**01Primeira classe - Contas**

**Transcrição**

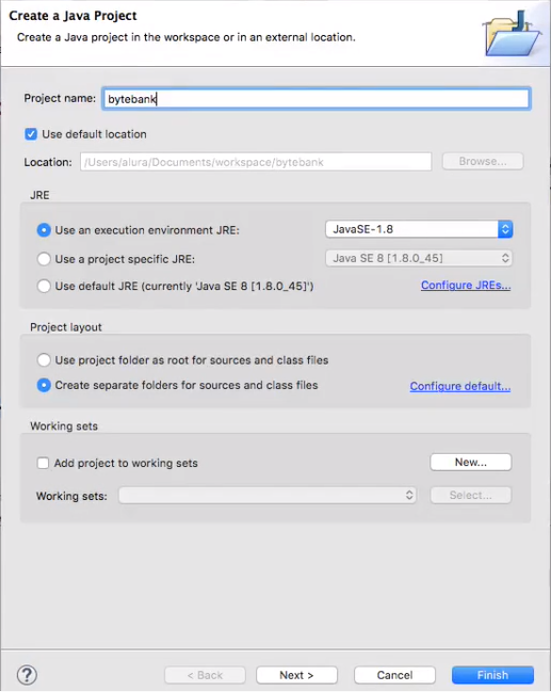
Faremos o **ByteBank**, um projeto de banco da **Alura**. Para isso, precisaremos criar o sistema que o compõe. Iremos descobrir quais dados serão necessários para esse sistema.

A menor unidade que iremos trabalhar é a *conta bancária*, e para isso, precisaremos fazer uma breve pesquisa para sabermos do que ela é composta.

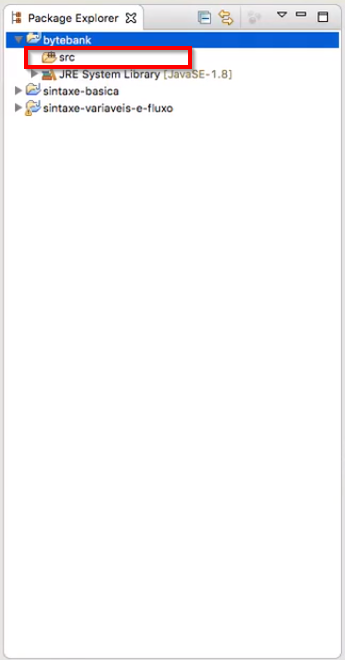
Se olharmos a definição de conta corrente no [WikiPedia](https://pt.wikipedia.org/wiki/Conta-corrente" \t "_blank), descobriremos uma série de referências acerca de juros e taxas, mas a questão que iremos nos atentar é que as contas bancárias acionam informações diferentes para fins variados.

Façamos uma analogia com uma seguradora de veículos: é importante que ela tenha informações como **marca**, **modelo**, **chassi** e a **cor** do carro. Mas é irrelevante para a seguradora saber quantos cilindros o carro tem ou a versão do motor. Já para uma empresa que fabrica veículos, esses dados fazem parte do **domínio** do problema.

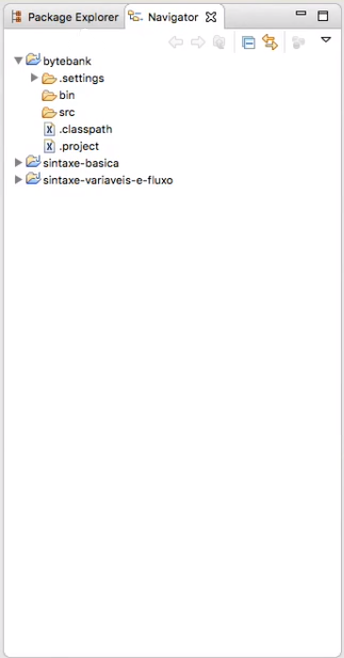
Criaremos um domínio para o nosso sistema baseado em um projeto de banco. No Eclipse, faremos um novo projeto Java intitulado bytebank. Depois, clique em "Finish".



Feito isso, será criado um novo projeto na área do *Package Explorer*. Ao clicarmos no projeto bytebank, veremos a pasta src.



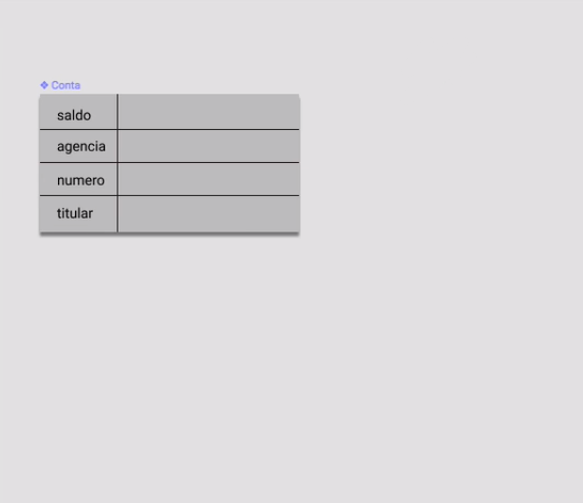
*Lembrando*: você pode selecionar "Window > Show View > Navigator". O *Navigator* possui uma ideia parecida com o Windows Explorer ou Finder, nele você visualizará mais diretórios internos. O *Package Explorer* oculta a visualização desses diretórios.



Em src, criaremos alguns códigos para guardarmos os dados sobre nossas contas bancárias. Uma característica interessante do Java é que ele nos possibilita a criação de **tipos**. Em um "tipo" do Java, armazenaremos todas as informações que constituem uma conta bancária.

Usaremos o [Figma](https://www.figma.com/" \t "_blank) - um *software* de criação de interface do usuário - para definirmos as informações básicas de uma conta bancária.

A nossa conta possui quatro características importantes para o nosso banco: **saldo**, **agência**, **número**, **titular**.

