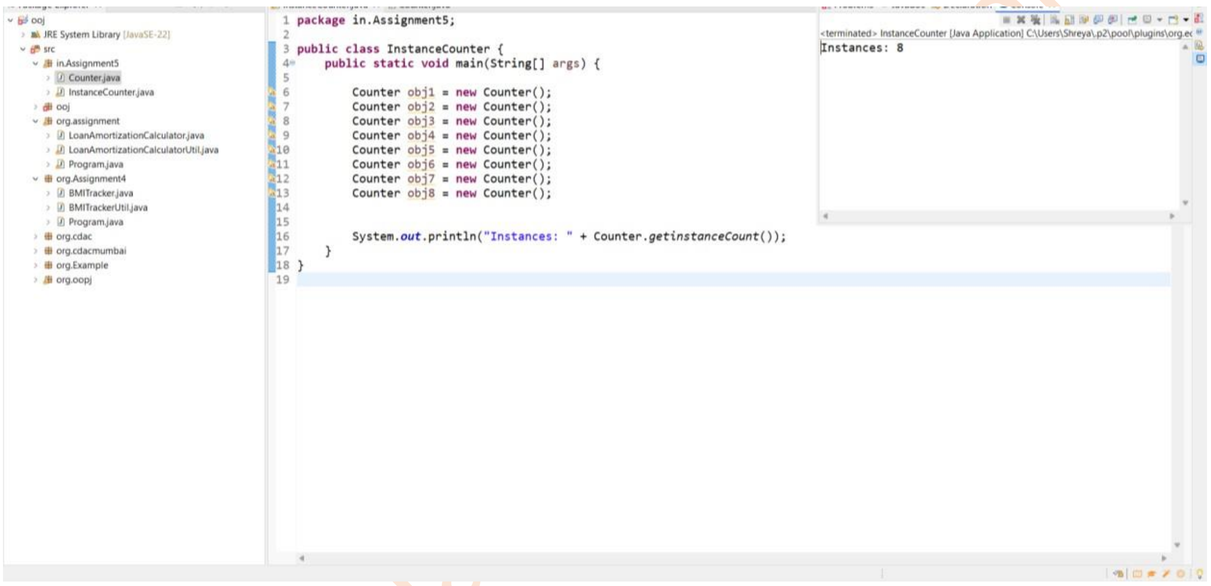
Note:

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.

```
package in.Assignment5;

public class Counter {

                    private static int instanceCount = 0;

                    public Counter() {
                              instanceCount++;
                    }

                    public static int getinstanceCount() {
                              return instanceCount;

                    }


                    @Override
                    public String toString() {
                              return "Total instances created: "  + instanceCount;
                    }

          }

package in.Assignment5;

public class InstanceCounter {
   public static void main(String[] args) {

      Counter obj1 = new Counter();
      Counter obj2 = new Counter();
      Counter obj3 = new Counter();
      Counter obj4 = new Counter();
      Counter obj5 = new Counter();
```

```
        Counter obj6 = new Counter();
        Counter obj7 = new Counter();
        Counter obj8 = new Counter();


        System.out.println("Instances: " + Counter.getinstanceCount());
    }
}
```



```
package in.Assignment5;

public class InstanceCounter {
    public static void main(String[] args) {

        Counter obj1 = new Counter();
        Counter obj2 = new Counter();
        Counter obj3 = new Counter();
        Counter obj4 = new Counter();
        Counter obj5 = new Counter();
        Counter obj6 = new Counter();
        Counter obj7 = new Counter();
        Counter obj8 = new Counter();


        System.out.println("Instances: " + Counter.getinstanceCount());
    }
}
```

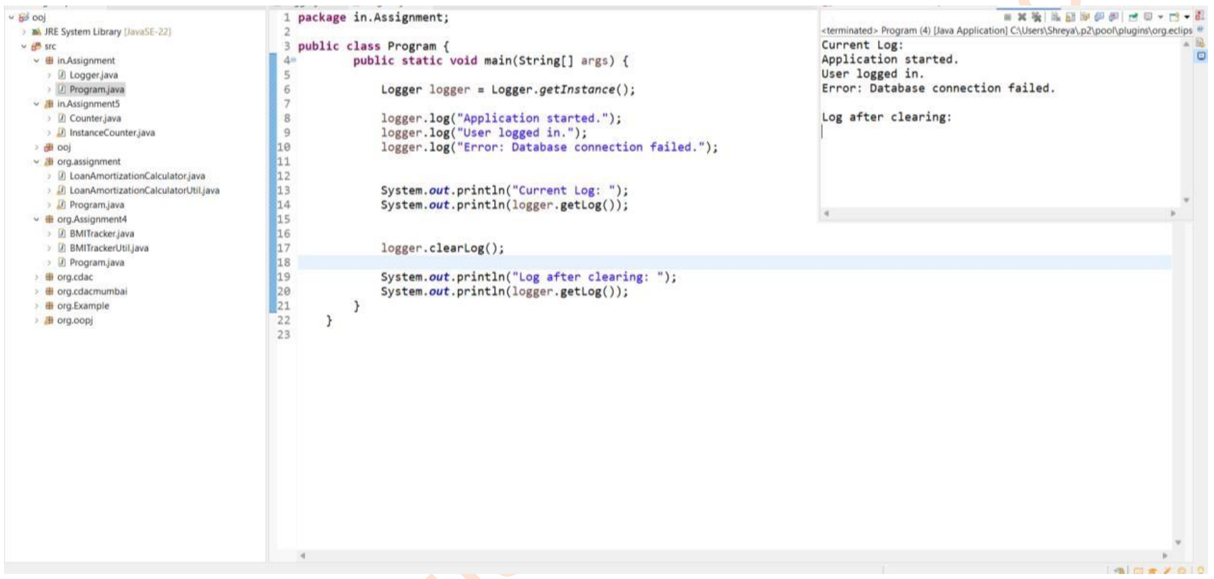
```
<terminated> InstanceCounter [Java Application] C:\Users\Shreya\.p2\pool\plugins\org.ec
Instances: 8
```

