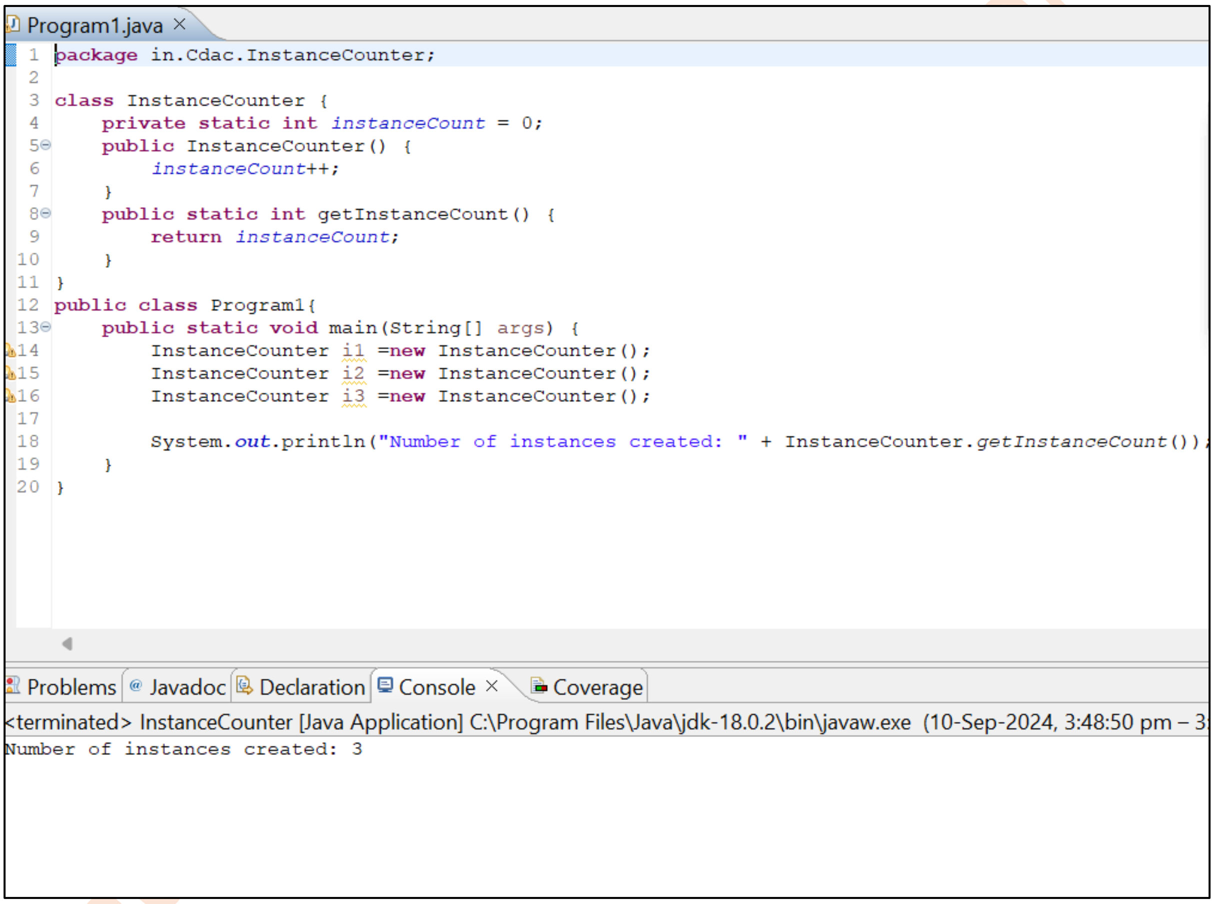
ASSIGNMENT NO.6

Note:

