ASSIGNMENT 5

**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.
4. Design and implement a class named InstanceCounter to track and count the number of instances created from this class.
5. **package** com.java;
6. **public** **class** InstanceCounter {
7. **private** **static** **int** *instanceCount* = 0;//static var track total no of instance created private to prevent modification.
8. **public** InstanceCounter() {
9. // Every time a new InstanceCounter object is created, the constructor is called. Inside the constructor, we increment the instanceCount by 1.
10. *instanceCount*++;
11. }
12. **public** **static** **int** getInstanceCount() {
13. // static method that returns the current value of instanceCount.
14. **return** *instanceCount*;
15. }
16. }

**package** com.java;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

InstanceCounter obj1 = **new** InstanceCounter();

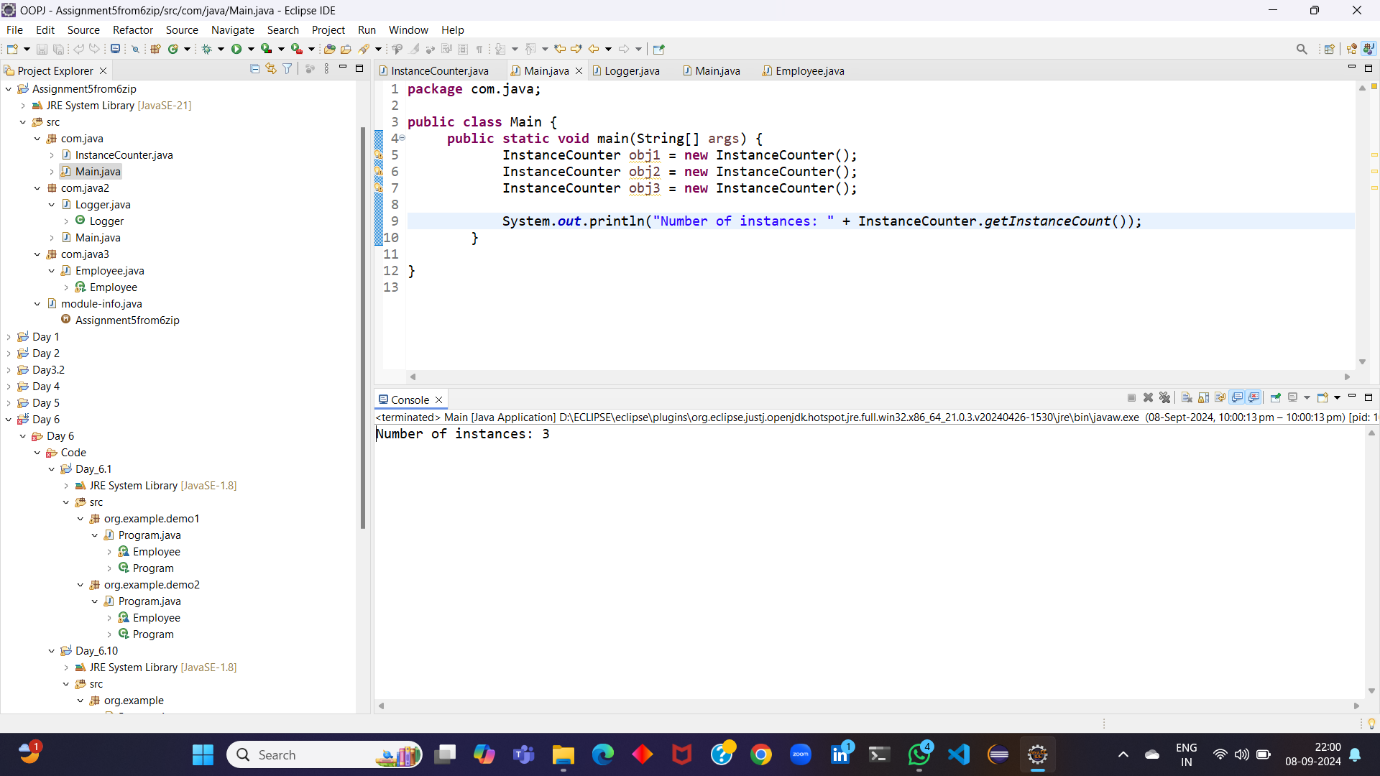
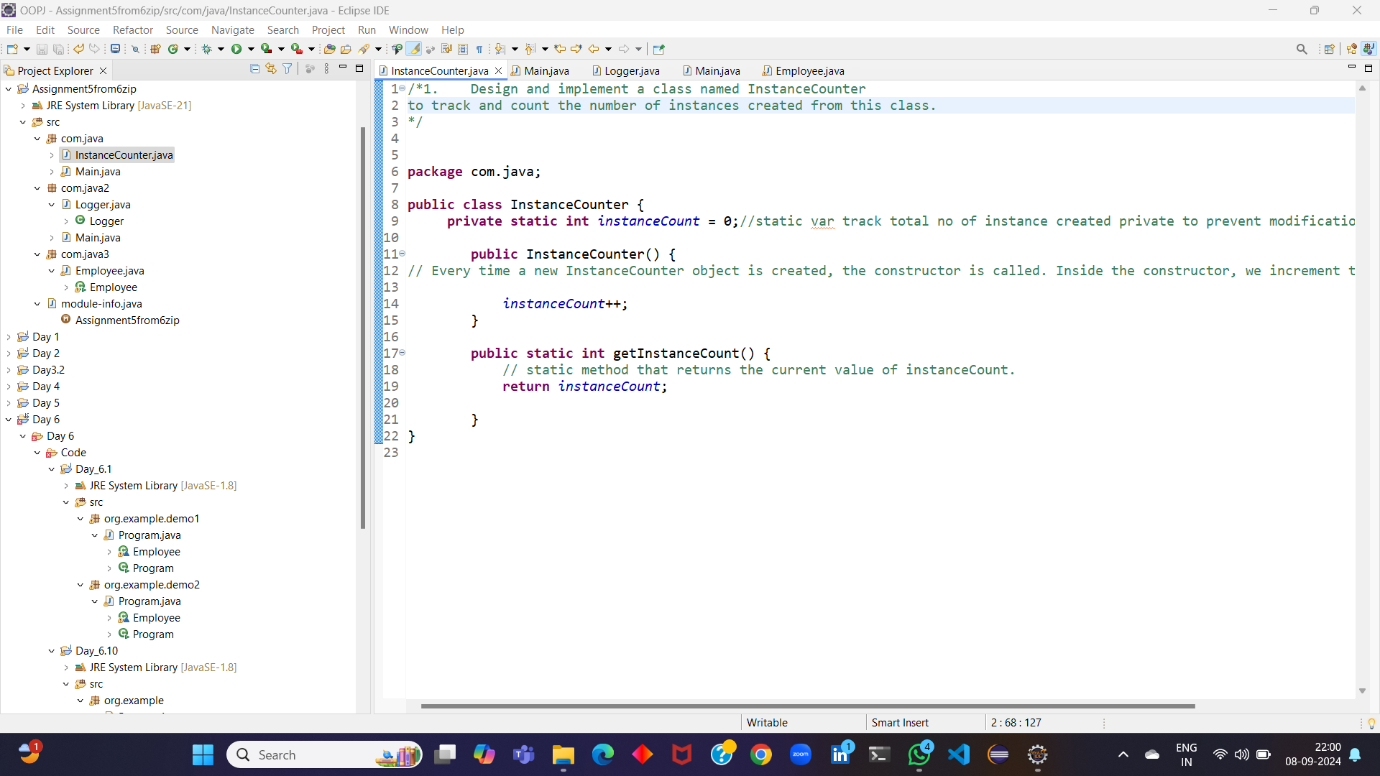
InstanceCounter obj2 = **new** InstanceCounter();

