

A **Engenharia de Software** surgiu no final da década de 60 para tentar solucionar os problemas gerados pela “**Crise do Software**”, no entanto, várias técnicas que foram desenvolvidas nos anos 70 e 80 não conseguiram resolver os problemas de produtividade e qualidade de software.

Isso veio a melhorar por volta dos anos 90 onde, empresas de grande porte e com sistema considerados de grande quantidade de informação e informatização, mais o aparecimento da Internet, fez com que cada vez mais o computador se tornasse uma ferramenta essencial de trabalho na vida das pessoas, assim, fez com que os programas ficassem cada dia mais poderosos, com mais recursos e consumindo mais recursos do computador que, forçou um aperfeiçoamento de alteração e expansão de sistemas e interação com outros sistemas para troca de dados, portabilidade entre diversas plataformas.

Dessa forma, verificou-se que o reuso de partes de sistemas para alavancar a produção de software se tornou uma peça-chave nesse contexto, ou seja, reduzindo as etapas de desenvolvimento de software. Uma das técnicas que a programação começou a oferecer foi a **Programação orientada a objetos**, fazendo com que grandes diferenciais da programação orientada a objetos em relação a outros paradigmas de programação que também permitem a definição de estruturas e, operações sobre essas estruturas está no conceito de:

- **herança**, mecanismo através do qual definições existentes podem ser facilmente estendidas. Juntamente com a **herança** deve ser enfatizada a importância do;
- **polimorfismo**, que permite selecionar funcionalidades que um programa irá utilizar de forma dinâmica, durante sua execução.

Esse paradigma de programação está ancorado basicamente em algumas definições tais como:

- ✓ classes;
- ✓ objetos;
- ✓ herança e;
- ✓ polimorfismo.

## Classes

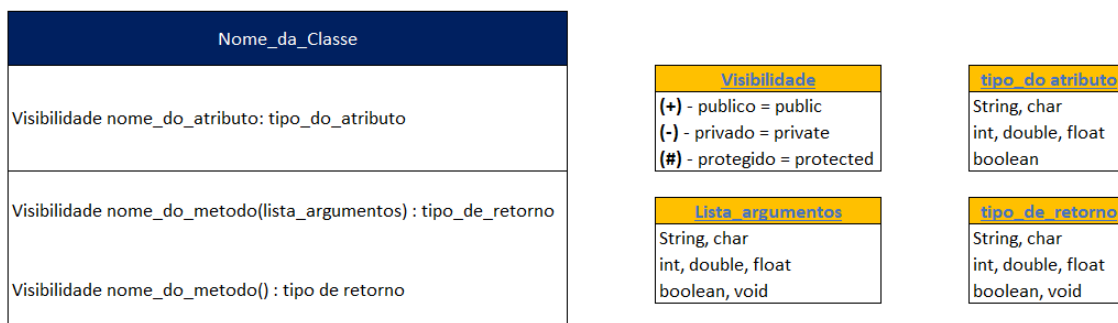
A palavra classe vem da taxonomia da biologia. Todos os seres vivos de uma mesma classe biológica têm uma série de atributos e comportamentos em comum, mas não são iguais, podem variar nos valores desses atributos e como realizam esses comportamentos, ou seja, uma classe é um **gabarito** para a **definição de objetos**. Através da definição de uma classe, descreve-se que

propriedades (**atributos**) que o objeto terá. Além dos atributos, a definição de uma classe descreve também qual o comportamento de objetos da classe, ou seja, que funcionalidades podem ser aplicadas a objetos da classe que podem ser descritas através dos métodos.