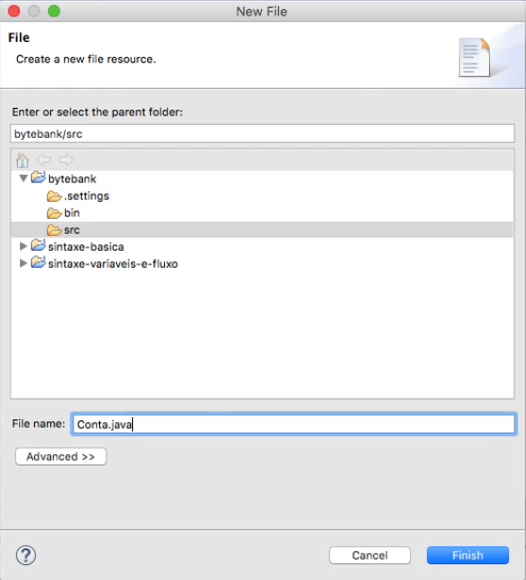


Veremos que no Java não serão guardados apenas os dados de conta bancária, mas também serão atribuídos comportamentos para a conta. Iremos definir algumas transações bancárias típicas, como realizar saques, depósitos e transferências.

O esquema que criamos no Figma pode ser reproduzido no Java, mas antes precisamos refletir: esse quadro com campos a serem preenchidos, pode ser caracterizado como uma conta? Não podemos realizar funções básicas de uma conta bancária apenas com esse quadro cinza, portanto, não se trata de uma conta, e sim, de uma especificação de um tipo conta.

Podemos fazer a seguinte analogia: a planta de uma casa não é uma casa, mas o esquema ou design de uma.

Recriaremos esse esquema de dados de conta bancária no Java. Em src, criaremos um novo arquivo chamado Conta.java.



Faremos a especificação dos componentes do tipo Conta.

tipo Conta:

saldo

agencia

numero

titularCopiar código

Obviamente, essa não é a sintaxe de Java. Precisaremos utilizar as regras da linguagem para realizarmos com sucesso a especificação do tipo Conta. Escreveremos ao lado de Conta a palavra class. Trata-se de uma palavra-chave que define um tipo. Para indicarmos que abrimos e fechamos um bloco de informações usaremos as chaves, {}.

class Conta {

saldo

agencia

numero

titular

}Copiar código

É também necessário declarar o tipo das outras categorias da conta. Para **saldo**, usaremos o tipo double, pois ele armazena dados de ponto flutuante.

Os componentes **agencia** e **numero** serão do tipo int. Para finalizar, **titular** será uma String, que guarda um conteúdo de texto.

Acionaremos o atalho "Ctrl + S" para salvarmos nosso trabalho. Com isso, já temos um código fonte válido em Java que já foi compilado. Ao observarmos a área *Navigator* e selecionarmos o diretório bin, veremos o arquivo Conta.class.

Porém, não podemos iniciar a execução do programa. Para que um programa possa ser iniciado em Java, ele precisa ter o ponto de entrada public static void main(String[]) args), a classe que criamos não possui esse ponto de entrada, pois trata-se de um arquivo de suporte.

Em uma aplicação Java é comum existirem vários arquivos que se comunicam, o primeiro a ser executado contém o método main com a "assinatura" (tipo de retorno e parâmetros) acima.

**02Instanciação, 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;

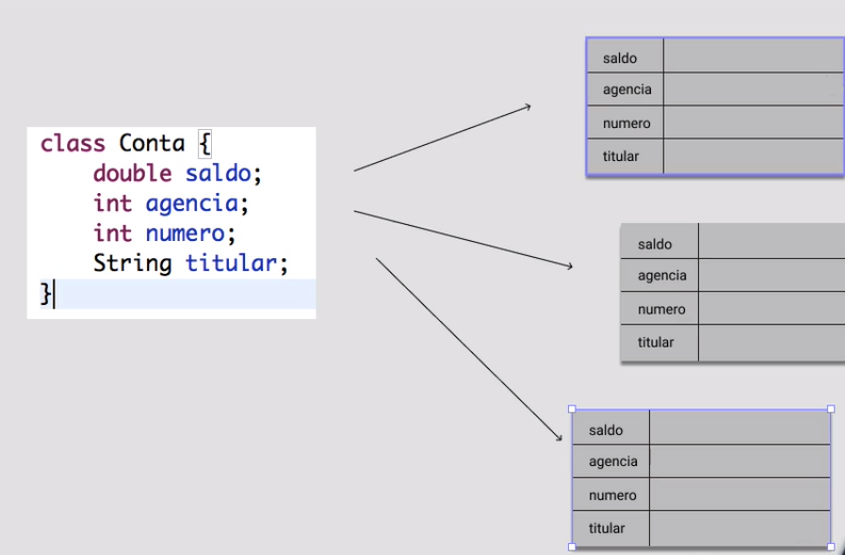
int numero;

String titular;

}Copiar código

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.

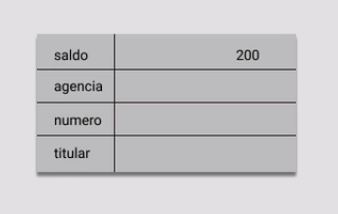
A tabela cinza, que é produzida através da classe Conta, chamaremos de **objeto** ou **instância** . A partir da especificação, construímos objetos ou instâncias do tipo Conta.



