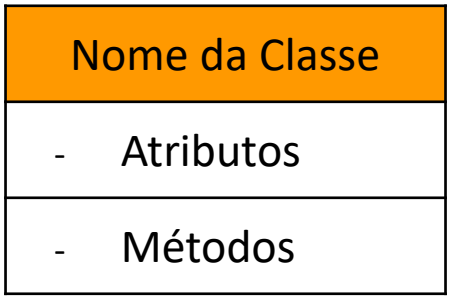
Programação Orientada a Objetos
Classes, Métodos e Atributos

Professor Isaac

CRIAÇÃO DE CLASSES EM JAVA

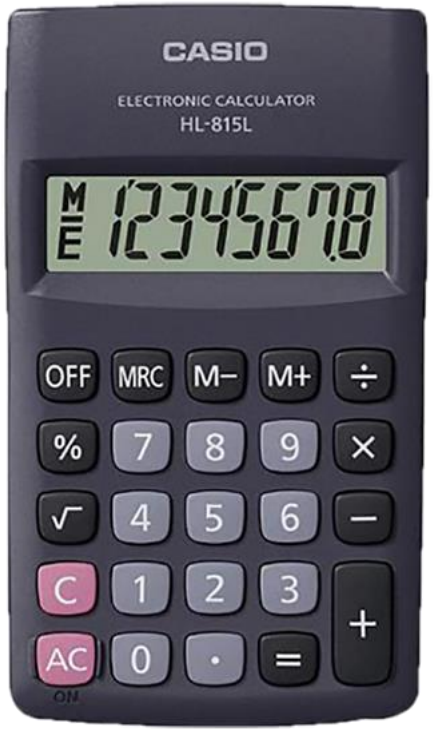
Estrutura de uma Classe

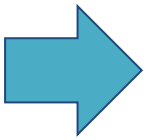


- Atributos são variáveis que armazenam informações do objeto.

- Métodos são as operações (funções) que o objeto pode realizar.

Exemplo de Classe





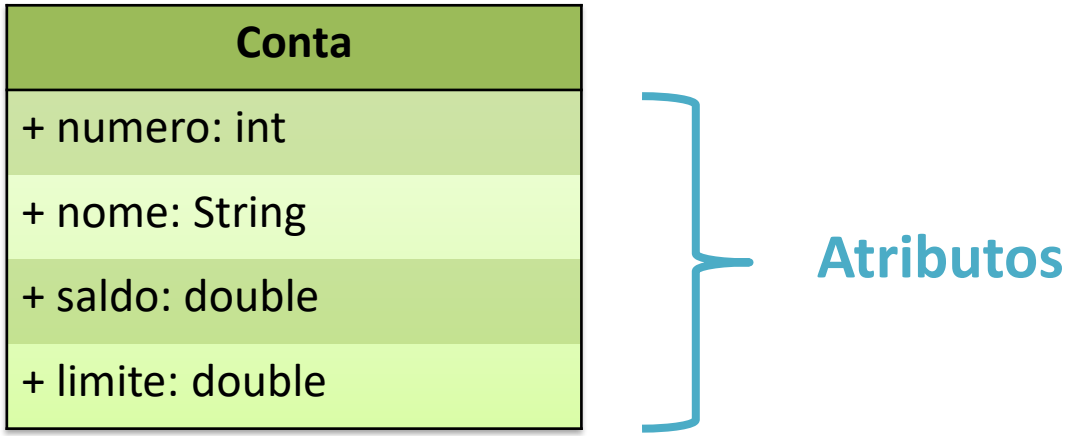
Calculadora
a: float b: float
Soma() Subtrai() Multiplica() Divide()

