Experiment-1

Student Name: Khyati singh **UID: 22BCS16405**

Branch: CSE **Section/Group: DL/901/A**

Semester: 6th **Date of Performance: 10/01/2025**

Subject Name: Java with Lab **Subject Code: 22CSH-359**

1) Aim: Create an application to save the employee information using arrays.

2) Objective: To develop a functional application that effectively utilizes arrays to store, manage, and retrieve employee information, enabling efficient data organization and manipulation within the application.

3) Algorithm:

1. **Initialize Employee Data**:
   - Create an array of Employee objects using the initializeEmployees method.
   - Each object stores details such as employee number, name, department, and salary components.

2. **Prompt User for Input**:
   - Use a Scanner to accept an employee number (empNo) from the user.

3. **Validate Input**:
   - Check if the input is numeric. If not, throw an IllegalArgumentException.

4. **Search for Employee**:
   - Iterate through the employees array.
   - Compare each Employee's empNo with the input value.

5. **Display Details**:
   - If a match is found:
     - Calculate the total salary using calculateSalary (Basic + HRA + DA - IT).
     - Display employee details using displayDetails.
   - If no match is found, print an appropriate message.

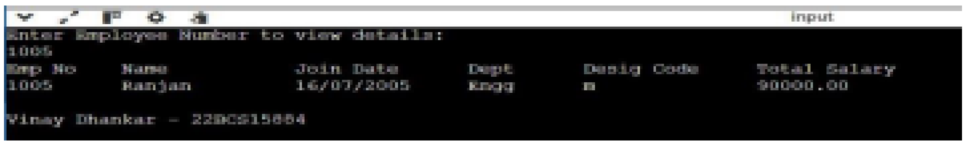
6. **Exception Handling**:
   - Handle invalid inputs and unexpected errors gracefully using try-catch.

7. **Terminate**:
   - Close resources and exit the program.

4) Implementation/Code:

```
import java.util.Scanner;
class Employee {
    int empNo;
    String empName;
    String joinDate;
    String desigCode;
    String department;
    double basic;
    double hra;
    double it;

    public Employee(int empNo, String empName, String joinDate, String desigCode, String
department, double basic, double hra, double it) {
        this.empNo = empNo;
        this.empName = empName;
        this.joinDate = joinDate;
        this.desigCode = desigCode;
        this.department = department;
        this.basic = basic;
        this.hra = hra;
        this.it = it;
    } public double getDA() {
        switch (desigCode) {
            case "e":
                return 20000;
            case "c":
                return 32000;
            case "k":
                return 12000;
            case "r":
                return 15000;
            case "m":
                return 40000;
            default:
                return 0; }
    }
    public double calculateSalary() {
        return basic + hra + getDA() - it;
    }
    public void displayDetails() {
        System.out.printf("%-10s%-15s%-15s%-10s%-15s%-15s%n",
                "Emp No", "Name", "Join Date", "Dept", "Desig Code", "Total Salary");
        System.out.printf("%-10d%-15s%-15s%-10s%-15s%-15.2f%n",
                empNo, empName, joinDate, department, desigCode, calculateSalary());
        System.out.println("\nNidhi Dhankar - 22BCS15886");
    }
}
public class Employees {
    public static void main(String[] args) {
        Employee[] employees = initializeEmployees();

        try (Scanner scanner = new Scanner(System.in)) {
            System.out.println("Enter Employee Number to view details:");

            if (!scanner.hasNextInt()) {
                throw new IllegalArgumentException("Invalid input. Please enter a numeric Employee
Number.");
            }
            int empNo = scanner.nextInt();
            boolean found = false;

            for (Employee emp : employees) {
                if (emp.empNo == empNo) {
                    emp.displayDetails();
                    found = true;
                    break;
                }
            }
            if (!found) {
                System.out.println("Employee not found.");
            } } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        }
    }
    private static Employee[] initializeEmployees() {
        return new Employee[] {
            new Employee(1001, "Ashish", "01/04/2009", "e", "R&D", 20000, 8000, 3000),
            new Employee(1002, "Sushma", "23/08/2012", "c", "PM", 30000, 12000,
            9000), new Employee(1003, "Rahul", "12/11/2008", "k", "Acct", 10000, 8000,
            1000),
            new Employee(1004, "Chahat", "29/01/2013", "r", "FrontDesk", 12000, 6000, 2000),
        new Employee(1005, "Ranjan", "16/07/2005", "m", "Engg", 50000, 20000, 20000), new
        Employee(1006, "Suman", "01/01/2000", "e", "Manufacturing", 23000, 9000, 4400), new
        Employee(1007, "Tanmay", "12/06/2006", "c", "PM", 29000, 12000, 10000) };
    }
}
```

5) Output:



```
Enter Employee Number to view details:
1005
Emp No     Name           Join Date      Dept       Desig Code      Total Salary
1005       Ranjan         16/07/2005     Engg       m               90000.00

Vinay Dhankar - 22BCS15884
```

6. Learning Outcome:

I. **Array Usage:** Learn to store, access, and iterate through arrays to manage related data like employee details.

II. **Switch-Case Logic**: Use switch-case to map designation codes to their respective roles and allowances efficiently.

III. **Input Validation**: Validate user inputs and use linear search to find data, handling invalid cases gracefully.

IV. **Real-World Application**: See how programming concepts can automate tasks like payroll processing effectively.

V. **Exception Handling:** Using try-catch blocks to handle Invalid input types (e.g., non-numeric values for Employee ID).