## Métodos

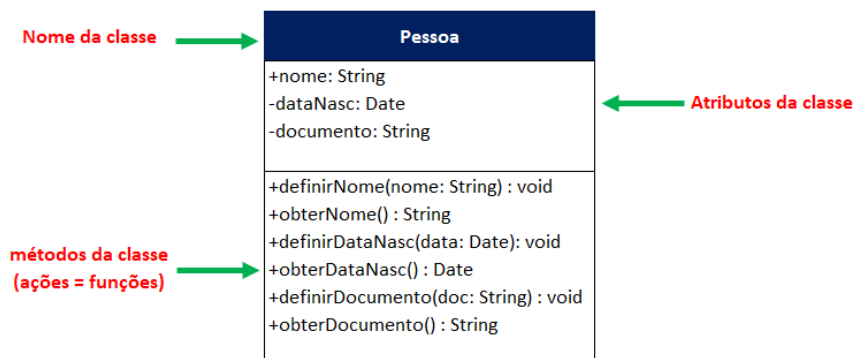
Um método nada mais é que o equivalente a um **procedimento** ou **função**, com a restrição que ele manipula apenas suas variáveis locais e os atributos que foram definidos para a classe. Uma vez que estejam definidas quais serão as classes que irão compor uma aplicação, assim como qual deve ser sua estrutura interna e comportamento, é possível criar essas classes em Java.

Na *Unified Modeling Language* (UML) – que veremos a seguir, a representação para uma classe no diagrama de classes é tipicamente expressa na forma **gráfica**, como mostrado na Figura a seguir.



Como se observa nessa figura, a especificação de uma classe é composta por três regiões:

- I. nome da classe;
- II. conjunto de atributos da classe e;
- III. conjunto de métodos da classe (funções).



O nome da **classe** é um identificador para a **classe**, que permite referenciá-la posteriormente, por exemplo, no momento da criação de um **objeto**, assim, o conjunto de **atributos** descreve as propriedades da classe e cada **atributo** é identificado por um nome e tem um **tipo** associado.

Os **métodos** definem as funcionalidades da classe, ou seja, o que será possível fazer com objetos dessa classe e cada **método** é especificado por uma **assinatura**, composta por:

1. Identificador para o **método** (o nome do método);
2. Tipo para o valor de retorno e;
3. Sua lista de argumentos, sendo cada argumento identificado por seu tipo e nome.

O modificador de visibilidade pode estar presente tanto para atributos como para métodos. Em princípio, três categorias de visibilidade podem ser definidas:

<u>publico</u>	Denotado em UML pelo símbolo +, assim, o atributo ou método de um objeto dessa classe pode ser acessado por qualquer outro objeto.
<u>privado</u>	Denotado em UML pelo símbolo -, assim, o atributo ou método de um objeto dessa classe não pode ser acessado por nenhum outro objeto.
<u>protegido</u>	Denotado em UML pelo símbolo #, assim, o atributo ou método de um objeto dessa classe poderá ser acessado apenas por objetos de classes que sejam derivadas dessa através do mecanismo de herança.

## Objetos

**Objetos** são instâncias de classes. É através deles que (**praticamente**) todo o processamento ocorre em sistemas programados com linguagens de programação orientadas a objetos.

O uso racional de **objetos**, obedecendo aos princípios associados à sua definição conforme estabelecido no paradigma de desenvolvimento orientado a objetos, é a chave para o desenvolvimento de sistemas complexos e eficientes.

Um **objeto** é um elemento que representa, no domínio da solução, alguma entidade (**abstrata ou concreta**) do domínio de interesse do problema sob análise.

**Objetos** similares são agrupados em classes. Quando se cria um **objeto**, esse **objeto** adquire um espaço em memória para armazenar seu estado (**os valores de seu conjunto de atributos, definidos pela classe**) e um conjunto de operações que podem ser aplicadas ao objeto (o conjunto de métodos definidos pela classe).

Um programa orientado a objetos é composto por um conjunto de objetos que interagem através de “**trocas de mensagens**”. Na prática, essa troca de mensagens traduz-se na aplicação de métodos a objetos.

### Exemplo-1:

Considere um **programa para um banco**, é bem fácil perceber que uma entidade extremamente importante para o nosso sistema é a **conta**. Nossa ideia aqui é generalizarmos alguma informação, juntamente com **funcionalidades** que toda **conta** deve ter.

O que toda conta tem e é importante para o desenvolvimento do sistema do banco:

1. número da conta;
2. nome do titular da conta e;
3. saldo.

O que toda conta faz e é importante, isto é, o que gostaríamos de "**pedir à conta**":

- sacar uma certa quantidade de valores;
- depositar uma certa quantidade;
- imprimir um extrato das transações da conta, nome do titular, número da conta e saldo.

Com isso, temos o projeto de uma conta bancária. Podemos pegar esse projeto e acessar seu saldo, correto?

