Exemplo de Programa Orientado a Objetos

Este programa não tem uma finalidade específica, a não ser a de esclarecer algumas dúvidas e servir de mostruário para quem está iniciando com Programação Orientação a Objetos (POO).

O programa é composto por duas classes, a classe principal que aqui leva o nome de "ExemploProgramaOrientadoObjetos" e a classe "Calculos" que é responsável pelos cálculos que serão efetuados no sistema.

Cada método da classe "Calculos" foi construído de uma maneira distinta, visando exemplificar algumas das formas que podem ser trabalhadas na POO.

O programa inteiro possui comentários, no entanto, é necessário conhecimento de lógica de programação e linguagem Java para se entender em profundidade o que este *software* se propõe a fazer.

A classe "Calculos"

```
1   package exemploprogramaorientadoobjeto;
2
3   public class Calculos {
4
5       private float num1;
6       private float num2;
7       private float total;
8
9       /**
10       * Método construtor.
11       */
12      public Calculos(){
13          num1 = 0;
14          num2 = 0;
15          total = 0;
16      }
17
18      /**
19       * Método construído para retornar o valor do atributo "total".
20       * @return total
21       */
22      public float getTotal(){
23          return(total);
24      }
25
26      /**
27       * Método construído para colocar valor no atributo "num1".
28       * @param v1
29       */
```

Figura 1: Classe "Calculos" - Parte 01 / 02

```
30      public void setNum1(float v1){
31          num1 = v1;
32      }
33
34      /**
35       * Método construído para colocar valor no atributo "num2".
36       * @param v1
37       */
38      public void setNum2(float v1){
39          num2 = v1;
40      }
41
42      /**
43       * Método construído para receber dois valores, efetuar um cáclulo e retornar o
44       * resultado sem usar os atributos da classe.
45       * @param v1
46       * @param v2
47       * @return (v1 * v2) / (v1 + v2)
48       */
49      public float calculo01(float v1, float v2){
50          float res;
51          res = (v1 * v2) / (v1 + v2);
52          return(res);
53      }
54
55      /**
56       * Método construído para efetuar um cálculo com valores previamente passados aos
57       * atributos "num1" e "num2" da classe e retornar o valor deste cálculo.
58       * @return num1 - 3 * num2
59       */
60      public float calculo02(){
61          total = num1 - 3 * num2;
62          return(total);
63      }
64
65      /**
66       * Método construído para efetuar um cálculo com valores previamente passados aos
67       * atributos "num1" e "num2" da classe. O resultado do cálculo será armazenado no
68       * atributo "total" e poderá ser obtido por meio da chamada ao método "getTotal".
69       */
70      public void calculo03(){
71          total = (num1 - num2) / num2;
72      }
73  }
```

Figura 2: Classe "Calculos" - Parte 02 / 02

Quando se programa orientado a objetos, o melhor a se fazer é planejar bem o *software* antes de começar a construí-lo.

Quando começar, comece pelos objetos que o comporão. E por último será a codificação da classe principal, onde uma boa parte dos objetos criados deverão ser chamados e onde se encontra o método ***main***.

A classe “ExemploProgramaOrientadoObjetos” (classe principal)

```
1  package exemploprogramaorientadoobjeto;
2
3  import java.util.Scanner;
4
5  public class ExemploProgramaOrientadoObjetos {
6
7      public static void main(String[] args) {
8          /**
9           * Declaração e criação do objeto "calc" da classe "Calculos".
10          */
11         Calculos calc;
12         calc = new Calculos();
13
14         /**
15          * Declaração e criação do objeto "sc" da classe "Scanner".
16          * Utilizado para a leitura de valores do teclado do computador.
17          */
18         Scanner sc;
19         sc = new Scanner(System.in);
20
21         /**
22          * Variáveis do programa.
23          */
24         float valor1, valor2, resultado = 0;
25         int opcao;
26
27         /**
28          * Menu apresentado ao usuário que utilizará o programa, para
29          * que este possa escolher o cálculo a ser efetuado ou sair do
30          * programa.
31          */
32         do{
33             System.out.println("Escolha o cálculo: ");
34             System.out.println("1 - Cálculo 01");
35             System.out.println("2 - Cálculo 02");
36             System.out.println("3 - Cálculo 03");
37             System.out.println("4 - Sair");
38             System.out.println("Opção: ");
39             opcao = sc.nextInt();
40
41             if(opcao != 4){
42                 // Leitura dos valores digitados pelo usuário.
43                 System.out.println("\nDigite o 1o. valor: ");
44                 valor1 = sc.nextFloat();
45                 System.out.println("\nDigite o 2o. valor: ");
46                 valor2 = sc.nextFloat();
47
48                 // Execução do cálculo escolhido pelo usuário.
49                 switch(opcao){
50                     case 1:
51                         resultado = calc.calculo01(valor1, valor2);
52                         break;
```

Figura 3: Classe ExemploProgramaOrientadoObjetos - Parte 01 / 02

```
53                      case 2:
54                          calc.setNum1(valor1);
55                          calc.setNum2(valor2);
56                          resultado = calc.calculo02();
57                          break;
58                      case 3:
59                          calc.setNum1(valor1);
60                          calc.setNum2(valor2);
61                          calc.calculo03();
62                          resultado = calc.getTotal();
63                          break;
64                  }
65
66                  // Apresentação do resultado do cálculo.
67                  System.out.println("\n\n==============================");
68                  System.out.println("Resultado: " + resultado);
69                  System.out.println("==============================\n\n");
70              }
71          }while(opcao != 4);
72      }
73  }
```

Figura 4: Classe ExemploProgramaOrientadoObjetos - Parte 02 / 02

Veja você que quando se programa utilizando POO, o *software* fica um pouco maior do que se fosse feito usando o paradigma estruturado. No entanto, veja também que o *software* fica muito mais organizado e mais fácil de ser entendido, tanto por quem faz, quanto por quem for dar manutenção.