

Variáveis Final (Constantes)

- São constantes do Java;
- São úteis para estabelecer valores padrões;
- Facilitam a mudança de valores no futuro;
- São usadas em parâmetros de métodos, quando o valor da variável constante não irá mudar no corpo do método;
- Isso ajuda a simplificar o controle de variáveis e pode melhorar a performance do aplicativo;
- Veja trechos de códigos a seguir;

Variáveis Final (Constantes)

Produto5.java ✕

```
1
2 public class Produto5 {
3
4     String nome;
5     String unidadeMedida;
6     float preco;
7     final float PRECO_MINIMO=0.1f;
8
9     public Produto5() {
10         this.unidadeMedida = "Kg";
11         this.preco = this.PRECO_MINIMO;
12     }
```

```
float totalizar(final float peso){
    return this.preco * peso;
}
```

Sobrecarga de Métodos

- É a capacidade de definição de métodos com o mesmo nome em uma classe;
- Normalmente esse recurso é usado para promover variações de comportamento, porém mantendo o mesmo nome, facilitando o aprendizado e a utilização desse classe;
- Também é conhecido como polimorfismo estático, pois a decisão da chamada do método é feita em tempo de compilação;
- O compilador consegue decidir qual das implementações chamar olhando a assinatura dos métodos (assinatura = nome + lista de parâmetros)

Sobrecarga de Métodos

Sobrecarga.java

```
1 public class Sobrecarga {
2
3     void m1(){
4         System.out.println("m1()");
5     }
6
7     void m1(int x){
8         System.out.println("m1(int x)");
9     }
10
11    void m1(int x, int y){
12        System.out.println("m1(int x, int y)");
13    }
14
15    void m1(int x, float y){
16        System.out.println("m1(int x, float y)");
17    }
18
19    void m1(float x, int y){
20        System.out.println("m1(float x, int y)");
21    }
22
23    void m1(float x, float y){
24        System.out.println("m1(float x, float y)");
25    }
26 }
```

- Todos os métodos possuem o mesmo nome, porém assinaturas diferentes;
- O tipo de retorno **não é considerado** na composição da assinatura.

Sobrecarga de Métodos

- Observe a relação das chamadas dos métodos com a implementação executada;
- A decisão de qual implementação será chamada é realizada no momento da compilação;

The screenshot displays an IDE window titled 'UsaSobrecarga.java'. The code defines a class 'UsaSobrecarga' with a 'main' method. Inside 'main', an instance 'sobrecarga' of 'Sobrecarga' is created, and six calls to 'sobrecarga.m1()' are made with different argument combinations. Arrows point from each call to a corresponding signature in a 'Console' window on the right. The console shows the execution of each 'm1' method with its specific parameters.

```
1 public class UsaSobrecarga {
2
3
4     public static void main(String[] args) {
5
6         Sobrecarga sobrecarga = new Sobrecarga();
7
8         sobrecarga.m1();
9         sobrecarga.m1(5);
10        sobrecarga.m1(5, 6);
11        sobrecarga.m1(5, 6.5f);
12        sobrecarga.m1(6.5f, 5);
13        sobrecarga.m1(5.5f, 6.5f);
14    }
15 }
```

Console Output:

```
<terminated> UsaSobrecarga [
m1()
m1(int x)
m1(int x, int y)
m1(int x, float y)
m1(float x, int y)
m1(float x, float y)
```

Sobrecarga de Métodos

```
Produto6.java x
1 public class Produto6 {
2
3     String nome;
4     String unidadeMedida;
5     float preco;
6     final float PRECO_MINIMO=0.1f;
7
8     public Produto6() {
9         this.unidadeMedida = "Kg";
10        this.preco = this.PRECO_MINIMO;
11    }
12
13    public Produto6(String nome) {
14        this();
15        this.nome = nome;
16    }
17
18    float totalizar(final float peso){
19        return this.preco * peso;
20    }
21
22    float totalizar(final int quantidade){
23        return this.preco * quantidade;
24    }
25 }
```

- A classe Produto6 possui duas implementações do método totalizar
- O preço do produto pode ser dado através de peso ou através de quantidade;