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;

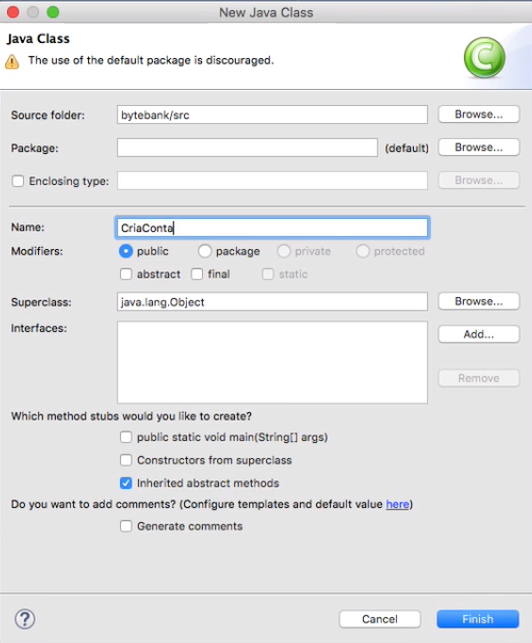
numero;

titular;

}

Copiar código

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) {

}

}Copiar código

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();

}

}Copiar código

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;

}

}Copiar código

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;

}

}Copiar código

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;

}

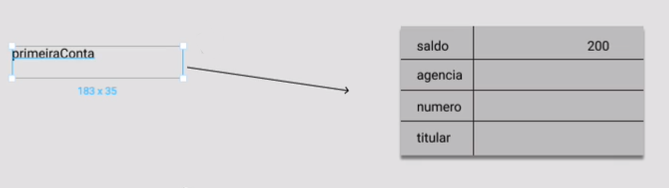
}Copiar código

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;

}

}Copiar código

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();

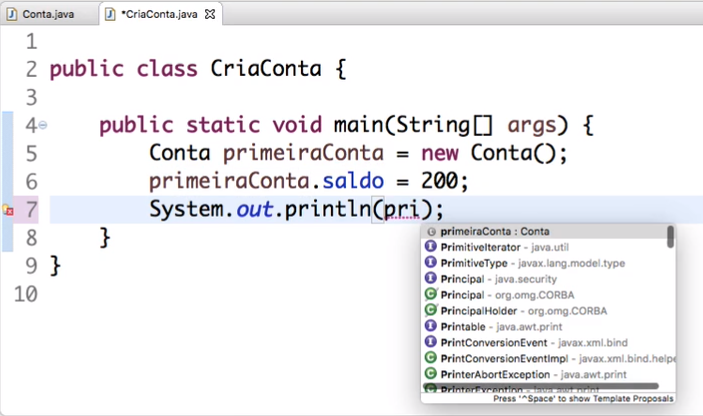
primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

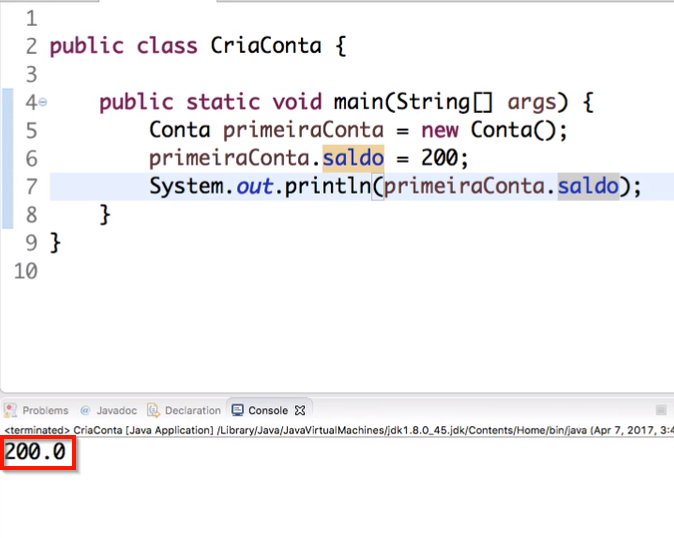
}

}Copiar código

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.



**03Características dos objetos**

Como chamamos, em orientação a objetos, as características de uma classe?

Parte superior do formulário

* Argumento
* Alternativa correta

Atributo

Correto, os atributos são as características que especificam uma classe.

* Alternativa correta

Variável

* Alternativa correta

Parametro

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**04Definindo tipos**

A partir do conhecimento adquirido de classes, leia as frases abaixo e responda a opção correta.

1. Uma classe é uma especificação de um tipo, definindo atributos e comportamentos.
2. Um objeto é uma instância de uma classe onde podemos definir valores para seus atributos.
3. Para criar uma instância é obrigatório preencher os valores de todos os atributos.
4. Para criar uma instância precisamos usar a palavra chave **new**

Parte superior do formulário

* Todas as opções estão corretas.
* Alternativa correta

A opção 1 é a única correta.

* Alternativa correta

A opção 3 é a única errada.

Certo, na criação de uma instância não precisamos preencher os atributos se não quisermos, isso faz com que a opção 3 esteja errada.

* Alternativa correta

As opções 2 e 3 estão erradas.

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**05Segunda Instância**

**Transcrição**

Iremos trabalhar mais com a primeiraConta. Lembrando que nosso código se encontra deste modo:

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

}

}Copiar código

Além de atribuirmos valores utilizando sinal =, podemos fazer uso de outros elementos da sintaxe básica do Java, como +=. Este elemento significa que o valor final de primeiraConta é aquele que já foi atribuído anteriormente (200) mais 100.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

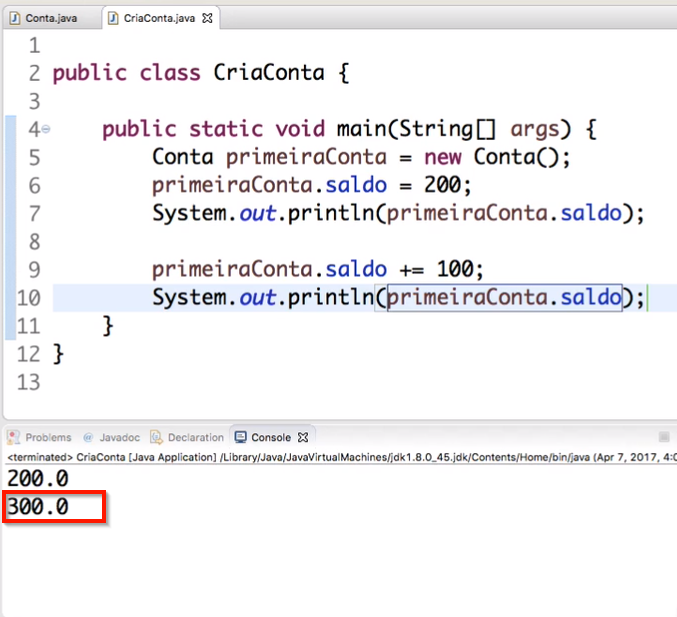
primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