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.**

```
package in.Assignment; public
class Logger {
    private static Logger loggerInstance = null;
private String logMessages;
    private Logger() {
        logMessages = "";
    }

    public static Logger getInstance() {
if (loggerInstance == null) {
loggerInstance = new Logger();         }
        return loggerInstance;
    }

    public void log(String message) {
        logMessages += message + "\n";
    }

    public String getLog() {
return logMessages;
    }

    public void clearLog() {
        logMessages = "";
    }

    @Override     public
String toString() {
        return "Logger: " + getLog();
    }
}

package in.Assignment;

public class Program {
                  public static void main(String[] args) {

                      Logger logger = Logger.getInstance();

                      logger.log("Application started.");
logger.log("User logged in.");
                      logger.log("Error: Database connection failed.");


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

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



```
package in.Assignment;

public class Program {
        public static void main(String[] args) {

            Logger logger = Logger.getInstance();

            logger.log("Application started.");
            logger.log("User logged in.");
            logger.log("Error: Database connection failed.");


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


            logger.clearLog();

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

```
<terminated> Program (4) [Java Application] C:\Users\Shreya\.p2\pool\plugins\org.eclips
Current Log:
Application started.
User logged in.
Error: Database connection failed.

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)`)**

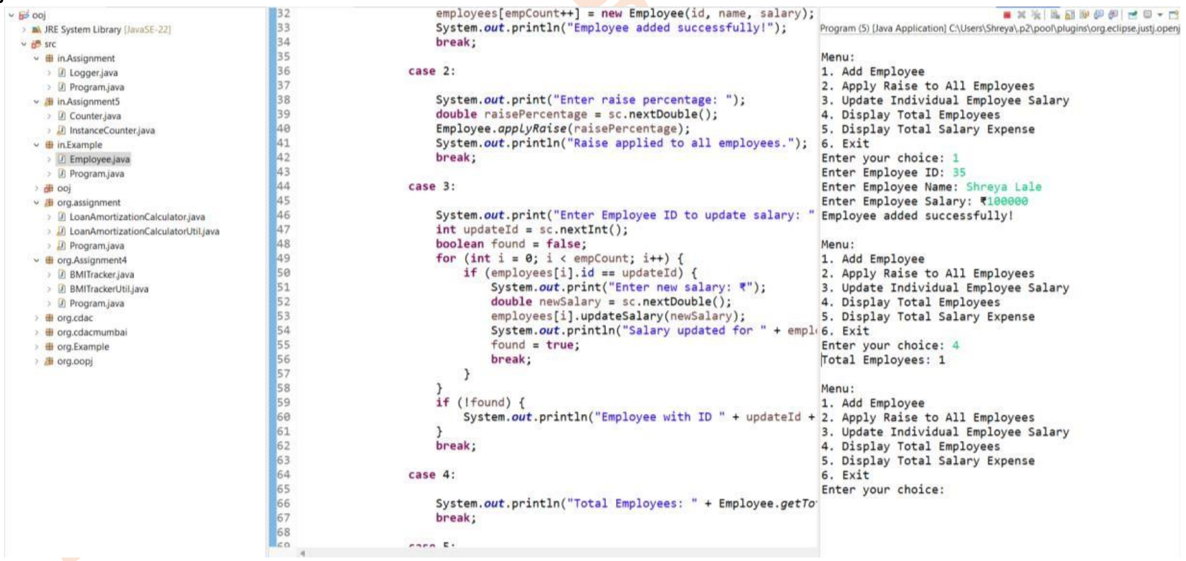
Understand the problem statement and use static and non-static fields and methods appropriately. Implement static and non-static initializers, constructors, getter and setter methods, and a `toString()` method to handle the initialization and representation of employee data.

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

```
package in.Example;

public class Employee {

    private static int totalEmployees = 0;
    private static double totalSalaryExpense = 0.0;

    int id;
    private String name;
    private double salary;

