ASSIGNMENT NO.6

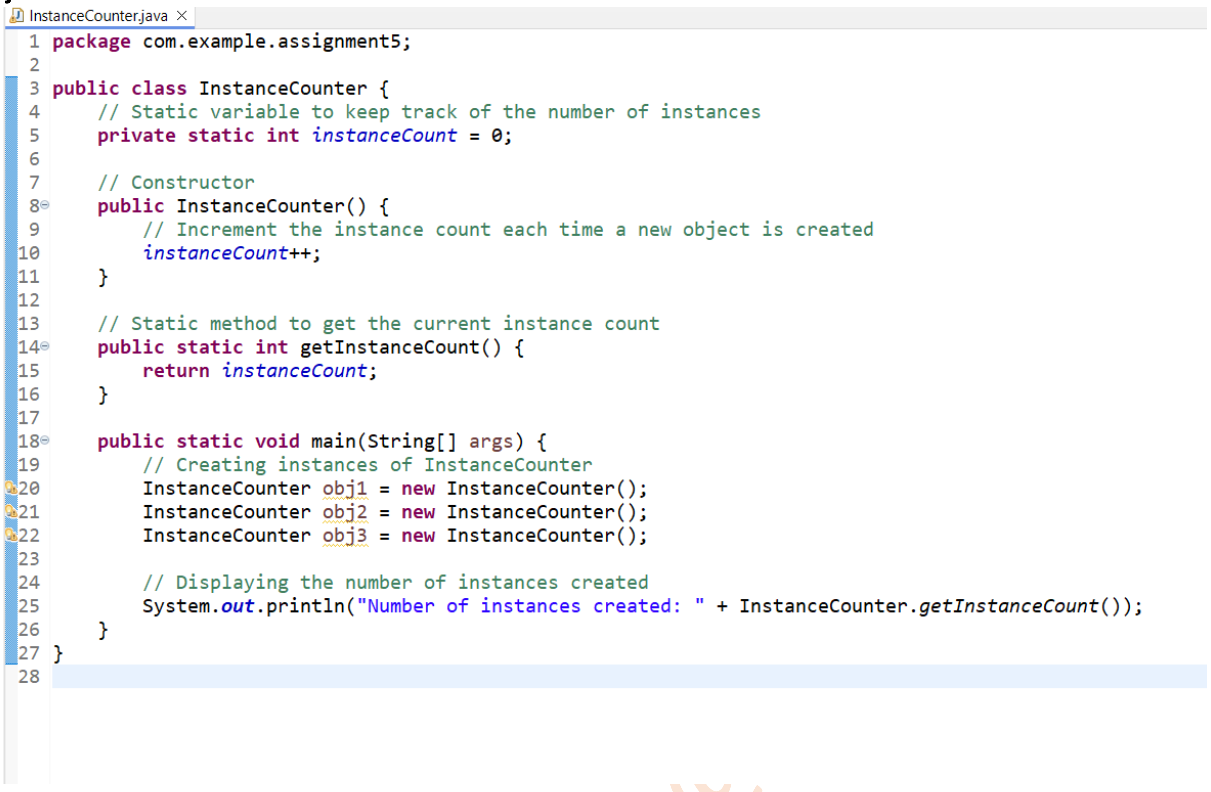
Note:

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.
   Ans:

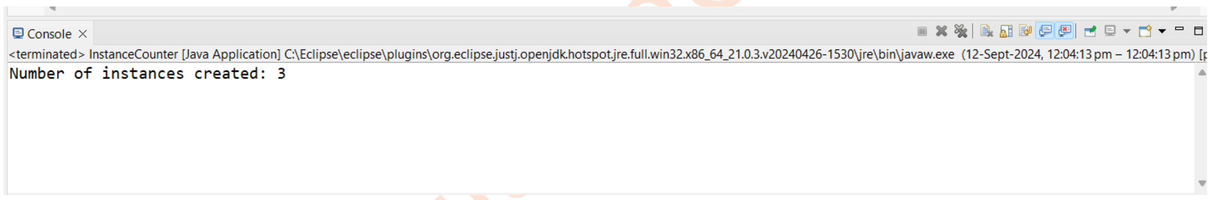
```
package com.example.assignment5;

public class InstanceCounter {
   // Static variable to keep track of the number of instances
   private static int instanceCount = 0;

   // Constructor
   public InstanceCounter() {
      // Increment the instance count each time a new object is created
      instanceCount++;
   }

   // Static method to get the current instance count
   public static int getInstanceCount() {
      return instanceCount;
   }

   public static void main(String[] args) {
      // Creating instances of InstanceCounter
      InstanceCounter obj1 = new InstanceCounter();
      InstanceCounter obj2 = new InstanceCounter();
      InstanceCounter obj3 = new InstanceCounter();

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

}



InstanceCounter.java

```java
package com.example.assignment5;

public class InstanceCounter {
    // Static variable to keep track of the number of instances
    private static int instanceCount = 0;

