

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.

Code:

```
package org.example.instancecounter;
```

```
public class Main {
```

```
    public static void main( String[] args ) {
        InstanceCounter i1 = new InstanceCounter();
        InstanceCounter i2 = new InstanceCounter();
        InstanceCounter i3 = new InstanceCounter();

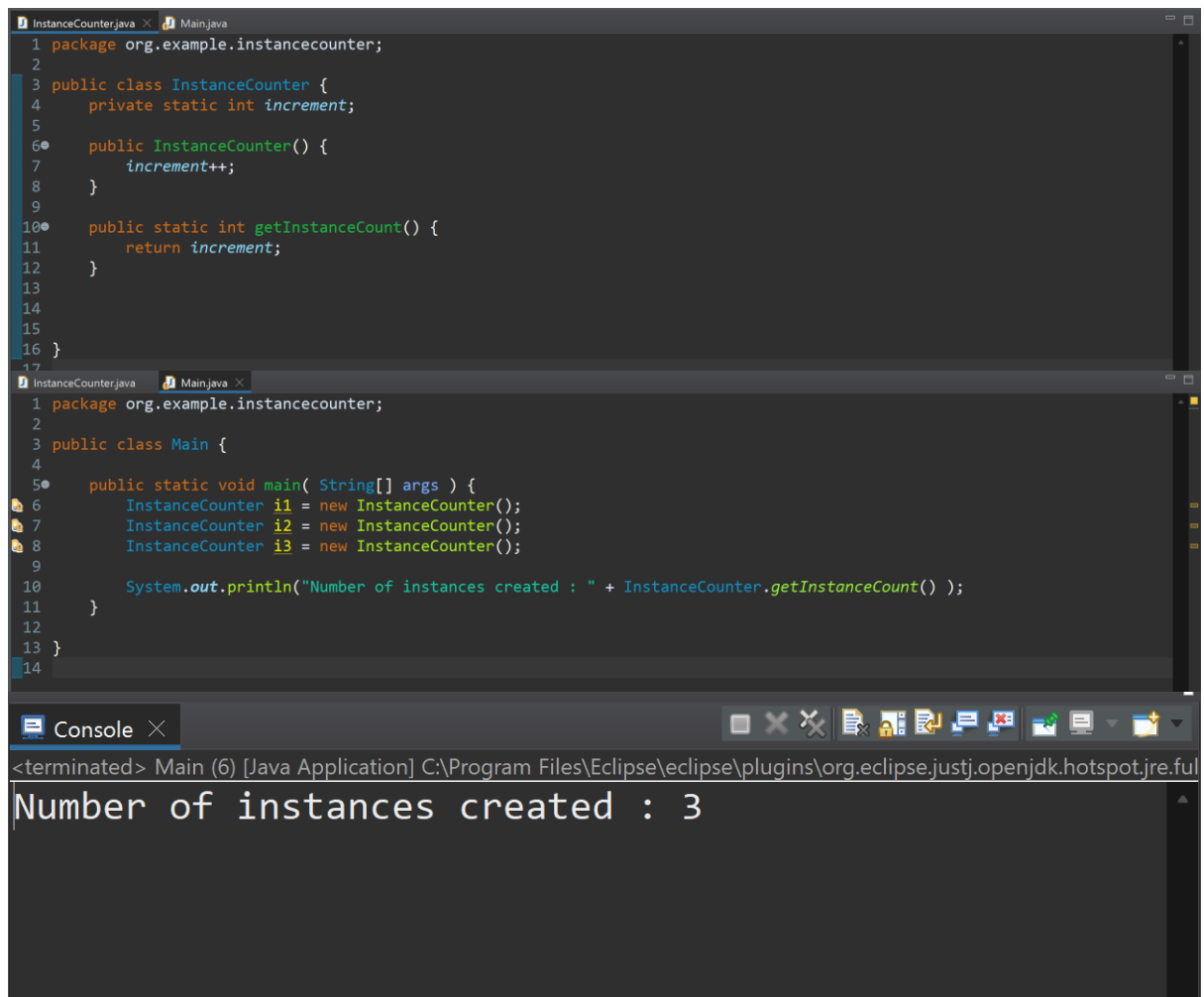
        System.out.println("Number of instances created : " +
InstanceCounter.getInstanceCount() );
    }
```

```
}
```

Class:

```
package org.example.instancecounter;
```

```
public class InstanceCounter {
    private static int increment;
    public InstanceCounter() {
        increment++;
    }
    public static int getInstanceCount() {
        return increment;
    }
}
```



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.

