

**Note:**

1. This assignment is designed to practice static fields, static initializers, and static methods.
  2. Understand the problem statement and use static and non-static wisely to solve the problem.
  3. Use constructors, proper getter/setter methods, and `toString()` wherever required.
1. Design and implement a class named `InstanceCounter` to track and count the number of instances created from this class.

Program:

**package** com.in.js;

```
class InstanceCounter {
    private static int Count = 0;

    public InstanceCounter() {
        Count++;
    }

    public static int getInstanceCount() {
        return InstanceCounter.Count;
    }
}

public class Main {
    public static void main(String[] args) {
        InstanceCounter i = new InstanceCounter();
        InstanceCounter i1 = new InstanceCounter();

        System.out.println(InstanceCounter.getInstanceCount());
    }
}
```

Output:

```
<terminated> Mail
2
```

2. Design and implement a class named `Logger` to manage logging messages for an application. The class should be implemented as a singleton to ensure that only one instance of the `Logger` exists throughout the application.

The class should include the following methods:

- `getInstance()` : Returns the unique instance of the `Logger` class.
- `log(String message)` : Adds a log message to the logger.
- `getLog()` : Returns the current log messages as a `String`.
- `clearLog()` : Clears all log messages.

Program:

Class file1: Logger

**package** com.Rahul.RS;

**import** java.util.ArrayList;

**public class** Logger {

**private static** Logger *reference* = **null**;

**private static int** *currentMessage* = 0;

**private** ArrayList<String> ar = **new** ArrayList<>();

// Singleton instance method

**public static** Logger getInstance() {

**if** (Logger.*reference* == **null**) {

            Logger.*reference* = **new** Logger();

        }

**return** *reference*;

    }

**public void** log(String message) {

        Logger.*currentMessage*++;

**this**.ar.add(message);

    }

// Method to retrieve the last log message

**public** String getLog() {

```
if (ar.size() > 0) {  
    return this.ar.get(Logger.currentMessage - 1);  
} else {  
    return "There is no Log Present";  
}  
}  
  
// Method to clear the logs  
  
public void clearLog() {  
    Logger.currentMessage = 0;  
    this.ar.clear();  
}  
  
// toString method to display all log messages  
  
@Override  
public String toString() {  
    if (ar.isEmpty()) {  
        return "No logs available.";  
    }  
    StringBuilder logs = new StringBuilder("Logs:\n");  
    for (String log : ar) {  
        logs.append(log).append("\n");  
    }  
    return logs.toString();  
}
```

```
}
```

Class file2 : program

```
package com.Rahul.RS;
```

```
public class program {
```

```
    public static void main(String[] args) {
```

```
        // Accessing the singleton Logger instance
```

```
        Logger logger = Logger.getInstance();
```

```
        // Adding log messages
```

```
        logger.log("Application started");
```

```
        logger.log("Performing some operations");
```

```
        // Display all log messages
```

```
        System.out.println(logger);
```

```
        // Clearing the log
```

```
        logger.clearLog();
```

```
        System.out.println("Log after clearing:");
```

```
        System.out.println(logger);
```

```
    }
```

```
}
```

Output:

```
<terminated> Program (2) [Java Application] C:\
Logs:
Application started
Performing some operations

Log after clearing:
No logs available.
```

- Design and implement a class named `Employee` to manage employee data for a company. The class should include fields to keep track of the total number of employees and the total salary expense, as well as individual employee details such as their ID, name, and salary.

The class should have methods to:

- Retrieve the total number of employees (`getTotalEmployees()`)
- Apply a percentage raise to the salary of all employees (`applyRaise(double percentage)`)
- Calculate the total salary expense, including any raises (`calculateTotalSalaryExpense()`)
- Update the salary of an individual employee (`updateSalary(double newSalary)`)

Understand the problem statement and use static and non-static fields and methods appropriately. Implement static and non-static initializers, constructors, getter and setter methods, and a `toString()` method to handle the initialization and representation of employee data.

Write a menu-driven program in the `main` method to test the functionalities.