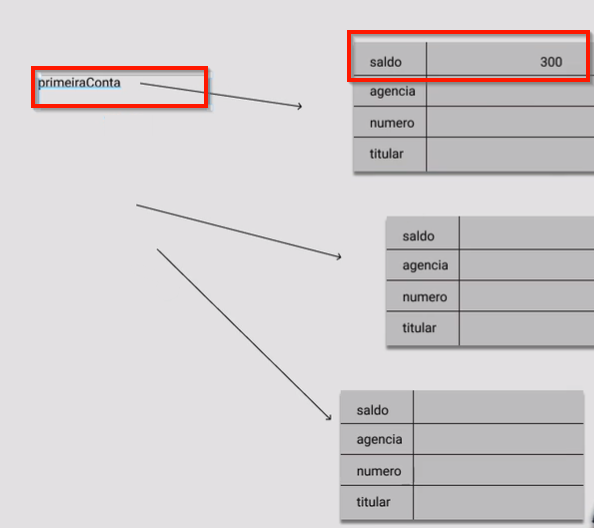
}

}Copiar código

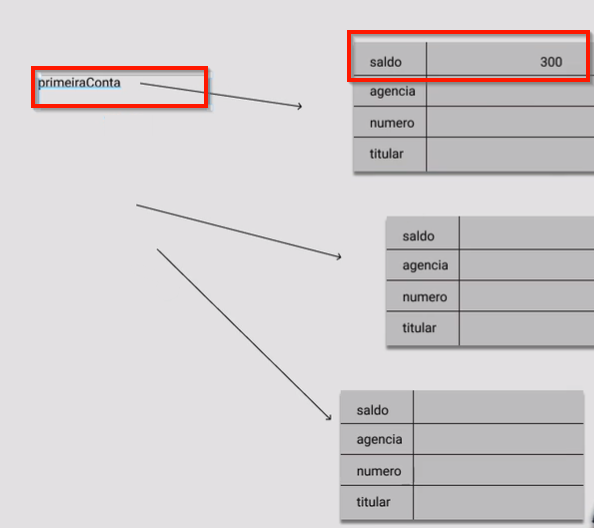
Ao executarmos a aplicação, veremos que será impresso o valor de 300.



O novo valor de primeiraConta.saldo é o resultado da soma do saldo anterior 200, mais 100. Chegando ao valor de 300. A variável primeiraConta é uma referência à uma conta específica, que chamamos de **objeto**.



Iremos modificar outras contas do nosso banco. Faremos uma referência à outra conta bancária através da referência segundaConta.



Como já vimos, daremos uma instanciação através do nosso código base embutido em Conta. Com isso, já temos os atributos criados (saldo, agência, número e titular) na conta nova que iremos trabalhar. Adicionaremos, também, a palavra chave new. A informação devolvida por new será guardada dentro da referência segundaConta, que será uma variável do tipo Conta.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

}

}Copiar código

Declararemos que em segundaConta há um saldo de 50.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

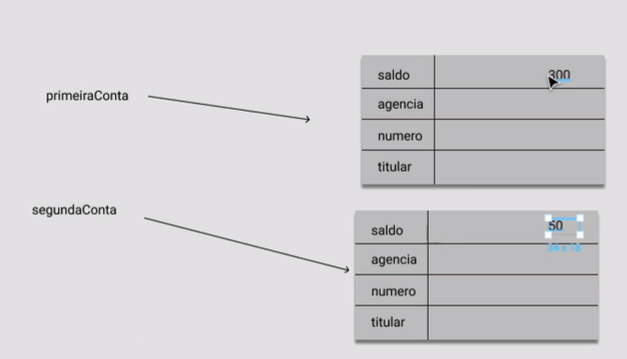
Conta segundaConta = new Conta();

segundaConta.saldo = 50;

}

}Copiar código

Portanto, segundaConta possui um saldo de 50 reais. Esse valor não possui qualquer ligação com o saldo de primeiraConta.



Para testarmos essa individualidade das contas, podemos pedir o saldo de primeiraConta e adicionar uma String "primeira conta tem". Faremos o mesmo procedimento com segundaConta.

Lembre-se de utilizar o + para a concatenação dos elementos.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

segundaConta.saldo = 50;

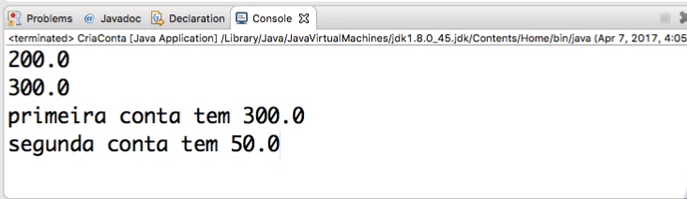
System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

}

}Copiar código

Ao executarmos a aplicação, veremos o saldos das contas apresentandos individualmente, pois são instâncias diferentes.



Não podemos simplesmente escrever no nosso código saldo = 50, pois o programa não achará a variável saldo no escopo, e ainda que achasse, existem múltiplos saldos devido a variedade de contas bancárias. Por isso, sempre devemos escrever a referência . atributo, ou seja, trabalhar de uma forma *orientada ao objeto*.

**06Valores default de atributos**

**Transcrição**

Vimos que os tipos de variáveis apresentadas no primeiro curso não possuíam valor padrão, ou seja, valor *default*. Não era possível se quer executar a aplicação com esse tipo de variável, pois não havia valores definidos.