Code:

Main:

```
package org.example.logger;
```

```
public class Main {
```

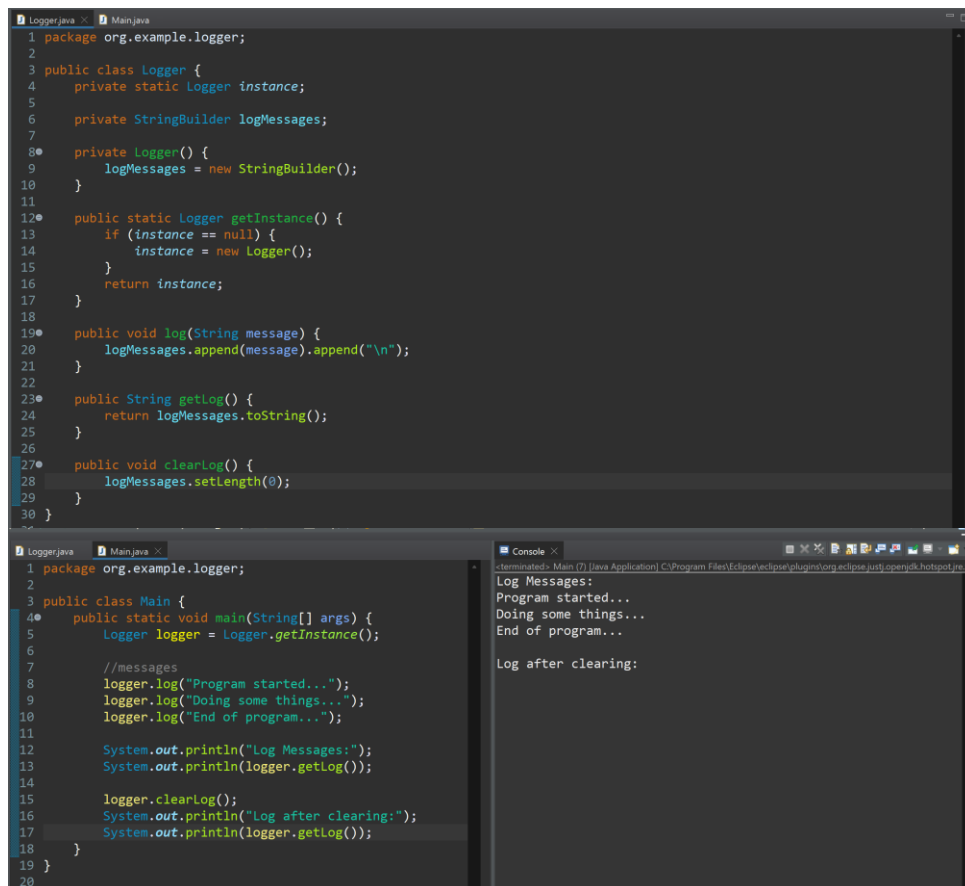
```
public static void main(String[] args) {
    Logger logger = Logger.getInstance();
```

```
//messages
logger.log("Program started...");
logger.log("Doing some things...");
logger.log("End of program...");

System.out.println("Log Messages:");
System.out.println(logger.getLog());

logger.clearLog();
System.out.println("Log after clearing:");
System.out.println(logger.getLog());
}
}
Class Logger:
package org.example.logger;
public class Logger {
    private static Logger instance;
    private StringBuilder logMessages;
    private Logger() {
        logMessages = new StringBuilder();
    }

    public static Logger getInstance() {
        if (instance == null) {
            instance = new Logger();
        }
        return instance;
    }
    public void log(String message) {
        logMessages.append(message).append("\n");
    }
    public String getLog() {
        return logMessages.toString();
    }
    public void clearLog() {
        logMessages.setLength(0);
    }
}
```



```

1 package org.example.logger;
2
3 public class Logger {
4     private static Logger instance;
5
6     private StringBuilder logMessages;
7
8     private Logger() {
9         logMessages = new StringBuilder();
10    }
11
12    public static Logger getInstance() {
13        if (instance == null) {
14            instance = new Logger();
15        }
16        return instance;
17    }
18
19    public void log(String message) {
20        logMessages.append(message).append("\n");
21    }
22
23    public String getLog() {
24        return logMessages.toString();
25    }
26
27    public void clearLog() {
28        logMessages.setLength(0);
29    }
30 }

```

```

1 package org.example.logger;
2
3 public class Main {
4     public static void main(String[] args) {
5         Logger logger = Logger.getInstance();
6
7         //messages
8         logger.log("Program started...");
9         logger.log("Doing some things...");
10        logger.log("End of program...");
11
12        System.out.println("Log Messages:");
13        System.out.println(logger.getLog());
14
15        logger.clearLog();
16        System.out.println("Log after clearing:");
17        System.out.println(logger.getLog());
18    }
19 }
20

