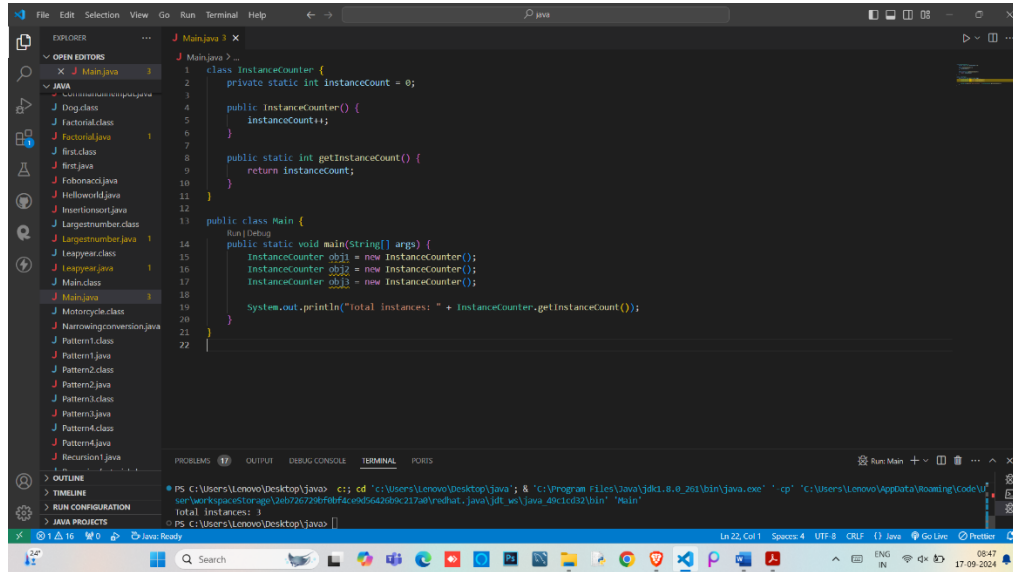


ASSIGNMENT 6

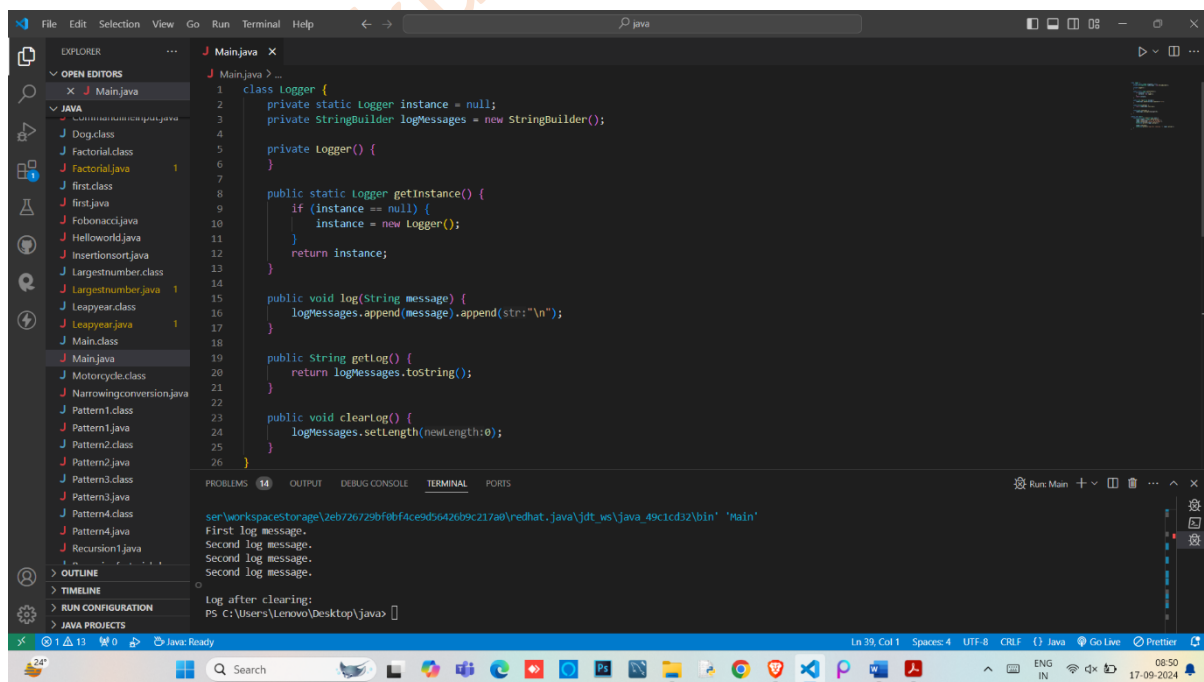
1. Design and implement a class named `InstanceCounter` to track and count the number of instances created from this class.



```
1 class InstanceCounter {
2     private static int instanceCount = 0;
3
4     public InstanceCounter() {
5         instanceCount++;
6     }
7
8     public static int getInstanceCount() {
9         return instanceCount;
10    }
11 }
12
13 public class Main {
14     public static void main(String[] args) {
15         InstanceCounter obj1 = new InstanceCounter();
16         InstanceCounter obj2 = new InstanceCounter();
17         InstanceCounter obj3 = new InstanceCounter();
18
19         System.out.println("Total instances: " + InstanceCounter.getInstanceCount());
20     }
21 }
22
```

