

Programação Orientada a Objeto

Classes, Atributos, Métodos e Objetos



Programação de Computadores II

Professor: Edwar Saliba Júnior

- 1) Java é uma linguagem orientada a objetos. Para que possamos fazer uso pleno das vantagens da Programação Orientada a Objetos (POO), devemos aprender e utilizar alguns conceitos de *software* que estruturam este paradigma da programação. Dentre os diversos conceitos existentes, vamos começar pelos básicos, ou seja, vamos entender o que são: classes, atributos, métodos e objetos.
- 2) **Classe:** Uma classe é um tipo de dado similar aos tipos primitivos, tais como: *int*, *float* e *double*. Tipos de dados são usados para declarar variáveis. Algumas particularidades das classes são:
 - Os nomes das classes sempre iniciam por uma letra maiúscula.
 - Sua estrutura interna é composta por três partes:
 1. **atributos** (também conhecidos por: propriedades ou campos): que especificam a espécie de dados que os objetos armazenam;
 2. **construtores:** que especificam como os objetos devem ser criados, e
 3. **métodos:** que especificam as operações que os objetos podem executar.
- 3) **Objeto:** Pode ser definido como uma classe que foi instanciada, ou seja, uma classe que foi construída e está pronta para ser usada.
- 4) Para exemplificar os conceitos de classe, atributos, métodos e objetos, vamos tomar como exemplo o um *software* que simula uma calculadora de quatro operações.
 - No *software* desenvolvido na Figura 1, em linguagem C, podemos ver o código de uma calculadora de 4 operações, desenvolvido sobre o paradigma da Programação Estruturada.

```

1  #include <stdio.h>
2
3  float adicao(float v1, float v2){
4      float total = 0;
5
6      total = v1 + v2;
7
8      return(total);
9  }
10
11 float subtracao(float v1, float v2){
12     float total = 0;
13
14     total = v1 - v2;
15
16     return(total);
17 }
18
19 float multiplicacao(float v1, float v2){
20     float total = 0;
21
22     total = v1 * v2;
23
24     return(total);
25 }
26
27 float divisao(float v1, float v2){
28     float total = 0;
29
30     total = v1 / v2;
31
32     return(total);
33 }
34
35 int main()
36 {
37     float val1, val2, res;
38     int op;
39
40     do{ /* Menu de operações. */
41
42         printf("\n\nOperações:\n");
43         printf("\n1 - adição");
44         printf("\n2 - subtração");
45         printf("\n3 - multiplicação");
46         printf("\n4 - divisão");
47         printf("\nEscolha uma opção: ");
48         scanf("%i",&op);
49     }while((op < 1) || (op > 4));
50
51     printf("\nDigite o primeiro valor: ");
52     scanf("%f",&val1);
53     printf("Digite o segundo valor: ");
54     scanf("%f",&val2);
55
56     switch(op){
57     case 1:
58         res = adicao(val1,val2);
59         break;
60     case 2:
61         res = subtracao(val1,val2);
62         break;
63     case 3:
64         res = multiplicacao(val1,val2);
65         break;
66     case 4:
67         if(val2 == 0)
68             printf("\n\nErro: Divisão por zero!\n\n");
69         else
70             res = divisao(val1,val2);
71         break;
72     default:
73         printf("\n\nOperação inválida!\n\n");
74     }
75
76     if(!(op == 4 && val2 == 0))
77         printf("\n\nO resultado é: %f \n\n",res);
78
79     return 0;
80 }

```

Figura 1: Calculadora em Linguagem C

- No *software* desenvolvido, em linguagem Java, podemos ver o código de uma calculadora de 4 operações, desenvolvido sobre o paradigma da POO. Pontos a serem observados:
 1. Existem dois arquivos dentro do *package* “calculadora”. Sendo que o primeiro é a classe “Calculadora” e o segundo é a classe “Main”, portadora do método “main”.
 2. Na classe Calculadora são criados 3 atributos.
 3. Cada atributo possui um método *get* e/ou *set* para seu preenchimento ou retorno de valor, visto que todos eles foram declarados como *private*, ou seja, são atributos que só podem ser acessados diretamente pela classe, ou por outras classes, através dos métodos *get* e *set*.
 4. Os métodos, ou seja, as operações que a Calculadora executa, são sempre declarados como métodos públicos.
 5. Já na classe *Main*, o arquivo é composto de estruturas e comandos já conhecidos, contudo, há também uma linha de comando criando e instanciando a classe Calculadora. Para que possamos usufruir de seus serviços.