InstanceCounter obj3 = **new** InstanceCounter();

System.***out***.println("Number of instances: " + InstanceCounter.*getInstanceCount*());

}

}



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 com.java2;

public class Logger {

private static Logger instance;//static variable that holds the reference to the single instance of the Logger class

private StringBuilder logMessages;//This is a private member variable that stores the log messages as a string.

private Logger() {//private const to prevent instance of class

logMessages = new StringBuilder();

}

public static Logger getInstance() {// This is a static method that returns the single instance of the Logger class.

if (instance == null) {//If the instance doesn't exist, it creates a new one.

instance = new Logger();

}

return instance;

}

public void log(String message) {//This method adds the given message to the log messages.

logMessages.append(message).append("\n");

}

public String getLog() {//This method returns the current log messages as a string.

return logMessages.toString();

}

public void clearLog() {//This method clears all the log messages.

logMessages.setLength(0);

}

**package** com.java2;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Logger logger = Logger.*getInstance*();

logger.log("This is a log message.");

logger.log("Another log message.");

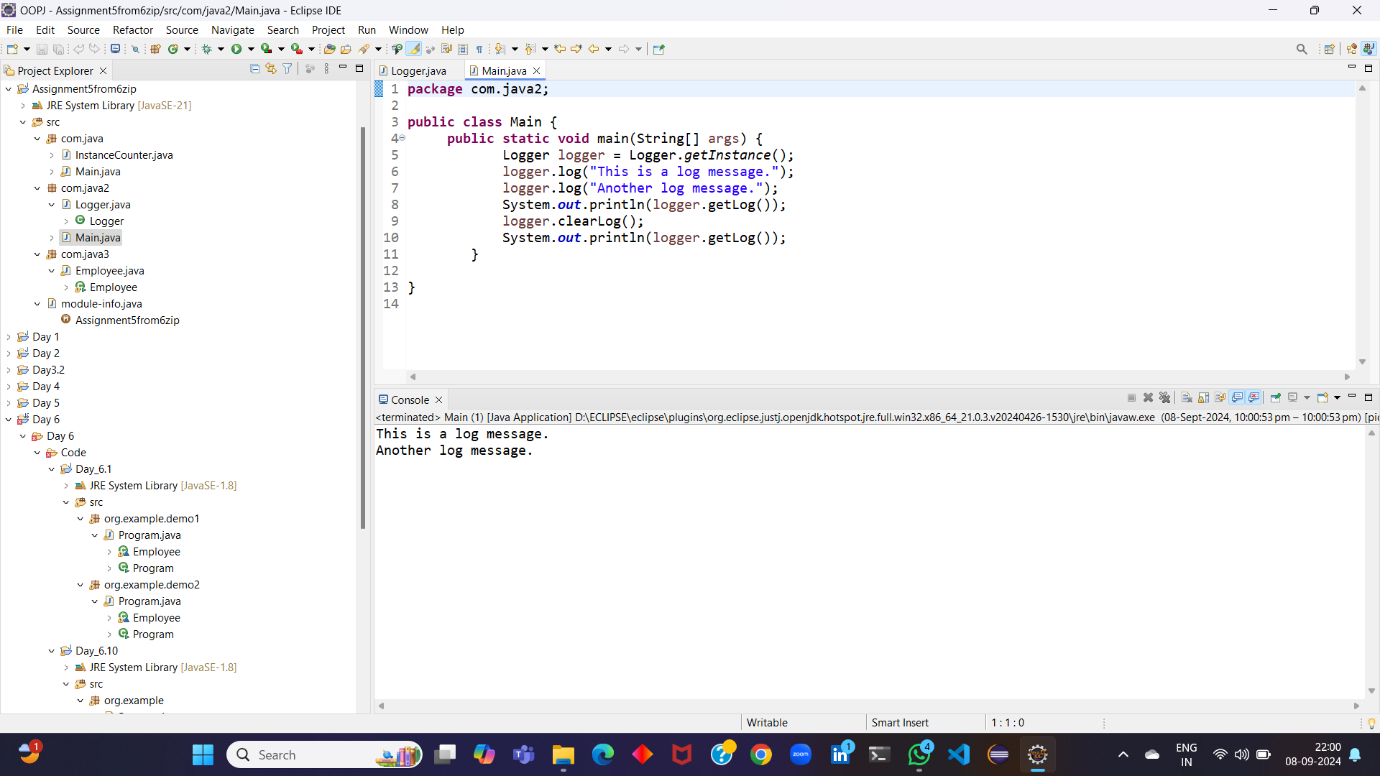
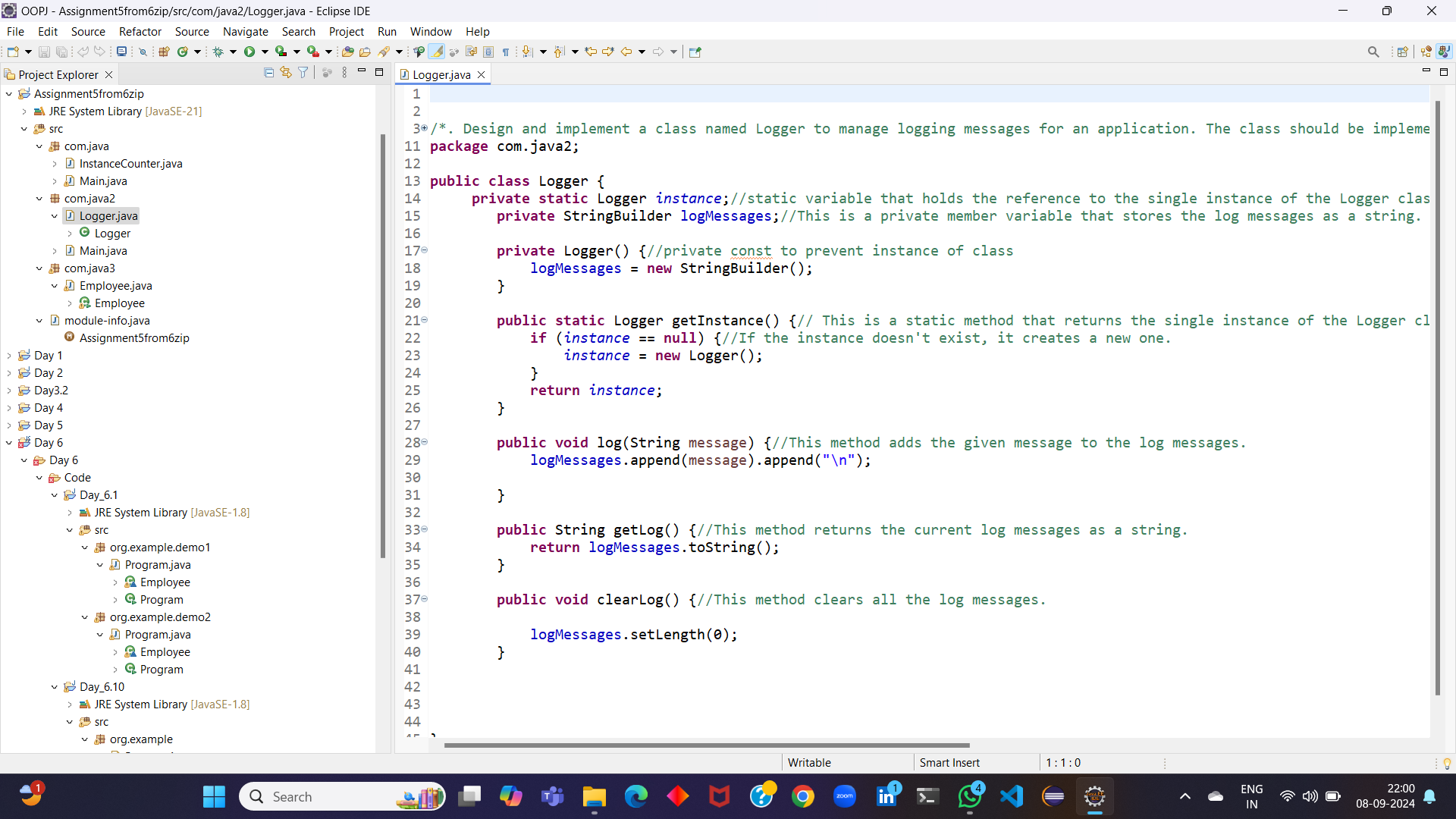
System.***out***.println(logger.getLog());

