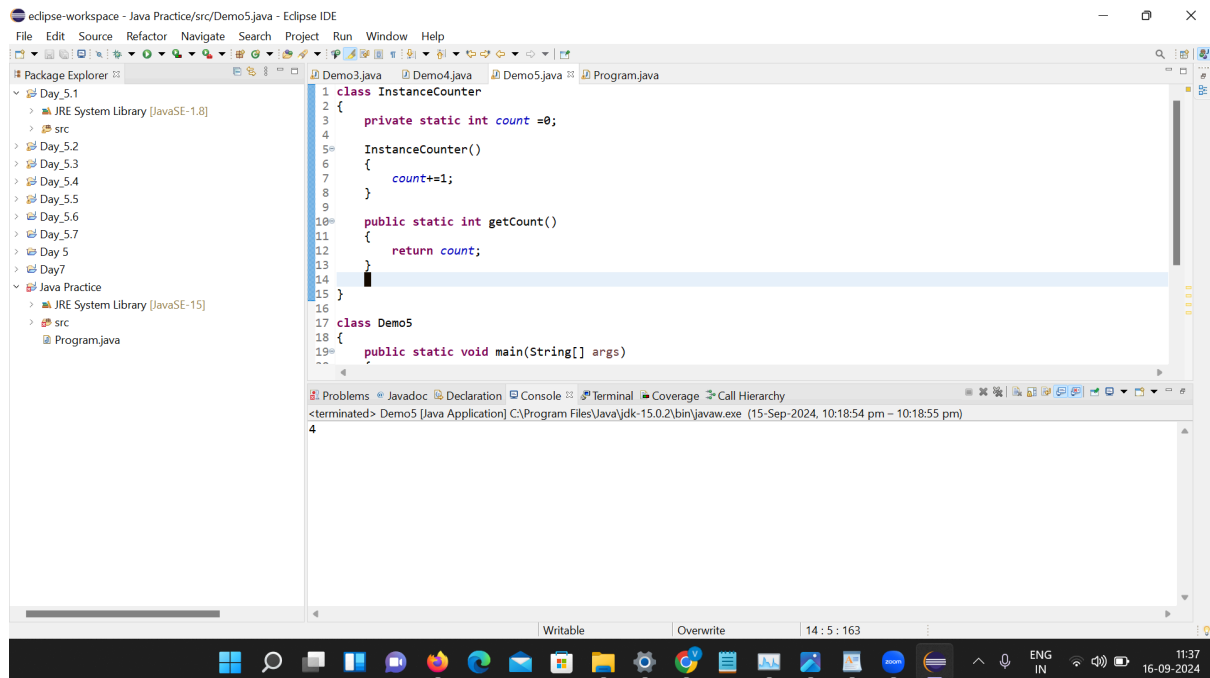


Note:

This assignment is designed to practice static fields, static initializers, and static methods. Understand the problem statement and use static and non-static wisely to solve the problem. Use constructors, proper getter/setter methods, and toString() wherever required.

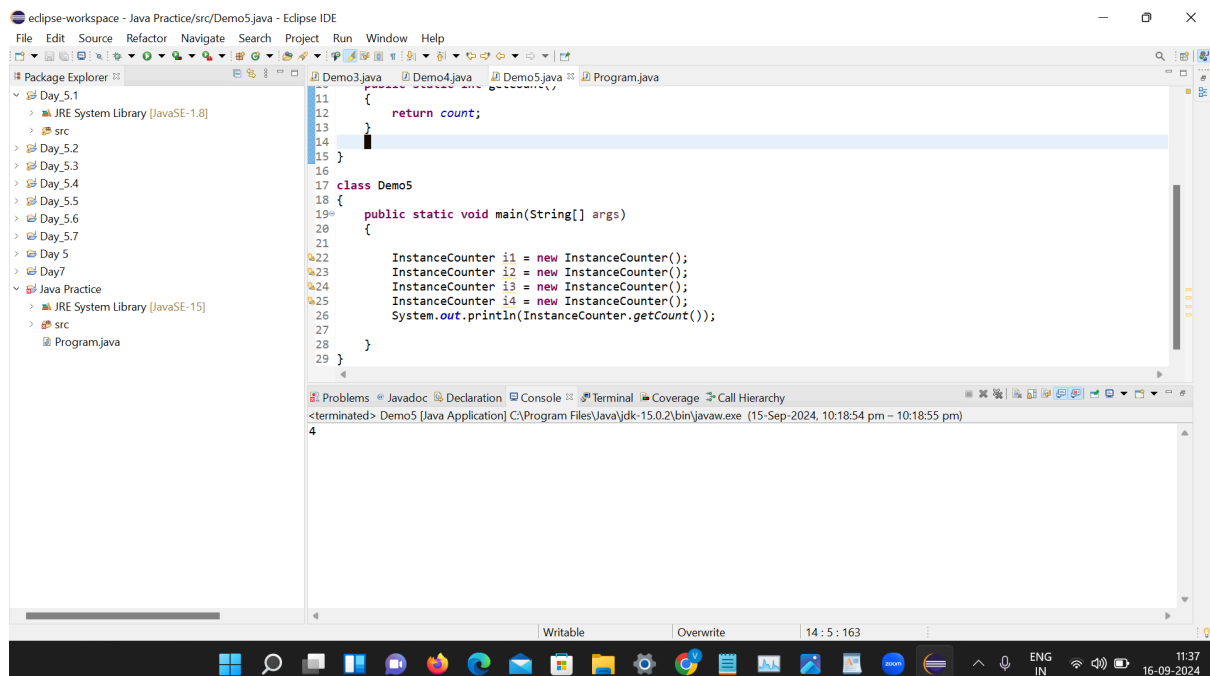
Design and implement a class named InstanceCounter to track and count the number of instances created from this class.