1. This assignment is designed to practice static fields, static initializers, and static methods.
2. Understand the problem statement and use static and non-static wisely to solve the problem.
3. Use constructors, proper getter/setter methods, and `toString()` wherever required.

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



```java
package in.Cdac.InstanceCounter;

class InstanceCounter {
    private static int instanceCount = 0;
    public InstanceCounter() {
        instanceCount++;
    }
    public static int getInstanceCount() {
        return instanceCount;
    }
}
public class Program1{
    public static void main(String[] args) {
        InstanceCounter i1 =new InstanceCounter();
        InstanceCounter i2 =new InstanceCounter();
        InstanceCounter i3 =new InstanceCounter();

        System.out.println("Number of instances created: " + InstanceCounter.getInstanceCount());
    }
}
```

```
<terminated> InstanceCounter [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 3:48:50 pm – 3
Number of instances created: 3
```

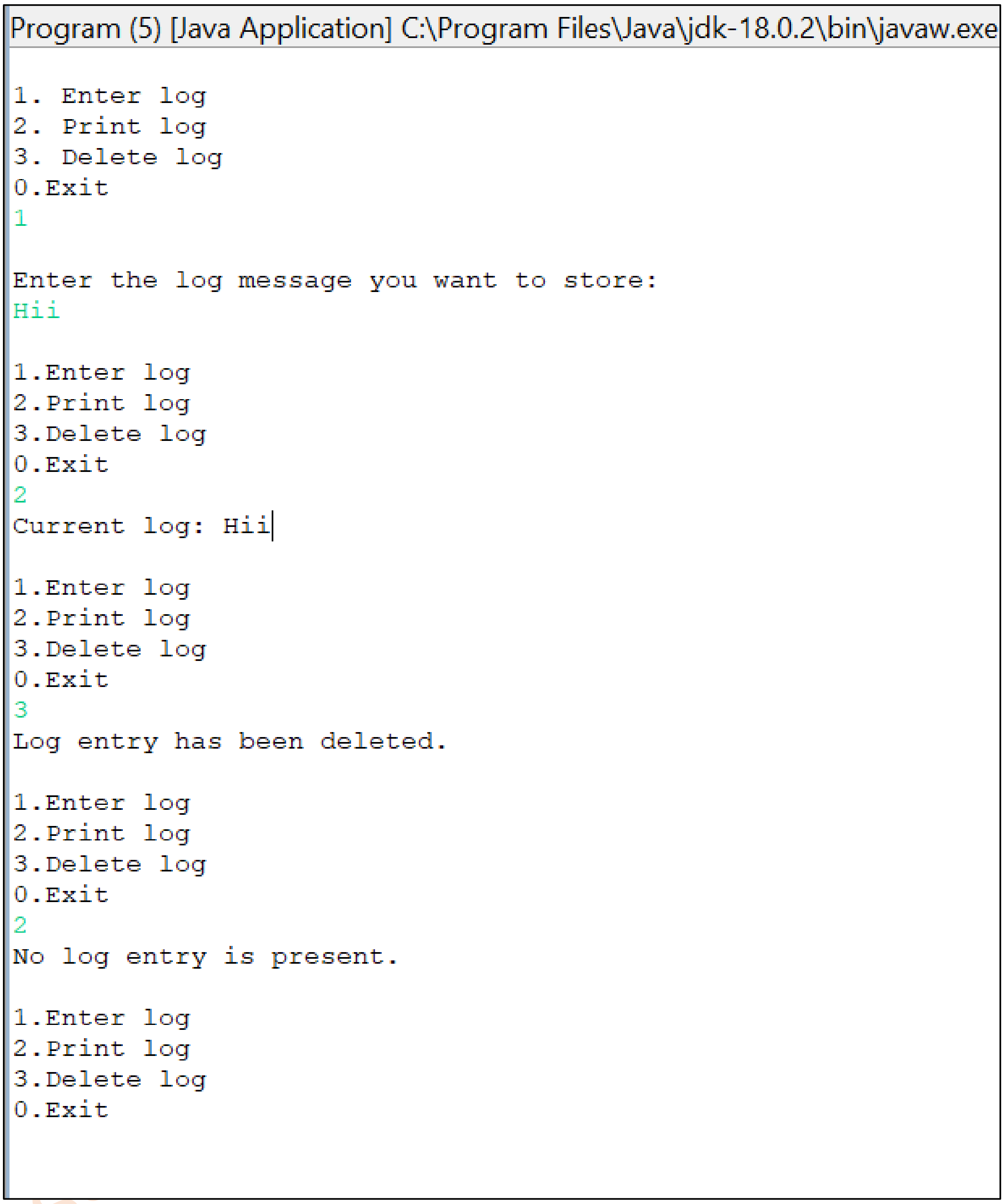
2. Design and implement a class named `Logger` to manage logging messages for an application. The class should be implemented as a singleton to ensure that only one instance of the `Logger` exists throughout the application.