    // Constructor
    public InstanceCounter() {
        // Increment the instance count each time a new object is created
        instanceCount++;
    }

    // Static method to get the current instance count
    public static int getInstanceCount() {
        return instanceCount;
    }

    public static void main(String[] args) {
        // Creating instances of InstanceCounter
        InstanceCounter obj1 = new InstanceCounter();
        InstanceCounter obj2 = new InstanceCounter();
        InstanceCounter obj3 = new InstanceCounter();

        // Displaying the number of instances created
        System.out.println("Number of instances created: " + InstanceCounter.getInstanceCount());
    }
}

```



Console

```
<terminated> InstanceCounter [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe  (12-Sept-2024, 12:04:13 pm – 12:04:13 pm) [
Number of instances created: 3
```

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:

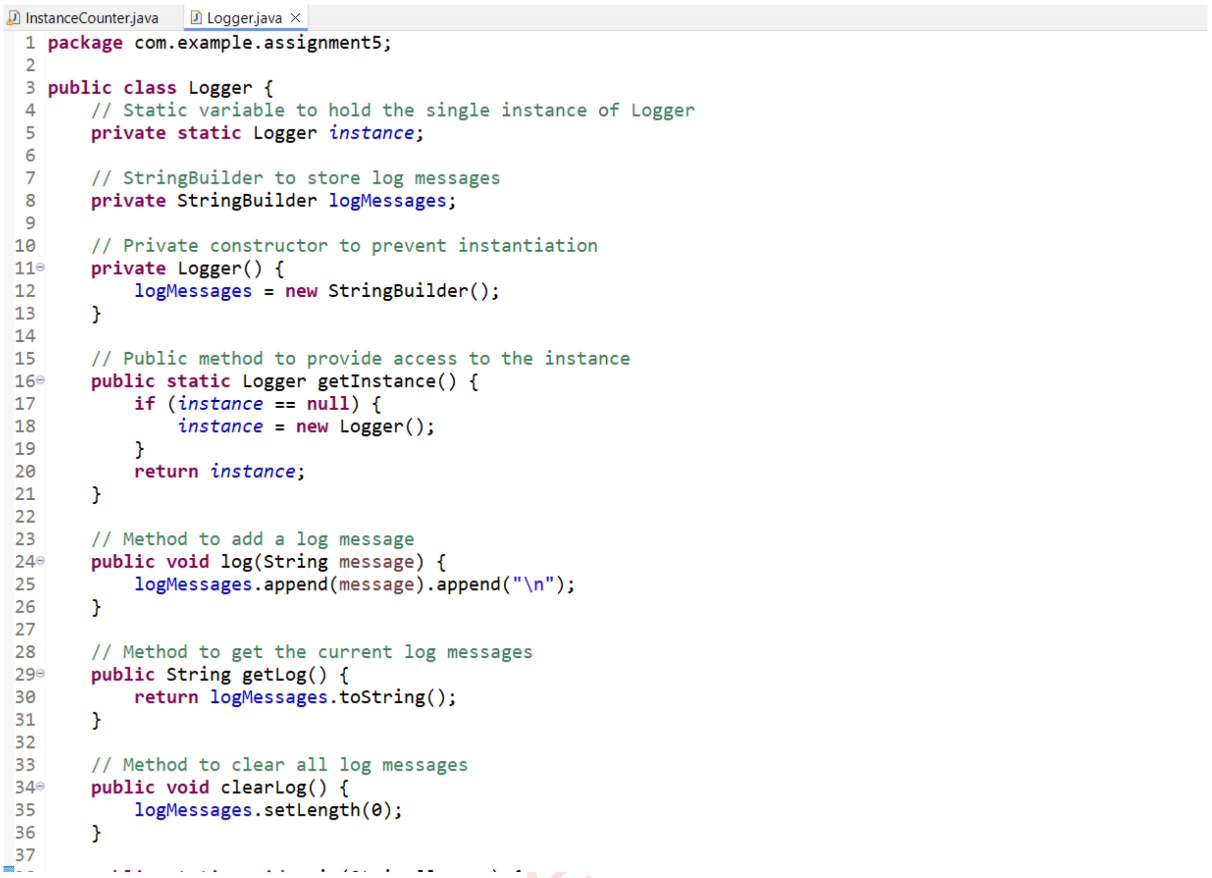
- **`getInstance()`**: Returns the unique instance of the `Logger` class.
- **`log(String message)`**: Adds a log message to the logger.
- **`getLog()`**: Returns the current log messages as a `String`.

`clearLog()`: Clears all log messages.
Ans:
package com.example.assignment5;

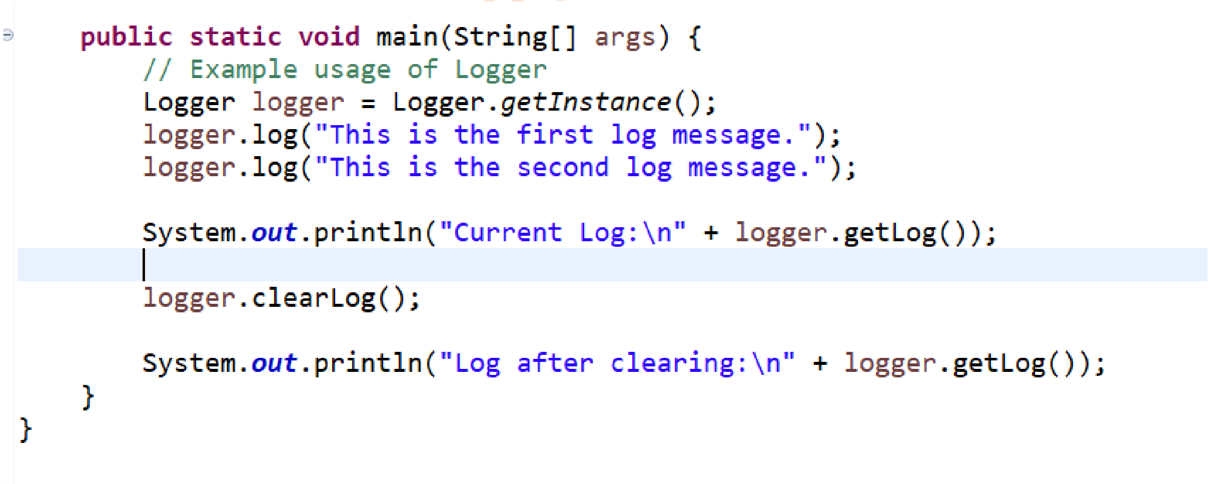
public class Logger {
        // Static variable to hold the single instance of Logger
  private static Logger *instance*;

