EXPERIMENT 1

Student Name: Kritika Sharma **UID: 22BCS14943**
Branch: BE – CSE **Section/Group: BCS_KRG_IOT-3B**

Semester: 6th **Date: 15/01/2025**
Subject Name: PBLJ With Lab **Subject Code: 22CSH-359**

1. **Aim:**
   Create an application to save 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:**
   - **Initialize Resources:**
   - Create a Scanner object for user input.
   - Initialize an ArrayList to store Employee objects.
   - **Display Menu:**
   - Continuously show a menu with three options:
     - Add Employees
     - Display Employees
     - Exit
   - **Handle User Choice:**
   - Prompt the user for a choice and validate the input.
   - Based on the user's choice, perform one of the following actions:
     - **Add Employees:**
       - Ask for the number of employees to add.
       - Loop through and collect employee details (ID, Name, Department, Salary).

- Create an Employee object for each set of details and add it to the ArrayList.
- Display a confirmation message and list all employees.

- **Display Employees:**
  - If the ArrayList is empty, show a message indicating no employees are available.
  - Otherwise, iterate through the ArrayList and display each employee's details using the Employee class's displayEmployee method.
- **Exit Program:**
  - Terminate the application with a goodbye message.

- **Loop Until Exit:**
- Continue showing the menu and processing user choices until the user selects the "Exit" option.

4. **Code:**

```
import java.util.ArrayList; import
java.util.Scanner; public class
EmployeeManagement {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      ArrayList<Employee> employees = new ArrayList<>();
      while (true) {           displayMenu();
int choice = getUserChoice(scanner);
switch (choice) {              case 1: // Add
Employees
```

```
                System.out.print("Enter the number of employees to add: ");
int numEmployees = scanner.nextInt();                scanner.nextLine();
// Consume newline                for (int i = 0; i < numEmployees; i++)
{                    System.out.println("\nAdding Employee " + (i + 1) +
":");                    addEmployee(scanner, employees);
                }
                System.out.println("\nAll employees added successfully!");
displayEmployees(employees);
                break;
            case 2: // Display Employees
displayEmployees(employees);
                break;
case 3: // Exit
                System.out.println("Exiting program. Goodbye!");
                scanner.close();
return;                default:
                System.out.println("Invalid choice. Please try again.");
            }
        }
    }
    private static void displayMenu() {
        System.out.println("\nMenu:");
```

```
        System.out.println("1. Add Employees");
        System.out.println("2. Display Employees");
        System.out.println("3. Exit");
        System.out.print("Choose an option: ");
    }
    private static int getUserChoice(Scanner scanner) {
while (!scanner.hasNextInt()) {
            System.out.print("Invalid input. Please enter a number: ");
            scanner.next();
        }
        return scanner.nextInt();
    }
    private static void addEmployee(Scanner scanner, ArrayList<Employee>
employees) {
        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 Department: ");
        String department = scanner.nextLine();
System.out.print("Enter Employee Salary: ");        double salary =
```

```
scanner.nextDouble();         employees.add(new Employee(id,
name, department, salary));
    }
    private static void displayEmployees(ArrayList<Employee> employees) {
        if (employees.isEmpty()) {
            System.out.println("No employees to display.");
        } else {
            System.out.println("\nEmployee Details:");
for (Employee emp : employees) {
emp.displayEmployee();

            }
        }
    }
} class Employee
{
    private int id;     private String name;     private String department;
private double salary;     public Employee(int id, String name, String
department, double salary) {
        this.id = id;
        this.name = name;
this.department = department;
this.salary = salary;
```

```
    }

    public void displayEmployee() {

    System.out.println("ID: " + id + ", Name: " + name + ", Department: " +
department + ", Salary: " + salary);

}

}
```

5. Output:

```
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> javac EmployeeManagement.java
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> java EmployeeManagement.java

Menu:
1. Add Employees
2. Display Employees
3. Exit
Choose an option: 1
Enter the number of employees to add: 2

Adding Employee 1:
Enter Employee ID: 01
Enter Employee Name: Dhruv Sorout
Enter Employee Department: IT
Enter Employee Salary: 150000

Adding Employee 2:
Enter Employee ID: 02
Enter Employee Name: Om
Enter Employee Department: IT
Enter Employee Salary: 200000

All employees added successfully!

Employee Details:
ID: 1, Name: Dhruv Sorout, Department: IT, Salary: 150000.0
ID: 2, Name: Om, Department: IT, Salary: 200000.0
```

6. Learning Outcomes:

- Understand OOP concepts like classes, objects, and encapsulation.
- Use ArrayList for dynamic data storage and management.
- Handle user input using Scanner effectively.
- Practice core Java programming skills.
- Build interactive, menu-driven console applications.