POO Java Alura

Nesta Aula aprenderemos sobre atributos e criação de objetos.

Veremos:

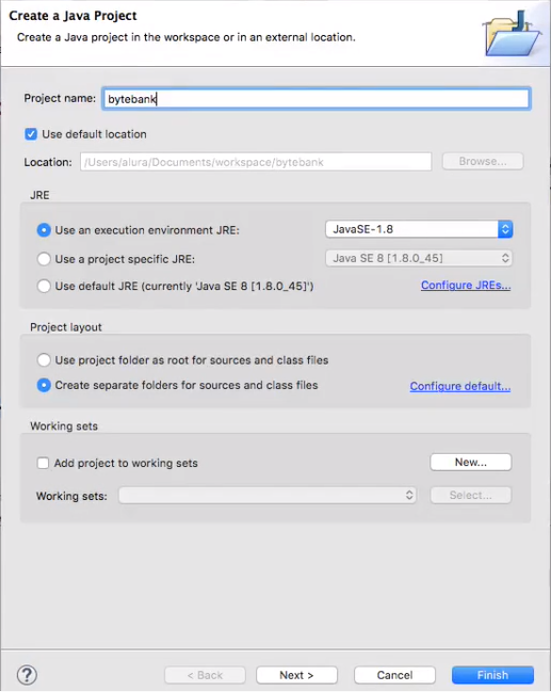
* Conhecendo a linguagem Java
* O que é uma classe
* Como criar uma classe
* Como criar um objeto ou instância de uma classe
* O que são atributos
* Como definir e visualizar valores de atributos
* Como funciona a atribuição de uma referência à uma variável

**Conhecendo Java < TODO >**

**Aprender sobre Classes em Java**

**{O que é uma classe}**

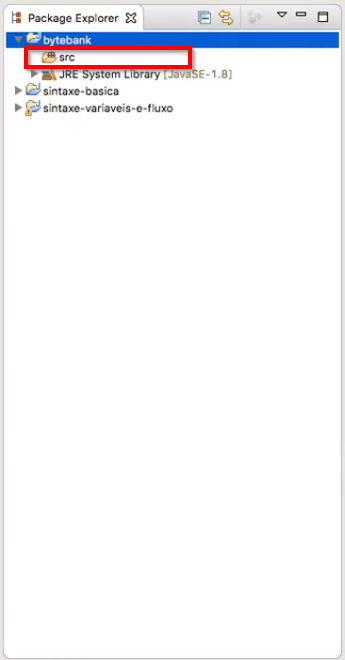
Criaremos um domínio para o nosso sistema baseado em um projeto de banco. No Eclipse, faremos um novo projeto Java intitulado bytebank. Depois, clique em "Finish".



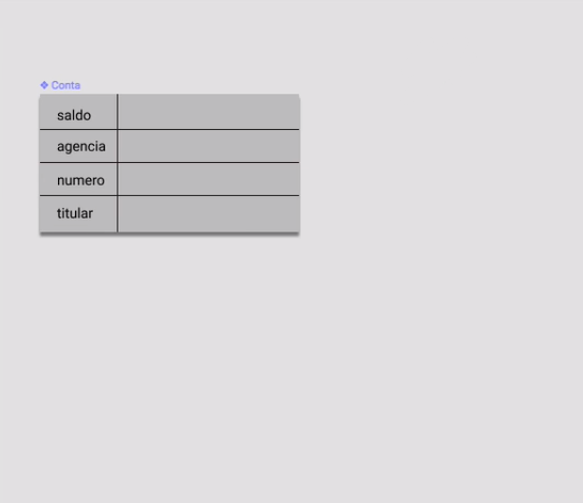
Feito isso, será criado um novo projeto na área do Package Explorer. Ao clicarmos no projeto bytebank, veremos a pasta src.

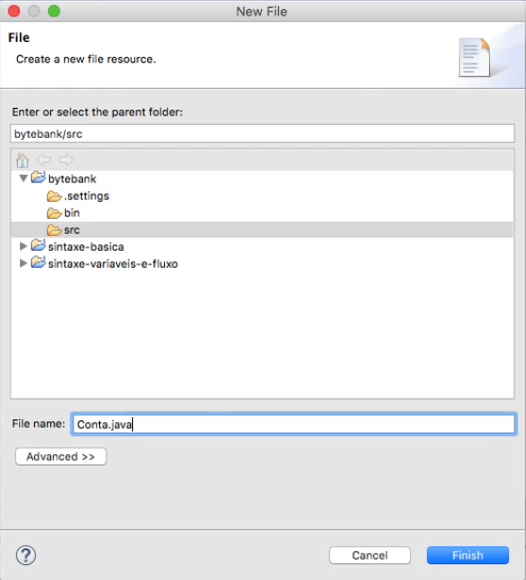
Lembrando: você pode selecionar **"Window > Show View >**Package Explorer **".**

O Package Explorer 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.

