Experiment 4

Student Name: Anirudh Singh Manhas
UID: 22BCS10576
Branch:BE-CSE
Section/Group:22BCS_KPIT-901A
Semester:6th
Date of Performance:21-2-25
Subject Name: Project based learning with Java
Subject Code:22CSH-359

1. **Aim:** Easy level : 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.

   Medium level: 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.

   Hard level: 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:** Following are the objectives:

- **Understanding List Operations:** To learn how to manipulate lists and understand basic list operations.
- **Application of Switch-Case:** Learn and apply the switch-case statement to determine specific outputs based on given conditions.
- **Developing Problem-Solving Skills Objective:** To enhance problem-solving and logical thinking by implementing simple algorithms to manipulate data in a list.
- **Dynamic Data Retrieval:** Simulate a real-world scenario of retrieving and displaying data dynamically based on input criteria.

3. **Implementation/Code:**
   Easy level:

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

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

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

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

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

   public static void addEmployee()
      { System.out.print("Enter Employee ID: ");
      int id = scanner.nextInt();
      scanner.nextLine(); // consume newline
      System.out.print("Enter Employee Name: ");
      String name = scanner.nextLine();
      System.out.print("Enter Employee Salary: ");
      double salary = scanner.nextDouble();
      employees.add(new Employee(id, name, salary));
      System.out.println("Employee added successfully!\n");
   }

   public static void updateEmployee()
      { System.out.print("Enter Employee ID to update: ");
      int id = scanner.nextInt();
      for (Employee emp : employees)
         { if (emp.id == id) {
```

```
            scanner.nextLine(); // consume newline
            System.out.print("Enter new Name: ");
            emp.name = scanner.nextLine();
            System.out.print("Enter new Salary: ");
            emp.salary = scanner.nextDouble();
            System.out.println("Employee updated successfully!\n");
            return;
        }
    }
    System.out.println("Employee not found!\n");
}

public static void removeEmployee()
    { System.out.print("Enter Employee ID to remove:
    "); int id = scanner.nextInt();
    employees.removeIf(emp -> emp.id == id);
    System.out.println("Employee removed successfully!\n");
}

public static void searchEmployee()
    { System.out.print("Enter Employee ID to search: ");
    int id = scanner.nextInt();
    for (Employee emp : employees)
        { if (emp.id == id) {
            System.out.println(emp + "\n");
            return;
        }
    }
    System.out.println("Employee not found!\n");
}

public static void displayEmployees()
    { if (employees.isEmpty()) {
        System.out.println("No employees found!\n");
        return;
    }
    for (Employee emp : employees)
        { System.out.println(emp);
```

```
        }
        System.out.println();
    }

    public static void main(String[] args)
        { while (true) {
            System.out.println("Employee 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 Employees");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1: addEmployee(); break;
                case 2: updateEmployee(); break;
                case 3: removeEmployee(); break;
                case 4: searchEmployee(); break;
                case 5: displayEmployees(); break;
                case 6: System.out.println("Exiting...\n"); return;
                default: System.out.println("Invalid choice!\n");
            }
        }
    }
}
```

Medium Level:

```
import java.util.*;

class Card {
    private String symbol;
    private String name;

    public Card(String symbol, String name) {
```

```
        this.symbol = symbol;
        this.name = name;
    }

    public String getSymbol()
        { return symbol;
    }

    @Override
    public String toString() {
        return "Card{" + "symbol='" + symbol + '\'' + ", name='" + name + '\'' + '}';
    }
}

public class Main {
    private static Collection<Card> cards = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void addCard()
        { System.out.print("Enter Card Symbol: ");
        String symbol = scanner.next();
        System.out.print("Enter Card Name: ");
        String name = scanner.next();
        cards.add(new Card(symbol, name));
        System.out.println("Card added successfully!\n");
    }

    public static void removeCard()
        { System.out.print("Enter Card Symbol to remove:
        "); String symbol = scanner.next();
        cards.removeIf(card -> card.getSymbol().equals(symbol));
        System.out.println("Card removed successfully!\n");
    }

    public static void searchCardsBySymbol()
        { System.out.print("Enter symbol to search: ");
        String symbol = scanner.next();
        boolean found = false;
```

