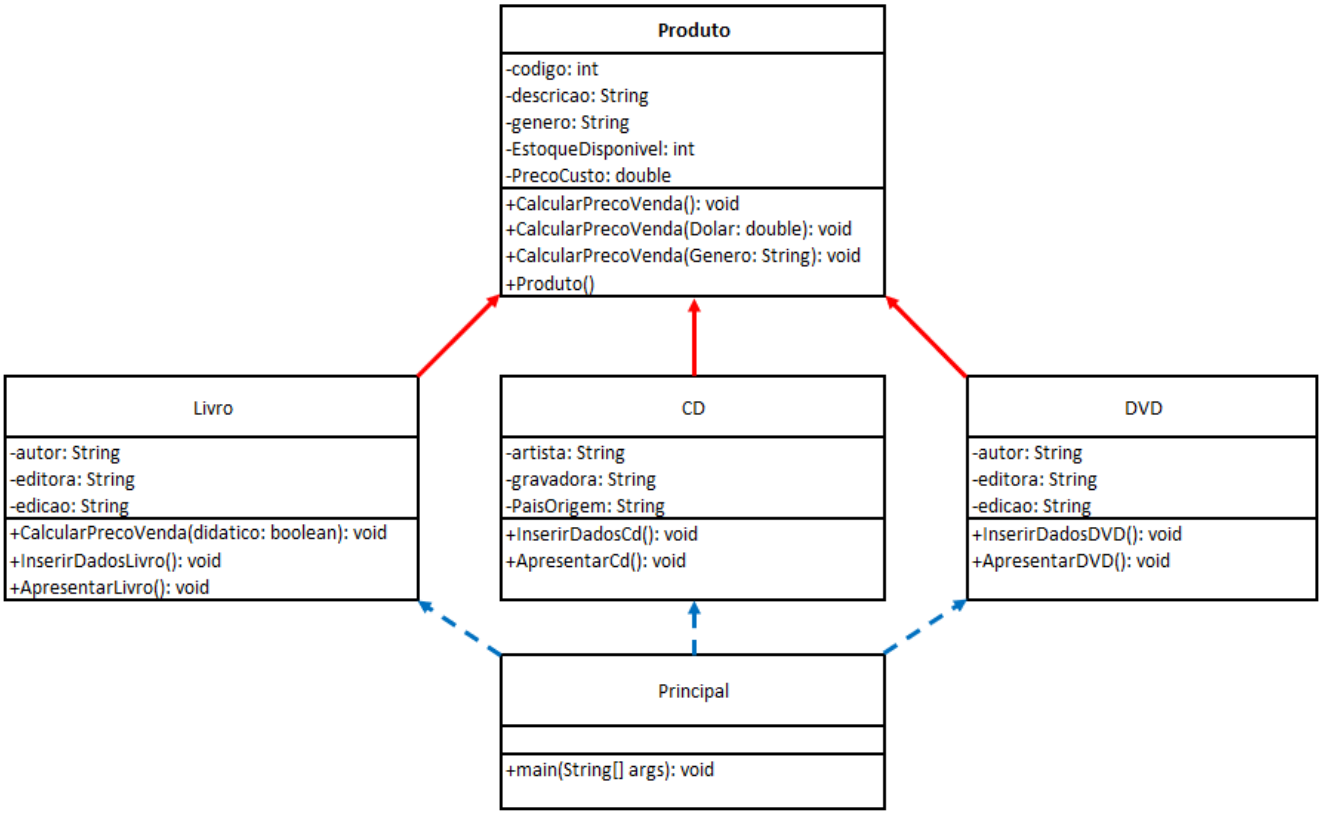
Orientação a objetos - Sobrecarga

A programação em sistemas desenvolvidos com Java é distribuída e organizada em métodos. Muitas vezes, os programadores se deparam com situações em que um método deve ser usado para finalidades semelhantes, mas com dados diferentes. Por exemplo, os produtos comercializados em uma livraria onde podemos ter Livros, CD's e DVD's, como representado no diagrama de classe abaixo:



O cálculo do preço de venda dos produtos da livraria depende de alguns fatores em determinadas épocas do ano, todos os produtos recebem a mesma porcentagem de acréscimo em relação ao preço de custo. Se o produto em questão for importado, é necessário considerar a cotação do dólar. Em algumas situações (promoções, por exemplo), é preciso atualizar todos os produtos de um determinado gênero. E, no caso específico dos livros didáticos, o cálculo do preço de venda é diferente dos demais.

Em outras linguagens de programação, como não é possível termos duas funções (blocos de código equivalentes a métodos) como o mesmo nome, nos depararíamos com a necessidade de criar funções nomeadas (calcularPrecoVenda1, calcularPrecoVenda2, calcularPrecoVenda3, calcularPrecoVenda4 ou calcularPrecoVendaNormal, calcularPrecoVendaImportado, calcularPrecoVendaPorGenero e

calcularPrecoVendaLivroDidatico e assim sucessivamente). Dessa forma, além de nomes extensos e muitas vezes estranhos, teríamos uma quantidade bem maior de nomes de funções para documentar no sistema.

Em Java, para situações desse tipo, usamos a sobrecarga que considera a identificação do método pela assinatura e não somente pelo nome. Como já vimos, a assinatura de um método é composta por:

- Nome do método e,
- Passagem de parâmetro.