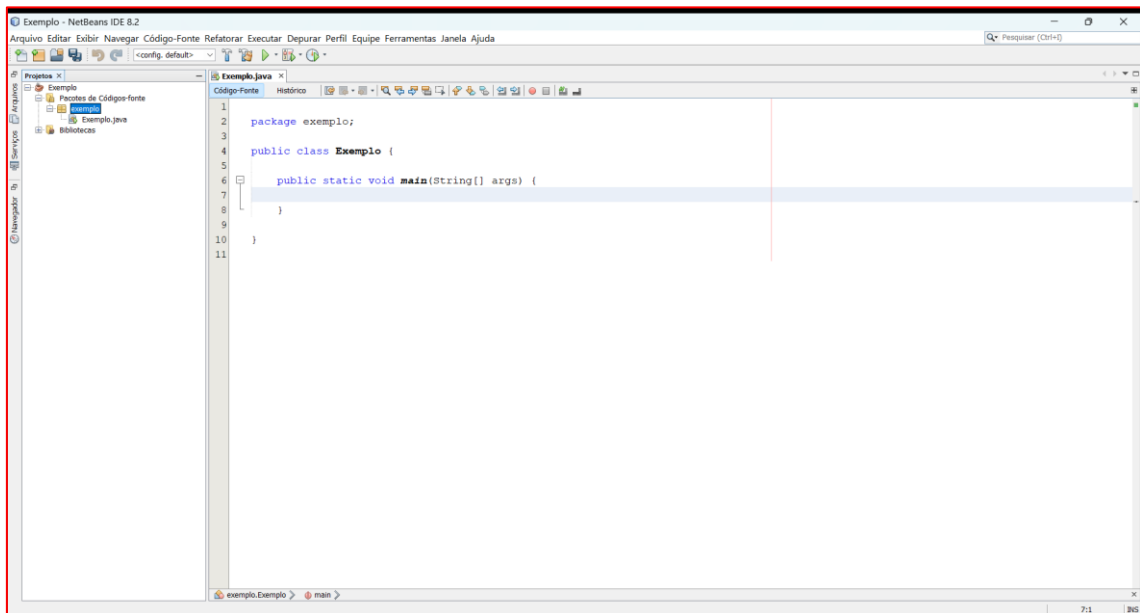
O que temos ainda é o projeto. Antes, precisamos construir uma conta, para poder acessar o que ela tem, e pedir a ela que faça algo.

### Criando a classe:

Conta	
+nroConta: String +nomeTitular: String +saldo: double	atributos
+Sacar(valor: double) : boolean +Depositar(valor: double) : void +Extrato() : void	

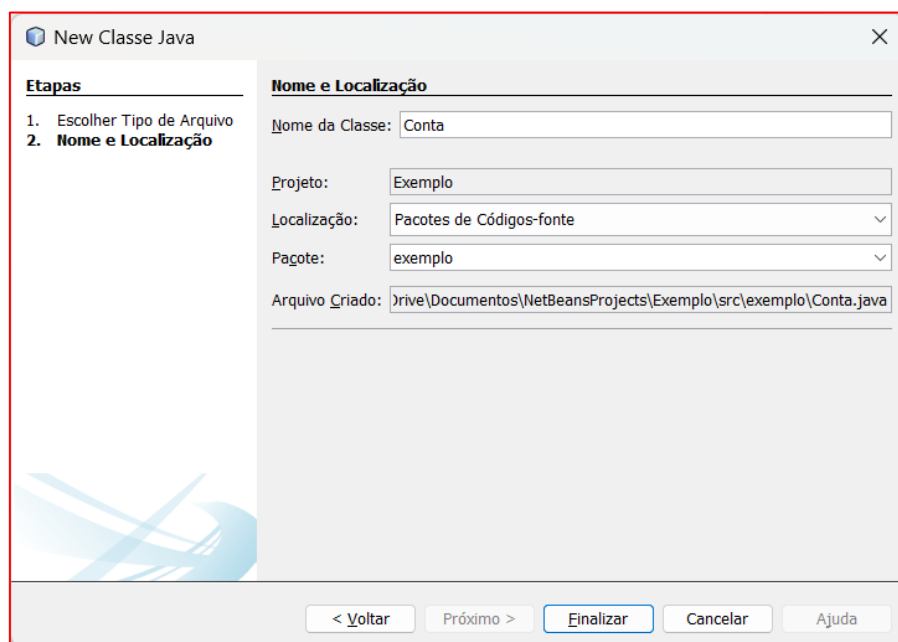
Agora, vamos transportar esse diagrama para o código no NetBeans:

