Question1: What is the output of the following program?

```java
public class Calculator {

    private int number1;
    private int number2;

    public Calculator(int nb1, int nb2) {
        number1 = nb1;
        number2 = nb2;
        System.out.println("Created Calculator object: ");
        display();
    }

    public void display() {
        System.out.println("Number1: " + number1);
        System.out.println("Number2: " + number2);
    }

    public void calculate() {
        int result = 100;
        for (int i=0; i<number1; i++) {
            result = result + number2;
        }
        System.out.println("Calculation result: " + result);
    }
}
```
public class Calculation
{
    public static void main(String[] args)
    {
        Calculator cal = new Calculator(3, 5);
        cal.calculate();
    }
}

Question 2:
Having a Moukhalafa class (see the UML class diagram), we would like to manage the driver violation of traffic regulations.

<table>
<thead>
<tr>
<th>Moukhalafa</th>
</tr>
</thead>
<tbody>
<tr>
<td>- carId : string</td>
</tr>
<tr>
<td>- driverId : int</td>
</tr>
<tr>
<td>- nbViolations : int</td>
</tr>
<tr>
<td>- totalAmount : int</td>
</tr>
<tr>
<td>+Moukhalafa(in car : string, in driver : int)</td>
</tr>
<tr>
<td>- speedViolation(in carSpeed : int, in maxAuthorizedSpeed : int) : void</td>
</tr>
<tr>
<td>- trafficLightViolation() : void</td>
</tr>
<tr>
<td>+updateViolationAmount(in violationType : int) : void</td>
</tr>
<tr>
<td>+display() : void</td>
</tr>
<tr>
<td>+hasViolation() : boolean</td>
</tr>
<tr>
<td>+hasGreaterAmountThan(in mk : Moukhalafa) : boolean</td>
</tr>
<tr>
<td>+ getTotalAmount() : int</td>
</tr>
<tr>
<td>+ getNbViolation() : int</td>
</tr>
</tbody>
</table>
A Moukhalafa has the following attributes:

- **carId**: the identification of the car.
- **driverId**: the identification of the driver
- **nbViolations**: the number of violations that were made by that driver in this car.
- **totalAmount**: the total amount of money that must be paid by the driver because of the violations.

The class Moukhalafa provides the following services:

- A **constructor** that accepts two parameters used to initialize the attributes `carId` and `driverId`.

  The attributes `nbViolations` and `totalAmount` are initialized to zero.

- **speedViolation** accepts two parameters: the current speed of the car and the maximum authorized speed. It updates the `totalAmount` by adding a new amount to it according to the following:
  
  - if the current speed exceeds the maximum authorized speed by a value between 10 Km and 25 Km then SR 300 will be added to `totalAmount`
  
  - if the current speed exceeds the maximum authorized speed by more than 25 Km then SR 500 will be added to `totalAmount`

- **trafficLightViolation** updates the `totalAmount` by adding SR 700.

- **updateViolationAmount** accepts one parameter: the type of violation. It updates the total amount according to the following:
  
  - if the type is zero the violation is a speed violation (hint: this needs reading data about the violation from the keyboard)
  
  - if the type is 1 the violation is a traffic light violation

  * **display** displays all information about a Moukhalafa object

  * **hasViolation** checks whether a violation was recorded.

  * **hasGreaterAmountThan** accepts one parameter of type Moukhalafa. It checks whether the
current Moukhalafa has a total amount that is greater than in the Moukhalafa passed as parameter.

**Question:** Implement the class Moukhalafa

**Question 3:**

We assume that the class Moukhalafa (in question 2) is written in Java.

Write in Java the program that processes the following tasks:

a. Creates an object `mklf` of type Moukhalafa.

b. Keeps repeating the following operations until the total amount of the violations exceeds SR5000 or the number of violations is greater than 12

1. Reads the type of violation

2. Updates the amount of violation accordingly

3. Display the number of violations and the average amount of a violation