Code

Class : `employee`

**package** com.in.Domain;

**public class** Employee {

// Static fields to track total number of employees and total salary expense

**private static int** *totalEmployees* = 0;

**private static double** *totalSalaryExpense* = 0.0;

// Non-static fields for individual employee details

**private int** id;

**private String** name;

**private double** salary;

// Static initializer to initialize total employees and salary expense

**static** {

System.*out*.println("Employee class loaded.");

```
}

// Non-static initializer to automatically update employee count and salary expense

{

    totalEmployees++;

    totalSalaryExpense += this.salary;

}

// Constructor to initialize individual employee details

public Employee(int id, String name, double salary) {

    this.id = id;

    this.name = name;

    this.salary = salary;

}

public int getId() {

    return id;

}

public void setId(int id) {

    this.id = id;

}

// Getter for employee name

public String getName() {

    return name;

}

public void setName(String name) {

    this.name = name;

}
```

```
public double getSalary() {  
    return salary;  
}  
  
public void updateSalary(double newSalary) {  
    totalSalaryExpense = totalSalaryExpense - this.salary + newSalary;  
    this.salary = newSalary;  
}  
  
public static int getTotalEmployees() {  
    return totalEmployees;  
}  
  
public static void applyRaise(double percentage, Employee[] employees) {  
    for (Employee emp : employees) {  
        double raiseAmount = emp.salary * (percentage / 100);  
        emp.updateSalary(emp.salary + raiseAmount);  
    }  
}  
  
public static double calculateTotalSalaryExpense() {  
    return totalSalaryExpense;  
}  
  
// Overriding toString method to print individual employee details  
@Override  
public String toString() {  
    return "Employee[ID=" + id + ", Name=" + name + ", Salary=" + salary + "];"  
}  
}
```

```

Class 2: program;
package com.in.Domain;
import java.util.*;
//import java.util.ArrayList;
//import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<Employee> employees = new ArrayList<>();

        // Initial employees for exploring more in details
        employees.add(new Employee(1, "Rahul", 50000));
        employees.add(new Employee(2, "Rohit", 60000));
        employees.add(new Employee(3, "Siddh", 70000));

        boolean exit = false;

        // Menu-driven program
        while (!exit) {
            System.out.println("\n Select your choice :");
            System.out.println("1. View Total Employees");
            System.out.println("2. View Total Salary Expense");
            System.out.println("3. Apply Raise to All Employees");
            System.out.println("4. Update Individual Employee Salary");
            System.out.println("5. View Employee Details");
            System.out.println("6. Add New Employee");
            System.out.println("7. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.println("Total Employees: " + Employee.getTotalEmployees());
                    break;
                case 2:
                    System.out.println("Total Salary Expense: " +
Employee.calculateTotalSalaryExpense());
                    break;
                case 3:
                    System.out.print("Enter raise percentage: ");
                    double percentage = scanner.nextDouble();
                    Employee.applyRaise(percentage, employees.toArray(new Employee[0]));
                    System.out.println("Applied " + percentage + "% raise to all employees.");

```



```

        break;
    case 4:
        System.out.print("Enter employee ID to update salary: ");
        int id = scanner.nextInt();
        System.out.print("Enter new salary: ");
        double newSalary = scanner.nextDouble();
        for (Employee emp : employees) {
            if (emp.getId() == id) {
                emp.updateSalary(newSalary);
                System.out.println("Updated salary for employee ID " + id);
                break;
            }
        }
        break;
    case 5:
        for (Employee emp : employees) {
            System.out.println(emp);
        }
        break;
    case 6:
        System.out.print("Enter new employee ID: ");
        int newId = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter new employee name: ");
        String newName = scanner.nextLine();
        System.out.print("Enter new employee salary: ");
        double newSalaryForNewEmp = scanner.nextDouble();
        Employee newEmployee = new Employee(newId, newName,
            newSalaryForNewEmp);
        employees.add(newEmployee);
        System.out.println("Added new employee: " + newEmployee);
        break;
    case 7:
        exit = true;
        break;
    default:
        System.out.println("Invalid option. Please choose again.");
    }
}

scanner.close();
}
}

```

Output:

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
Choose an option: 1
Total Employees: 3
```

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
Choose an option: 2
Total Salary Expense: 0.0
```

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
Choose an option: 3
Enter raise percentage: 20
Applied 20.0% raise to all employees.
```

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
Choose an option: 4
Enter employee ID to update salary: 1
Enter new salary: 65000
Updated salary for employee ID 1
```

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
Choose an option: 5
Employee[ID=1, Name=Rahul, Salary=65000.0]
Employee[ID=2, Name=Rohit, Salary=72000.0]
Employee[ID=3, Name=Siddh, Salary=84000.0]
```

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
```

```
6. ADD NEW EMPLOYEE
7. Exit
Choose an option: 6
Enter new employee ID: 4
Enter new employee name: Rishab
Enter new employee salary: 45000
Added new employee: Employee[ID=4, Name=Rishab, Salary=45000.0]
```

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
Choose an option: 5
Employee[ID=1, Name=Rahul, Salary=65000.0]
Employee[ID=2, Name=Rohit, Salary=72000.0]
Employee[ID=3, Name=Siddh, Salary=84000.0]
Employee[ID=4, Name=Rishab, Salary=45000.0]
```

```
Select your choice :
1. View Total Employees
2. View Total Salary Expense
3. Apply Raise to All Employees
4. Update Individual Employee Salary
5. View Employee Details
6. Add New Employee
7. Exit
Choose an option: 7
```

sandeepkulange@gmail.com