1. Criar um projeto **Exemplo**

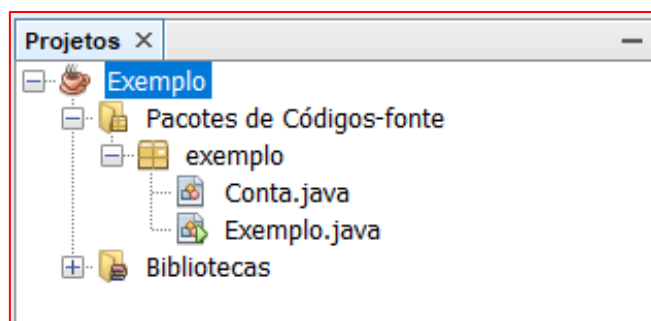


2. Com o pacote do projeto conforme figura anterior, após criar o projeto Exemplo, vamos criar a classe **conta**

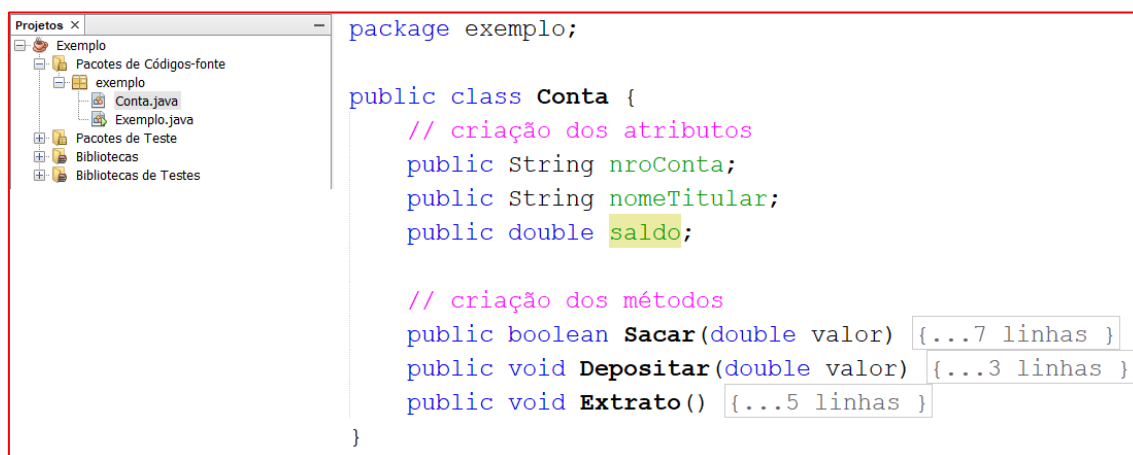
2.1. Com o botão direito do mouse no pacote do projeto, selecione Novo ou New, em seguida clique em Classe Java ou Java Class e será apresentado a tela abaixo, onde vamos digitar o nome da classe **Conta**.



3. Observando a estrutura do projeto no NetBeans, temos:



4. Agora podemos clicar no arquivo Conta.java e passar do diagrama **Conta** criado para o código.



Dentro da classe, também declararemos o que cada conta faz e como isto é feito - **os comportamentos que cada classe tem, isto é, o que ela faz**. Por exemplo, de que maneira que uma Conta saca dinheiro? Especificaremos isso dentro da própria classe **Conta**, e não em um local desatrelado das informações da própria Conta. É por isso que essas "**funções**" são chamadas de métodos. Pois é a maneira de fazer uma operação com um objeto.

Queremos criar um método que **saca** uma determinada quantidade e tem como retorno (verdadeiro – sim, há saldo na conta ou falso – não, há saldo suficiente para o saque), informação para quem acionar esse método:

```
public boolean Sacar(double valor) {
    if (valor >= this.saldo) {
        this.saldo -= valor;
        return true;
    }
    return false;
}
```

Quando alguém pedir para **sacar**, ele também vai dizer quanto quer sacar. Por isso precisamos declarar o método com algo dentro dos parênteses - **o que vai aí dentro é chamado de argumento do método (ou parâmetro)**. Essa variável é uma variável comum, chamada também de temporária ou local, pois, ao final da execução desse método, ela deixa de existir.