Atributos

Métodos

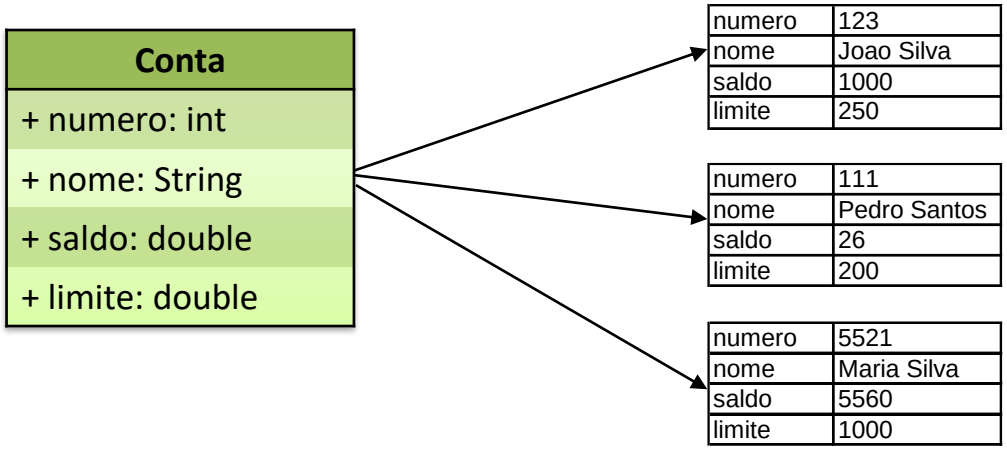
Conta Bancária - Classe

- Perceba que uma conta bancária possui estrutura bem definida.



Conta Bancária – Classe

- Gostaríamos na verdade que a especificação de uma conta pudesse ser utilizada quantas vezes fossem necessárias



Criação de uma Classe em Java

- A criação de uma classe em Java usa a palavra “class”.

- Sintaxe:

```
<tipo de acesso>    class    <NomeDaClasse> {
        // atributos e métodos da classe
}
```

- Inicialmente todas as classes que criaremos serão de acesso público (**public**).

Codificando a classe Conta

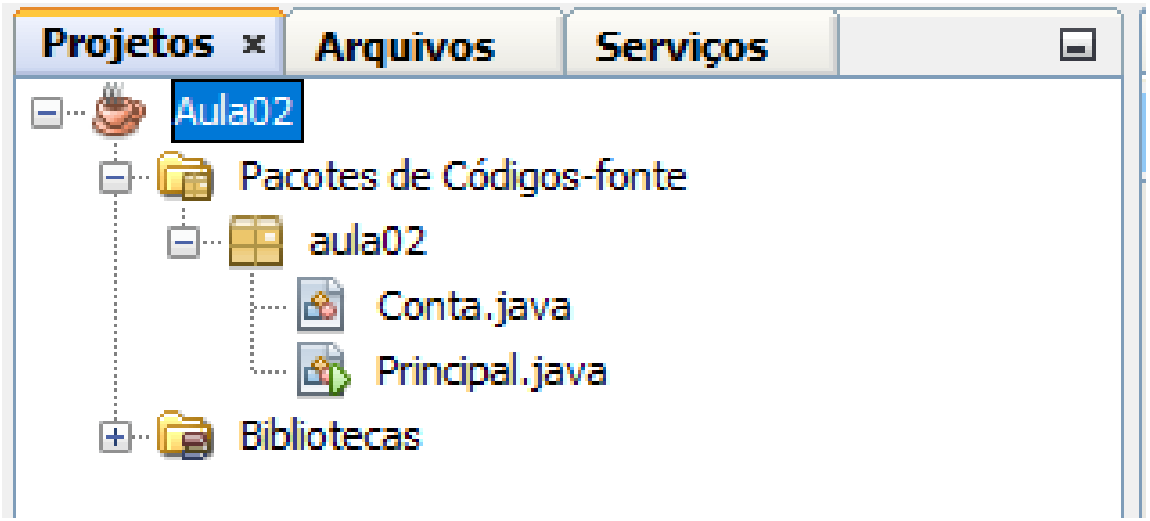
Conta
+ numero: int
+ nome: String
+ saldo: double
+ limite: double

```
public class Conta {
        public int numero;
        public String nome;
        public double saldo;
        public double limite;
}
```