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 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

titular

class Conta {

double saldo;

int agencia;

int numero;

String titular;

}

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, {}.

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, a Conta.class.

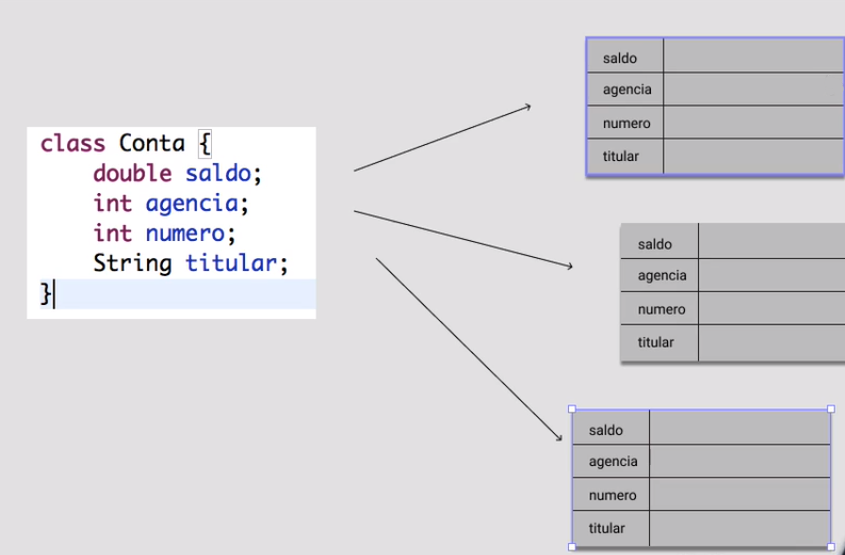
Porém, não podemos iniciar a execução do programa. Para que um programa possa ser iniciado em Java, ele precisa ter o ponto de entrada **public static void main(String[]) args)**, a classe que criamos não possui esse ponto de entrada, pois trata-se de um arquivo de suporte.

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.

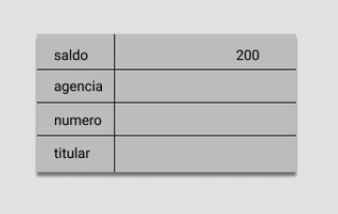
Temos nossa primeira classe Conta, que possui quatro características: saldo, agencia, numero e titular. O nome que damos para tais características é atributos.

Diremos que as contas do ByteBank possuem quatro atributos.

Ainda não temos uma conta bancária, pois não podemos fazer operações básicas que envolvem uma conta. O que possuímos é 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.

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. Voltando ao código, adicionaremos o public, pois frequentemente o Eclipse irá inseri-lo automaticamente.

public class Conta {

double saldo;

int agencia;

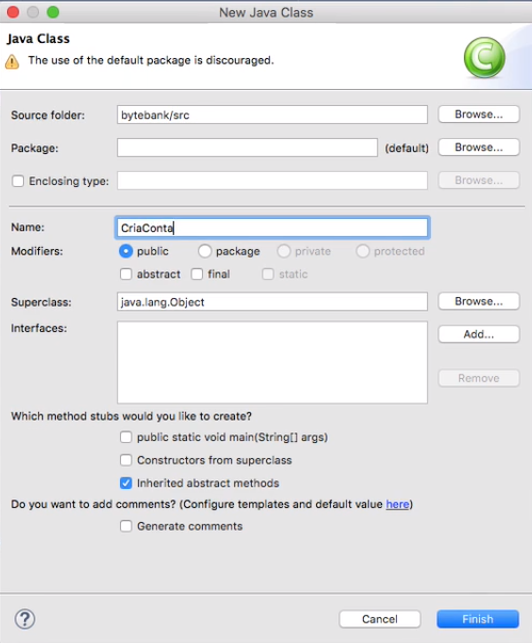
int conta;

String titular;

}

**Criando uma Classe de Teste**

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 ().

