02

# Instanciação, atributos e referências

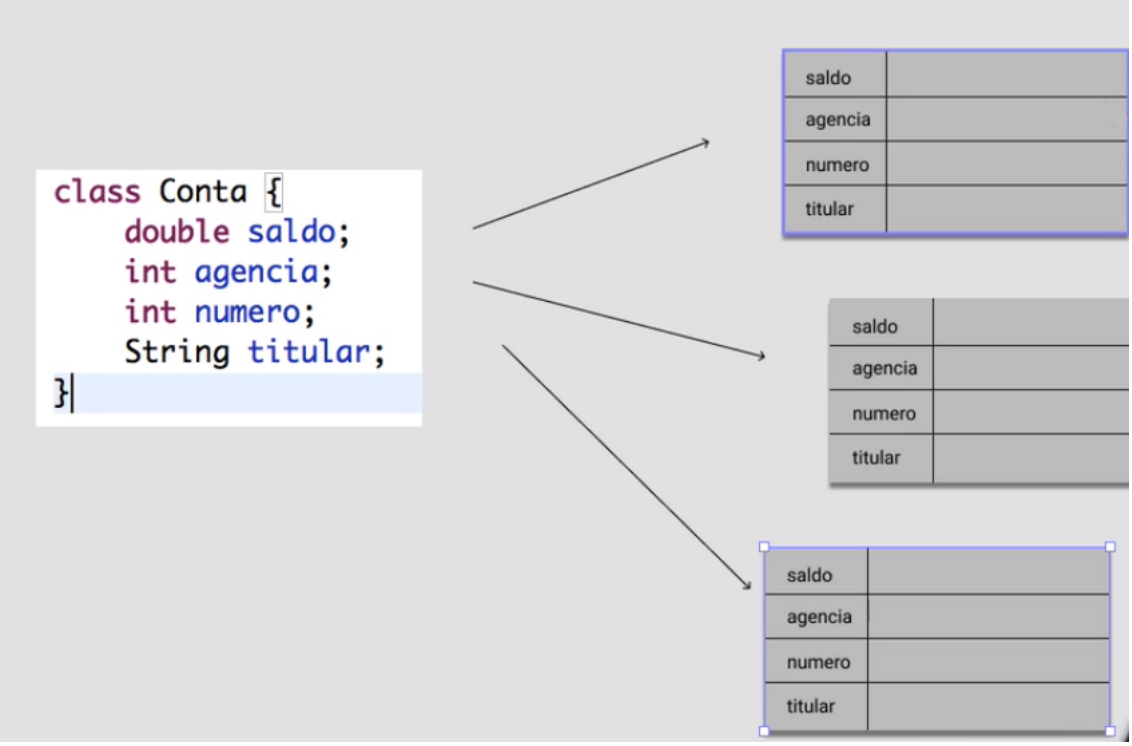
### Transcrição

Temos nossa primeira classe `Conta`, que possui quatro características: `saldo`, `agencia`, `numero` e `titular`. O nome que damos para tais características é **atributos**. Diremos que as contas do *ByteBank* possuem quatro atributos.

```
class Conta {
    double saldo;
    int agencia;
    int numero;
    String titular;
}
```

Ainda não temos uma conta bancária, pois não podemos fazer operações básicas que envolvem uma conta. O que possuímos é - como o quadro cinza feito no Figma - uma especificação de conta. A partir dessa especificação, podemos construir várias contas bancárias individuais.

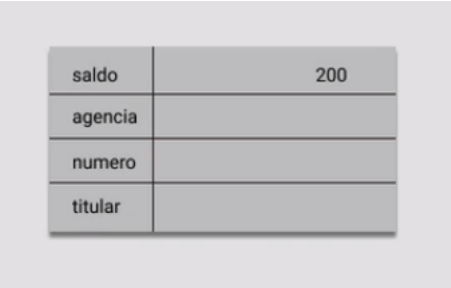
A tabela cinza, que é produzida através da classe `Conta`, chamaremos de **objeto** ou **instância**. A partir da especificação, construímos objetos ou instâncias do tipo `Conta`.




Da mesma forma que o processo de retirar do papel a planta de uma casa chama-se "construção", no caso da linguagem orientada a objeto, fala-se em "instanciação".

Dada uma classe `Conta`, instanciamos um objeto do tipo `Conta`. Discutiremos esses conceitos com mais profundidade no decorrer do curso.

Na atual condição do nosso código, não podemos depositar dinheiro na nossa conta bancária, mas podemos inserir um texto no campo "saldo", incluindo um valor de `200` reais.