logger.clearLog();

System.***out***.println(logger.getLog());

}

}



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.

/\*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 com.java3;

import java.util.Scanner;

public class Employee {

private static int totalEmployees = 0;

private static double totalSalaryExpense = 0;

private 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 void applyRaise(double percentage) {

totalSalaryExpense \*= (1 + percentage / 100);

}

public static double calculateTotalSalaryExpense() {

return totalSalaryExpense;

}

public void updateSalary(double newSalary) {

totalSalaryExpense -= this.salary;

this.salary = newSalary;

totalSalaryExpense += this.salary;

}

public String toString() {

return "Employee ID: " + id + ", Name: " + name + ", Salary: " + salary;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Employee[] employees = new Employee[100]; // Assuming a maximum of 100 employees

int choice;

do {

System.out.println("1. Add Employee");

System.out.println("2. Apply Raise");

System.out.println("3. Calculate Total Salary Expense");

System.out.println("4. Update Employee Salary");

System.out.println("5. Display Employee Information");

System.out.println("6. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.print("Enter Employee ID: ");

int id = scanner.nextInt();

System.out.print("Enter Employee Name: ");

String name = scanner.next();

System.out.print("Enter Employee Salary: ");

double salary = scanner.nextDouble();

employees[totalEmployees] = new Employee(id, name, salary);

System.out.println("Employee added successfully.");

break;

case 2:

System.out.print("Enter percentage raise: ");

double percentage = scanner.nextDouble();

Employee.applyRaise(percentage);

System.out.println("Raise applied successfully.");

break;

case 3:

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

break;

case 4:

System.out.print("Enter Employee ID to update: ");

int updateId = scanner.nextInt();

for (int i = 0; i < totalEmployees; i++) {

if (employees[i].id == updateId) {

System.out.print("Enter new salary: ");

double newSalary = scanner.nextDouble();

employees[i].updateSalary(newSalary);

System.out.println("Salary updated successfully.");

break;

}

}

break;

case 5:

System.out.println("Employee Information:");

for (int i = 0; i < totalEmployees; i++) {

System.out.println(employees[i]);

}

break;

case 6:

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

break;

default:

System.out.println("Invalid choice.");

}

} while (choice != 6);

}

}

