**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.

Code:

public class InstanceCounter {

    private static int instanceCount = 0;

    // Constructor

    public InstanceCounter() {

        instanceCount++; // Increment count on instance creation

    }

    // Method to get instance count

    public static int getInstanceCount() {

        return instanceCount;

    }

    public static void main(String[] args) {

        // Creating instances of the class

        InstanceCounter obj1 = new InstanceCounter();

        InstanceCounter obj2 = new InstanceCounter();

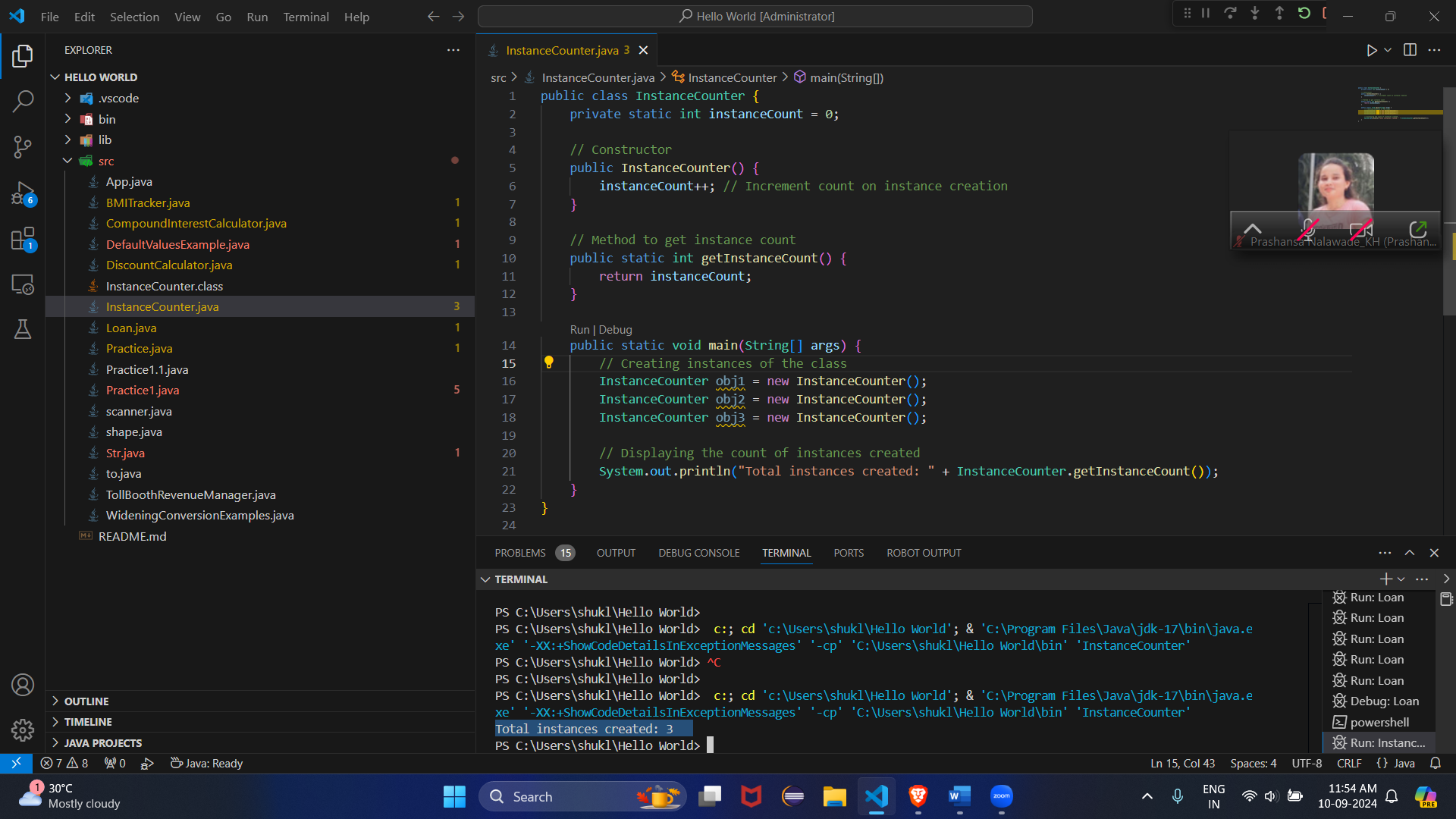
        InstanceCounter obj3 = new InstanceCounter();

        // Displaying the count of instances created

        System.out.println("Total instances created: " + InstanceCounter.getInstanceCount());

    }

}



1. 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.

Code:

import java.util.ArrayList;

import java.util.List;

public class Logger {

// Private static variable to hold the single instance of the Logger class

private static Logger instance;

// List to store log messages

private List<String> logMessages;

// Private constructor to prevent instantiation from outside

private Logger() {

logMessages = new ArrayList<>();

}

// Public static method to get the single instance of Logger

public static synchronized Logger getInstance() {

if (instance == null) {

instance = new Logger(); // Create the instance if it doesn't exist

}

return instance;

}

// Method to add a log message

public void log(String message) {

logMessages.add(message);

}

// Method to get all log messages

public String getLog() {

return String.join("\n", logMessages);

}

// Method to clear all log messages

public void clearLog() {

logMessages.clear();

}

// Main method to test the Logger class

public static void main(String[] args) {

// Get the single instance of Logger

Logger logger = Logger.getInstance();

// Add log messages

logger.log("Application started.");

logger.log("User logged in.");

logger.log("An error occurred.");

// Print all log messages

System.out.println("Log Messages:");

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

// Clear log messages

logger.clearLog();

System.out.println("Log Messages after clearing:");

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

}

}

A screenshot of a computer screen

Description automatically generated

1. 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.

public class Employee {

    private static int totalEmp = 0;

    private static double salarayexp = 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;

        totalEmp++;

        salarayexp += salary;

    }

    public static int gettotalEmp(){

        return totalEmp;

    }

    public static void applyRaise(double percentage) {

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

    }

    public static double calculateTotalSalaryExpense() {

        return salarayexp;

    }

    // Method to update salary of an employee

    public void updateSalary(double newSalary) {

        salarayexp = salarayexp - this.salary + newSalary;

        this.salary = newSalary;

    }

    public String toString() {

        return "Employee [ID=" + id + ", Name=" + name + ", Salary=" + salary + "]";

    }

    public static void main(String[] args) {

        Employee emp1 = new Employee(1, "Alice", 50000);

        Employee emp2 = new Employee(2, "Bob", 60000);

        System.out.println(emp1);

        System.out.println(emp2);

    }

}

A screenshot of a computer

Description automatically generated