**Assignment No-6**

**Name:** Yash Bandu Dhole.

**Center:** Juhu.

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

**Code:**

package problem1;

class demo {

// static field

public static int *counter* = 0;

demo() {

demo.*counter*++;

System.***out***.println("Instance created: " + demo.*counter*);

}

}

public class instanceCounter {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

demo d = new demo();

demo d1 = new demo();

demo d2 = new demo();

}

}

**o/p:**

Instance created: 1

Instance created: 2

Instance created: 3

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:**

package problem2;

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 class singletonPattern {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Logger logger = Logger.*getInstance*();

logger.log("first logg");

logger.log("second logg");

System.***out***.println("log Messages: ");

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

logger.clearLog();

System.***out***.println("log cleared");

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

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

}

}

**o/p:**

log Messages:

first logg

second logg

log cleared

Current log:

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.

**Code:**

package problem3;

import java.util.Scanner;

class Employee {

// Static fields

private static int *totalEmployees* = 0;

private static double *totalSalaryExpense* = 0;

// Non-static fields

private int employeeID;

private String employeeName;

private double salary;

// Static initializer

static {

*totalEmployees* = 0;

*totalSalaryExpense* = 0;

}

// Constructor

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

this.employeeID = id;

this.employeeName = name;

this.salary = salary;

*totalEmployees*++; // Increment total employees

*totalSalaryExpense* += salary; // Add employee's salary to total expense

}

public static int getTotalEmployees() {

return *totalEmployees*;

}

public static void applyRaise(double percentage, Employee[] employees) {

for (Employee emp : employees) {

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

emp.salary += raiseAmount;

*totalSalaryExpense* += raiseAmount;

}

}

public static double calculateTotalSalaryExpense() {

return *totalSalaryExpense*;

}

public void updateSalary(double newSalary) {

*totalSalaryExpense* -= this.salary;

this.salary = newSalary;

*totalSalaryExpense* += newSalary;

}

public int getEmployeeID() {

return employeeID;

}

public String getEmployeeName() {

return employeeName;

}

public double getSalary() {

return salary;

}

public void setEmployeeName(String employeeName) {

this.employeeName = employeeName;

}

// toString method to display employee details

*@Override*

public String toString() {

return "Employee ID: " + employeeID + ", Name: " + employeeName + ", Salary: " + salary;

}

}

public class empMain {

public static void main(String[] args) {

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

Employee[] employees = new Employee[4];

employees[0] = new Employee(1, "Yash Dhole", 50000);

employees[1] = new Employee(2, "Saurabh Pawar", 60000);

employees[2] = new Employee(3, "Kshitij Deshpande", 55000);

employees[3] = new Employee(4, "Vaibhav Kale", 75000);

boolean exit = false;

while (!exit) {

System.***out***.println("\n--- Employee Management System ---");

System.***out***.println("1. View total employees");

System.***out***.println("2. Apply salary raise to all employees");

System.***out***.println("3. Calculate total salary expense");

System.***out***.println("4. Update salary for an employee");

System.***out***.println("5. View all employees");

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

System.***out***.print("Choose an option: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

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

break;

case 2:

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

double percentage = scanner.nextDouble();

Employee.*applyRaise*(percentage, employees);

System.***out***.println("Raise applied to all employees.");

break;

case 3:

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

break;

case 4:

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

int empID = scanner.nextInt();

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

double newSalary = scanner.nextDouble();

for (Employee emp : employees) {

if (emp.getEmployeeID() == empID) {

emp.updateSalary(newSalary);

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

break;

}

}

break;

case 5:

System.***out***.println("\n--- Employee Details ---");

for (Employee emp : employees) {

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

}

break;

case 6:

exit = true;

break;

default:

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

break;

}

}

scanner.close();

}

}

**O/p:**

--- Employee Management System ---

1. View total employees

2. Apply salary raise to all employees

3. Calculate total salary expense

4. Update salary for an employee

5. View all employees

6. Exit

Choose an option: 5

--- Employee Details ---

Employee ID: 1, Name: Yash Dhole, Salary: 50000.0

Employee ID: 2, Name: Saurabh Pawar, Salary: 60000.0

Employee ID: 3, Name: Kshitij Deshpande, Salary: 55000.0

Employee ID: 4, Name: Vaibhav Kale, Salary: 75000.0

--- Employee Management System ---

1. View total employees

2. Apply salary raise to all employees

3. Calculate total salary expense

4. Update salary for an employee

5. View all employees

6. Exit

Choose an option: 4

Enter Employee ID to update salary: 3

Enter new salary: 65000

Salary updated.

--- Employee Management System ---

1. View total employees

2. Apply salary raise to all employees

3. Calculate total salary expense

4. Update salary for an employee

5. View all employees

6. Exit

Choose an option: 5

--- Employee Details ---

Employee ID: 1, Name: Yash Dhole, Salary: 50000.0

Employee ID: 2, Name: Saurabh Pawar, Salary: 60000.0

Employee ID: 3, Name: Kshitij Deshpande, Salary: 65000.0

Employee ID: 4, Name: Vaibhav Kale, Salary: 75000.0

--- Employee Management System ---

1. View total employees

2. Apply salary raise to all employees

3. Calculate total salary expense

4. Update salary for an employee

5. View all employees

6. Exit

Choose an option: 6