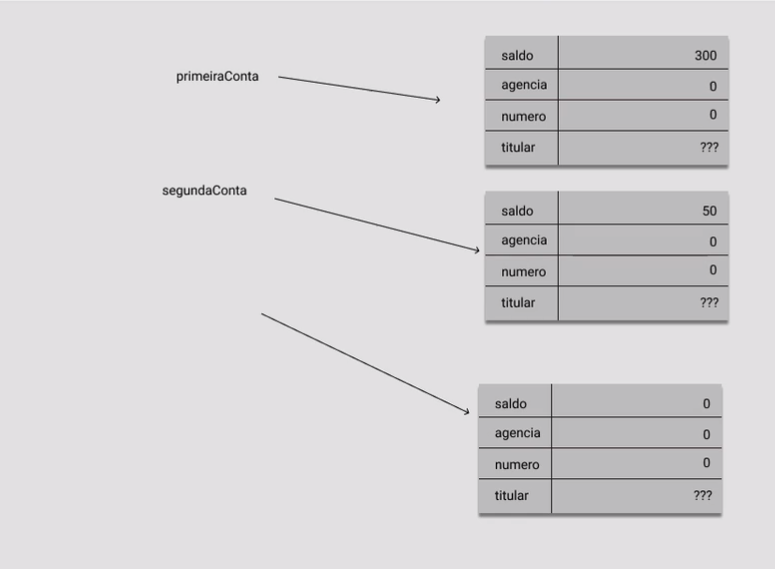
O tipos de variáveis que estamos trabalhando na classe Conta, não são as mesmas que ficam dentro do método main. As variáveis que estamos nos referindo ficam diretamente na classe.

Essas variáveis nós chamamos de *atributos* ou características de um objeto, comparativamente, elas são similares às especificações da planta de uma casa: se terá quatro quartos, uma sala, um banheiro. No caso de uma conta bancária, seus atributos seriam agência, conta, titular. Esses atributos podem ser chamados de **campo** ou **propriedade**.

Esse tipo de variável especial que possui significado para objetos tem um comportamento diferente. Quando acionamos a palavra-chave new e o Java instancia o objeto, todo os campos são zerados.

Para entendermos melhor essa ideia, observaremos os cartões cinzas que representam as especificações das nossas contas bancárias.

Percebam que os campos que não possuem valores estipulados estão zerados. Como titular é um tipo String, ou seja, não numérico, utilizamos as interrogações.



Para vermos como isso está representado no Eclipse, iremos na nossa classe CriaConta e solicitaremos a impressão do valor de agencia e numero de primeiraConta.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

segundaConta.saldo = 50;

System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

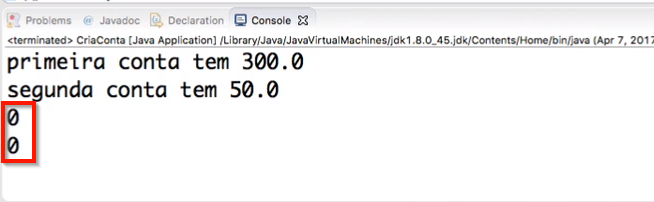
System.out.println(primeiraConta.agencia);

System.out.println(primeiraConta.numero);

}

}Copiar código

Ao solicitarmos a execução da aplicação, teremos o resultado dos valores de agencia e numero, que como sabemos, será 0 para ambos os casos. Percebam que não houve erros na execução, mesmo não existindo valores definidos para estes atributos.



Quando o Java constrói objetos, todos os seus atributos são zerados. 0 é o valor *default* de vários tipos numéricos, como int, double e long. No caso do tipo boolean o valor é *false*.

Poderíamos configurar valores padrão diferentes de zero, mas isso não seria interessante no nosso caso. Por exemplo, poderíamos dizer o valor de agencia sempre será 42 para todas as contas bancárias.

public class Conta {

double saldo;

int agencia = 42;

int numero;

String titular;

}Copiar código

A agência de todas as contas bancárias será 42. Podemos verificar isso solicitando a impressão de agencia da segundaConta e da primeiraConta.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

segundaConta.saldo = 50;

System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

System.out.println(primeiraConta.agencia);

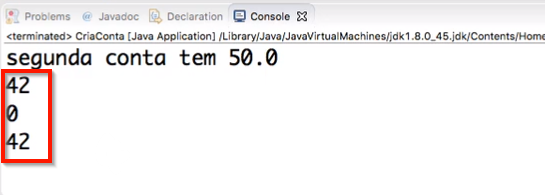
System.out.println(primeiraConta.numero);

System.out.println(segundaConta.agencia);

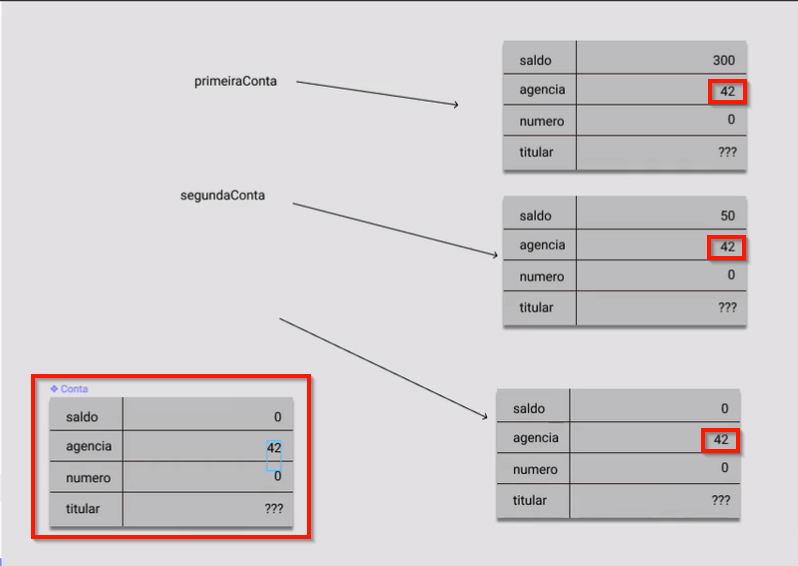
}

}Copiar código

O resultado será 42 para ambos os casos.



Configuramos o valor 42 como default para todas as agências. Podemos representar essa ideia tomando como configuração padrão o cartão cinza no canto esquerdo da tela.



