**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.

**InstanceCounter Class**

**package main;**

**public class InstanceCounter {**

**private static int *instanceCount* = 0;**

**static {**

**System.*out*.println("Static initializer executed.");**

**}**

**public InstanceCounter() {**

***instanceCount*++;**

**}**

**public static int getInstanceCount() {**

**return *instanceCount*;**

**}**

**@Override**

**public String toString() {**

**return "InstanceCounter [getClass()=" + getClass() + ", hashCode()=" + hashCode() + ", toString()="**

**+ super.toString() + "]";**

**}**

**}**

**Testing InstanceCounter Class**

**package main;**

**public class Program {**

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

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

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

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

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

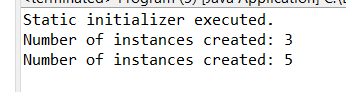
**InstanceCounter obj4 = new InstanceCounter();**

**InstanceCounter obj5 = new InstanceCounter();**

**System.*out*.println("Number of 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.

Logger Class

**package** main;

**public** **class** Logger {

**private** **static** Logger *instance*;

**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);

}

}

Testing Logger Class

**package** main;

**public** **class** Program {

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

Logger logger = Logger.*getInstance*();

logger.log("Application started.");

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

logger.log("Error: Unable to fetch data.");

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

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

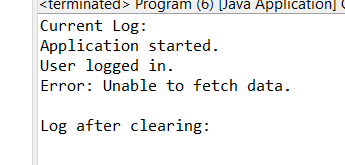
logger.clearLog();

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

System.***out***.println(logger.getLog()); // Should print an empty string

}

}



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.

Employee Class

**package** main;

**public** **class** Employee {

**private** **static** **int** *totalEmployees* = 0;

**private** **static** **double** *totalSalaryExpense* = 0.0;

**private** **int** id;

**private** String name;

**private** **double** salary;

**static** {

System.***out***.println("Static initializer block executed.");

}

{

*totalEmployees*++;

}

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

**this**.id = id;

**this**.name = name;

setSalary(salary);

}

**public** **static** **int** getTotalEmployees() {

**return** *totalEmployees*;

}

**public** **static** **double** getTotalSalaryExpense() {

**return** *totalSalaryExpense*;

}

**public** **static** **void** applyRaise(**double** percentage) {

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

}

**public** **static** **double** calculateTotalSalaryExpense() {

**return** *totalSalaryExpense*;

}

**public** **void** updateSalary(**double** newSalary) {

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

**this**.salary = newSalary;

}

**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) {

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";

}

}

Testing Employee Class

**package** main;

**public** **class** Program {

**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);

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

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

Employee.*applyRaise*(10);

System.***out***.println("Total Salary Expense after 10% raise: ₹" + Employee.*calculateTotalSalaryExpense*());

emp1.updateSalary(70000);

System.***out***.println(emp1);

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

}

}