Usamos a palavra-chave **this** para mostrar que esse é um atributo, e não uma simples variável.

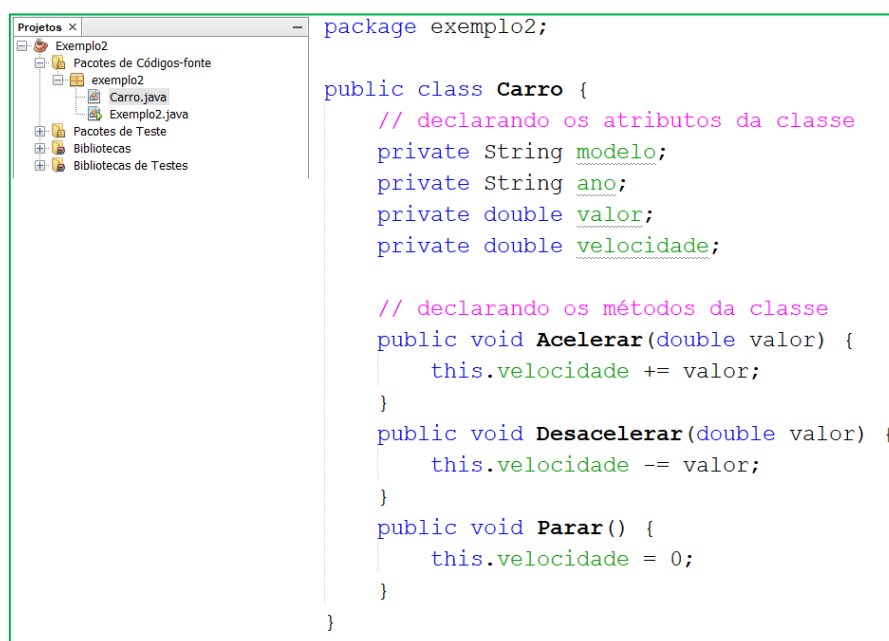
**this.saldo -= valor;**

Observe que não usamos uma variável auxiliar e, além disso, usamos a abreviação **-=** para deixar o método bem simples. O **-=** subtrai a quantidade ao valor antigo do saldo e guarda no próprio saldo, o valor resultante.

Da mesma forma, temos o método para depositar alguma quantia:

```
public void Depositar(double valor) {
    this.saldo += valor;
}
public void Extrato() {
    System.out.println("Nro da conta: " + this.nroConta);
    System.out.println("Nome do titular: " + this.nomeTitular);
    System.out.println("Sado da conta: " + this.saldo);
}
```

### Exemplo-2:



```
package exemplo2;

public class Carro {
    // declarando os atributos da classe
    private String modelo;
    private String ano;
    private double valor;
    private double velocidade;

    // declarando os métodos da classe
    public void Acelerar(double valor) {
        this.velocidade += valor;
    }
    public void Desacelerar(double valor) {
        this.velocidade -= valor;
    }
    public void Parar() {
        this.velocidade = 0;
    }
}
```

**Exemplo prático-1:**

O preço, ao consumidor, de um carro novo é a soma do custo de fábrica com a porcentagem do distribuidor e com os impostos, ambos aplicados ao custo de fábrica. As porcentagens encontram-se na tabela a seguir. Faça um projeto em Java que receba o custo de fábrica de um carro e mostre o preço ao consumidor. Para isso, crie uma **classe fabrica** que apresente os percentuais do distribuidor e a porcentagem dos impostos para o carro.