```

```

Log Messages:
Program started...
Doing some things...
End of program...

Log after clearing:

```

- 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:

Employee class:

```
package org.example.employee;
```

```
public class Employee {
```

```
    private static int totalEmployees = 0;
    private static double totalSalaryExpense = 0.0;
    private int id;
    private String name;
    private double salary;
```

```
    static {
        totalEmployees = 0;
        totalSalaryExpense = 0.0;
    }
```

```
    {
        this.id = 0;
        this.name = "";
        this.salary = 0.0;
    }
```

```
    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
        totalEmployees++;
        totalSalaryExpense += salary;
    }
```

```
    public int getId() {
        return id;
    }
```

```
    public void setId(int id) {
        this.id = id;
    }
```

```
    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }
```

```
    public double getSalary() {
        return salary;
    }
```

```
//Update the salary of an individual employee (updateSalary(double newSalary))
public void updateSalary(double newSalary) {
    totalSalaryExpense -= this.salary;
    this.salary = newSalary;
    totalSalaryExpense += newSalary;
}
//Retrieve the total number of employees (getTotalEmployees())
public static int getTotalEmployees() {
    return totalEmployees;
}
//Calculate the total salary expense, including any raises
(calculateTotalSalaryExpense())
public static double calculateTotalSalaryExpense() {
    return totalSalaryExpense;
}
//Apply a percentage raise to the salary of all employees (applyRaise(double
percentage))
public static void applyRaise(double percentage, Employee[] employees) {
    for (Employee employee : employees) {
        double raiseAmount = employee.salary * (percentage / 100);
        employee.updateSalary(employee.salary + raiseAmount);
    }
}
@Override
public String toString() {
    return "Employee [ID: " + id + ", Name: " + name + ", Salary: ₹" + salary + "];"
}
}
```

Main class:

```
package org.example.employee;
import java.util.Scanner;

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

        Employee[] employees = {
            new Employee(101, "John", 50000),
            new Employee(102, "Shyam", 60000),
            new Employee(103, "Priti", 55000)
        };
        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Display all employees");
            System.out.println("2. Apply a raise to all employees");
        }
    }
}
```

```

System.out.println("3. Display total number of employees");
System.out.println("4. Display total salary expense");
System.out.println("5. Update an employee's salary");
System.out.println("6. Exit");

int choice = scanner.nextInt();
switch (choice) {
    case 1:
        for (Employee emp : employees) {
            System.out.println(emp);
        }
        break;

    case 2:
        System.out.println("Enter percentage raise: ");
        double percentage = scanner.nextDouble();
        Employee.applyRaise(percentage, employees);
        System.out.println("Raise applied to all employees.");
        break;

    case 3:
        System.out.println("Total employees: " + Employee.getTotalEmployees());
        break;

    case 4:
        System.out.println("Total salary expense: ₹" +
Employee.calculateTotalSalaryExpense());
        break;

    case 5:
        System.out.println("Enter employee ID to update salary: ");
        int id = scanner.nextInt();
        System.out.println("Enter new salary: ");
        double newSalary = scanner.nextDouble();
        for (Employee emp : employees) {
            if (emp.getId() == id) {
                emp.updateSalary(newSalary);
                System.out.println("Salary updated for employee ID: " + id);
                break;
            }
        }
        break;

    case 6:
        System.out.println("Exiting program.");
        System.exit(0);

    default:
        System.out.println("Invalid choice, try again.");
}
}
}
}
}

```

ASSIGNMENT NO.5

```
Employee.java x Main.java Console
1 package org.example.employee;
2
3 public class Employee {
4
5     private static int totalEmployees = 0;
6     private static double totalSalaryExpense = 0.0;
7     private int id;
8     private String name;
9     private double salary;
10
11     static {
12         totalEmployees = 0;
13         totalSalaryExpense = 0.0;
14     }
15     {
16         this.id = 0;
17         this.name = "";
18         this.salary = 0.0;
19     }
20     public Employee(int id, String name, double salary) {
21         this.id = id;
22         this.name = name;
23         this.salary = salary;
24         totalEmployees++;
25         totalSalaryExpense += salary;
26     }
27
28     public int getId() {
29         return id;
30     }
31
32     public void setId(int id) {
33         this.id = id;
34     }
35
36     public String getName() {
37         return name;
38     }
39
40     public void setName(String name) {
41         this.name = name;
42     }
43
44     public double getSalary() {
45         return salary;
46     }
47     //Update the salary of an individual employee (updateSalary(double newSalary))
48     public void updateSalary(double newSalary) {
49         totalSalaryExpense -= this.salary;
50         this.salary = newSalary;
51         totalSalaryExpense += newSalary;
52     }
53     //Retrieve the total number of employees (getTotalEmployees())
54     public static int getTotalEmployees() {
55         return totalEmployees;
56     }
57     //Calculate the total salary expense, including any raises (calculateTotalSalaryExpense())
58     public static double calculateTotalSalaryExpense() {
59         return totalSalaryExpense;
60     }
61     //Apply a percentage raise to the salary of all employees (applyRaise(double percentage))
62     public static void applyRaise(double percentage, Employee[] employees) {
63         for (Employee employee : employees) {
64             double raiseAmount = employee.salary * (percentage / 100);
65             employee.updateSalary(employee.salary + raiseAmount);
66         }
67     }
68     @Override
69     public String toString() {
70         return "Employee [ID: " + id + ", Name: " + name + ", Salary: ₹" + salary + "];";
71     }
72 }
73 }
74
```


