**Assignment 5**

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

package org.solution.ques1;

public class InstanceCounter {

private static int *instanceCount* = 0;

public InstanceCounter(){

*instanceCount*++;

}

public static int getInstanceCount() {

return *instanceCount*;

}

public static void main(String[] args) {

InstanceCounter obj1 = new InstanceCounter();

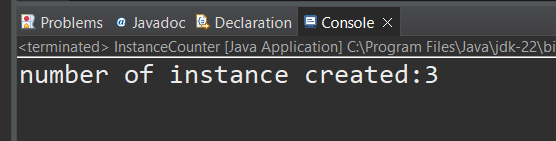
InstanceCounter obj2 = new InstanceCounter();

InstanceCounter obj3 = new InstanceCounter();

System.*out*.println("number of instance created:" + 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.

public class Logger {

private static Logger loggerInstance = null;

private StringBuilder logMessages;

private Logger() {

logMessages = new StringBuilder();

}

public static Logger getInstance() {

if (loggerInstance == null) {

loggerInstance = new Logger(); // Create instance only if not already created

}

return loggerInstance;

}

public void log(String message) {

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

}

public String getLog() {

return logMessages.toString();

}

public void clearLog() {

logMessages.setLength(0); // Clears the StringBuilder

}

public static void main(String[] args) {

Logger logger = Logger.getInstance();

logger.log("This is the first log message.");

logger.log("This is the second log message.");

System.out.println("Current log messages:\n" + logger.getLog());

logger.clearLog();

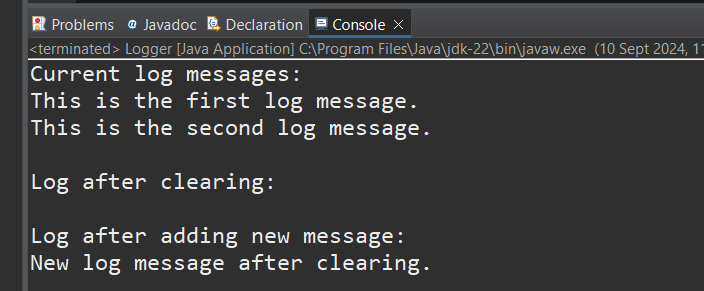
System.out.println("Log after clearing:\n" + logger.getLog());

logger.log("New log message after clearing.");

System.out.println("Log after adding new message:\n" + 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.

package org.solution.ques3;

import java.util.ArrayList;

import java.util.Scanner;

public class Employee {

private static int *totalEmployees* = 0;

private static double *totalSalaryExpense* = 0;

private int employeeID;

private String name;

private double salary;

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

this.employeeID = employeeID;

this.name = name;

this.salary = salary;

*totalEmployees*++;

*totalSalaryExpense* += salary;

}

public static int getTotalEmployees() {

return *totalEmployees*;

}

public static double calculateTotalSalaryExpense() {

return *totalSalaryExpense*;

}

public static void applyRaise(double percentage, ArrayList<Employee> employees) {

for (Employee emp : employees) {

double raise = emp.salary \* (percentage / 100);

emp.salary += raise;

*totalSalaryExpense* += raise;

}

}

public void updateSalary(double newSalary) {

*totalSalaryExpense* = *totalSalaryExpense* - this.salary + newSalary;

this.salary = newSalary;

}

*@Override*

public String toString() {

return "ID: " + employeeID + ", Name: " + name + ", Salary: " + salary;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

ArrayList<Employee> employees = new ArrayList<>();

while (true) {

System.***out***.println("\n1. Add Employee 2. Total Employees 3. Apply Raise 4. Total Salary 5. Update Salary 6. Show Employees 7. Exit");

int choice = sc.nextInt();

switch (choice) {

case 1:

System.***out***.print("ID: ");

int id = sc.nextInt();

sc.nextLine();

System.***out***.print("Name: ");

String name = sc.nextLine();

System.***out***.print("Salary: ");

double salary = sc.nextDouble();

employees.add(new Employee(id, name, salary));

break;

case 2:

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

break;

case 3:

System.***out***.print("Raise %: ");

double percentage = sc.nextDouble();

Employee.*applyRaise*(percentage, employees);

break;

case 4:

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

break;

case 5:

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

int updateID = sc.nextInt();

for (Employee emp : employees) {

if (emp.employeeID == updateID) {

System.***out***.print("New Salary: ");

emp.updateSalary(sc.nextDouble());

}

}

break;

case 6:

employees.forEach(System.***out***::println);

break;

case 7:

sc.close();

return;

default:

System.***out***.println("Invalid choice. Try again.");

}

}

}

}