```
    // StringBuilder to store log messages
    private StringBuilder logMessages;

    // Private constructor to prevent instantiation
    private Logger() {
        logMessages = new StringBuilder();
    }

    // Public method to provide access to the instance
    public static Logger getInstance() {
        if (instance == null) {
            instance = new Logger();
        }
        return instance;
    }

    // Method to add a log message
    public void log(String message) {
        logMessages.append(message).append("\n");
    }

    // Method to get the current log messages
    public String getLog() {
        return logMessages.toString();
    }

    // Method to clear all log messages
    public void clearLog() {
        logMessages.setLength(0);
    }

    public static void main(String[] args) {
        // Example usage of Logger
        Logger logger = Logger.getInstance();
        logger.log("This is the first log message.");
        logger.log("This is the second log message.");

        System.out.println("Current Log:\n" + logger.getLog());

        logger.clearLog();

        System.out.println("Log after clearing:\n" + logger.getLog());
    }
}
```

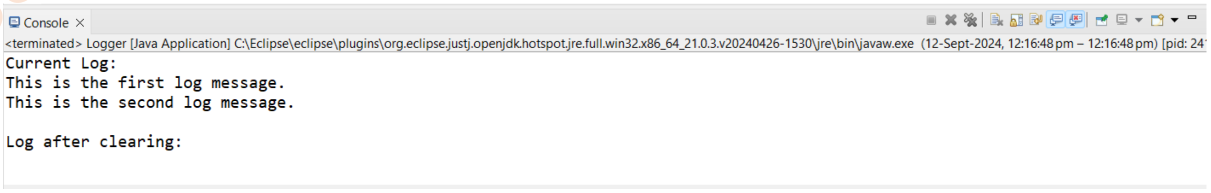
ASSIGNMENT NO.6



```java
package com.example.assignment5;

public class Logger {
    // Static variable to hold the single instance of Logger
    private static Logger instance;

    // StringBuilder to store log messages
    private StringBuilder logMessages;

    // Private constructor to prevent instantiation
    private Logger() {
        logMessages = new StringBuilder();
    }

    // Public method to provide access to the instance
    public static Logger getInstance() {
        if (instance == null) {
            instance = new Logger();
        }
        return instance;
    }

    // Method to add a log message
    public void log(String message) {
        logMessages.append(message).append("\n");
    }

    // Method to get the current log messages
    public String getLog() {
        return logMessages.toString();
    }

    // Method to clear all log messages
    public void clearLog() {
        logMessages.setLength(0);
    }

```



```java
    public static void main(String[] args) {
        // Example usage of Logger
        Logger logger = Logger.getInstance();
        logger.log("This is the first log message.");
        logger.log("This is the second log message.");

        System.out.println("Current Log:\n" + logger.getLog());

        logger.clearLog();

        System.out.println("Log after clearing:\n" + logger.getLog());
    }
}
```



```
<terminated> Logger [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (12-Sept-2024, 12:16:48 pm – 12:16:48 pm) [pid: 24
Current Log:
This is the first log message.
This is the second log message.

