DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment 4

Student Name: Abhishek **UID: 22BCS11779**
Branch: CSE **Section: 631-A**
Semester: 6th **DOP:18-02-25**
Subject: Java **Subject Code:22CSH-359**

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

Objective: 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.

Algorithm:

- Initialize an empty ArrayList<Employee>.
- Display menu with options: Add, Update, Remove, Search, List All, Exit.
- Input choice from user.
- If choice is "Add":
  - Input employee ID, Name, and Salary.
  - Create new Employee and add to list.
- If choice is "Update":
  - Input employee ID.
  - Find employee by ID.
  - Update Name and Salary. If choice is "Remove":
-
  - Input employee ID.
  - Find employee by ID and remove from list.
- If choice is "Search":
  - Input employee ID.
  - Find employee by ID and display details. If choice is "List All":
-
  - Display all employees in the list.
- If choice is "Exit":
  - End the program.
- Repeat until "Exit" is selected.

Code:

```
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 EmployeeManagementSystem {

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

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

        while (true) {
            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. List All Employees");
            System.out.println("6. Exit");
System.out.print("Enter your choice: ");
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:
listAllEmployees();
break;
            case 6:
                System.out.println("Exiting the
system.");                   return;              default:
                System.out.println("Invalid choice. Please try again.");
            }
        }
    }

    // Add an employee
    private 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();

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

    // Update employee details     private
static void updateEmployee() {
        System.out.print("Enter Employee ID to update: ");
        int id = scanner.nextInt();
        scanner.nextLine();  // consume newline
Employee employee = findEmployeeById(id);

        if (employee != null) {
            System.out.print("Enter new Name: ");
employee.name = scanner.nextLine();
System.out.print("Enter new Salary: ");
employee.salary = scanner.nextDouble();
            System.out.println("Employee updated successfully.");
        } else {
            System.out.println("Employee with ID " + id + " not found.");
        }
```

```
    }

    // Remove an employee
    private static void removeEmployee() {
        System.out.print("Enter Employee ID to remove: ");
int id = scanner.nextInt();
        Employee employee = findEmployeeById(id);

        if (employee != null) {
employees.remove(employee);
            System.out.println("Employee removed successfully.");
        } else {
            System.out.println("Employee with ID " + id + " not found.");         }
    }

    // Search for an employee by ID
private static void searchEmployee() {
        System.out.print("Enter Employee ID to search: ");
int id = scanner.nextInt();
        Employee employee = findEmployeeById(id);

        if (employee != null) {
            System.out.println("Employee found: " + employee);
        } else {
            System.out.println("Employee with ID " + id + " not found.");
        }
    }

    // List all employees
    private static void listAllEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees in the system.");
} else {
            System.out.println("List of all employees:");
for (Employee employee : employees) {
                System.out.println(employee);
            }
        }
    }

    // Helper method to find employee by ID     private
static Employee findEmployeeById(int id) {         for
(Employee employee : employees) {
            if (employee.id == id) {
                return employee;
            }
}
        return null;
    }
```

```
}
```

Learning Outcomes:

1. Demonstrate: Apply key concepts to real-world scenarios to showcase understanding.
2. Analyze: Critically evaluate information, identify patterns, and draw meaningful conclusions.
3. Create: Develop original work, including presentations, reports, or projects, to exhibit comprehension and skills.
4. Communicate: Convey ideas and findings effectively through oral and written communication.
5. Collaborate: Contribute to group projects and exhibit strong teamwork capabilities in a collaborative environment.

Output

```
Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. List All Employees
6. Exit
Enter your choice: 1
Enter Employee ID: 610
Enter Employee Name: Section B
Enter Employee Salary: 25000
Employee added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. List All Employees
6. Exit
Enter your choice: 5
List of all employees:
ID: 610, Name: Section B, Salary:

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

2.

Objective: 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.

Code:

```
import java.util.*;

class Card {
    private String symbol;
    private int value;

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

    public String getSymbol() {
        return symbol;
    }

    public int getValue() {
        return value;
    }

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

public class CardCollection {
    public static void main(String[] args) {
        Collection<Card> cardCollection = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        // Adding cards



System.out.print("Enter the number of cards to add: ");
        int numCards = scanner.nextInt();
        scanner.nextLine(); // Consume newline
```

```
        for (int i = 0; i < numCards; i++) {
            System.out.print("Enter card " + (i + 1) + " symbol: ");
            String symbol = scanner.nextLine();
            System.out.print("Enter card " + (i + 1) + " value: ");
            int value = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            cardCollection.add(new Card(symbol, value));
        }

        // Display all cards
        System.out.println("\nAll Cards:");
        for (Card card : cardCollection) {
            System.out.println(card);
        }

        // Finding cards by symbol
        System.out.print("\nEnter symbol to search for cards: ");
        String searchSymbol = scanner.nextLine();
        System.out.println("Cards with symbol '" + searchSymbol + "':");
        boolean found = false;
        for (Card card : cardCollection) {
            if (card.getSymbol().equalsIgnoreCase(searchSymbol)) {
                System.out.println(card);
                found = true;
            }
        }
        if (!found) {
            System.out.println("No cards found with the symbol '" + searchSymbol + "'.");
        }

        scanner.close();
    }
}
```


Enter the number of cards to add: 2
Enter card 1 symbol: 2
Enter card 1 value: 13
Enter card 2 symbol: 1
Enter card 2 value: 15
All Cards:
Card{symbol='2', value=13}
Card{symbol='1', value=15}
Enter symbol to search for cards: 2
Cards with symbol '2':
Card{symbol='2', value=13}
...Program finished with exit code 0
Press ENTER to exit console.

CU
CHANDIGARH UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING