Lock seatLock = new ReentrantLock();

    public SeatReservationSystem(int totalSeats)
    {
        this.seatsAvailable = totalSeats;
    }

    public void reserveSeat(String userName, int requestedSeats)
    {
        seatLock.lock();
        try
        {
            if (seatsAvailable >= requestedSeats)
            {
                System.out.println(userName + " successfully reserved " +
                requestedSeats + " seat(s). Seats left: " + (seatsAvailable - requestedSeats));
                seatsAvailable -= requestedSeats;
            }
        }
    }
}
```

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        }
        else
        {
            System.out.println("Apologies, " + userName + "! Only " +
seatsAvailable + " seat(s) remaining.");
        }
    }
    finally
    {
        seatLock.unlock();
    }
}
}
```

```
class Passenger extends Thread
{
    private SeatReservationSystem reservationSystem;
    private String userName;
    private int requestedSeats;

    public Passenger(SeatReservationSystem system, String name, int seats,
int priorityLevel)
    {
        this.reservationSystem = system;
        this.userName = name;
        this.requestedSeats = seats;
        setPriority(priorityLevel);
    }

    @Override
    public void run()
    {
        reservationSystem.reserveSeat(userName, requestedSeats);
    }
}
```

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}

public class TicketReservationApp
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter total number of seats available: ");
        int totalSeats = input.nextInt();
        SeatReservationSystem system = new
        SeatReservationSystem(totalSeats);

        System.out.print("Enter number of passengers: ");
        int passengerCount = input.nextInt();
        input.nextLine();

        Passenger[] passengers = new Passenger[passengerCount];

        for (int i = 0; i < passengerCount; i++)
        {
            System.out.print("Enter passenger name: ");
            String name = input.nextLine();
            System.out.print("Enter number of seats required: ");
            int seats = input.nextInt();
            System.out.print("Select priority level (1 - High, 2 - Standard): ");
            int priority = input.nextInt();
            input.nextLine();

            int assignedPriority = (priority == 1) ? Thread.MAX_PRIORITY :
            Thread.NORM_PRIORITY;
            passengers[i] = new Passenger(system, name, seats,
            assignedPriority);
        }
    }
}
```

Name: Akshat dua

Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        for (Passenger passenger : passengers)
        {
            passenger.start();
        }

        input.close();
    }
}
```

4. Output:

```
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Exit
Enter your choice: **1**
Enter Id of the Employee: **101**
Enter name of the Employee: **Alice**
Enter salary of the Employee: **50000**
Employee added successfully!

Enter your choice: **4**
Enter ID to search: **101**
ID: 101, Name: Alice, Salary: 50000.0

Enter your choice: **3**
Enter ID to Remove: **101**
Employee removed successfully!

Enter your choice: **4**
Enter ID to search: **101**
Employee with ID 101 not found.

Enter your choice: **5**
Exiting program...
```

Name: Akshat dua
Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
1. Add Card
2. Find Cards by Symbol
3. Display All Cards
4. Exit
Enter your choice: **1**
Enter Card Symbol: **Hearts**
Enter Card Value: **Ace**
Card added successfully!

Enter your choice: **1**
Enter Card Symbol: **Spades**
Enter Card Value: **King**
Card added successfully!

Enter your choice: **2**
Enter Symbol to search: **Hearts**
Cards under Hearts: [Ace]

Enter your choice: **3**
Hearts: [Ace]
Spades: [King]

Enter your choice: **4**
Exiting...
```

Name: Akshat dua
Uid: 22bcs10591



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
Enter total number of seats available: **5**
Enter number of passengers: **3**

Enter passenger name: **Alice**
Enter number of seats required: **2**
Select priority level (1 - High, 2 - Standard): **1**

Enter passenger name: **Bob**
Enter number of seats required: **3**
Select priority level (1 - High, 2 - Standard): **2**

Enter passenger name: **Charlie**
Enter number of seats required: **1**
Select priority level (1 - High, 2 - Standard): **1**
```

```
Alice successfully reserved 2 seat(s). Seats left: 3
Bob successfully reserved 3 seat(s). Seats left: 0
Apologies, Charlie! Only 0 seat(s) remaining.
```

6. Learning Outcome:

- **Employee Management System:** Learned how to use **ArrayList** for dynamic object storage and perform **CRUD operations** efficiently.
- **Card Collection System:** Practiced **HashMap & HashSet** for structured data storage and retrieval with **computeIfAbsent()**.
- **Ticket Reservation System:** Gained hands-on experience with **multithreading, locks (ReentrantLock), and thread synchronization** to prevent race conditions.
- **User Interaction & Input Handling:** Improved skills in handling **user input, loops, and conditionals** while ensuring smooth execution in interactive Java applications.

Name: Akshat dua

Uid: 22bcs10591