   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.

```
package in.Cdac.Logger;

public class Logger {
	private static Logger l1;
	private static String message;

	static {
		Logger.l1 = null;
		Logger.message = "";
	}

	private Logger() { // private constructor to restrict creation of
instance from outside the class

	}

	public static Logger getInstance() {
		if (Logger.l1 == null) { // check if there is no instance
inside reference l1
			Logger.l1 = new Logger(); // only create new instance
of Logger Class if there is no instance present in
										// reference
l1
		}
		return l1;
	}

	public void log(String message) {

		Logger.message = message;

	}

	public String getLog() {
		return Logger.message;
	}

	public void clearLog() {
		Logger.message = "";

	}
}
package in.Cdac.Logger;

import java.util.Scanner;

public class Program {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int choice;

		Logger lg = Logger.getInstance();

		System.out.println("\n1. Enter log\n2. Print log\n3. Delete
log\n0.Exit");
		while ((choice = sc.nextInt()) != 0) {

			switch (choice) {
```

```
                              case 1:
                                        sc.nextLine();
                                        System.out.println("\nEnter the log message you
want to store: ");
                                        String s1 = sc.nextLine();
                                        lg.log(s1);
                                        break;

                              case 2:
                                        if (lg.getLog() == "")
                                                  System.out.println("No log entry is
present.");
                                        else
                                                  System.out.println("Current log: " +
lg.getLog());
                                        break;

                              case 3:
                                        lg.clearLog();
                                        System.out.println("Log entry has been
deleted.");
                                        break;

                              }
                              System.out.println("\n1.Enter log\n2.Print
log\n3.Delete log\n0.Exit");

                    }

                    sc.close();

          }
}
```



```
Program (5) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe

1. Enter log
2. Print log
3. Delete log
0.Exit
1

Enter the log message you want to store:
Hii

1.Enter log
2.Print log
3.Delete log
0.Exit
2
Current log: Hii

1.Enter log
2.Print log
3.Delete log
0.Exit
3
Log entry has been deleted.

1.Enter log
2.Print log
3.Delete log
0.Exit
2
No log entry is present.

1.Enter log
2.Print log
3.Delete log
0.Exit
```

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:

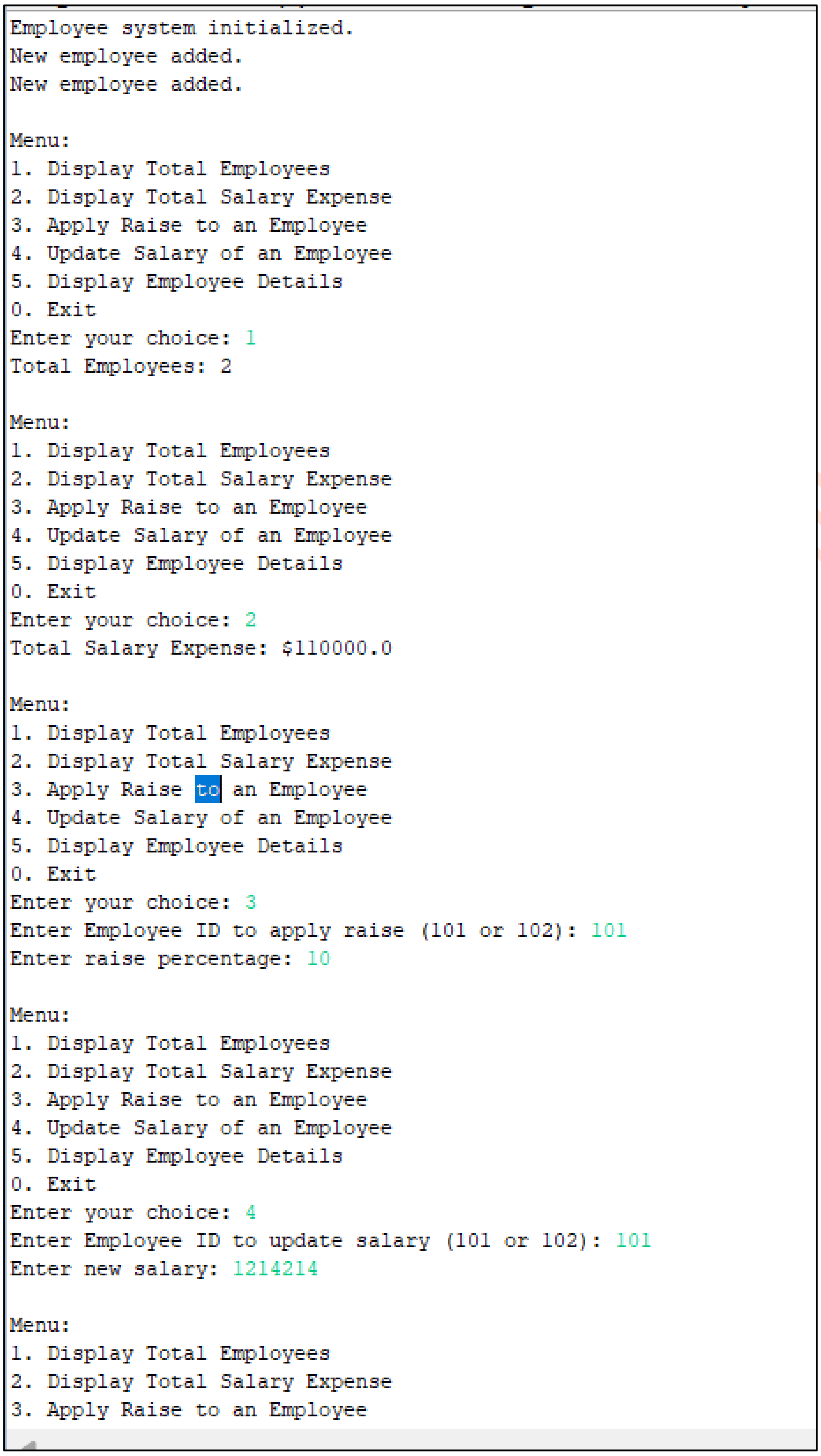
