

# ASSIGNMENT 5

Kunal More

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

## P1. Java

```
package Five.Assignment;

public class p1
{
    public static void main(String a[])
    {
        instanceCount obj1 = new instanceCount()
        ;
        instanceCount obj2 = new instanceCount()
        ;
        instanceCount obj3 = new instanceCount()
        ;
        instanceCount obj4 = new instanceCount()
        ;
        instanceCount obj5 = new instanceCount()
        ;
        instanceCount obj6 = new instanceCount()
        ;
        instanceCount obj7 = new instanceCount()
        ;
        instanceCount obj8 = new instanceCount()
        ;
        instanceCount obj9 = new instanceCount()
        ;

        System.out.println("INSTANCES "+ instanceCount.getInstance())
    }
}
```

# ASSIGNMENT 5

## InstanceCount.java

```
package Five.Assignment;

public class instanceCount
{
    public static int instancecount = 0
;
    public instanceCount()
    {
        instancecount++
        ;
    }

    public static int getinstance()
    {
        return instancecount
        ;
    }

    public String toString()
    {
        return "Total Instances created : " +
instancecount
        ;
    }
}
```

# ASSIGNMENT 5

The screenshot shows the Eclipse IDE interface. The code editor displays a Java file named p1.java with the following content:

```
3 public class p1
4
5•     public static void main(String a[])
6         instanceCount obj1 = new instanceCount()
7         instanceCount obj2 = new instanceCount()
8         instanceCount obj3 = new instanceCount()
9         instanceCount obj4 = new instanceCount()
10        instanceCount obj5 = new instanceCount()
11        instanceCount obj6 = new instanceCount()
12        instanceCount obj7 = new instanceCount()
13        instanceCount obj8 = new instanceCount()
14        instanceCount obj9 = new instanceCount()
15
16        System.out.println("INSTANCES " + instanceCount.getinsta
17
18
```

The console tab at the bottom shows the output of the program:

```
Console × Problems Debug Shell
<terminated> p1 (6) [Java Application] C:\Users\ACER\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2
INSTANCES 9
```

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.

## P2.java

```
package Five.Assignment;

class Logger {
    private static Logger LoggerInstance = null;
    private String logMessages;
    private Logger() {
        logMessages = "";
    }

    public static Logger getInstance() {
        if (LoggerInstance == null) {
            LoggerInstance = new Logger();
        }
        return LoggerInstance;
    }
}
```

# ASSIGNMENT 5

```
public void log(String message) {
    logMessages += message + "\n";
}
public String getLog() {
    return logMessages;
}
public void clearLog() {
    logMessages = "";
}
@Override
public String toString() {
    return "Logger: " + getLog();
}
```

## Log.java

```
package Five.Assignment;

public class log
{
    public static void main(String[] args)
    {
        Logger logger = Logger.getInstance()
        ;
        logger.log("Application started.")
        ;
        logger.log("User logged in.")
        ;
        logger.log("Error: Database connection failed.")
        ;
        System.out.println("Current Log: ")
        ;
        System.out.println(logger.getLog())
        ;
        logger.clearLog()
        ;
        System.out.println("Log after clearing: ")
        ;
    }
}
```

# ASSIGNMENT 5

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

}
```

## Output 😊

The screenshot shows the Eclipse IDE interface. On the left, there's a package explorer with several Java files listed. In the center, a code editor window displays a Java class named 'Logger'. The code includes static methods for getting an instance and retrieving log messages, along with a method to clear the log. A tooltip for the static variable 'loggerInstance' is visible, pointing to its definition in the class. On the right, a terminal window titled 'Console' shows the output of running the application. It prints 'User logged in.' followed by 'Error: Database connection failed.' and then 'Log after clearing:'.

```
> log.java
> p1.java
> p2.java
> p3.java
# four.assignment
# nine.Sep
# org.example
# three.assignmen
assWorkPractise
ay_9.1
ur.assignment.mam
sting

10*     public static Logger getInstance() {
11         if (LoggerInstance == null) {
12             LoggerInstance = new Logger();
13         }
14         return LoggerInstance;
15     }
16     public
17     logMess
18     }
19     }
20*     public String getLog() {
21         return logMessages;
22     }
23*     public void clearLog() {
24         logMessages = "";
```

Console X Problems Debug Shell  
<terminated> log [Java Application] C:\Users\ACER\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v2  
User logged in.  
Error: Database connection failed.  
  
Log after clearing:

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.

# ASSIGNMENT 5

Write a menu-driven program in the `main` method to test the functionalities.

Employee.java

