Experiment- 4

Name: Karanpreet Singh
Branch: CSE
Semester: 6th
Subject: Java

UID: 22BCS12898
Section:22BCS_DL-901/B
Date of Performance: 13/02/2025
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: 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.

3. Algorithm:

i. Define Data Structure:
- Create an Employee class/structure with:
- Integer id
- String name
- Double salary

ii. Initialize Data Storage:
- Create an empty list (e.g., ArrayList<Employee>) to hold employee objects.

iii. Main Loop:
- Repeat until the user chooses to exit:

a. Display Menu Options:
- "1. Add Employee"
- "2. Update Employee"
- "3. Remove Employee"
- "4. Search Employee"
- "5. Display All Employees"
- "0. Exit"

b. Input Choice:
- Read the user's menu option (e.g., as an integer).

iv. Process User Choice:
- If choice is1 (Add Employee):

1. Prompt the user to enter Employee ID.
2. Prompt the user to enter Employee Name.
3. Prompt the user to enter Employee Salary.
4. Create a new Employee object with the provided details.
5. Add the new employee to the list.
6. Display a success message.

If choice is 2 (Update Employee):

1. Prompt the user to enter the Employee ID to update.
2. Search for the employee in the list using the given ID.
3. If the employee exists:
   - Prompt the user to enter the new Name.
   - Prompt the user to enter the new Salary.
   - Update the employee's name and salary.
   - Display a success message.

4. Else:
   - Display a "not found" message.
- If choice is 3 (Remove Employee):

1. Prompt the user to enter the Employee ID to remove.
2. Search for the employee in the list using the given ID.
3. If the employee exists:
   - Remove the employee from the list.
   - Display a success message.

4. Else:
   - Display a "not found" message.
- If choice is 4 (Search Employee):

1. Prompt the user to enter the Employee ID to search.
2. Search for the employee in the list using the given ID.
3. If the employee exists:
   - Display the employee's details.

4. Else:
   - Display a "not found" message.
   - If choice is 5 (Display All Employees):

1. If the employee list is empty:
   - Display a message indicating no employees to show.
2. Else:
   - Iterate over the list and display each employee's details.
   - If choice is 0 (Exit):
   - Terminate the program loop.

v. End Program

4. Code:

```
import java.util.ArrayList;
import java.util.Scanner;
class Employee {
   private int id;
   this.id = id;
      this.name = name;
      this.salary = salary;
   }
```

```
    public int getId() { return id; }
    public String getName() { return name;
    } public double getSalary() { return
    salary; }
    public void setName(String name) { this.name = name; }
    public void setSalary(double salary) { this.salary = salary; }
    @Override
    public String toString() {
        return "Employee ID: " + id + ", Name: " + name + ", Salary: " + salary;
    }
}
public class EmployeeManagement {
        private String
        name;
        private double salary;
        public Employee(int id, String name, double salary) {

    private static ArrayList<Employee> employees = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        int choice;
        do {
            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. Display All Employees");
            System.out.println("0. Exit");
            System.out.print("Enter your choice:
            "); choice = scanner.nextInt();
            scanner.nextLine();
            switch(choice) {
                case 1: addEmployee(); break;
                case 2: updateEmployee();
                break; case 3:
                removeEmployee(); break; case
                4: searchEmployee(); break;
                case 5: displayEmployees();
                break;
                case 0: System.out.println("Exiting..."); break;
                default: System.out.println("Invalid choice. Try again.");
            }
        } while(choice != 0);
    }
    private static void addEmployee() {
        System.out.print("Enter Employee ID:
        "); int id = scanner.nextInt();
        scanner.nextLine();
```

```
    System.out.print("Enter Employee Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter Employee Salary: ");
    double salary = scanner.nextDouble();
    scanner.nextLine();
    employees.add(new Employee(id, name, salary));
    System.out.println("Employee added successfully.");
}
private static void updateEmployee() {
    System.out.print("Enter Employee ID to update: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    Employee emp = findEmployeeById(id);
    if(emp != null) {
        System.out.print("Enter new Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter new Salary: ");
        double salary = scanner.nextDouble();
        scanner.nextLine();
        emp.setName(name);
        emp.setSalary(salary);
        System.out.println("Employee updated successfully.");

    } else {
        System.out.println("Employee not found.");
    }
 }
 private static void removeEmployee() {
    System.out.print("Enter Employee ID to remove: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    Employee emp = findEmployeeById(id);
    if(emp != null) {
        employees.remove(emp);
        System.out.println("Employee removed
        successfully.");
    } else {
        System.out.println("Employee not found.");
    }
 }
private static void searchEmployee() {
    System.out.print("Enter Employee ID to search: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    Employee emp = findEmployeeById(id);
    if(emp != null) {
            System.out.println("Employee found: " + emp);
        } else {
            System.out.println("Employee not found.");
        }
    }
```

```
    private static void displayEmployees() {
        if(employees.isEmpty()) {
            System.out.println("No employees to display.");
        } else {
            System.out.println("Employee List:");
            for(Employee emp : employees) {
                System.out.println(emp);
            }
        }
    }
    private static Employee findEmployeeById(int
        id) { for(Employee emp : employees) {
            if(emp.getId() ==
                id) { return
                emp;
            }
        }
        return null;
    }
}
```

5. Output:

```
Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
0. Exit
Enter your choice: 5
Employee List:
Employee ID: 17209, Name: Vishwas, Salary: 1500000.0
Employee ID: 17134, Name: Rajat, Salary: 1150000.0
```

6. Learning Outcomes:

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