Log after clearing:
```

3. 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)`)

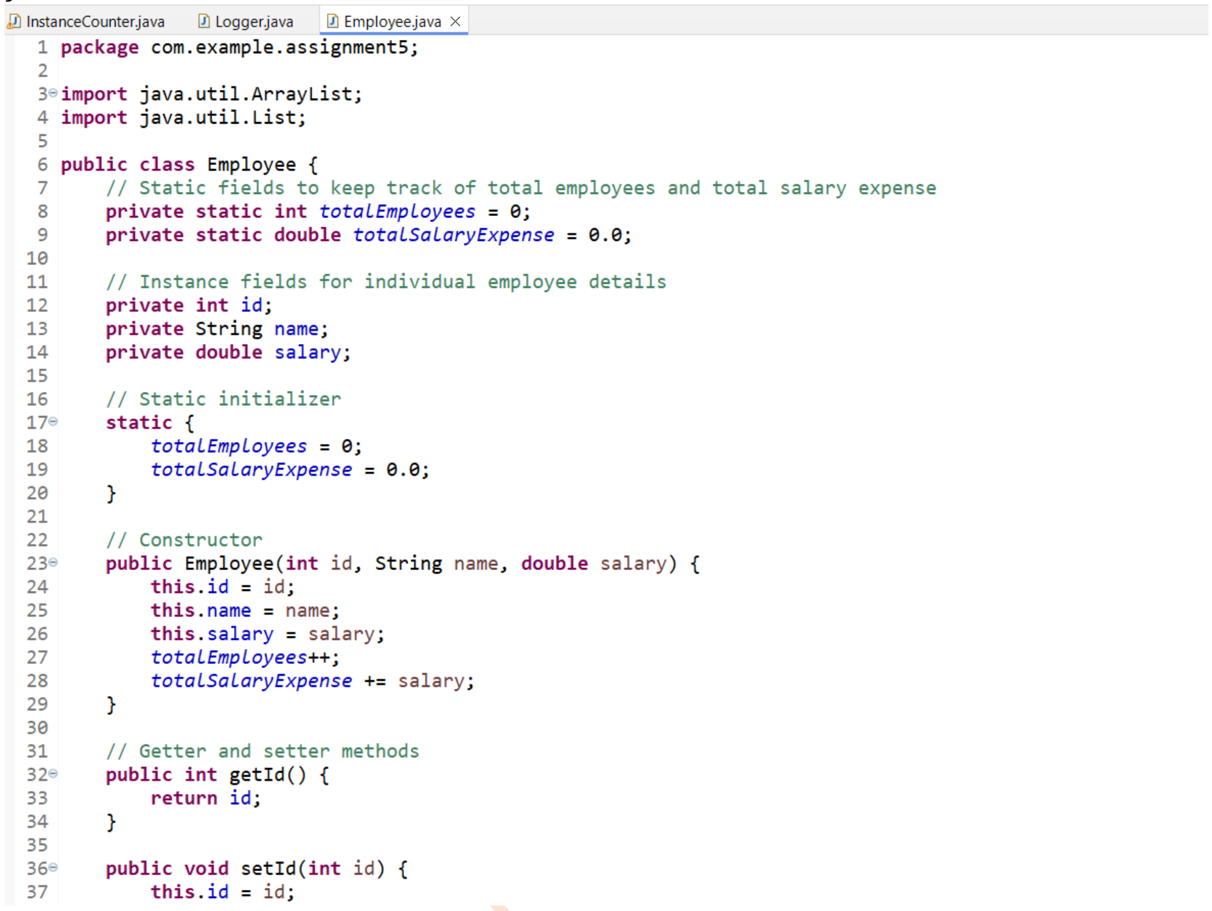
Understand the problem statement and use static and non-static fields and methods appropriately. Implement static and non-static initializers, constructors, getter and setter methods, and a `toString()` method to handle the initialization and representation of employee data.

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

Ans:

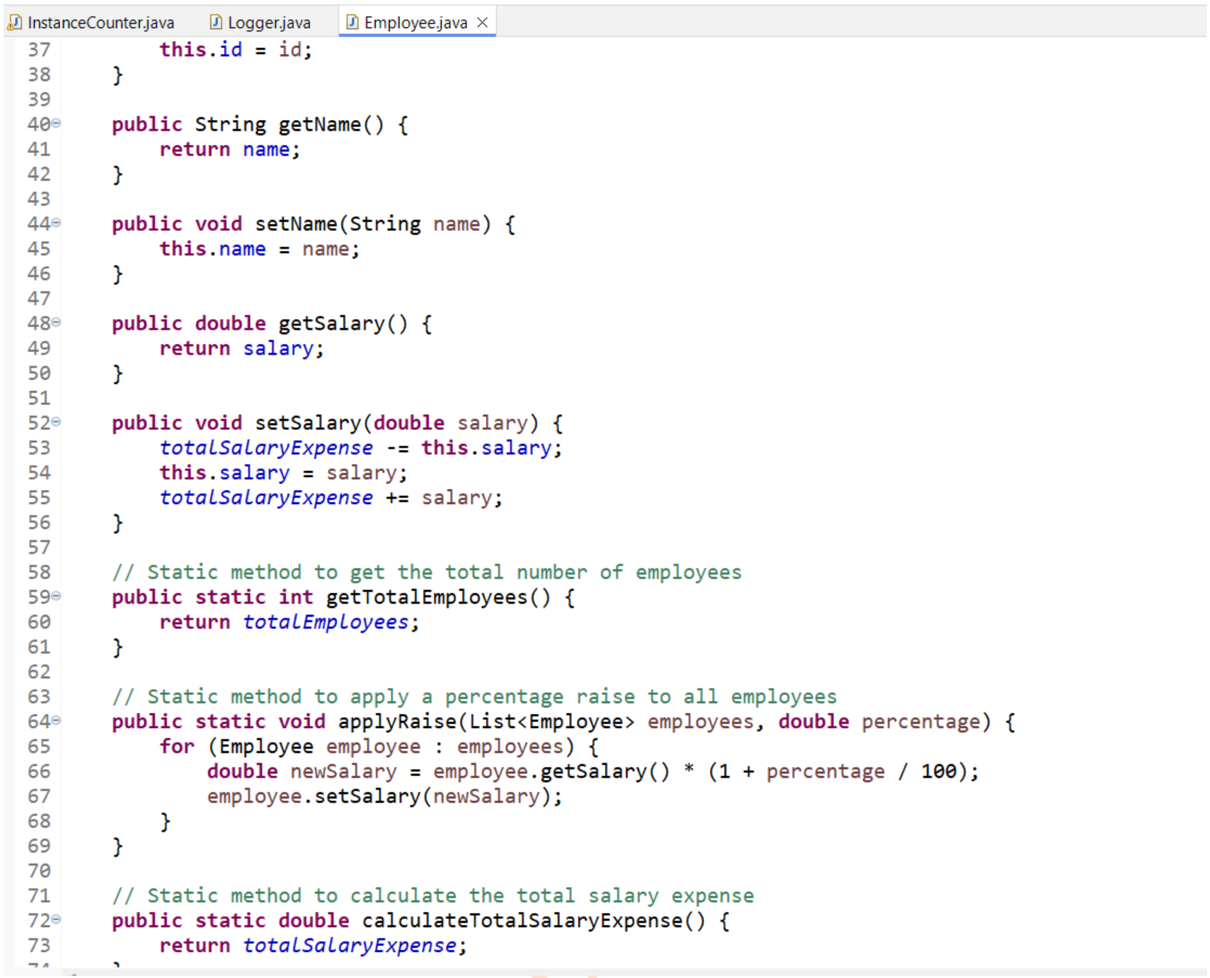
```
package com.example.assignment5;