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.



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

ser/workspaces/storage/2eb726729bfb4f4ced9d6426b9c217a0/redhat_java_jdt_ws/java_49c1cd32/bin' 'Main'

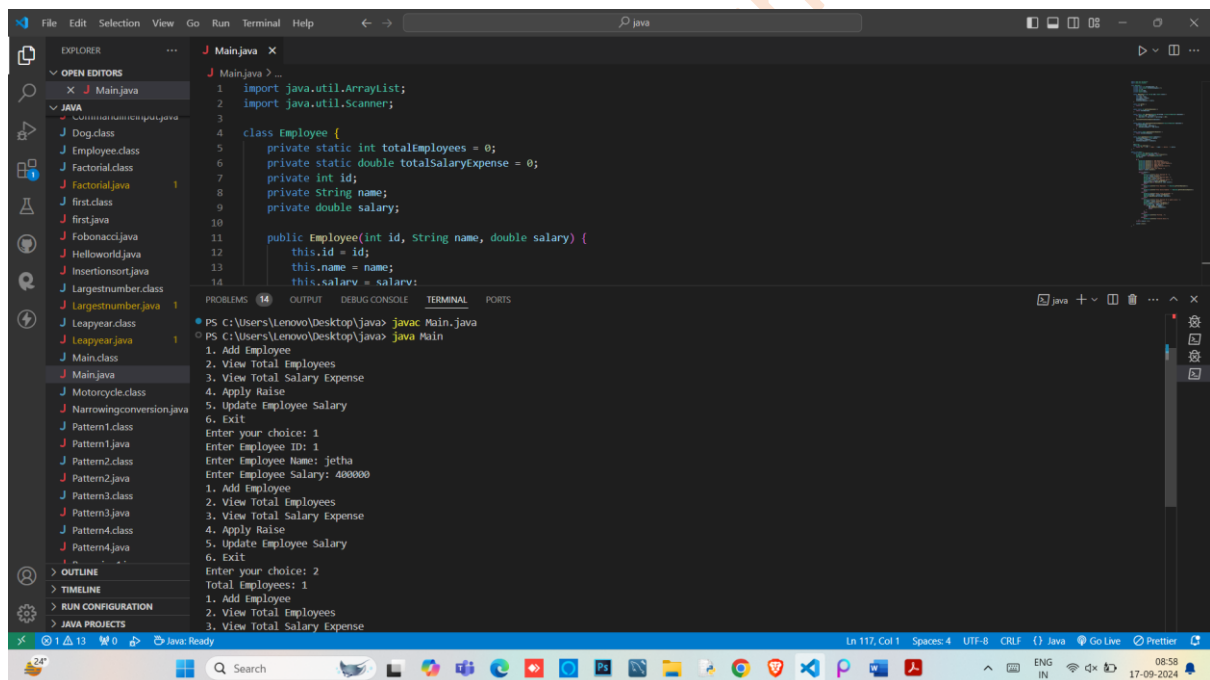
First log message.
Second log message.
Second log message.