Podemos alterar o valor no objeto em si. Como por exemplo, redefiniremos um novo valor de agencia para segundaConta, que será 146.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

segundaConta.saldo = 50;

System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

System.out.println(primeiraConta.agencia);

System.out.println(primeiraConta.numero);

System.out.println(segundaConta.agencia);

segundaConta.agencia = 146;

System.out.println("agora a segunda conta está na agencia " + segundaConta.agencia);

}

}Copiar código

Com isso, modificamos o valor de agencia de segundaConta para 146.

**07Definindo valor de atributos**

Jonas criou uma classe do tipo Pessoa para representar um personagem de um jogo que está criando.

Observe a classe que ele criou:

public class Pessoa {

String nome;

int idade;

int peso;

}Copiar código

Qual das opções abaixo é a correta para criar um objeto e definir um valor para seus atributos?

Parte superior do formulário

* new Pessoa();
* Pessoa.nome = "Jonny"
* Pessoa.idade = 34;
* Alternativa correta
* Pessoa heroi = new Pessoa();

heroi.nome = "Jonny";

Opção correta! O objeto é criado e sua referência é atribuída à variável heroi, em seguida, atributo nome está definido no lugar correto de forma que a atribuição seja feita com sucesso.

* Alternativa correta
* Pessoa pessoa = new Pessoa();

pessoa = nome, "João"

* Alternativa correta
* new Pessoa();
* nome = "Jonny";
* idade = 45;

peso = 100;

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

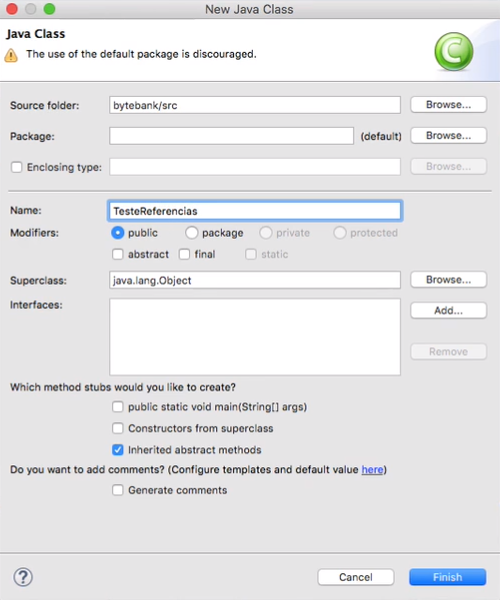
Sim

Não muito

**08Referências vs Objetos**

**Transcrição**

Nos atentaremos para um detalhe que causa confusão mesmo em programadores mais experientes. Criaremos uma nova classe chamada TesteReferencias.



Nesta nova classe, adicionaremos a main, e criaremos uma conta chamada primeiraConta. Estipularemos o valor de 300 para saldo.

public class TesteReferencias {

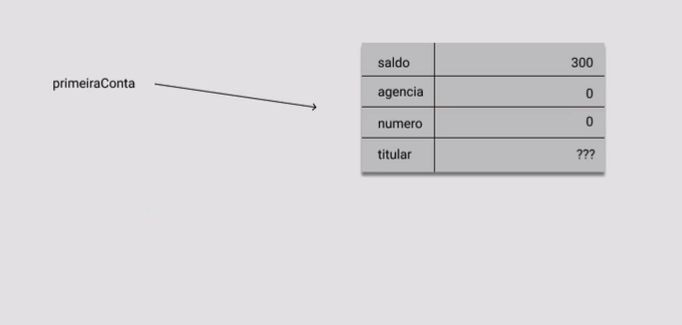
public static void main(String [] args) {

Conta primeiraConta = new Conta();

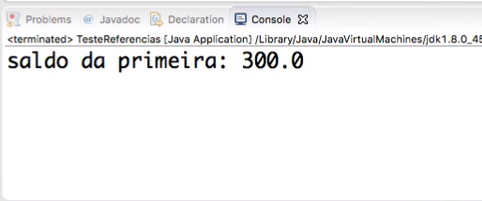
primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);Copiar código

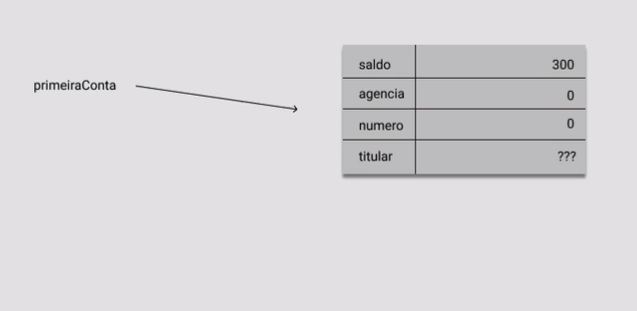
Temos a seguinte representação no cartão cinza:



Executaremos o programa e verificaremos que o código está funcional.



Nos atentaremos para uma questão importante: a diferença entre o tipo (Conta) e a variável desse tipo (primeiraConta). A variável não é um objeto Conta, e sim, uma indicação a um objeto específico, uma *referência* de um objeto. Sua representação gráfica seria a flecha que referencia o objeto.



Observem a seguinte declaração:

public class TesteReferencias {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);

Conta segundaConta = primeiraConta;

}

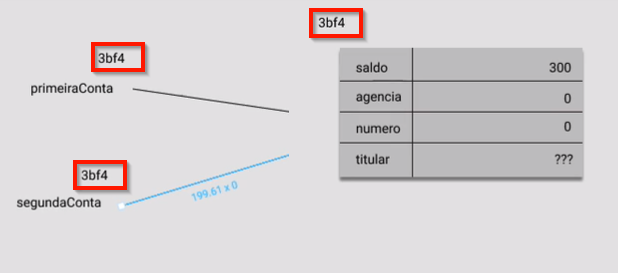
}Copiar código

A princípio, podemos pensar que esta declaração gera uma cópia da primeiraConta para a segundaConta e teríamos uma espécie de "clone de objeto".

Lembrem-se = no Java copia o que está na direita e cola na esquerda.