EXERCÍCIO 01

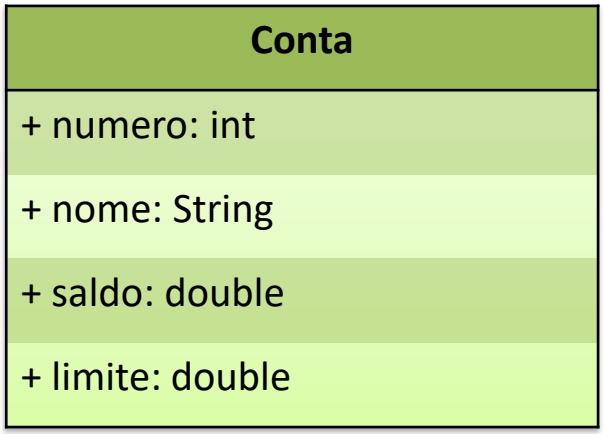
Exercício 1 - parte 1

- Crie um projeto no **Netbeans** com a classe **Principal**.
- Crie uma nova classe chamada **Conta** dentro do mesmo pacote da classe principal.



Exercício 1 - parte 1

- Dentro da classe crie os quatro atributos conforme diagrama de classes abaixo.




```
public class Conta {
        public int numero;
        public String nome;
        public double saldo;
        public double limite;
}
```

INSTANCIANDO UM OBJETO

Instanciando um objeto

- Uma classe pode ter vários objetos.
- Para instanciar um objeto de uma classe usamos a palavra “new”.
- Deve ser criada uma variável com o nome da classe e depois instanciada.

Sintaxe:

<NomeDaClasse> <nomeDoObjeto> = new < NomeDaClasse> (<parâmetros>);

- Os parâmetros são opcionais.

Instanciando um objeto Conta

```
public class Principal {
    public static void main(String[] args) {
        Conta conta01 = new Conta();
    }
}
```

- O comando **new Conta()**, realmente criamos (instanciamos) um **objeto** do tipo **Conta** e dissemos que a variável **conta01** representará este **objeto**

Acessar atributos de um objeto

- O acesso aos atributos de um objeto pode ser feito utilizando o nome do objeto, seguido de ponto (.) e seguido do atributo.

Sintaxe:

<nomeDoObjeto>.<nomeDoAtributo>

- O atributo pode ser acessado para conhecer seu valor ou para atribuir um valor

Acessando o objeto conta01

```
public class Principal {
    public static void main(String[] args) {
        Conta conta01 = new Conta();

        conta01.numero = 123;
        conta01.nome = "Joao Silva";
        conta01.saldo = 1000;
        conta01.limite = 250;
    }
}
```

Exercício 1 - parte 2

Na classe **Principal**:

- Instancie (**crie**) um objeto da classe **Conta**.
- Acesse os atributos do objeto que você criou, modificando seus valores.
- Use System.out.println() para mostrar os valores dos atributos.