The screenshot shows the Eclipse IDE with the following code in `Demo5.java`:

```
1 class InstanceCounter
2 {
3     private static int count = 0;
4
5     InstanceCounter()
6     {
7         count++;
8     }
9
10    public static int getCount()
11    {
12        return count;
13    }
14 }
15
16
17 class Demo5
18 {
19     public static void main(String[] args)
20     {
21
22     }
```

The Package Explorer on the left shows the project structure with `src` containing `Program.java`. The Console at the bottom shows the output of the program, which is currently empty.



The screenshot shows the Eclipse IDE with the following code in `Demo5.java`:

```
11 {
12     return count;
13 }
14 }
15
16
17 class Demo5
18 {
19     public static void main(String[] args)
20     {
21
22         InstanceCounter i1 = new InstanceCounter();
23         InstanceCounter i2 = new InstanceCounter();
24         InstanceCounter i3 = new InstanceCounter();
25         InstanceCounter i4 = new InstanceCounter();
26         System.out.println(InstanceCounter.getCount());
27     }
28 }
29 }
```

The Package Explorer on the left shows the project structure with `src` containing `Program.java`. The Console at the bottom shows the output of the program, which is currently empty.

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.

The screenshot shows the Eclipse IDE with the 'Logger' class implemented. The Package Explorer on the left lists various Java files, including 'Demo6.java'. The main editor displays the code for 'Logger.java'.

```
1 class Logger
2 {
3     private static Logger instance = null;
4
5     private StringBuilder logMessages;
6
7     private Logger()
8     {
9         logMessages = new StringBuilder();
10    }
11
12    public static synchronized Logger getInstance()
13    {
14        if(instance==null)
15        {
16            instance = new Logger();
17        }
18        return instance;
19    }
20 }
```

The Console window at the bottom shows the output of running 'Demo6.java':

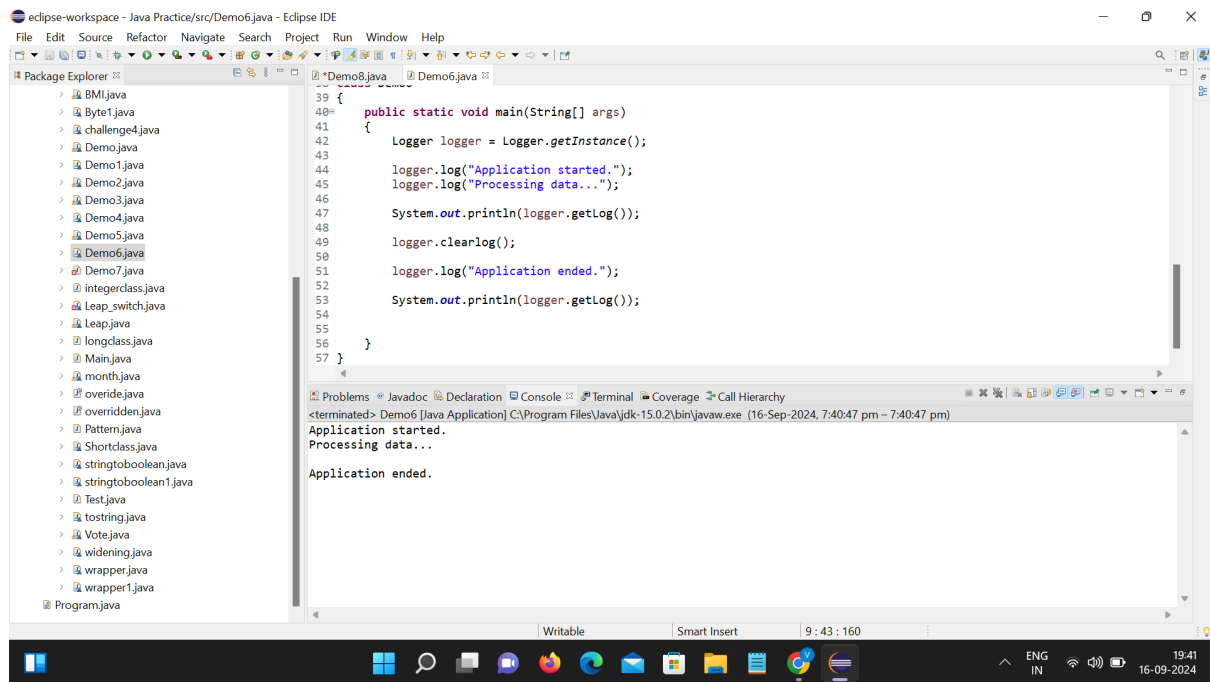
```
<terminated> Demo6 [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe (16-Sep-2024, 7:40:47 pm - 7:40:47 pm)
Application started.
Processing data...
Application ended.
```

The screenshot shows the Eclipse IDE with the 'Demo6' class implemented. The Package Explorer on the left lists various Java files, including 'Demo6.java'. The main editor displays the code for 'Demo6.java'.

```
21 }
22
23 public void log(String Messages)
24 {
25     System.out.println(Messages);
26 }
27
28 public String getLog()
29 {
30     return logMessages.toString();
31 }
32
33 public void clearlog()
34 {
35     logMessages.setLength(0);
36 }
37
38 class Demo6
39 {
```

The Console window at the bottom shows the output of running 'Demo6.java':

```
<terminated> Demo6 [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe (16-Sep-2024, 7:40:47 pm - 7:40:47 pm)
Application started.
Processing data...
Application ended.
```



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.