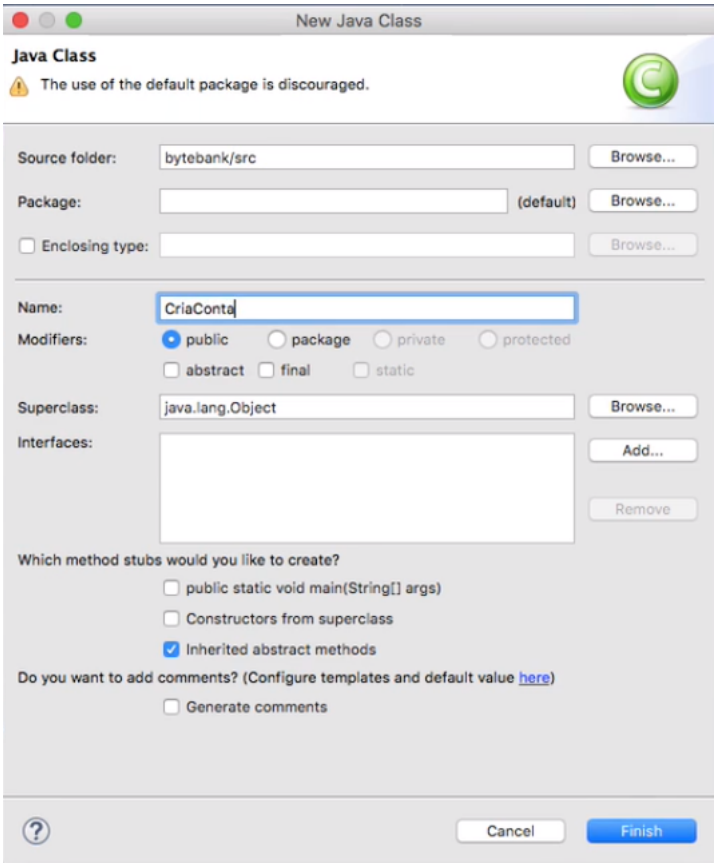



Tratando-se de um objeto do tipo `Conta`, podemos alterar um de seus atributos. Não podemos alterar a especificação do que é uma conta, mas as contas de forma individual.

Voltando ao código, adicionaremos o `public`, pois frequentemente o Eclipse irá inseri-lo automaticamente. Veremos com mais atenção o que isso significa no decorrer do curso.

```
public class Conta {
    saldo;
    agencia;
    numero;
    titular;
}
```

Na área *Package Explorer* selecionaremos a pasta `default value` e criaremos uma nova classe chamada `CriaConta`.




Essa nova classe faz exatamente o que seu nome anuncia: *cria contas*. Ela testará se o nosso código inicial está de fato funcionando como o esperado.

Na `CriaConta` incluiremos o `main` e, pressionando "Ctrl + Space", será incluído o método `public static void main` que é condição essencial para inicializar uma aplicação, ou seja, rodar o programa.

```
public class CriaConta {

    public static void main(String[] args) {

    }
}
```

Nosso objetivo é "tirar do papel" o código fonte e construir contas bancárias funcionais a partir dele. Para isso, adicionaremos a palavra-chave `new` e ao lado dela, adicionaremos o nome da classe que servirá para a criação de objetos. Neste caso, será a classe `Conta`. Vamos inserir também dois parênteses `()`. Não se preocupe! Entenderemos sua função posteriormente.

```
public class CriaConta {

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

Temos um programa que, dentro da nossa `main`, instancia - ou constrói - um objeto do tipo `Conta`. Neste ponto, o programa já pode ser executado, embora ainda esteja simples e não gere nenhum resultado, é um programa válido e é isso que queríamos construir.

Nosso próximo passo é escrever no código que o valor do saldo de uma conta específica vale `200`. Podemos incluir essa informação no código da seguinte maneira:

```
public class CriaConta {

    public static void main(String[] args) {
     new Conta();
     saldo = 200;
    }
}
```

Veremos que a palavra `saldo` não é identificada, pois no nosso contexto não existe nenhuma variável denominada `saldo`. Precisamos também criar um mecanismo de referenciação, ou seja, queremos assinalar que o saldo de `200` reais é referente à uma conta específica. Podemos fazer isso guardando o retorno de `new Conta()` em uma variável. Chamaremos essa variável de `primeiraConta`.