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

    public static int getTotalEmployees() {
        return totalEmployees;
    }

    public static double calculateTotalSalaryExpense() {
        return totalSalaryExpense;
    }

    public static void applyRaise(double percentage) {
        totalSalaryExpense += totalSalaryExpense * (percentage / 100);
    }

    public void updateSalary(double newSalary) {
        totalSalaryExpense -= this.salary;
        this.salary = newSalary;
        totalSalaryExpense += this.salary;
    }
    public String getName() {
        return name;
    }

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


    public double getSalary() {
        return salary;
    }

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


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


     package in.Example;

     import java.util.Scanner;


public class Program {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Employee[] employees = new Employee[100];
        int empCount = 0;

        int choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Add Employee");
            System.out.println("2. Apply Raise to All Employees");
            System.out.println("3. Update Individual Employee Salary");
            System.out.println("4. Display Total Employees");
            System.out.println("5. Display Total Salary Expense");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
```

```
choice = sc.nextInt();

switch (choice) {
  case 1:

    System.out.print("Enter Employee ID: ");
    int id = sc.nextInt();
    sc.nextLine();
    System.out.print("Enter Employee Name: ");
    String name = sc.nextLine();
    System.out.print("Enter Employee Salary: ₹");
    double salary = sc.nextDouble();
    employees[empCount++] = new Employee(id, name, salary);
    System.out.println("Employee added successfully!");
    break;

  case 2:

    System.out.print("Enter raise percentage: ");
    double raisePercentage = sc.nextDouble();
    Employee.applyRaise(raisePercentage);
    System.out.println("Raise applied to all employees.");
    break;

  case 3:

    System.out.print("Enter Employee ID to update salary: ");
    int updateId = sc.nextInt();
    boolean found = false;
    for (int i = 0; i < empCount; i++) {
      if (employees[i].id == updateId) {
        System.out.print("Enter new salary: ₹");
        double newSalary = sc.nextDouble();
        employees[i].updateSalary(newSalary);
        System.out.println("Salary updated for " + employees[i].getName());
        found = true;
        break;
      }
    }
    if (!found) {
      System.out.println("Employee with ID " + updateId + " not found.");
    }
    break;

  case 4:
```

```
                System.out.println("Total Employees: " + Employee.getTotalEmployees());
                break;

            case 5:

                System.out.println("Total Salary Expense: ₹" +
Employee.calculateTotalSalaryExpense());
                break;

            case 6:

                System.out.println("Exiting...");
                break;

            default:
                System.out.println("Invalid choice. Please try again.");
                break;
          }
        } while (choice != 6);

        sc.close();
    }
}
```



```
ooj
  JRE System Library [JavaSE-22]
  src
    in.Assignment
      Logger.java
      Program.java
    in.Assignment5
      Counter.java
      InstanceCounter.java
    in.Example
      Employee.java
      Program.java
    ooj
    org.assignment
      LoanAmortizationCalculator.java
      LoanAmortizationCalculatorUtil.java
      Program.java
    org.Assignment4
      BMITracker.java
      BMITrackerUtil.java
      Program.java
    org.cdac
    org.cdacmumbai
    org.Example
    org.oopj
```

```
                employees[empCount++] = new Employee(id, name, salary);
                System.out.println("Employee added successfully!");
                break;

            case 2:

                System.out.print("Enter raise percentage: ");
                double raisePercentage = sc.nextDouble();
                Employee.applyRaise(raisePercentage);
                System.out.println("Raise applied to all employees.");
                break;

            case 3:

                System.out.print("Enter Employee ID to update salary: "
                int updateId = sc.nextInt();
                boolean found = false;
                for (int i = 0; i < empCount; i++) {
                    if (employees[i].id == updateId) {
                        System.out.print("Enter new salary: ₹");
                        double newSalary = sc.nextDouble();
                        employees[i].updateSalary(newSalary);
                        System.out.println("Salary updated for " + empl
                        found = true;
                        break;
                    }
                }
                if (!found) {
                    System.out.println("Employee with ID " + updateId +
                }
                break;

            case 4:

                System.out.println("Total Employees: " + Employee.getTo
                break;

            case 5:
```

```
Program (5) [Java Application] C:\Users\Shreya\.p2\pool\plugins\org.eclipse.justj.openj

Menu:
1. Add Employee
2. Apply Raise to All Employees
3. Update Individual Employee Salary
4. Display Total Employees
5. Display Total Salary Expense
6. Exit
Enter your choice: 1
Enter Employee ID: 35
Enter Employee Name: Shreya Lale
Enter Employee Salary: ₹100000
Employee added successfully!

Menu:
1. Add Employee
2. Apply Raise to All Employees
3. Update Individual Employee Salary
4. Display Total Employees
5. Display Total Salary Expense
6. Exit
Enter your choice: 4
Total Employees: 1

Menu:
1. Add Employee
2. Apply Raise to All Employees
3. Update Individual Employee Salary
4. Display Total Employees
5. Display Total Salary Expense
6. Exit
Enter your choice:
```