Assim, é possível definir os métodos com o mesmo nome (calcularPrecoVenda, como no diagrama) e alternar a passagem de parâmetros. Observe então como ficaria os métodos na classe Produto.

```
public void CalcularPrecoVenda() {
    // aplicando aumento no preco de 20%
    this.setPrecoVenda(this.getPrecoCusto() * 1.2);
}

public void CalcularPrecoVenda(double Dolar) {
    // aplicando aumento no preco pelo cotacao Dolar
    this.setPrecoVenda(this.getPrecoCusto() * Dolar);
}

public void CalcularPrecoVenda(String genero) {
    // aplicando aumento no preco dependente do genero
    if (this.getGenero().equals(genero)) {
        this.setPrecoVenda(this.getPrecoCusto() * 1.20);
    }
}
```

E na classe Livro ficaria assim:

```
public void CalcularPrecoVenda(boolean didatico) {
    // aplicando aumento no preco se livro for didatico
    if (didatico) {
        this.setPrecoVenda(this.getPrecoCusto() * 1.10);
    }
}
```

Os pontos importantes na codificação anterior são:

- A diferenciação das assinaturas dos métodos se baseia na quantidade e no tipo de dado dos parâmetros.
- A sobrecarga pode ser realizada na mesma classe ou em subclasses, e os conceitos de herança são aplicados na utilização dos objetos. Em nosso exemplo, um objeto do tipo Livro tem quatro métodos CalcularPrecoVenda(), já na classe CD e DVD temos somente três.

No método principal (main), teremos o que mostra a figura abaixo quando digitarmos o (.) ponto no objeto Livro:

```
public static void main(String[] args) {
    Livro livro = new Livro();

    livro.Calcular
                CalcularPrecoVenda(): void
                CalcularPrecoVenda(boolean didatico): void
                CalcularPrecoVenda(double Dolar): void
                CalcularPrecoVenda(String genero): void
}
```

O Java identificará o método que dever ser executado de acordo com a chamada realizada, como mostra o exemplo abaixo:

```
livro.CalcularPrecoVenda(1.9);
```

Como está sendo passado um valor do tipo double, será executado o método CalcularPrecoVenda que considera a cotação do dólar.

Exemplo-1: Faça um programa que receba o número de horas trabalhadas, o valor do salário-mínimo e o número de horas extras trabalhadas, calcule e mostre o salário a receber, seguindo as regras abaixo. Construa um método onde é possível passar o código do funcionário para cálculo específico.

1) A hora trabalhada equivale 1/48 do salário-mínimo

2) A hora extra vale 1/40 do salário-mínimo

3) O salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada

4) A quantia a receber pelas horas extras equivale ao número de horas extras trabalhadas multiplicado pelo valor da hora extra

5) A salário a receber equivale ao salário bruto mais a quantia a receber pelas horas extras.

```
package sobrecarga;

import javax.swing.JOptionPane;

public class SobreCarga {

    public static void main(String[] args) {
        FolhaPagto folha;

        // instanciando a classe e passando os
        // parametros para o construtor da classe
        folha = new FolhaPagto(140, 1320, 80);

        JOptionPane.showMessageDialog(null, folha.CalculaFolha("74ABF23-J"));

    }
}
```

```
package sobrecarga;

public class FolhaPagto {
    // declarando os atributos da classe
    private double NHTrab;
    private double salMinimo;
    private double NHExtras;

    //Construtor da classe
    public FolhaPagto() {

    }
    // aplicando a sobrecarga
    public FolhaPagto(double NHTrab, double salMinimo, double NHExtras) {
        this.NHTrab = NHTrab;
        this.salMinimo = salMinimo;
        this.NHExtras = NHExtras;
    }

    // metodo da classe
    public String CalculaFolha() {
        String aux = "\nFolha de Pagto\n\n";

        double vHTrab = this.salMinimo / 48;
        double vHExtra = this.salMinimo / 40;
        double salBruto = (vHTrab * this.NHTrab);
        double quantiaHExtras = (vHExtra * this.NHExtras);
        double salReceber = (salBruto + quantiaHExtras);

        aux += "Hora Trabalhada R$ "      + vHTrab        + "\n";
        aux += "Hora Extra Trabalhada R$ "+ vHExtra       + "\n";
        aux += "Sal Bruto R$ "            + salBruto      + "\n";
        aux += "Quantia a receber HE R$ " + quantiaHExtras+ "\n";
        aux += "Salario a receber R$ "    + salReceber    + "\n";

        return aux;
    }
    // aplicando a sobrecarga
    public String CalculaFolha(String codigoFunc) {
        String aux = "Funcionario não encontrado!";

        if (codigoFunc.equals("74ABF23-J")) {
            aux = "\nFolha de Pagto\n\n";
```

```
            double vHTrab = this.salMinimo / 48;
            double vHExtra = this.salMinimo / 40;
            double salBruto = (vHTrab * this.NHTrab);
            double quantiaHExtras = (vHExtra * this.NHExtras);
            double salReceber = (salBruto + quantiaHExtras);

            aux += "Hora Trabalhada R$ "      + vHTrab        + "\n";
            aux += "Hora Extra Trabalhada R$ "+ vHExtra       + "\n";
            aux += "Sal Bruto R$ "            + salBruto      + "\n";
            aux += "Quantia a receber HE R$ " + quantiaHExtras+ "\n";
            aux += "Salario a receber R$ "    + salReceber    + "\n";
        }

        return aux;
    }

    // metodos get's e set's (modificadores)
    public double getNHTrab() {
        return NHTrab;
    }
    public void setNHTrab(double NHTrab) {
        this.NHTrab = NHTrab;
    }
    public double getSalMinimo() {
        return salMinimo;
    }
    public void setSalMinimo(double salMinimo) {
        this.salMinimo = salMinimo;
    }
    public double getNHExtras() {
        return NHExtras;
    }
    public void setNHExtras(double NHExtras) {
        this.NHExtras = NHExtras;
    }
}
```