Vejamos a seguir (Figura 2) como ficou o código da classe Calculadora, construído no paradigma da Orientação a Objetos.

```

6 package calculadora;
7
8 /**
9  *
10 * @author Edwar Saliba Júnior
11 */
12 public class Calculadora {
13     private float valor1;
14     private float valor2;
15     private float resultado;
16
17     public Calculadora() {
18         this.valor1 = 0;
19         this.valor2 = 0;
20     }
21
22     public float getResultado() {
23         return resultado;
24     }
25
26     public void setValor1(float valor1) {
27         this.valor1 = valor1;
28     }
29
30     public void setValor2(float valor2) {
31         this.valor2 = valor2;
32     }
33
34     public void adicao() {
35         this.resultado = this.valor1 + this.valor2;
36     }
37     public void subtracao() {
38         this.resultado = this.valor1 - this.valor2;
39     }
40
41     public void multiplicacao() {
42         this.resultado = this.valor1 * this.valor2;
43     }
44
45     public void divisao() {
46         if(valor2 == 0)
47             System.out.print("\n\nErro: divisão por zero!\n\n");
48         else
49             this.resultado = this.valor1 + this.valor2;
50     }
51 }
52

```

Figura 2: Classe Calculadora

E o código da classe *Main* (Figura 3).

```

6 package calculadora;
7
8 import java.util.Scanner;
9
10 /**
11  *
12  * @author Edwar Saliba Júnior
13  */
14 public class Main {
15
16     /**
17      * @param args the command line arguments
18      */
19     public static void main(String[] args) {
20         float val1, val2;
21         int op;
22         Scanner e = new Scanner(System.in);
23         Calculadora calc = new Calculadora();
24
25         do{ /* Menu de operações. */
26             System.out.printf("\n\nOperações:\n");
27             System.out.printf("\n1 - adição");
28             System.out.printf("\n2 - subtração");
29             System.out.printf("\n3 - multiplicação");
30             System.out.printf("\n4 - divisão");
31             System.out.printf("\nEscolha uma opção: ");
32             op = e.nextInt();
33         }while((op < 1) || (op > 4));
34
35         System.out.printf("\nDigite o primeiro valor: ");
36         val1 = e.nextFloat();
37         calc.setValor1(val1);
38         System.out.printf("Digite o segundo valor: ");
39         val2 = e.nextFloat();
40         calc.setValor2(val2);
41
42         switch(op){
43             case 1:
44                 calc.adicao();
45                 break;
46             case 2:
47                 calc.subtracao();
48                 break;
49             case 3:
50                 calc.multiplicacao();
51                 break;
52             case 4:
53                 calc.divisao();
54                 break;
55             default:
56                 System.out.printf("\n\nOperação inválida!\n\n");
57         }
58
59         if(!(op == 4 && val2 == 0))
60             System.out.printf("\n\nO resultado é: %f \n\n", calc.getResultado());
61     }
62
63 }

```

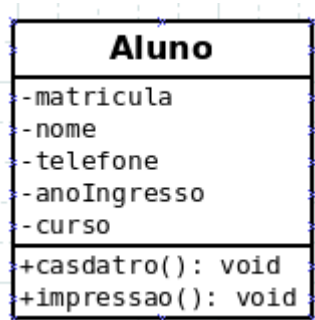
Figura 3: Classe Main

Como implementar esse sistema no paradigma OO?

- Primeiro passo: a construção de qualquer sistema, seja qual for o paradigma, requer o estudo do problema a ser solucionado.
- Segundo passo: propor uma solução computacional para o problema. Para isso, é preciso:
 - identificar as entidades a serem modeladas;
 - modelar computacionalmente tais entidades. Na POO, essas entidades correspondem naturalmente a *classes* de *objetos*.

Exemplo de uma classe através de UML (Unified Modeling Language¹)

1. A seguir podemos ver a classe Aluno, onde podemos identificar facilmente, na parte superior da figura, o nome da classe, no meio da figura, os atributos e na parte inferior, os métodos que compõem a classe Aluno.



Exercícios:

1. Construa um *software* utilizando POO, que seja capaz de cadastrar 3 alunos, com os atributos e os métodos definidos na classe Aluno. O *software* também deverá ser capaz de imprimir todos os alunos cadastrados ou somente um deles, a escolha do usuário.
2. Construa uma *software* utilizando POO, que possua uma classe chamada Tempo. Esta classe deverá possuir três atributos do tipo inteiro: hora, min e seg.²
 - a) Crie um método construtor que inicialize os membros de dados com zeros.
 - b) Crie um segundo construtor que inicialize os membros de dados com os valores recebidos como argumento.
 - c) Crie um método para imprimir a hora no formato: hh:mm:ss.
 - d) Crie um método **soma**, que soma dois objetos da classe Tempo passados como argumentos e coloca o resultado no objeto do qual é membro.
 - e) Crie uma sobrecarga do método **soma**, que soma o objeto que recebe como argumento com o objeto do qual é membro e retorna um novo objeto com o total.
 - f) Crie um método que subtraia dois campos horas (maior - menor) e retorne o número de segundos entre elas. A função recebe dois objetos da classe Tempo passados como argumentos.
 - g) Crie um *software* que demonstre o funcionamento de todos os métodos criados para a classe Tempo.

¹ UML (*Unified Modeling Language*) – A linguagem de modelagem unificada é uma metodologia utilizada para o planejamento de sistemas computacionais através de diagramas e figuras.

² MIZRAHI, V. V. **Treinamento em Linguagem C++**. 2. ed. São Paulo: Pearson, 2006, p. 42.