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

```
public class InstanceCounter {
    // Static field to count instances
    private static int instanceCount = 0;

    // Static initializer
    static {
        instanceCount = 0;
    }

    // Constructor
    public InstanceCounter() {
        instanceCount++;
    }

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

    @Override
    public String toString() {
        return "InstanceCounter{" + "instances created=" + instanceCount + '}';
    }
}
```

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.

```
public class Logger {

    // Static field for the unique instance

    private static Logger instance = null;
```

```
// Log message storage
private StringBuilder logMessages;

// Private constructor (singleton)
private Logger() {
    logMessages = new StringBuilder();
}

// Static method to get the single instance
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 get log messages
public String getLog() {
    return logMessages.toString();
}
```

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

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.

```
public class Employee {
    // Static fields for employee count and total salary expense
    private static int totalEmployees = 0;
    private static double totalSalaryExpense = 0;

    // Instance fields for employee details
    private int id;
    private String name;
    private double salary;
```

```
// Static initializer
static {
    totalEmployees = 0;
    totalSalaryExpense = 0;
}

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

// Static method to get total employees
public static int getTotalEmployees() {
    return totalEmployees;
}

// Static method to apply a raise to all employees
public static void applyRaise(double percentage) {
    totalSalaryExpense += totalSalaryExpense * (percentage / 100);
}
```

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

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

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

public String getName() {
    return name;
}

public double getSalary() {
    return salary;
}
```

```
    // toString() method for employee representation
    @Override
    public String toString() {
        return "Employee{id=" + id + ", name='" + name + "', salary=" + salary + '}';
    }
}
```