CUSTO DE FABRICA	% DISTRIBUIDOR	% IMPOSTOS
Até R\$ 21.000,00	5	isento
Entre R\$ 21.000,00 e R\$ 52.000,00	10	15
Acima de R\$ 52.000,00	15	20

```
package exemplo;

public class Fabrica {
    // declarando os atributos da classe
    public double custo;
    public double custoFabrica;
    public double custoImpostos;

    // metodos da classe
    public double valorDistribuidor() {

        if (this.custo <= 21000) {
            this.custoFabrica = this.custo * 0.05;
        } else if (this.custo > 21000 && this.custo <= 52000) {
            this.custoFabrica = this.custo * 0.10;
        } else {
            this.custoFabrica = this.custo * 0.15;
        }
        return this.custoFabrica;
    }

    public double valorImpostos() {

        if (this.custo <= 21000) {
            this.custoImpostos = 0;
        } else if (this.custo > 21000 && this.custo <= 52000) {
            this.custoImpostos = this.custo * 0.15;
        } else {
            this.custoImpostos = this.custo * 0.20;
        }
        return this.custoImpostos;
    }
}
```



```

package exemplo;

public class Exemplo {

    public static void main(String[] args) {
        Fabrica fabrica = new Fabrica();
        int i;

        // vamos executar os valores para 10 carros
        for (i = 0; i < 10; i++) {
            // valor aleatorio para o carro
            fabrica.custo = 1 + Math.random() * 60000;

            // estabelece o valor do distribuidor
            fabrica.valorDistribuidor();

            // estabelece o valor do imposto
            fabrica.valorImpostos();

            // Relatorio final
            System.out.println("Custo do carro: R$ " + String.format("%.2f", fabrica.custo));
            System.out.println("Custo do distribuidor: R$ " + String.format("%.2f", fabrica.custoFabrica));
            System.out.println("Custo do imposto: R$ " + String.format("%.2f", fabrica.custoImpostos));
            System.out.println("");
        }
    }
}

```

### Exemplo prático-2:

Faça um projeto em Java que receba o valor de uma dívida e mostre uma tabela com os seguintes dados:

- Valor da dívida;
- Valor dos juros;
- Quantidade de parcelas e;
- Valor da parcela.

Os juros e a quantidade de parcelas seguem a tabela:

QUANTIDADE DE PARCELAS	% DE JUROS SOBRE O VALOR INICIAL DA DÍVIDA
1	0
3	10
6	15
9	20
12	25

Exemplo de Saída:

VALOR DA DÍVIDA	VALOR DOS JUROS	QUANTIDADE DE PARCELAS	VALOR DA PARCELA
R\$ 1.000,00	0	1	R\$ 1.000,00
R\$ 1.100,00	100	3	R\$ 366,67
R\$ 1.150,00	150	6	R\$ 191,67

```

package exemplo;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Exemplo {

    public static void main(String[] args) {
        CalculaDivida calc = new CalculaDivida();
        Scanner sc = new Scanner(System.in);
        double valorDivida;

        try {

            System.out.println("");
            System.out.print("Digite o valor da dívida: ");
            valorDivida = sc.nextDouble();
            calc.calcular(valorDivida);
            calc.montaTabela();

        } catch (InputMismatchException ex) {
            System.out.println("Erro de digitação!");
        }

    }
}

```

```

package exemplo;
public class CalculaDivida {
    // declarando os atributos
    public double [][] tabela = new double[5][4];

    // declarando os metodos da classe
    public void calcular(double valor) {
        int i, parc;
        double indice;

        this.tabela[0][0] = valor;
        this.tabela[0][1] = 0;
        this.tabela[0][2] = 1;
        this.tabela[0][3] = valor;
        indice = 0.10;
        parc = 3;

        for (i = 1; i < 5; i++) {
            this.tabela[i][0] = valor + (valor * indice);
            this.tabela[i][1] = valor * indice;
            this.tabela[i][2] = parc;
            this.tabela[i][3] = (valor + (valor * indice)) / parc;
            indice += 0.05;
            parc += 3;
        }

    }

    public void montaTabela() {
        int i;

        System.out.println(" ");
        System.out.println("VALOR DÉVIDA \t VALOR DOS JUROS \t QTDE PARCELAS \t VALOR PARCELA");
        for (i = 0; i < 5; i++) {
            System.out.print("R$ " + String.format("%.2f", this.tabela[i][0]) + "\t ");
            System.out.print("R$ " + String.format("%.2f", this.tabela[i][1]) + "\t\t\t");
            System.out.print((int)this.tabela[i][2] + "\t ");
            System.out.print("R$ " + String.format("%.2f", this.tabela[i][3]));
            System.out.println("");
        }
        System.out.println("\n");
    }
}

```