A questão é que não há um objeto Conta à direita, e sim uma **referência**. O que copiamos é a referência para um mesmo objeto.

Pense da seguinte forma: existe uma espécie de Id dos objetos, que chamaremos de 3bf4. A variável primeiraConta possui o valor 3bf4, fazendo referência ao Id do objeto. Quando declaramos que primeiraConta = segundaConta, na verdade estamos copiando esse Id 3bf4 que é a referência, e não o objeto em si.



O que temos são duas referências para o mesmo objeto. É como se duas cartas fossem endereçadas ao mesmo local. Embora sejam cartas diferentes, possuem o mesmo destino.

Veremos qual é o resultado dessa dupla referenciação no nosso código. Faremos o sysout no saldo de segundaConta.

public class TesteReferencias {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);

Conta segundaConta = primeiraConta;

}

}Copiar código

Ao executarmos o código, veremos que o programa irá imprimir o valor 300, pois temos duas variáveis, e não dois objetos.

Prosseguiremos com um desafio mais complicado. Primeiramente, vamos incluir mais 100 reais em sengundaConta.

public class TesteReferencias {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);

Conta segundaConta = primeiraConta;

System.out.println("saldo da segunda conta: " + segundaConta.saldo);

}

}Copiar código

Ao executarmos a aplicação teremos um saldo de 400 reais para segundaConta. Feito isso, iremos acionar o sysout no saldo de primeiraConta.

public class TesteReferencias {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);

Conta segundaConta = primeiraConta;

System.out.println("saldo da segunda conta: " + segundaConta.saldo);

segundaConta.saldo += 100;

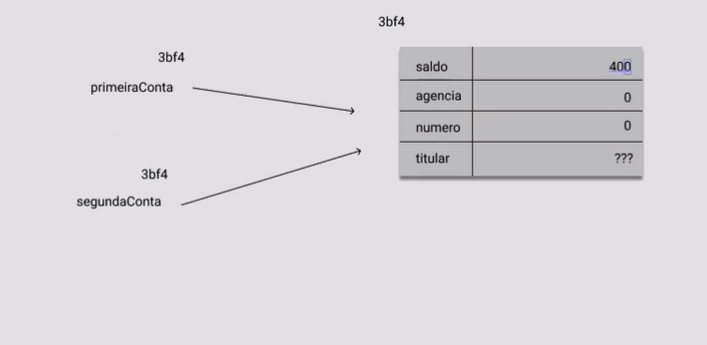
System.out.println("saldo da segunda conta " + segundaConta.saldo);

System.out.println(primeiraConta.saldo);

}

}Copiar código

Quando executarmos o programa teremos um saldo de 300 ou 400? Continuaremos tendo duas referências para apenas um objeto, portanto, o saldo será de 400.



Podemos verificar se primeiraConta possui as mesmas informações de segundaConta, fazendo um if.

public class TesteReferencias {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);

Conta segundaConta = primeiraConta;

System.out.println("saldo da segunda conta: " + segundaConta.saldo);

segundaConta.saldo += 100;

System.out.println("saldo da segunda conta " + segundaConta.saldo);

System.out.println(primeiraConta.saldo);

if(primeiraConta == segundaConta) {

System.out.println("é a mesma conta");

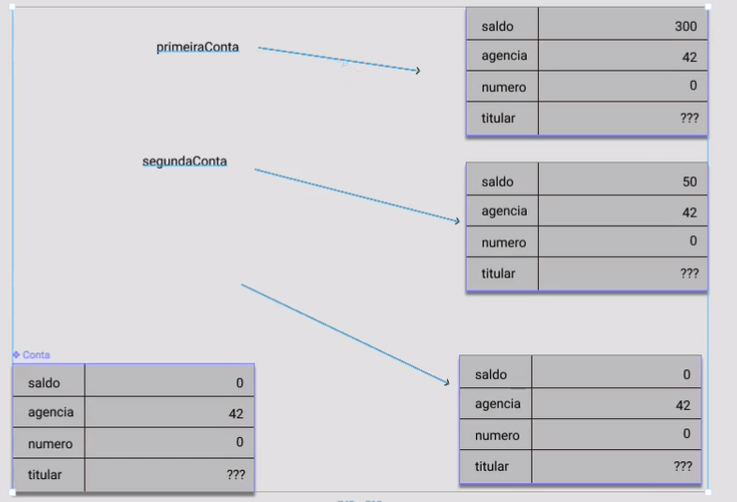
}

}

}Copiar código

Os números de referência são iguais, portanto, são a mesma conta, fazem referência ao mesmo objeto neste código.

Se formos na classe CriaConta que fizemos nas aulas passadas, teremos as variáveis primeiraConta e segundaConta referenciando objetos diferentes. Há dois news no nosso código, um para cada referência. Podemos perceber a individualidade das referências na representação gráfica do código anterior.



Na classe CriaConta, iremos fazer um if e um else para vermos como o nosso código se comporta.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

segundaConta.saldo = 50;

System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

System.out.println(primeiraConta.agencia);

System.out.println(primeiraConta.numero);

System.out.println(segundaConta.agencia);

segundaConta.agencia = 146;

System.out.println("agora a segunda conta está na agencia " + segundaConta.agencia);

if(primeiraConta == segundaConta) {

System.out.println("mesma conta");

} else {

System.out.println("contas diferentes");

}

}

}Copiar código

O resultado da execução dessa aplicação será contas diferentes. O sinal == irá **comparar referências**, e não objetos.