Métodos

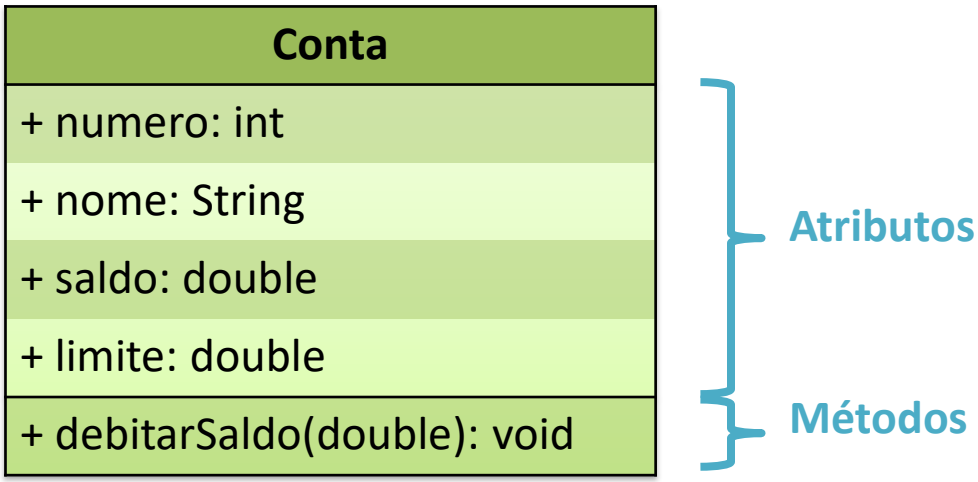
Criando métodos

Criação de métodos

- A sintaxe para criação de um método tem alguns itens obrigatórios e outros opcionais:

```
<tipo de acesso> <tipo de modificador> <tipo do retorno> <nomeDoMetodo>  (<parâmetros>){
        <comandos> // operações realizadas pelo método
}
```

- **Obrigatórios**: tipo do retorno, nome do método, parênteses, chaves e comandos
- **Opcionais**: tipo de acesso, tipo de modificador, parâmetros



Tipo de Acesso

- Quanto ao tipo de acesso e tipo de modificador, a omissão indica que serão usados aqueles padronizados na linguagem Java.
- Você irá aprendendo aos poucos sobre eles. Mas, no momento destacarei um tipo de acesso: **public**
- Quando definimos que um método é **public** isto indica que ele poderá ser usado por qualquer classe que compõe o seu programa

Tipo de Modificador

- Um tipo de modificador a ser destacado é o **static**
- O modificador static nada mais é do que uma garantia de que uma variável ou método terá apenas uma referência na memória
- Um método static pode ser chamado sem necessidade de ter um objeto atrelado a ele

Tipo de Retorno

- O retorno é a saída do método, o que irá retornar quando voltar para quem a chamou o método
- O tipo de retorno pode ser um tipo primitivo ou uma classe

Tipo de métodos quanto ao retorno e aos parâmetros

- Método SEM retorno e SEM parâmetros
- Método COM retorno e SEM parâmetros
- Método SEM retorno e COM parâmetros
- Método COM retorno e COM parâmetros

MétodoSEM retorno e SEM parâmetros

- Os métodos que não tem valor de retorno e também não recebem parâmetros utilizam a palavra "*void*" para indicar sem retorno (ou retorno vazio) e os parênteses vazios para indicar sem parâmetros:

```
void <nomeDoMétodo> ( ) {
        <comandos>
}
```

Método COM retorno e SEM parâmetros

- O **retorno** de um método é o valor que é retornado **desse método para aquele que o chamou**
- No método para retornar o valor deve ser usado o comando "*return*":

```
<tipo> < nomeDoMétodo> ( ) {
        <comandos>
        return <valor>;
}
```

Método SEM retorno e COM parâmetros

- Os **parâmetros** de um método são os valores que são enviados **do método que chamou para o outro método**

```
void < nomeDoMétodo>(        <tipo1> <nome1>,
                             <tipo2> <nome2>, ... ,
                             <tipoN> <nomeN>) {
    <comandos>
}
```

Método COM retorno e COM parâmetros

- O **retorno** de um método é o valor que é retornado **desse método para aquele que o chamou** usando o comando return
- Os **parâmetros** de um método são os valores que são enviados **do método que chamou para o outro método**

```
<tipo> < nomeDoMétodo> (<tipo1> <nome1>,
                        <tipo2> <nome2>, ... ,
                        <tipoN> <nomeN>) {
    <comandos>
    return <valor>;
}
```

