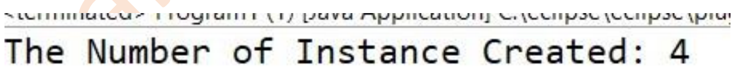
# ASSIGNMENT NO. 06

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

**CODE:**

```
package org.assignment5;

class InstanceCounter{
        private static int count = 0;

        public InstanceCounter(){
                count++;
        }

        public static int displayCount() {
                return count++;
        }
}
public class Program1 {

        public static void main(String[] args) {
                InstanceCounter inst1 = new InstanceCounter();
                InstanceCounter inst2 = new InstanceCounter();
                InstanceCounter inst3 = new InstanceCounter();
                InstanceCounter inst4 = new InstanceCounter();


                System.out.println("The Number of Instance Created: 
"+InstanceCounter.displayCount());
        }
}
```

**OUTPUT:**



```
The Number of Instance Created: 4
```

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.

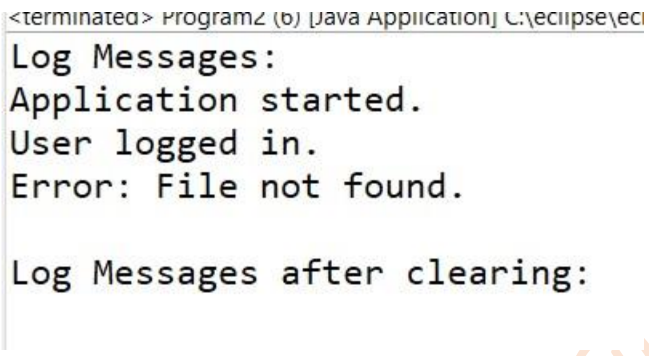
- **`getLog()`**: Returns the current log messages as a `String`.
- **`clearLog()`**: Clears all log messages.

**CODE:**

```
package org.assignment5;

class Logger {
   private static Logger instance;

   private StringBuilder logMessages;

   private Logger() {
      logMessages = new StringBuilder();
   }
   public static synchronized Logger getInstance() {
      if (instance == null) {
         instance = new Logger();
      }
      return instance;
   }

   public void log(String message) {
      logMessages.append(message).append(System.lineSeparator());
   }

   public String getLog() {
      return logMessages.toString();
   }


   public void clearLog() {
      logMessages.setLength(0); // Clear the StringBuilder
   }
}

public class Program2 {

         public static void main(String[] args) {
                  Logger logger = Logger.getInstance();


               logger.log("Application started.");
               logger.log("User logged in.");
               logger.log("Error: File not found.");


               System.out.println("Log Messages:");
               System.out.println(logger.getLog());
```

```
                logger.clearLog();


                System.out.println("Log Messages after clearing:");
                System.out.println(logger.getLog());
            }
    }
```

**OUTPUT:**



```
<terminated> Program2 (6) [Java Application] C:\eclipse\ec
Log Messages:
Application started.
User logged in.
Error: File not found.

Log Messages 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.

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

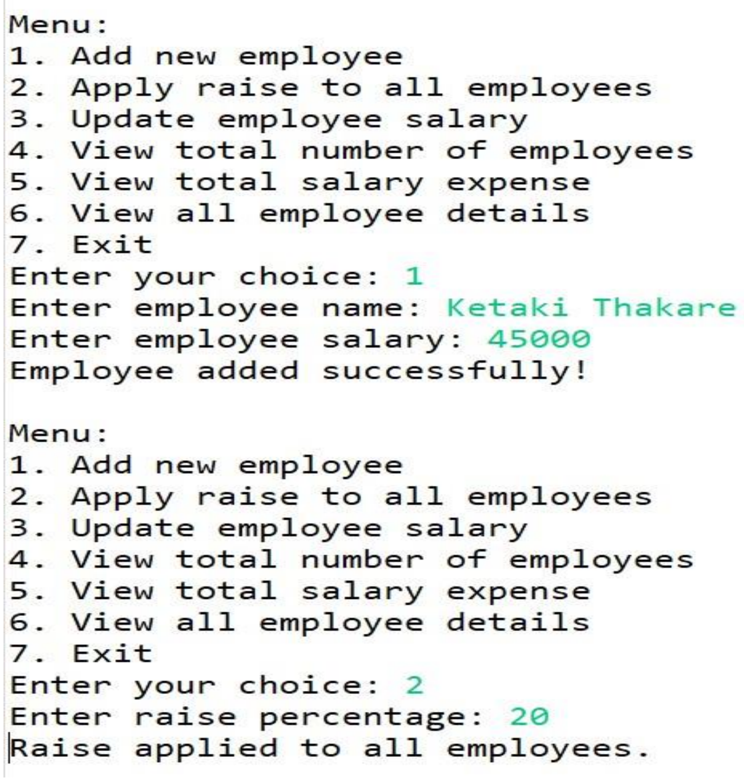
**CODE:**