De volta à classe TesteReferencias, acionaremos o sysout para primeiraConta.

public class TesteReferencias {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);

Conta segundaConta = primeiraConta;

System.out.println("saldo da segunda conta: " + segundaConta.saldo);

segundaConta.saldo += 100;

System.out.println("saldo da segunda conta " + segundaConta.saldo);

System.out.println(primeiraConta.saldo);

if(primeiraConta == segundaConta) {

System.out.println("é a mesma conta");

}

System.out.println(primeiraConta);

}

}Copiar código

Ao executarmos o programa, veremos que o Java irá imprimir Conta@15db9742. Estamos fazendo uma referência a um objeto do tipo Conta, e que este objeto está dentro de uma "gaveta de memória" identificada por essa sequência numérica, estamos falando da mesma ideia do "Id" que vimos anteriormente com o 3bf4. Faremos o mesmo procedimento com segundaConta.

public class TesteReferencias {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 300;

System.out.println("saldo da primeira: " + primeiraConta.saldo);

Conta segundaConta = primeiraConta;

System.out.println("saldo da segunda conta: " + segundaConta.saldo);

segundaConta.saldo += 100;

System.out.println("saldo da segunda conta " + segundaConta.saldo);

System.out.println(primeiraConta.saldo);

if(primeiraConta == segundaConta) {

System.out.println("é a mesma conta");

}

System.out.println(primeiraConta);

System.out.println(segundaConta);

}

}Copiar código

O resultado impresso será o mesmo, pois são duas referências apontando para o mesmo objeto.



Faremos o mesmo procedimento na classe CriaConta para que vejamos a diferença de comportamento de um código para o outro.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.Saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

segundaConta.saldo = 50;

System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

System.out.println(primeiraConta.agencia);

System.out.println(primeiraConta.numero);

System.out.println(segundaConta.agencia);

segundaConta.agencia = 146;

System.out.println("agora a segunda conta está na agencia " + segundaConta.agencia);

if(primeiraConta == SegundaConta) {

System.out.println("mesma conta");

} else {

System.out.println("contas diferentes");

}

System.out.println(primeiraConta);

System.out.println(segundaConta);

}

}Copiar código

Quando iniciamos a aplicação, veremos que o Java imprimiu os valores Conta@15db972 e Conta@6d06d69c diferentes , afinal, estamos trabalhando com referências orientadas para dois objetos diferentes.

Façamos um teste: colocaremos a mesma quantidade de saldo e a mesmo número em agencia nas duas contas, e verificaremos se a numeração Id continua diferente.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = new Conta();

segundaConta.saldo = 300;

System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

segundaConta.agencia = 146;

System.out.println(primeiraConta.agencia);

System.out.println(primeiraConta.numero);

System.out.println(segundaConta.agencia);

segundaConta.agencia = 146;

System.out.println("agora a segunda conta está na agencia " + segundaConta.agencia);

if(primeiraConta == segundaConta) {

System.out.println("mesma conta");

} else {

System.out.println("contas diferentes");

}

System.out.println(primeiraConta);

System.out.println(segundaConta);

}

}Copiar código

Temos o saldo de 300 para ambas as contas, bem como a agencia de número 146, e a numeração das contas continua diferente, sendo Conta@15db9742 e Conta@6d06d69c. O Java não irá comparar objetos, e sim **referências**.

**09Referências de objetos**

Usando o aprendizado sobre referências e atribuição de valores vamos definir uma classe abaixo.

public class Conta {

double saldo;

}Copiar código

A partir desta classe, diga o que imprime o código:

public class Teste {

public static void main(String [] args) {

Conta minhaConta = new Conta();

minhaConta.saldo = 500.0;

Conta outraConta = minhaConta;

outraConta.saldo += 1000.0;

System.out.println(minhaConta.saldo);

}

}

Parte superior do formulário

* Imprime 500.0
* Alternativa correta

Imprime 1000.0

* Alternativa correta

Imprime 1500.0

Imprime 1500, pois as duas referências (minhaConta e outraConta) apontam para o mesmo objeto, fazendo com que a quantidade 1000 seja somada à anterior 500.

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**10Mão na massa: Criando as primeiras classes**

Vamos começar a implementar nosso projeto!

1) Crie um novo projeto, no curso usamos o nome "**bytebank**".

2) Declare sua primeira classe do tipo **Conta** e defina os atributos que julgar necessários.

public class Conta{

double saldo;

int agencia;

int numero;

String titular;

}Copiar código

3) Crie mais uma classe, no curso utilizamos o nome "**CriaConta**".

4) Na classe **CriaConta**, crie seu método main e instancie uma **Conta**.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

}

}Copiar código

5) Altere o saldo da **primeiraConta** para 200 e imprima esse valor para confirmar que está tudo ok.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

}

}Copiar código

6) Faça uma operação e adicione mais 100 ao saldo da **primeiraConta**:

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

}

}Copiar código

Nossa saída deve ser 200 e 300.

7) Cria mais uma referência para o objeto conta atual, no nosso caso, **segundaConta** e modifique o saldo para 50. Nosso código deve ficar assim:

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

System.out.println(primeiraConta.saldo);

primeiraConta.saldo += 100;

System.out.println(primeiraConta.saldo);

Conta segundaConta = primeiraConta;

segundaConta.saldo = 50;

System.out.println("primeira conta tem " + primeiraConta.saldo);

System.out.println("segunda conta tem " + segundaConta.saldo);

}

}Copiar código

8) Como falado nos vídeos, **primeiraConta** e **segundaConta** são referências, quando imprimimos, vemos um código um tanto estranho, para isso basta adicionar:

System.out.println(primeiraConta);Copiar código

**Opinião do instrutor**

Caso tenha dúvidas, fale conosco no fórum!

**11O que aprendemos?**

Nessa aula aprendemos sobre atributos e criação de objetos.

Vimos:

* O que é uma classe
* Como criar uma classe
* Como criar um objeto ou instância de uma classe
* O que são atributos
* Como definir e visualizar valores de atributos
* Como funciona a atribuição de uma referência à uma variável

Vamos ver mais sobre as classes na próxima aula! Agora não só suas características mas também seu comportamento. Vamos que só está no começo! Espero você lá!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**12Arquivos do projeto atual**

No link abaixo, você encontra o projeto até o momento atual do curso.

<https://github.com/alura-cursos/Curso-Java-parte-2-Introdu-o-Orienta-o-a-Objetos/archive/capitulo2.zip>

**Opinião do instrutor**

Lembre-se de usar o fórum em caso de dúvidas.