ASSIGNMENT NO.5

```

1 package org.example.employee;
2
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8
9         Employee[] employees = {
10             new Employee(101, "John", 50000),
11             new Employee(102, "Shyam", 60000),
12             new Employee(103, "Priti", 55000)
13         };
14         while (true) {
15             System.out.println("\nMenu:");
16             System.out.println("1. Display all employees");
17             System.out.println("2. Apply a raise to all employees");
18             System.out.println("3. Display total number of employees");
19             System.out.println("4. Display total salary expense");
20             System.out.println("5. Update an employee's salary");
21             System.out.println("6. Exit");
22
23             int choice = scanner.nextInt();
24             switch (choice) {
25                 case 1:
26                     for (Employee emp : employees) {
27                         System.out.println(emp);
28                     }
29                     break;
30
31                 case 2:
32                     System.out.println("Enter percentage raise: ");
33                     double percentage = scanner.nextDouble();
34                     Employee.applyRaise(percentage, employees);
35                     System.out.println("Raise applied to all employees.");
36                     break;
37
38                 case 3:
39                     System.out.println("Total employees: " + Employee.getTotalEmployees());
40                     break;
41
42                 case 4:
43                     System.out.println("Total salary expense: ₹" + Employee.calculateTotalSalaryExpense());
44                     break;
45
46                 case 5:
47                     System.out.println("Enter employee ID to update salary: ");
48                     int id = scanner.nextInt();
49                     System.out.println("Enter new salary: ");
50                     double newSalary = scanner.nextDouble();
51                     for (Employee emp : employees) {
52                         if (emp.getId() == id) {
53                             emp.updateSalary(newSalary);
54                             System.out.println("Salary updated for employee ID: " + id);
55                             break;
56                         }
57                     }
58                     break;
59
60                 case 6:
61                     System.out.println("Exiting program.");
62                     System.exit(0);
63
64                 default:
65                     System.out.println("Invalid choice, try again.");
66             }
67         }
68     }
69 }
70
71
72 }
73

```

ASSIGNMENT NO.5

```
Menu:
1. Display all employees
2. Apply a raise to all employees
3. Display total number of employees
4. Display total salary expense
5. Update an employee's salary
6. Exit

1
Employee [ID: 101, Name: John, Salary: ₹50000.0]
Employee [ID: 102, Name: Shyam, Salary: ₹60000.0]
Employee [ID: 103, Name: Priti, Salary: ₹55000.0]

Menu:
1. Display all employees
2. Apply a raise to all employees
3. Display total number of employees
4. Display total salary expense
5. Update an employee's salary
6. Exit

2
Enter percentage raise:
10
Raise applied to all employees.

Menu:
1. Display all employees
2. Apply a raise to all employees
3. Display total number of employees
4. Display total salary expense
5. Update an employee's salary
6. Exit

3
Total employees: 3

Menu:
1. Display all employees
2. Apply a raise to all employees
3. Display total number of employees
4. Display total salary expense
5. Update an employee's salary
6. Exit

4
Total salary expense: ₹181500.0

Menu:
1. Display all employees
2. Apply a raise to all employees
3. Display total number of employees
4. Display total salary expense
5. Update an employee's salary
6. Exit

5
Enter employee ID to update salary:
101
Enter new salary:
65000
Salary updated for employee ID: 101

Menu:
1. Display all employees
2. Apply a raise to all employees
3. Display total number of employees
4. Display total salary expense
5. Update an employee's salary
6. Exit

1
Employee [ID: 101, Name: John, Salary: ₹65000.0]
Employee [ID: 102, Name: Shyam, Salary: ₹60000.0]
Employee [ID: 103, Name: Priti, Salary: ₹55000.0]
```