- Retrieve the total number of employees (`getTotalEmployees()`)
- Apply a percentage raise to the salary of all employees (`applyRaise(double percentage)`)
- Calculate the total salary expense, including any raises (`calculateTotalSalaryExpense()`)
- Update the salary of an individual employee (`updateSalary(double 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.

```
package in.Cdac.Employee;

import java.util.Scanner;

class Employee {
	private static int totalEmployees = 0;
	private static double totalSalaryExpense = 0.0;
	private int id;
	private String name;
	private double salary;

	static {
		System.out.println("Employee system initialized.");
	}

	{
		totalEmployees++;
		System.out.println("New employee added.");
	}

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

	public static int getTotalEmployees() {
		return totalEmployees;
	}

	public void applyRaise(double percentage) {
		double raiseAmount = (salary * percentage) / 100;
		totalSalaryExpense -= salary;
		salary += raiseAmount;
		totalSalaryExpense += salary;
	}
	public static double calculateTotalSalaryExpense() {
		return totalSalaryExpense;
	}

	public void updateSalary(double newSalary) {
		totalSalaryExpense -= salary;
		salary = newSalary;
		totalSalaryExpense += salary;
	}

	public int getId() {
		return id;
	}

	public String getName() {
		return name;
	}

	public double getSalary() {
		return salary;
	}

	public String toString() {
		return "Employee [\nid=" + id + "\nname=" + name +
"\nsalary=" + salary + "\n]";
	}
}

class EmployeeUtil {
	private static Scanner sc = new Scanner(System.in);

	public void displayTotalEmployees() {
		System.out.println("Total Employees: " +
Employee.getTotalEmployees());
	}

	public void displayTotalSalaryExpense() {
		System.out.println("Total Salary Expense: $" +
Employee.calculateTotalSalaryExpense());
	}

	public void applyRaiseToEmployee(Employee emp1, Employee emp2) {
		System.out.print("Enter Employee ID to apply raise (101 or
102): ");
		int empId = sc.nextInt();
		System.out.print("Enter raise percentage: ");
		double raise = sc.nextDouble();
		if (empId == emp1.getId()) {
			emp1.applyRaise(raise);
		} else if (empId == emp2.getId()) {
			emp2.applyRaise(raise);
		} else {
			System.out.println("Employee not found.");
		}
	}

	public void updateSalaryOfEmployee(Employee emp1, Employee emp2)
{
		System.out.print("Enter Employee ID to update salary (101 or 
102): ");
		int empId = sc.nextInt();
		System.out.print("Enter new salary: ");
		double newSalary = sc.nextDouble();
		if (empId == emp1.getId()) {
			emp1.updateSalary(newSalary);
		} else if (empId == emp2.getId()) {
			emp2.updateSalary(newSalary);
		} else {
			System.out.println("Employee not found.");
		}
	}

	public void displayEmployeeDetails(Employee emp1, Employee emp2) 
{
		System.out.print("Enter Employee ID to view details (101 or 
102): ");
