

Nome: Lucas Ertel Soares Ra: 2096960

Refatoração código progressão aritmética

## Code Smell

```
public void calculaSomaPA ()
{
    int j, n, a1, an, r, t, s;
    PrintStream w = System.out;
    Scanner sc = new Scanner(System.in);
    do {
        w.println("Digite o número de termos da PA:");
        n = sc.nextInt();
    } while (n < 2);
    do {
        w.println("Digite o primeiro termo da PA:");
        a1 = sc.nextInt();
    } while (a1 < 1);
    do {
        w.println("Digite a razão da PA:");
        r = sc.nextInt();
    } while (r < 1);
    an = a1 + (n - 1) * r;
    s = ((a1 + an) * n) / 2;
    for (j = 1; j <= n; j++) {
        t = a1 + (j - 1) * r;
        w.printf("a%d=%d\n", j, t);
    }
    w.printf("A soma é:%d\n", s);
}
```

Clean Code  
métodos curtos  
Nomes claros

SOLID  
Single Responsibility

Refatoração

Link do projeto Github

[Click aqui](#)

Aqui foi separado em métodos cada uma com seu propósito geral

```
package ArithmeticProgression;

import java.io.PrintStream;
import java.util.Scanner;

public class ArithmeticProgression {
    private final Scanner read = new Scanner(System.in);
    private final PrintStream write = System.out;

    public ArithmeticProgression() {
    }

    public void initArithmeticProgression() {
        int firstterm, amountterms, reason;

        firstterm = firstTermAp();
        amountterms = amountTermsAp();
        reason = reasonAp();

        sumPa(firstterm, amountterms, reason);
    }
}
```

## 1 - Inicia a progressão aritmética

```
public void initArithmeticProgression() {  
    int firstterm, amountterms, reason;  
  
    firstterm = firsttermAp();  
  
    amountterms = amounttermsAp();  
  
    reason = reasonAp();  
  
    sumPa(firstterm, amountterms, reason);  
}
```

## 2-Método quantidade de termos que não pode ser menor que dois

```
public int amounttermsAp() {  
    int n;  
  
    do {  
  
        write.println(x: "Digite o número de termos da PA:");  
  
        n = read.nextInt();  
  
    } while (n < 2);  
  
    return n;  
}
```

### 3 - Método responsável por pedir a quantidade de termos

```
public int firsttermAp() {
    int a1 ;

    do{
        write.println(x: "Digite o primeiro termo da

        a1 = read.nextInt();

    }while(a1 < 1);

    return a1;
}
```

### 4- Método responsável por pedir a razão da progressão aritmética .

```
public int reasonAp() {
    int r;

    do{

        write.println(x: "Digite a razao da PA:");

        r = read.nextInt();

    }while(r < 1);

    return r;
}
```

### 5 - Método com algoritmo que faz a soma da pa

```
public void sumPa(int firstterm,int amounterms ,int reason) {
    int an, sum;

    an = firstterm + (amounterms -1) * reason;

    sum = ((firstterm + an) * amounterms)/2;

    sequencelistAp(firstterm,amounterms,reason);

    write.printf(format: "A soma é:%d\n",args:sum);
}
```

### 6- Método que lista a sequência da soma progressão.

```

public void sequencelistAp(int firstterm,int amountterms ,int reason){
    int t;
    for (int index = 1; index <= amountterms; index++) {

        t = firstterm + (index - 1) * reason;

        write.printf(format: "a%d=%d\n", args:index, args:t);
    }
}

```

## 7 Método main que executa os métodos e chama o objeto da classe

```

* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
*/
package ArithmeticProgression;

/**
 *
 * @author lucas
 */
public class Main {

    public static void main(String[] args) {
        ArithmeticProgression arithmeticProg = new ArithmeticProgression();

        arithmeticProg.initArithmeticProgression();
    }

}

```

## 8 Resultado com teste

Output - ArithmeticProgressionCalc (run) X

```

run:
Digite o primeiro termo da PA:
2
Digite o número de termos da PA:
5
Digite a razao da PA:
1
a1=2
a2=3
a3=4
a4=5
a5=6
A soma é:20
BUILD SUCCESSFUL (total time: 11 seconds)

```

"É fazendo que se aprende a fazer aquilo que se deve aprender a fazer."(Aristóteles)