```
package Five.Assignment;

public class Employee{
    private static int totalEmployees = 0;
    private static double totalSalaryExpense = 0.0;
//  ASSIGNMENT NO.6
    int id;
    private String name;
    private double salary;
    public void Employee(int id, String name, double
salary) {
        this.id = id;
        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) {
        totalSalaryExpense += totalSalaryExpense *
(percentage / 100);
    }
    public void updateSalary(double newSalary) {
        totalSalaryExpense -= this.salary;
        this.salary = newSalary;
        totalSalaryExpense += this.salary;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
```

# ASSIGNMENT 5

```
this.name = name;
}

public double getSalary() {
    return salary;
}
public void setSalary(double salary) {
    this.salary = salary;
// ASSIGNMENT NO.6
}

@Override
public String toString() {
    return "Employee ID: " + id + ", Name: " + name + ",\nSalary: ₹" + salary;
}
}
```

Program.java

```
package Five.Assignment;

import java.util.Scanner;

public class program {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Employee[] employees = new Employee[100];
        int empCount = 0;

        int choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Add Employee");
            System.out.println("2. Apply Raise to All Employees");
            System.out.println("3. Update Individual Employee Salary");
```

# ASSIGNMENT 5

```
        System.out.println("4. Display Total  
Employees");  
        System.out.println("5. Display Total Salary  
Expense");  
        System.out.println("6. Exit");  
        System.out.print("Enter your choice: ");  
        choice = sc.nextInt();  
switch (choice) {  
    case 1:  
  
        System.out.print("Enter Employee ID:  
");  
        int id = sc.nextInt();  
        sc.nextLine();  
        System.out.print("Enter Employee Name:  
");  
        String name = sc.nextLine();  
        System.out.print("Enter Employee  
Salary: ₹");  
        double salary = sc.nextDouble();  
        employees[empCount++] = new  
Employee();  
        System.out.println("Employee added  
successfully!");  
        break;  
  
    case 2:  
  
        System.out.print("Enter raise  
percentage: ");  
        double raisePercentage =  
sc.nextDouble();  
        Employee.applyRaise(raisePercentage);  
        System.out.println("Raise applied to  
all employees.");  
        break;  
  
    case 3:  
  
        System.out.print("Enter Employee ID to  
update salary: ");  
        int updateId = sc.nextInt();  
        Employee.updateSalary(updateId, salary);  
        System.out.println("Salary updated successfully!");  
        break;  
}  
System.out.println("Program ended");
```

# ASSIGNMENT 5

```
        boolean found = false;
        for (int i = 0; i < empCount; i++) {
            if (employees[i].id == updateId) {
                System.out.print("Enter new
salary: ₹");
                double newSalary =
sc.nextDouble();

employees[i].updateSalary(newSalary);
                System.out.println("Salary
updated for " + employees[i].getName());
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Employee with
ID " + updateId + " not found.");
        }
        break;
case 4:
    System.out.println("Total Employees: "
+ Employee.getTotalEmployees());
    break;

case 5:
    System.out.println("Total Salary
Expense: ₹" + Employee.calculateTotalSalaryExpense());
    break;

case 6:
    System.out.println("Exiting...");
    break;

default:
    System.out.println("Invalid choice.
Please try again.");
    break;
}
```

# ASSIGNMENT 5

```
        } while (choice != 6);

        sc.close();
    }
}
```

OUTPUT 😊

The screenshot shows the Eclipse IDE interface. On the left, there's a file tree with several Java files like 'Assignment.java', 'Employee.java', and 'EmployeeManagement.java'. The main area displays Java code for a menu system. The code includes a menu with options 1 through 6 and logic for adding employees. The console tab at the bottom shows the execution of the program. It prompts for employee details (ID 13, Name Kunal, Salary 808080), adds the employee successfully, and then displays the menu again with option 4 selected (Display Total Employees). The total number of employees is shown as 0.

```
14     System.out.println("1. Add Employee"),
15     System.out.println("2. Apply Raise to All Employees");
16     System.out.println("3. Update Individual Employee Salary");
17     System.out.println("4. Display Total Employees");
18     System.out.println("5. Display Total Salary Expense");
19     System.out.println("6. Exit");
20     System.out.print("Enter your choice: ");

Enter Employee ID: 13
Enter Employee Name: Kunal
Enter Employee Salary: ₹808080
Employee added successfully!

Menu:
1. Add Employee
2. Apply Raise to All Employees
3. Update Individual Employee Salary
4. Display Total Employees
5. Display Total Salary Expense
6. Exit
Enter your choice: 4
Total Employees: 0

Menu:
1. Add Employee
2. Apply Raise to All Employees
3. Update Individual Employee Salary
4. Display Total Employees
5. Display Total Salary Expense
6. Exit
Enter your choice:
```