Experiment 1.1

Student Name: jayanth **UID: 22BCS11651**
Branch: CSE **Section:901-ML-B**
Semester: 6th **DOP:08/01/2025**
Subject: Java **Subject Code:22CSH-359**

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

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.

Algorithm:

Step 1: Initialize the Program

- Start the program.
- Define an array of structures to store employee information.
- Each structure will include fields such as Employee ID, Name, Age, and Department.

Step 2: Define Functions

1. **Add Employee Information**:
   - Prompt the user to enter details for an employee (ID, Name, Age, Department).
   - Store the entered details in the next available position in the array.
   - Check for array overflow (i.e., maximum number of employees).
2. **Display All Employee Information**:
   - Iterate through the array and print all stored employee details.
   - Handle cases where no employees are stored.
3. **Search for an Employee**:
   - Prompt the user to enter the Employee ID.
   - Search the array for a matching ID.
   - Display the employee's details if found, otherwise print a message indicating the ID is not found.
4. **Exit Application**:
   - Provide an option to exit the program.

Step 3: Display Menu

- Display a menu with options to:
  1. Add Employee
  2. View All Employees
  3. Search for an Employee
  4. Exit

Step 4: Handle User Input

- Use a loop to repeatedly display the menu and prompt the user for a choice.
- Call the appropriate function based on the user's selection.
- Ensure input validation for numeric values and string lengths.

Step 5: Terminate Program

- Exit the loop when the user selects the Exit option.

Code:

```
import java.util.Scanner;

class Employee {
    private String name;
    private int id;
    private String department;

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

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public String getDepartment() {
        return department;
    }

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

public class EmployeeManagementApp {
    private static Employee[] employees = new Employee[100]; // Array to store employee objects
    private static int employeeCount = 0; // To track the number of employees

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\nEmployee Management System");
            System.out.println("1. Add Employee");
            System.out.println("2. Display Employees");
            System.out.println("3. Search Employee by ID");
```

```
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    addEmployee(scanner);
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    searchEmployeeById(scanner);
                    break;
                case 4:
                    System.out.println("Exiting the application.");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 4);

        scanner.close();
    }

    private static void addEmployee(Scanner scanner) {
        if (employeeCount >= employees.length) {
            System.out.println("Employee list is full. Cannot add more employees.");
            return;
        }

        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 Department: ");
        String department = scanner.nextLine();

        employees[employeeCount] = new Employee(name, id, department);
        employeeCount++;
        System.out.println("Employee added successfully!");
    }

    private static void displayEmployees() {
        if (employeeCount == 0) {
            System.out.println("No employees to display.");
            return;
        }

        System.out.println("\nList of Employees:");
        for (int i = 0; i < employeeCount; i++) {
```

```
            System.out.println(employees[i]);
        }
    }

    private static void searchEmployeeById(Scanner scanner) {
        System.out.print("Enter Employee ID to search: ");
        int idToSearch = scanner.nextInt();

        boolean found = false;

        for (int i = 0; i < employeeCount; i++) {
            if (employees[i].getId() == idToSearch) {
                System.out.println("Employee found: " + employees[i]);
                found = true;
                break;
            }
        }

        if (!found) {
            System.out.println("Employee with ID " + idToSearch + " not found.");
        }
    }
}
```

Output:

```
Employee Management System
1. Add Employee
2. Display Employees
3. Search Employee by ID
4. Exit
Enter your choice: 1
Enter Employee ID: 11651
Enter Employee Name: JAYANTH
Enter Department: CSE
Employee added successfully!

Employee Management System
1. Add Employee
2. Display Employees
3. Search Employee by ID
4. Exit
Enter your choice: 2

List of Employees:
Employee ID: 11651, Name: JAYANTH, Department: CSE

Employee Management System
1. Add Employee
2. Display Employees
3. Search Employee by ID
4. Exit
Enter your choice: 3
Enter Employee ID to search: 11651
Employee found: Employee ID: 11651, Name: JAYANTH, Department: CSE

Employee Management System
1. Add Employee
2. Display Employees
3. Search Employee by ID
4. Exit
Enter your choice: 4
Exiting the application.
```

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.