```
package org.assignment5;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class Employee {
    private static int totalEmployees = 0;
    private static double totalSalaryExpense = 0.0;

    private int employeeID;
    private String name;
    private double salary;
    static {
        totalEmployees = 0;
        totalSalaryExpense = 0.0;
    }
    public Employee(String name, double salary) {
        this.employeeID = ++totalEmployees;
        this.name = name;
        this.salary = salary;
        totalSalaryExpense += salary;
    }
    public int getEmployeeID() {
        return employeeID;
    }

    public String getName() {
        return 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 raise = this.salary * (percentage / 100);
        setSalary(this.salary + raise);
    }

    @Override
    public String toString() {
        return "Employee ID: " + employeeID + ", Name: " + name + ", Salary: " + salary;
    }
}

public class Program3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Employee> employeeList = new ArrayList<>();

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add new employee");
            System.out.println("2. Apply raise to all employees");
            System.out.println("3. Update employee salary");
            System.out.println("4. View total number of employees");
            System.out.println("5. View total salary expense");
            System.out.println("6. View all employee details");
            System.out.println("7. Exit");

            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter employee name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter employee salary: ");
                    double salary = scanner.nextDouble();
                    employeeList.add(new Employee(name, salary));
                    System.out.println("Employee added successfully!");
                    break;

                case 2:
                    System.out.print("Enter raise percentage: ");
                    double percentage = scanner.nextDouble();
                    for (Employee emp : employeeList) {
                        emp.applyRaise(percentage);
                    }
                    System.out.println("Raise applied to all employees.");
                    break;

                case 3:
                    System.out.print("Enter employee ID to update salary: ");
                    int id = scanner.nextInt();
                    Employee employeeToUpdate = null;
                    for (Employee emp : employeeList) {
                        if (emp.getEmployeeID() == id) {
                            employeeToUpdate = emp;
                            break;
                        }
                    }
                    if (employeeToUpdate != null) {
                        System.out.print("Enter new salary: ");
                        double newSalary = scanner.nextDouble();
                        employeeToUpdate.setSalary(newSalary);
                        System.out.println("Salary updated successfully.");
                    } else {
                        System.out.println("Employee not found.");
                    }
                    break;

                case 4:
                    System.out.println("Total number of employees: " +
Employee.getTotalEmployees());
                    break;

                case 5:
                    System.out.println("Total salary expense: " +
Employee.calculateTotalSalaryExpense());
                    break;

                case 6:
                    for (Employee emp : employeeList) {
                        System.out.println(emp);
                    }
                    break;

                case 7:
                    System.out.println("Exiting program...");
                    scanner.close();
                    return;

                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
}
```

**OUTPUT:**

```
Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 1
Enter employee name: Ketaki Thakare
Enter employee salary: 45000
Employee added successfully!

Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 2
Enter raise percentage: 20
Raise applied to all employees.

Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 3
Enter employee ID to update salary: 101
Employee not found.

Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 4
Total number of employees: 1

Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 5
Total salary expense: 54000.0
```

```
Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 6
Employee ID: 1, Name: Ketaki Thakare, Salary: 54000.0

Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 7
Exiting program...
```



```
Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 1
Enter employee name: Ketaki Thakare
Enter employee salary: 45000
Employee added successfully!

Menu:
1. Add new employee
2. Apply raise to all employees
3. Update employee salary
4. View total number of employees
5. View total salary expense
6. View all employee details
7. Exit
Enter your choice: 2
Enter raise percentage: 20
Raise applied to all employees.
```