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

```
class InstanceCounter {
    private static int instanceCount = 0;

    // Static initializer block
    static {
        System.out.println("InstanceCounter class loaded.");
    }

    // Constructor increments the instance count
    public InstanceCounter() {
        instanceCount++;
    }

    // Static method to retrieve the instance count
    public static int getInstanceCount() {
        return instanceCount;
    }

    @Override
    public String toString() {
        return "InstanceCounter with instance count: " + instanceCount;
    }
}
```

Que2) 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.

```
class Logger {
    private static Logger instance;
    private StringBuilder logMessages;

    // Static initializer block
    static {
        System.out.println("Logger class loaded.");
    }

    // Private constructor ensures only one instance is created
    private Logger() {
        logMessages = new StringBuilder();
    }

    // Static method to return the unique instance of Logger
    public static Logger getInstance() {
        if (instance == null) {
            instance = new Logger();
        }
        return instance;
    }

    // Method to log messages
    public void log(String message) {
        logMessages.append(message).append("\n");
    }
```

```
    // Method to retrieve all log messages
    public String getLog() {
        return logMessages.toString();
    }

    // Method to clear all log messages
    public void clearLog() {
        logMessages.setLength(0);
    }
}
```

Que3) 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.

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

    private int id;
```

```
private String name;
private double salary;

// Static initializer to initialize static fields
static {
    System.out.println("Employee class loaded.");
}

// Constructor to initialize an employee
public Employee(int id, String name, double salary) {
    this.id = id;
    this.name = name;
    this.salary = salary;
    totalEmployees++;
    totalSalaryExpense += salary;
}

// Static method to retrieve the total number of employees
public static int getTotalEmployees() {
    return totalEmployees;
}

// Static method to calculate the total salary expense
public static double calculateTotalSalaryExpense() {
    return totalSalaryExpense;
}

// Static method to apply a percentage raise to all employees
public static void applyRaise(double percentage, Employee[] employees) {
    for (Employee emp : employees) {
        double raiseAmount = emp.salary * (percentage / 100);
```

```
            emp.updateSalary(emp.salary + raiseAmount);
        }
    }

    // Method to update the salary of an individual employee
    public void updateSalary(double newSalary) {
        totalSalaryExpense -= this.salary;
        this.salary = newSalary;
        totalSalaryExpense += newSalary;
    }

    // Getter and Setter methods
    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    // toString method to display employee details
    @Override
    public String toString() {
        return "Employee ID: " + id + ", Name: " + name + ", Salary: " + salary;
    }
}
import java.util.Scanner;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Logger instance
        Logger logger = Logger.getInstance();

        // Employee array
        Employee[] employees = new Employee[3];
        employees[0] = new Employee(1, "John", 50000);
        employees[1] = new Employee(2, "Jane", 60000);
        employees[2] = new Employee(3, "Mike", 55000);

        // Menu-driven program
        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Create InstanceCounter object");
            System.out.println("2. Log a message");
            System.out.println("3. Show log messages");
            System.out.println("4. Clear log messages");
            System.out.println("5. Display total employees");
            System.out.println("6. Apply raise to all employees");
            System.out.println("7. Display employee details");
            System.out.println("8. Display total salary expense");
            System.out.println("9. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
```

```
            new InstanceCounter();
            System.out.println("Current instance count: " + InstanceCounter.getInstanceCount());
            logger.log("Created a new InstanceCounter object.");
            break;
        case 2:
            System.out.print("Enter a log message: ");
            scanner.nextLine();  // Consume newline
            String message = scanner.nextLine();
            logger.log(message);
            System.out.println("Message logged.");
            break;
        case 3:
            System.out.println("Log Messages:");
            System.out.println(logger.getLog());
            break;
        case 4:
            logger.clearLog();
            System.out.println("Log cleared.");
            break;
        case 5:
            System.out.println("Total employees: " + Employee.getTotalEmployees());
            break;
        case 6:
            System.out.print("Enter raise percentage: ");
            double percentage = scanner.nextDouble();
            Employee.applyRaise(percentage, employees);
            System.out.println("Applied raise to all employees.");
            break;
        case 7:
            for (Employee emp : employees) {
                System.out.println(emp);
```

```
                }
                break;
            case 8:
                System.out.println("Total salary expense: " + Employee.calculateTotalSalaryExpense());
                break;
            case 9:
                System.out.println("Exiting...");
                System.exit(0);
                break;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
  }
}
```