		int empId = sc.nextInt();
		if (empId == emp1.getId()) {
			System.out.println(emp1);
		} else if (empId == emp2.getId()) {
			System.out.println(emp2);
		} else {
			System.out.println("Employee not found.");
		}
	}

	public static int menuList() {
		System.out.println("\nMenu:");
		System.out.println("1. Display Total Employees");
		System.out.println("2. Display Total Salary Expense");
		System.out.println("3. Apply Raise to an Employee");
		System.out.println("4. Update Salary of an Employee");
		System.out.println("5. Display Employee Details");
		System.out.println("0. Exit");
		System.out.print("Enter your choice: ");
		return sc.nextInt();
	}
}

public class Program {
	public static void main(String[] args) {
		int choice;
		Employee emp1 = new Employee(101, "John Doe", 50000);
		Employee emp2 = new Employee(102, "Jane Smith", 60000);
		EmployeeUtil util = new EmployeeUtil();

		while ((choice = EmployeeUtil.menuList()) != 0) {
			switch (choice) {
				case 1:
					util.displayTotalEmployees();
					break;
				case 2:
					util.displayTotalSalaryExpense();
					break;
				case 3:
					util.applyRaiseToEmployee(emp1, emp2);
```

```
                                        break;
                                case 4:
                                        util.updateSalaryOfEmployee(emp1, emp2);
                                        break;
                                case 5:
                                        util.displayEmployeeDetails(emp1, emp2);
                                        break;
                                default:
                                        System.out.println("Invalid choice, please 
try again.");
                        }
                }

        }
}
```



```
Employee system initialized.
New employee added.
New employee added.

Menu:
1. Display Total Employees
2. Display Total Salary Expense
3. Apply Raise to an Employee
4. Update Salary of an Employee
5. Display Employee Details
0. Exit
Enter your choice: 1
Total Employees: 2

Menu:
1. Display Total Employees
2. Display Total Salary Expense
3. Apply Raise to an Employee
4. Update Salary of an Employee
5. Display Employee Details
0. Exit
Enter your choice: 2
Total Salary Expense: $110000.0

Menu:
1. Display Total Employees
2. Display Total Salary Expense
3. Apply Raise to an Employee
4. Update Salary of an Employee
5. Display Employee Details
0. Exit
Enter your choice: 3
Enter Employee ID to apply raise (101 or 102): 101
Enter raise percentage: 10

Menu:
1. Display Total Employees
2. Display Total Salary Expense
3. Apply Raise to an Employee
4. Update Salary of an Employee
5. Display Employee Details
0. Exit
Enter your choice: 4
Enter Employee ID to update salary (101 or 102): 101
Enter new salary: 1214214

Menu:
1. Display Total Employees
2. Display Total Salary Expense
3. Apply Raise to an Employee
```