**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.

package org.cdac;

public class InstanceCounter {

private static int *instanceCount* =0;

public InstanceCounter() {

*instanceCount*++;

}

public static int getInstanceCount() {

return *instanceCount*;

}

*@Override*

public String toString() {

return "InstanceCounter object created, total instance:"+*instanceCount*;

}

public static void main(String args []) {

InstanceCounter obj1 =new InstanceCounter();

System.***out***.println(obj1.toString());

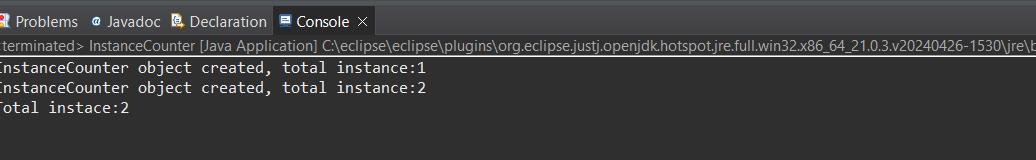
InstanceCounter obj2 = new InstanceCounter();

System.***out***.println(obj2.toString());

System.***out***.println("Total instace:" + 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 message

package org.cdac.copy;

public class Logger {

private static Logger *instance* = null;

private StringBuilder logMessages;

private Logger() {

logMessages = new StringBuilder();

}

public static Logger getInstance() {

if (*instance* == null) {

*instance* = new Logger();

}

return *instance*;

}

public void log(String message) {

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

}

public String getLog() {

return logMessages.toString();

}

public void clearLog() {

logMessages.setLength(0);

}

public static void main(String[] args) {

Logger logger = Logger.*getInstance*();

logger.log("Application started.");

logger.log("User login.");

logger.log("User action performed.");

System.***out***.println("Log messages:");

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

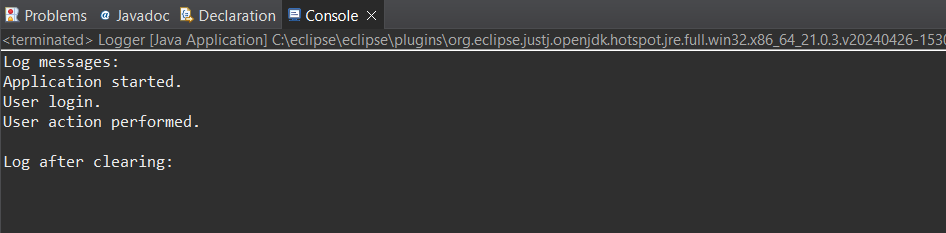
logger.clearLog();

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

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

}

}



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.

package org.cdac.copy.copy;

import java.util.ArrayList;

import java.util.List;

public class Employee {

private static int *totalEmployees* = 0;

private static double *totalSalaryExpense* = 0;

private int id;

private String name;

private double salary;

static {

*totalEmployees* = 0;

*totalSalaryExpense* = 0;

}

public Employee(int id, String name, double salary) {

this.id = id;

this.name = name;

this.salary = salary;

*totalEmployees*++;

*totalSalaryExpense* += salary;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

*totalSalaryExpense* -= this.salary;

this.salary = salary;

*totalSalaryExpense* += salary;

}

public static int getTotalEmployees() {

return *totalEmployees*;

}

public static double calculateTotalSalaryExpense() {

return *totalSalaryExpense*;

}

public void applyRaise(double percentage) {

double raiseAmount = salary \* (percentage / 100);

setSalary(salary + raiseAmount);

}

*@Override*

public String toString() {

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

}

public static void main(String[] args) {

List<Employee> employeeList = new ArrayList<>();

Employee emp1 = new Employee(101, "rushi", 50000);

Employee emp2 = new Employee(102, "snhea", 60000);

employeeList.add(emp1);

employeeList.add(emp2);

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

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

for (Employee emp : employeeList) {

emp.applyRaise(10);

}

System.***out***.println("\nAfter 10% raise:");

for (Employee emp : employeeList) {

System.***out***.println(emp);

}

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

emp1.setSalary(70000);

System.***out***.println("\nAfter updating Alice's salary:");

for (Employee emp : employeeList) {

System.***out***.println(emp);

}

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

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

}

}