```
public class CriaConta {

    public static void main(String[] args) {
     primeiraConta = new Conta();
     saldo = 200;
    }
}
```

A variável `primeiraConta` também não será compilada, pois ela não existe no contexto. Mas quando formos declará-la ela será do tipo `Conta`.

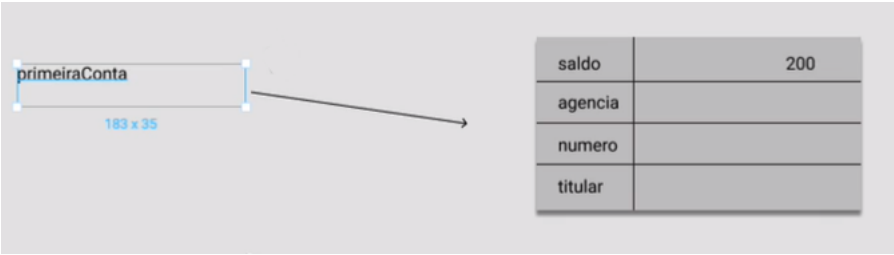
```
public class CriaConta {

    public static void main(String[] args) {
        Conta primeiraConta = new Conta();
        saldo = 200;
    }
}
```

Pode soar estranho ter de escrever o tipo `Conta` à direita e à esquerda da variável, mas existe o conceito de polimorfismo no Java que será desenvolvido futuramente durante o curso.

Algumas ideias ficam nebulosas nos primeiros passos no Java, mas precisamos seguir essa curva de aprendizado para começarmos a escrever códigos básicos.

Conseguimos referenciar uma conta específica. É comum ter a impressão de que a palavra-chave `new` nos devolve o objeto em si, e de que a variável `primeiraConta` contém o objeto, mas isso nunca ocorre.

No Java, assim como em outras linguagens, um objeto **nunca** está dentro de uma variável. O que tem dentro de uma variável é somente uma indicação a um objeto específico, uma *referência* a um objeto específico.



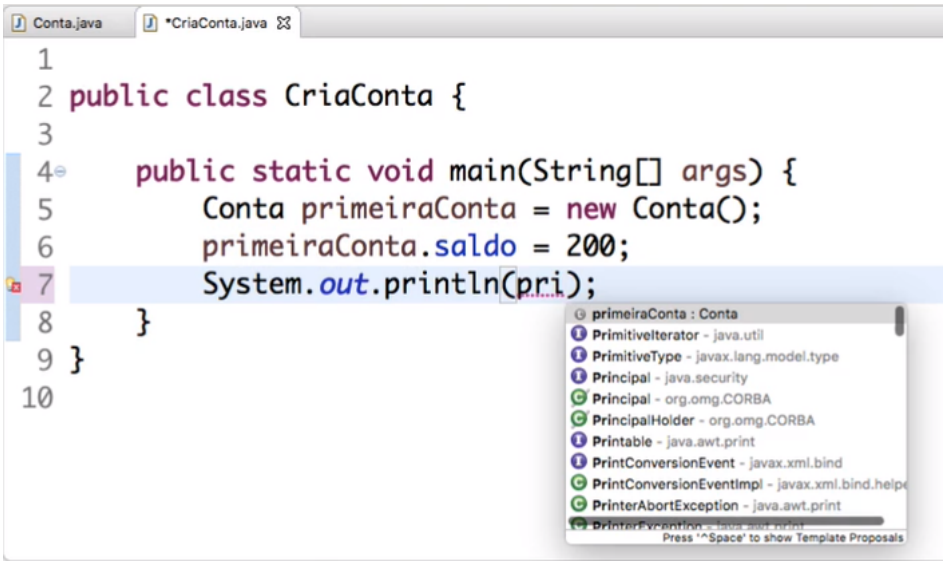

No nosso código, especificaremos que o valor de `saldo` a ser exibido é referente à `primeiraConta`. A navegação entre `primeiraConta` e `saldo` ocorre através do caractere `.`

```
public class CriaConta {

    public static void main(String[] args) {
    Conta primeiraConta = new Conta();
    primeiraConta.saldo = 200;
    }
}
```

Ao rodar o programa, nada acontece. O que falta é *mostrar* o valor do saldo na tela. Vamos imprimir o valor `200` acessando o atributo `saldo` de `primeiraConta`. Através do atalho `sysout` e depois "Ctrl + Space", será mostrado o `System.out.println()`, e então imprimiremos a variável `primeiraConta` acessando o seu `saldo` através do caractere `.`.

```
public class CriaConta {

    public static void main(String[] args) {
    Conta primeiraConta = new Conta();
    primeiraConta.saldo = 200;
    System.out.println(primeiraConta.saldo);
    }
}
```

Uma dica sobre o atalho "Ctrl+ Space": toda a vez que possui variáveis com nomes grandes, podemos acionar esse atalho e sugestões serão apresentadas.




Ao executarmos aplicação, o resultado será o valor do saldo da conta específica referenciada, ou seja, `200` .