Log after clearing:
PS C:\Users\Lenovo\Desktop\javas >

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.



```
File Edit Selection View Go Run Terminal Help
Main.java
EXPLORER
  OPEN EDITORS
    Main.java
  JAVA
    Dog.class
    Employee.class
    Factorial.class
    Factorial.java
    first.class
    first.java
    Fobonacci.java
    Helloword.java
    Insertionsort.java
    Largestnumber.class
    Largestnumber.java
    Leapyear.class
    Leapyear.java
    Main.class
    Main.java
    Motorcycle.class
    Narrowingconversion.java
    Pattern1.class
    Pattern1.java
    Pattern2.class
    Pattern2.java
    Pattern3.class
    Pattern3.java
    Pattern4.class
    Pattern4.java
  OUTLINE
  TIMELINE
  RUN CONFIGURATION
  JAVA PROJECTS
Main.java
1 import java.util.ArrayList;
2 import java.util.Scanner;
3
4 class Employee {
5     private static int totalEmployees = 0;
6     private static double totalSalaryExpense = 0;
7     private int id;
8     private String name;
9     private double salary;
10
11     public Employee(int id, String name, double salary) {
12         this.id = id;
13         this.name = name;
14         this.salary = salary;
15     }
16 }
17
18 public class Main {
19     public static void main(String[] args) {
20         Scanner sc = new Scanner(System.in);
21         int choice;
22         do {
23             System.out.println("1. Add Employee\n2. View Total Employees\n3. View Total Salary Expense\n4. Apply Raise\n5. Update Employee Salary\n6. Exit\nEnter your choice: ");
24             choice = sc.nextInt();
25             switch (choice) {
26                 case 1:
27                     System.out.println("Enter Employee ID: ");
28                     int id = sc.nextInt();
29                     System.out.println("Enter Employee Name: ");
30                     String name = sc.next();
31                     System.out.println("Enter Employee Salary: ");
32                     double salary = sc.nextDouble();
33                     Employee emp = new Employee(id, name, salary);
34                     ArrayList<Employee> employees = new ArrayList<>();
35                     employees.add(emp);
36                     break;
37                 case 2:
38                     System.out.println("Total Employees: " + employees.size());
39                     break;
39                 case 3:
40                     System.out.println("Total Salary Expense: " + calculateTotalSalaryExpense(employees));
41                     break;
42                 case 4:
43                     System.out.println("Enter Raise Percentage: ");
44                     double raise = sc.nextDouble();
45                     applyRaise(employees, raise);
46                     break;
47                 case 5:
48                     System.out.println("Enter Employee ID: ");
49                     int id = sc.nextInt();
50                     System.out.println("Enter New Salary: ");
51                     double newSalary = sc.nextDouble();
52                     updateSalary(id, newSalary);
53                     break;
54                 case 6:
55                     break;
56             }
57         } while (choice != 6);
58     }
59 }
60
61 private static double calculateTotalSalaryExpense(ArrayList<Employee> employees) {
62     double totalSalaryExpense = 0;
63     for (Employee emp : employees) {
64         totalSalaryExpense += emp.salary;
65     }
66     return totalSalaryExpense;
67 }
68
69 private static void applyRaise(ArrayList<Employee> employees, double raise) {
70     for (Employee emp : employees) {
71         emp.salary = emp.salary * (1 + raise);
72     }
73 }
74
75 private static void updateSalary(int id, double newSalary) {
76     for (Employee emp : employees) {
77         if (emp.id == id) {
78             emp.salary = newSalary;
79         }
80     }
81 }
```