Importante

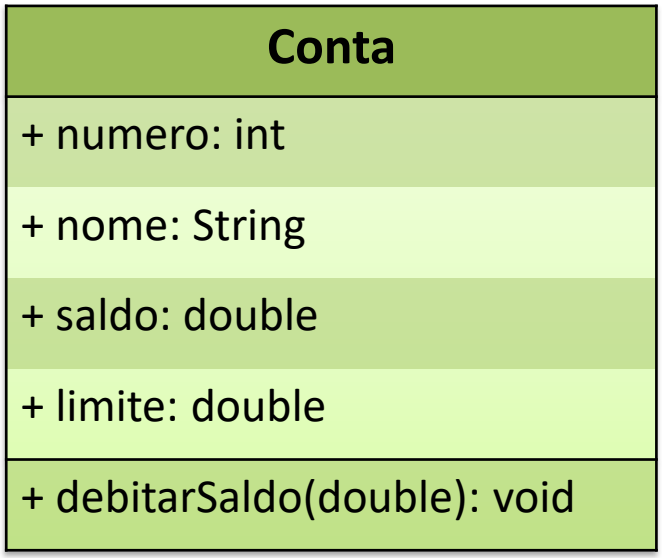
- **Apenas um** valor pode ser retornado por um método
- **Um ou mais parâmetros** podem ser recebidos pelo método
- **Qualquer método** também pode chamar outro método passando argumentos e recebendo retorno

Adicionando Métodos a Classe

- Anteriormente foi dito que as classes definem os atributos e comportamentos dos objetos que ela representa.
- Os métodos dão a classe o poder de executar determinadas ações.
- Por exemplo, podemos criar métodos para creditar valores ao saldo, debitar valores, transferir valores entre contas e etc.

Adicionando método a classe Conta

- Vamos adicionar o método debitarSaldo. Este método simplesmente deverá receber um valor e debitar do saldo total



Codificando Métodos na Classe

```
public class Conta {

    private int numero;
    private String nome;
    private double saldo;
    private double limite;

    public void debitarSaldo(double quantidade) {
        this.saldo = this.saldo - quantidade;
    }
}
```

- Perceba que quando vou acessar um atributo da classe usamos a palavra chave “***this***” *(mais adiante veremos que é opcional).*

Exercício 1 - parte 3

- Dentro da classe Conta crie o método **debitarSaldo()** conforme diagrama de classes abaixo.

Conta
+ numero: int
+ nome: String
+ saldo: double
+ limite: double
+ debitarSaldo(double): boolean

```
public class Conta {

    public int numero;
    public String nome;
    public double saldo;
    public double limite;

    public boolean debitarSaldo(double quantidade) {
        this.saldo = this.saldo - quantidade;
        return true;
    }
}
```

Chamada de métodos

Chamando Métodos sobre Objetos

- Para chamar um método basta informar o nome do variável que representa o objeto seguido por um ponto (.) e o nome do método a ser invocado.

<nomeDoObjeto>.<nomeDoMétodo> (<argumentos>);

- Exemplo: **conta01.debitarSaldo(150);**

Chamando Métodos da classe Conta

```
public class Principal {
    public static void main(String[] args) {
        Conta conta01 = new Conta();

        conta01.numeroConta = 123;
        conta01.nome = "Joao Silva";
        conta01.saldo = 1000;
        conta01.limite = 250;

        conta01.debitarSaldo(150);
    }
}
```

