Experiment – 4

Student Name: Lokesh Yadav
UID: 22BCS16604
Branch: BE-CSE
Section/Group:IOT-631/A
Semester: 6th
Dateof Performance:14-02-25
Subject Name: Java
Subject Code: 22CSH-352

1. Aim:

(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

2. Objective:

(A): The objective is to develop a Java program that manages employee records using an `ArrayList`, allowing users to:

- **Add** employee details (ID, Name, Salary).
- **Update** existing employee details.
- **Remove** an employee from the list.
- **Search** for an employee by their ID.
- Provide an interactive console-based menu for seamless user interaction.

(B): To create a Java program that maintains a collection of playing cards using the `Collection` interface, enabling users to:

- **Add** cards with a specific symbol and value.
- **Retrieve** all cards belonging to a specific symbol.
- Use an efficient data structure (`ArrayList`) to store and manage card details dynamically.
- Provide a simple and user-friendly interface for managing card collections.

3. Algorithm :

(A) :

- **Initialize** an `ArrayList` to store employee details (ID, Name, Salary).
- **Display Menu** with the following options:
  - Add Employee
  - Update Employee
  - Remove Employee
  - Search Employee
  - Exit
- **If user chooses "Add Employee"**:

- Prompt the user for Employee ID, Name, and Salary.
- Create a new `Employee` object and add it to the `ArrayList`.
- Display a success message.

➢ **If user chooses "Update Employee"**:
- Ask for the Employee ID.
- Search the `ArrayList` for a matching ID.
- If found, prompt the user for new Name and Salary, and update the object.
- Display a success message; if not found, show an error message.

➢ **If user chooses "Remove Employee"**:
- Ask for Employee ID.
- Remove the employee from the `ArrayList` if found.
- Display success message; if not found, show an error message.

➢ **If user chooses "Search Employee"**:
- Ask for Employee ID.
- Search for and display the employee details if found, otherwise show an error message.

➢ **Repeat the process** until the user selects "Exit".

(B):

➢ **Initialize** a Collection (e.g., ArrayList<Card>) to store cards.

➢ **Display Menu** with the following options:
- Add Card
- Find Cards by Symbol
- Exit

➢ **If user chooses "Add Card"**:
- Prompt the user for Card Symbol (e.g., Hearts, Spades).
- Prompt for Card Value (e.g., Ace, King, 10).
- Create a Card object and store it in the collection.
- Display a success message.

➢ **If user chooses "Find Cards by Symbol"**:
- Ask for a symbol (e.g., Hearts).
- Search for and display all cards with the given symbol.
- If no cards are found, display an appropriate message.

➢ **Repeat the process** until the user selects "Exit".

4. Code/Implementation :

CODE (A):

```
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 "ID: " + id + ", Name: " + name + ", Salary: " + salary;
    }
}

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

    public static void addEmployee() {
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        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!");
    }

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

    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!");
    }

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

    public static void main(String[] args) {
        while (true) {
            System.out.println("\n1. Add Employee\n2. Update Employee\n3. Remove Employee\n4. Search Employee\n5. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> addEmployee();
                case 2 -> updateEmployee();
                case 3 -> removeEmployee();
                case 4 -> searchEmployee();
                case 5 -> {
                    System.out.println("Exiting...");
                    return;
                }
                default -> System.out.println("Invalid choice! Try again.");
            }
        }
    }
}
```

CODE (B):

```
import java.util.*;

class Card {
    String symbol;
    String value;

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

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

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

    public static void addCard() {
        System.out.print("Enter Card Symbol (e.g., Hearts, Spades): ");
        String symbol = scanner.next();
        System.out.print("Enter Card Value (e.g., Ace, King, 10): ");
        String value = scanner.next();
        cards.add(new Card(symbol, value));
        System.out.println("Card added successfully!");
    }

    public static void findCardsBySymbol() {
        System.out.print("Enter Symbol to find cards: ");
        String symbol = scanner.next();
        System.out.println("Cards with symbol " + symbol + ":");
        for (Card card : cards) {
            if (card.symbol.equalsIgnoreCase(symbol)) {
                System.out.println(card);
            }
        }
    }

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

```
            System.out.println("\n1. Add Card\n2. Find Cards by Symbol\n3. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1 -> addCard();
                case 2 -> findCardsBySymbol();
                case 3 -> {
                    System.out.println("Exiting...");
                    return;
                }
                default -> System.out.println("Invalid choice! Try again.");
            }
        }
    }
}
```

5.Output:

```
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Exit
Choose an option: 1
Enter Employee ID: 101
Enter Name: John
Enter Salary: 16604
Employee added successfully!

1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Exit
Choose an option:
```

```
1. Add Card
2. Find Cards by Symbol
3. Exit
Choose an option: 1
Enter Card Symbol (e.g., Hearts, Spades): Hearts
Enter Card Value (e.g., Ace, King, 10): 4
Card added successfully!

1. Add Card
2. Find Cards by Symbol
3. Exit
Choose an option:
```

6. Learning Outcomes:

- **Collection & Data Management** – Use `ArrayList` and `Collection` to store and manage dynamic data.

- **Object-Oriented Programming** – Apply concepts like classes, objects, and encapsulation.
- **CRUD Operations & Searching** – Implement add, update, remove, and search functionalities.
- **Iteration & Processing** – Utilize loops and iterators to manipulate data efficiently.
- **User Interaction & Input Handling** – Develop interactive console applications with `Scanner`.