import java.util.ArrayList;
import java.util.List;

public class Employee {
   // Static fields to keep track of total employees and total salary expense
   private static int totalEmployees = 0;
   private static double totalSalaryExpense = 0.0;

   // Instance fields for individual employee details
   private int id;
   private String name;
   private double salary;

   // Static initializer
   static {
      totalEmployees = 0;
      totalSalaryExpense = 0.0;
   }

   // Constructor
   public Employee(int id, String name, double salary) {
      this.id = id;
      this.name = name;
      this.salary = salary;
```

```
    totalEmployees++;
    totalSalaryExpense += salary;
}

// Getter and setter methods
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;
}

public void setSalary(double salary) {
    totalSalaryExpense -= this.salary;
    this.salary = salary;
    totalSalaryExpense += salary;
}

// Static method to get the total number of employees
public static int getTotalEmployees() {
    return totalEmployees;
}

// Static method to apply a percentage raise to all employees
public static void applyRaise(List<Employee> employees, double percentage) {
    for (Employee employee : employees) {
        double newSalary = employee.getSalary() * (1 + percentage / 100);
        employee.setSalary(newSalary);
    }
}

// Static method to calculate the total salary expense
public static double calculateTotalSalaryExpense() {
    return totalSalaryExpense;
    }

    // toString method to represent employee data
    @Override
    public String toString() {
        return "Employee [ID=" + id + ", Name=" + name + ", Salary=" + salary + "]";
    }

    // Main method to test the functionalities
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee(1, "Alice", 50000));
        employees.add(new Employee(2, "Bob", 60000));
        employees.add(new Employee(3, "Charlie", 70000));

        System.out.println("Total Employees: " + Employee.getTotalEmployees());
        System.out.println("Total Salary Expense: " +
Employee.calculateTotalSalaryExpense());

        System.out.println("\nApplying a 10% raise to all employees...");
        Employee.applyRaise(employees, 10);