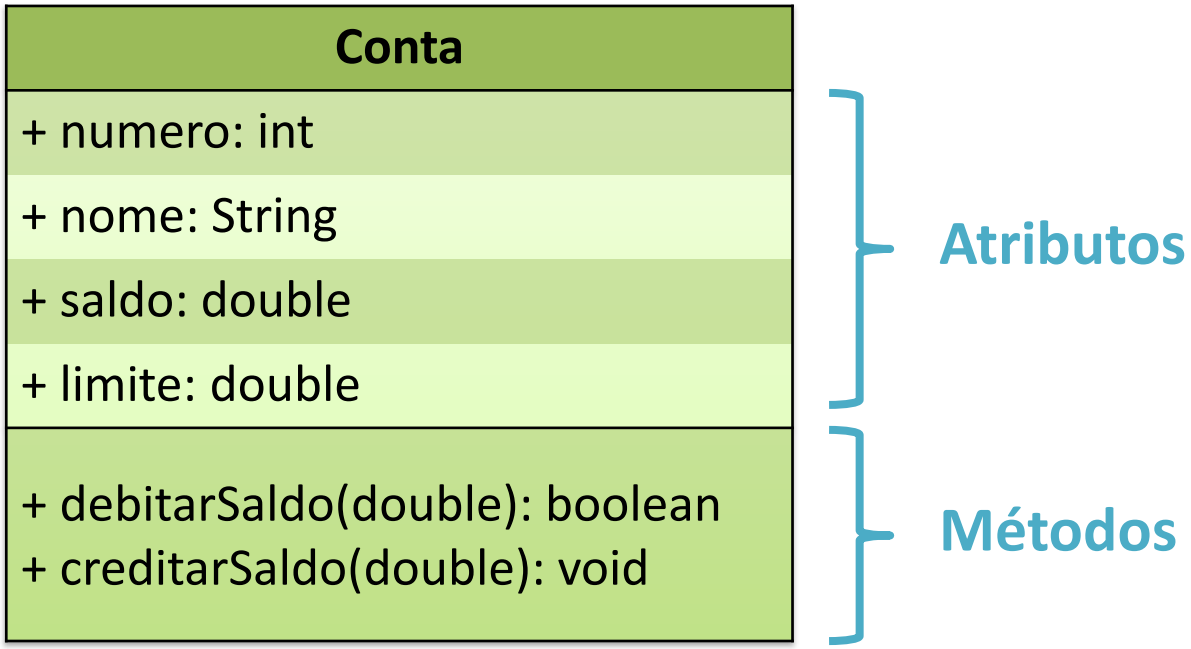
Exercício 1 - parte 4

Na classe **Principal**:

- Use o método **debitarSaldo()** do objeto Conta com um valor que você deseja debitar do saldo.
- Use **println()** ou **printf()** para mostrar os valores dos atributos (Obs: use um **println** ou **printf()** antes e um depois do método **debitarSaldo()** para verificar que o valor do atributo foi modificado).

Adicionando Métodos na classe Conta

Adicionando Métodos na classe Conta



Utilizando os novos métodos

```
public class Principal {
    public static void main(String[] args) {
        Conta conta01 = new Conta();
        conta01.numeroConta = 123;
        conta01.nome = "Joao Silva";
        conta01.saldo = 1000;
        conta01.limite = 250;

        conta01.creditarSaldo(150);

        if(conta01.debitarSaldo(150)){
            System.out.println("Debito realizado com sucesso");
        } else {
            System.out.println("Saldo indisponível");
        }
    }
}
```

Exercício 1 - parte 5

Na classe **Conta**:

- Modifique o método **debitarSaldo()** para que não seja possível sacar um valor maior que o saldo, conforme exemplo mostrado nos dois slides anteriores.
- Crie o método **creditarSaldo().**

Na classe **Principal**:

- Use o método **creditarSaldo()**.
- Use o método **debitarSaldo()** conforme exemplo do slide anterior.
- Use **System.out.println()** para mostrar os valores dos atributos.
- Use valores maiores e menores que o saldo no método **debitarSaldo(),** para verificar o funcionamento da lógica implementada no método.

Exercício 1 - parte 6

Na classe **Principal**:

- Crie mais 2 objetos chamados conta02 e conta03.
- Acesse os atributos dos objetos conta02 e conta03 que você criou, e modifique os valores dos atributos (O conteúdo deverá ser diferente).
- Use o método **creditarSaldo()** nos dois objetos conta02 e conta03.
- Use o método **debitarSaldo()** nos dois objetos conta02 e conta03.
- Use **System.out.println()** para mostrar os valores dos atributos dos objetos conta01, conta02 e conta03.

EXERCÍCIO 02

Exercício 02

- Passe todos os atributos da classe **Conta** para **private** e crie os métodos **getters** e **setters** desse atributos.
- Mantenha os mesmos métodos do exercício anterior.
- Execute os mesmos testes no **Main**, porém use os métodos getters e setters.