```
    for (Card card : cards) {
        if (card.getSymbol().equals(symbol))
            { System.out.println(card);
            found = true;
        }
    }
    if (!found) {
        System.out.println("No cards found with the given symbol!\n");
    }
}

public static void displayCards()
    { if (cards.isEmpty()) {
        System.out.println("No cards available!\n");
        return;
    }
    for (Card card : cards)
        { System.out.println(card);
    }
    System.out.println();
}

public static void main(String[] args)
    { while (true) {
        System.out.println("Card Collection Manager");
        System.out.println("1. Add Card");
        System.out.println("2. Remove Card");
        System.out.println("3. Search Cards by Symbol");
        System.out.println("4. Display All Cards");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
            case 1: addCard(); break;
            case 2: removeCard(); break;
            case 3: searchCardsBySymbol(); break;
            case 4: displayCards(); break;
```

```
                case 5: System.out.println("Exiting...\n"); return;
                default: System.out.println("Invalid choice!\n");
            }
        }
    }
}
```

Hard level:

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

class TicketBookingSystem
    { private int availableSeats;
    private final Lock lock = new ReentrantLock();

    public TicketBookingSystem(int seats)
        { this.availableSeats = seats;
    }

    public void bookSeat(String customerType, int seats)
        { lock.lock();
        try {
            if (availableSeats >= seats) {
                System.out.println(customerType + " booked " + seats + " seat(s)
successfully.");
                availableSeats -= seats;
            } else {
                System.out.println(customerType + " failed to book. Not enough seats
available.");
            }
        } finally
            { lock.unlock();
        }
    }
}

class Customer extends Thread {
```

```
    private TicketBookingSystem system;
    private int seats;
    private String customerType;

    public Customer(TicketBookingSystem system, int seats, String customerType, int
priority) {
        this.system = system;
        this.seats = seats;
        this.customerType = customerType;
        setPriority(priority);
    }

    @Override
    public void run()
        { system.bookSeat(customerType,
        seats);
    }
}

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

        System.out.print("Enter number of customers: ");
        int numCustomers = scanner.nextInt();

        for (int i = 0; i < numCustomers; i++)
            { System.out.print("Enter customer type (VIP/Regular): ");
            String type = scanner.next();
            System.out.print("Enter number of seats to book: ");
            int seats = scanner.nextInt();
            int priority = type.equalsIgnoreCase("VIP") ? Thread.MAX_PRIORITY :
Thread.NORM_PRIORITY;
            new Customer(system, seats, type + " Customer " + (i + 1), priority).start();
        }
```

```
        scanner.close();
    }
}
```

4. Output:

Easy:

```
Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display Employees
6. Exit
Enter your choice: 1
Enter Employee ID: 1
Enter Employee Name: harleen
Enter Employee Salary: 50000
Employee added successfully!

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display Employees
6. Exit
Enter your choice: 4
Enter Employee ID to search: 1
Employee ID: 1, Name: harleen, Salary: 50000.0

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display Employees
6. Exit
Enter your choice: 5
Employee ID: 1, Name: harleen, Salary: 50000.0

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display Employees
6. Exit
Enter your choice:
```

Medium:

```
Card Collection Manager
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Display All Cards
5. Exit
Enter your choice: 1
Enter Card Symbol: @
Enter Card Name: ace
Card added successfully!

Card Collection Manager
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Display All Cards
5. Exit
Enter your choice: 1
Enter Card Symbol: #
Enter Card Name: heart
Card added successfully!

Card Collection Manager
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Display All Cards
5. Exit
Enter your choice: 4
Card{symbol='@', name='ace'}
Card{symbol='#', name='heart'}

Card Collection Manager
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Display All Cards
5. Exit
Enter your choice: 3
Enter symbol to search: @
Card{symbol='@', name='ace'}
Card Collection Manager
1. Add Card
2. Remove Card
3. Search Cards by Symbol
4. Display All Cards
5. Exit
Enter your choice:
```

Hard:

```
Enter total number of available seats: 4
Enter number of customers: 4
Enter customer type (VIP/Regular): vip
Enter number of seats to book: 4
vip Customer 1 booked 4 seat(s) successfully.
Enter customer type (VIP/Regular): regular
Enter number of seats to book: 5
Enter customer type (VIP/Regular): regular Customer 2 failed to book. Not enough seats available.
```

5. Learning Outcomes: Following are the learning outcomes:

- **Understanding Java Fundamentals:** I developed Improved understanding of basic Java concepts such as arrays, strings,, and conditionals.
- **Working with Data Structures:** I Learnt how to store and manipulate data using data structures or related information.
- **Switch Case Usage :** I Developed skills in using switch-case statements to determine designations and calculate allowances based on specific conditions.
- **Command-Line Applications :** I Learnt how to build simple command-line applications that can take user inputs dynamically and produce desired outputs.