        System.out.println("Total Salary Expense after raise: " +
Employee.calculateTotalSalaryExpense());

        System.out.println("\nEmployee Details:");
        for (Employee employee : employees) {
            System.out.println(employee);
        }

        System.out.println("\nUpdating salary of employee with ID 2 to 65000...");
        employees.get(1).setSalary(65000);

        System.out.println("Total Salary Expense after update: " +
Employee.calculateTotalSalaryExpense());

        System.out.println("\nEmployee Details after update:");
        for (Employee employee : employees) {
            System.out.println(employee);
        }
    }
}
```

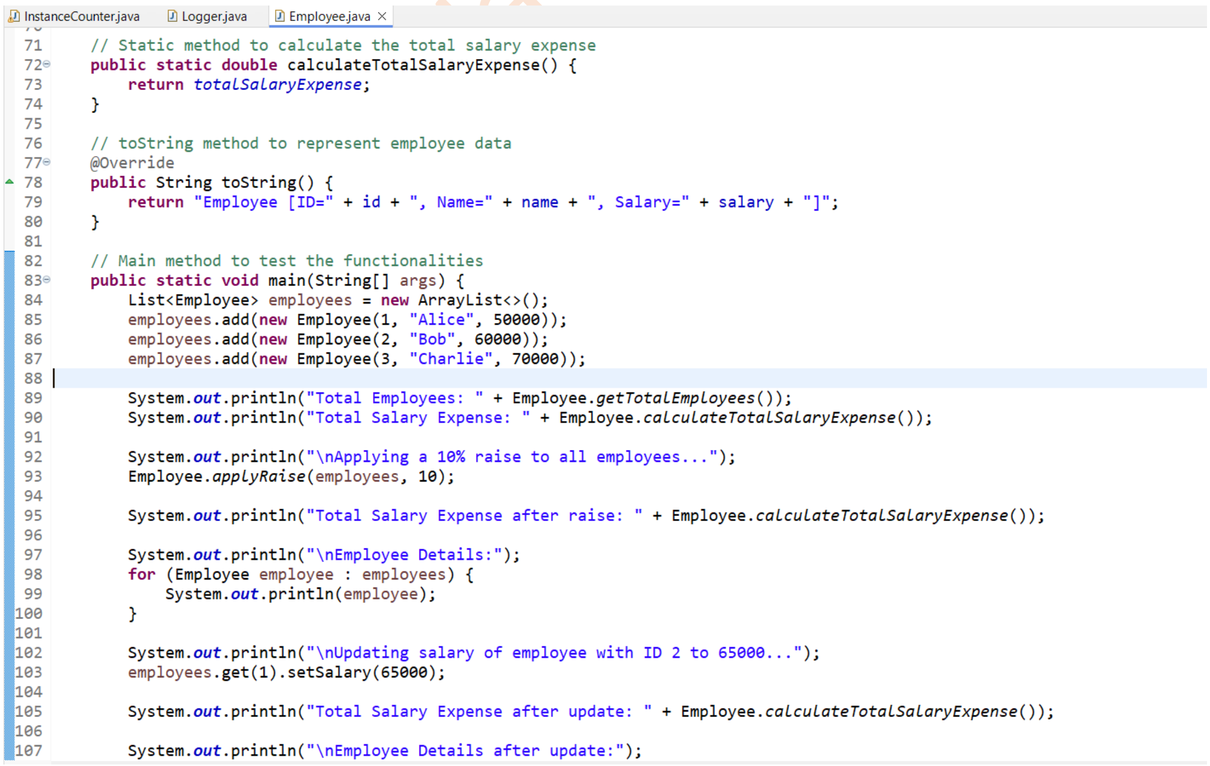


```java
package com.example.assignment5;

import java.util.ArrayList;
import java.util.List;

public class Employee {
    // Static fields to keep track of total employees and total salary expense
    private static int totalEmployees = 0;
    private static double totalSalaryExpense = 0.0;

    // Instance fields for individual employee details
    private int id;
    private String name;
    private double salary;

    // Static initializer
    static {
        totalEmployees = 0;
        totalSalaryExpense = 0.0;
    }

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

    // Getter and setter methods
    public int getId() {
        return id;
    }

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

ASSIGNMENT NO.6

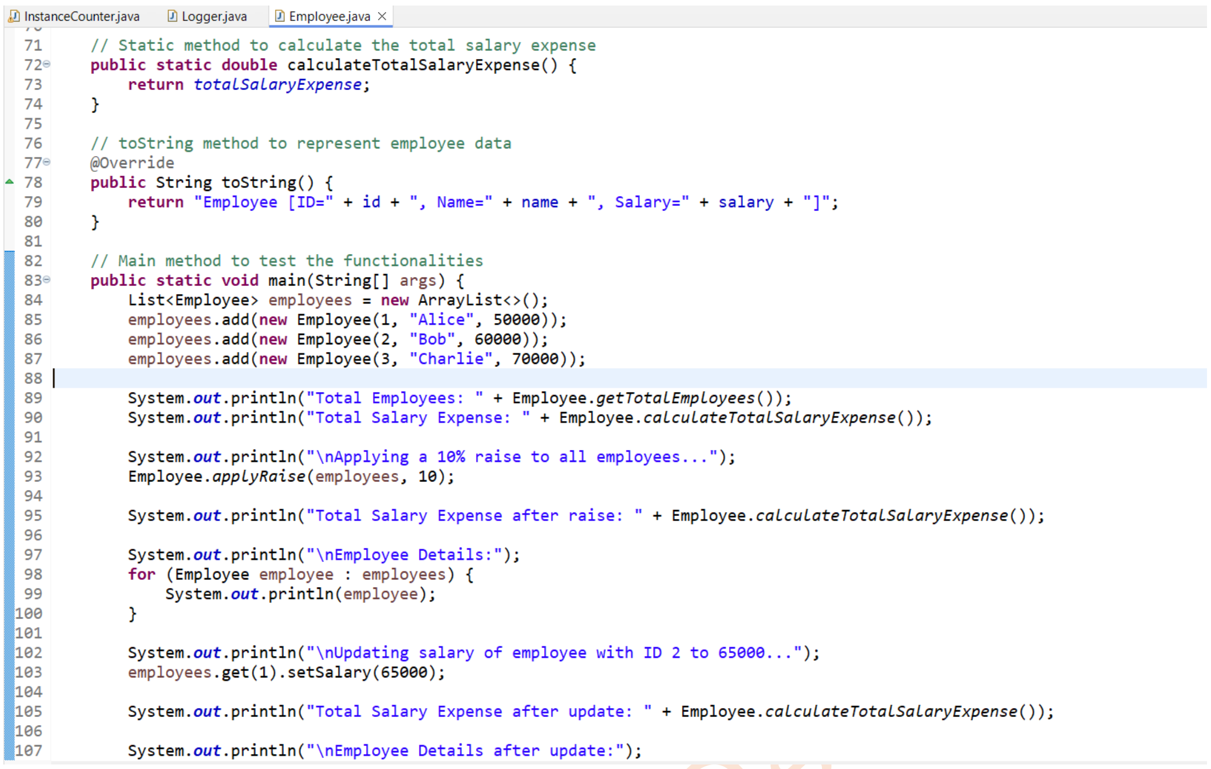


```java
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        totalSalaryExpense -= this.salary;
        this.salary = salary;
        totalSalaryExpense += salary;
    }

    // Static method to get the total number of employees
    public static int getTotalEmployees() {
        return totalEmployees;
    }

    // Static method to apply a percentage raise to all employees
    public static void applyRaise(List<Employee> employees, double percentage) {
        for (Employee employee : employees) {
            double newSalary = employee.getSalary() * (1 + percentage / 100);
            employee.setSalary(newSalary);
        }
    }

    // Static method to calculate the total salary expense
    public static double calculateTotalSalaryExpense() {
        return totalSalaryExpense;
```

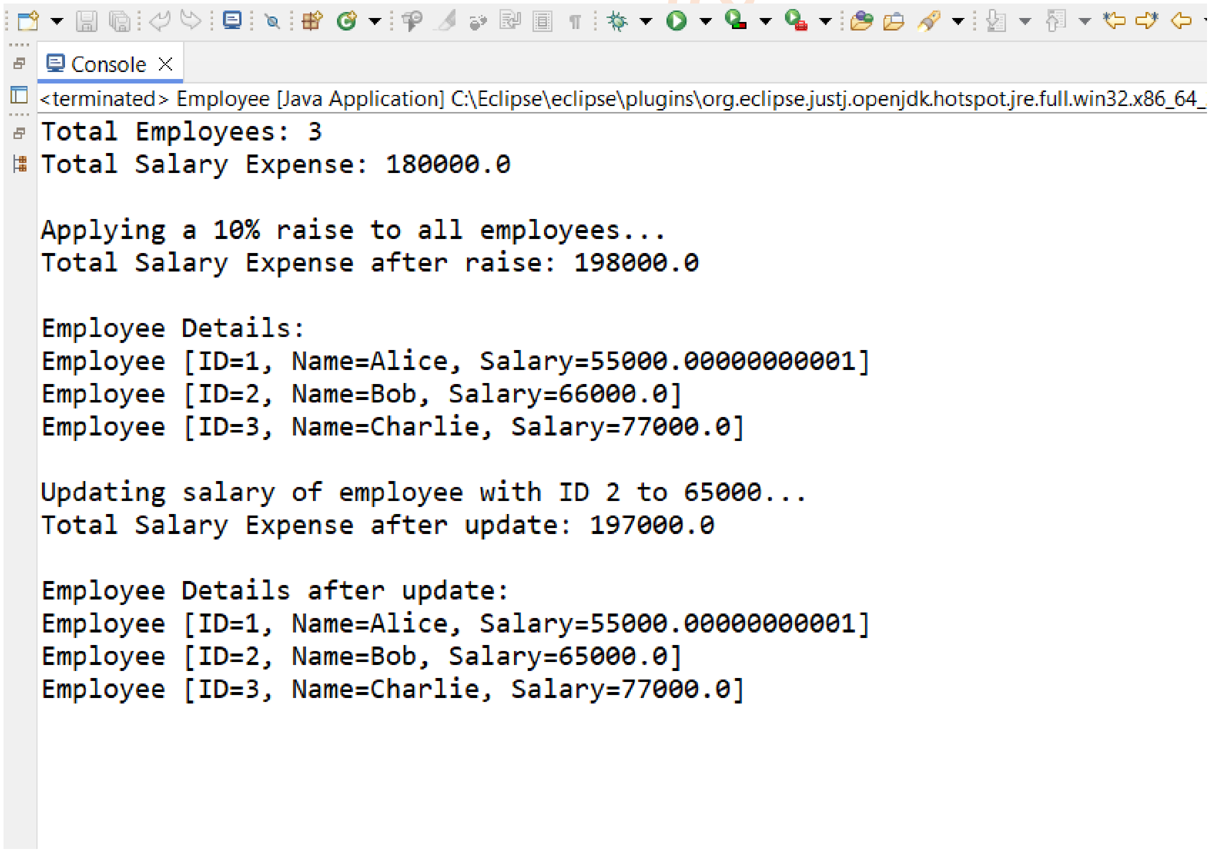


```java
    // Static method to calculate the total salary expense
    public static double calculateTotalSalaryExpense() {
        return totalSalaryExpense;
    }

    // toString method to represent employee data
    @Override
    public String toString() {
        return "Employee [ID=" + id + ", Name=" + name + ", Salary=" + salary + "]";
    }

    // Main method to test the functionalities
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee(1, "Alice", 50000));
        employees.add(new Employee(2, "Bob", 60000));
        employees.add(new Employee(3, "Charlie", 70000));

        System.out.println("Total Employees: " + Employee.getTotalEmployees());
        System.out.println("Total Salary Expense: " + Employee.calculateTotalSalaryExpense());

        System.out.println("\nApplying a 10% raise to all employees...");
        Employee.applyRaise(employees, 10);

        System.out.println("Total Salary Expense after raise: " + Employee.calculateTotalSalaryExpense());

        System.out.println("\nEmployee Details:");
        for (Employee employee : employees) {
            System.out.println(employee);
        }

        System.out.println("\nUpdating salary of employee with ID 2 to 65000...");
        employees.get(1).setSalary(65000);

        System.out.println("Total Salary Expense after update: " + Employee.calculateTotalSalaryExpense());

        System.out.println("\nEmployee Details after update:");
```

ASSIGNMENT NO.6



```java
    // Static method to calculate the total salary expense
    public static double calculateTotalSalaryExpense() {
        return totalSalaryExpense;
    }

    // toString method to represent employee data
    @Override
    public String toString() {
        return "Employee [ID=" + id + ", Name=" + name + ", Salary=" + salary + "]";
    }

    // Main method to test the functionalities
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee(1, "Alice", 50000));
        employees.add(new Employee(2, "Bob", 60000));
        employees.add(new Employee(3, "Charlie", 70000));

        System.out.println("Total Employees: " + Employee.getTotalEmployees());
        System.out.println("Total Salary Expense: " + Employee.calculateTotalSalaryExpense());

        System.out.println("\nApplying a 10% raise to all employees...");
        Employee.applyRaise(employees, 10);

        System.out.println("Total Salary Expense after raise: " + Employee.calculateTotalSalaryExpense());

        System.out.println("\nEmployee Details:");
        for (Employee employee : employees) {
            System.out.println(employee);
        }

        System.out.println("\nUpdating salary of employee with ID 2 to 65000...");
        employees.get(1).setSalary(65000);

        System.out.println("Total Salary Expense after update: " + Employee.calculateTotalSalaryExpense());

        System.out.println("\nEmployee Details after update:");
```



```
<terminated> Employee [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_
Total Employees: 3
Total Salary Expense: 180000.0

Applying a 10% raise to all employees...
Total Salary Expense after raise: 198000.0

Employee Details:
Employee [ID=1, Name=Alice, Salary=55000.00000000001]
Employee [ID=2, Name=Bob, Salary=66000.0]
Employee [ID=3, Name=Charlie, Salary=77000.0]

Updating salary of employee with ID 2 to 65000...
Total Salary Expense after update: 197000.0

Employee Details after update:
Employee [ID=1, Name=Alice, Salary=55000.00000000001]
Employee [ID=2, Name=Bob, Salary=65000.0]
Employee [ID=3, Name=Charlie, Salary=77000.0]
```