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 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, que será**  será do tipo Conta.

public class CriaConta {

public static void main(String[] args) {

**Conta primeiraConta = new Conta();**

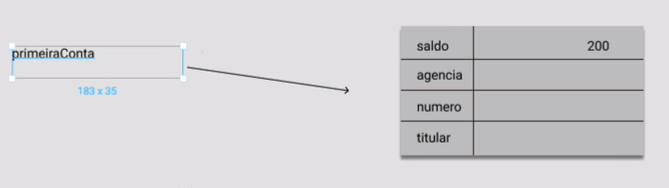
saldo = 200;

}

}

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 “ponto”.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

**primeiraConta.saldo = 200;**

}

}

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 “ponto”.

public class CriaConta {

public static void main(String[] args) {

Conta primeiraConta = new Conta();

primeiraConta.saldo = 200;

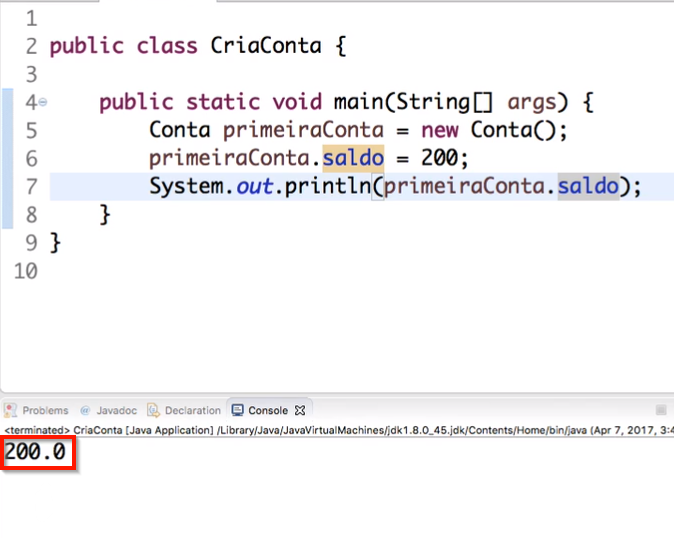
System**.**out**.**println**(primeiraConta.saldo);**

}

}

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.



**Algumas Operações em Java**

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);

}

}

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);

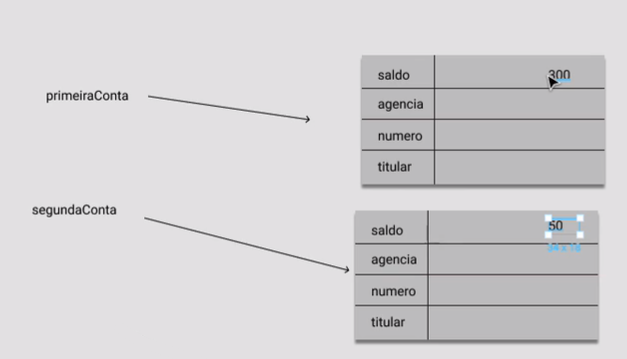
Conta segundaConta = new Conta();

segundaConta.saldo = 50;

}

}

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;

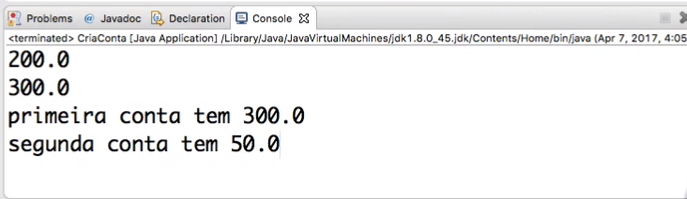
System.out.println(primeiraConta.saldo);

}

}

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.

Ao executarmos a aplicação, veremos os 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.

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

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;

}

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();

}

}

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);

}

}

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);

}

}

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);

}

}

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

System.out.println(primeiraConta);

**Revisão**

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

1) Escreva o método deposita, que recebe um valor do tipo double. Lembre-se que quando nosso método não retorna nenhum valor, devemos colocar a palavra especial void, ficando assim:

public void deposita(double valor){

}

2) Complete seu método deposita, fazendo com que some o valor recebido ao saldo da conta, não se esqueça de utilizar o this

//dentro da class Conta

public void deposita(double valor){

this.saldo += valor;

}

3) Crie agora o método saca, que também recebe um valor do tipo double, nesse caso, nosso método deve retornar um boolean, portanto não se esqueça do return.

public boolean saca(double valor){

if(this.saldo >= valor){

this.saldo -= valor;

return true;

} else {

return false;

}

}

4) Escreva agora o método transfere, ele deve também retornar um booleano mas agora ele receberá uma Conta que será o destino, e o valor a ser transferido.

public boolean transfere(double valor, Conta destino){

if(this.saldo >= valor){

this.saldo -= valor;

destino.deposita(valor);

return true;

}

return false;

}

5) Não se esqueça de testar os seus métodos, para isso, crie uma classe (utilizamos o nome TestaMetodos) e escreva o main, veja se está tudo dando certo!

6) Fica como desafio alterar o método transfere para reutilizar nosso código implementado em saca para simplificar o código.

Até o momento, nosso código da classe Conta está assim:

public class Conta {

double saldo;

int agencia;

int numero;

String titular;

public void deposita(double valor) {

this.saldo = this.saldo + valor;

}

public boolean saca(double valor) {

if(this.saldo >= valor) {

this.saldo -= valor;

return true;

} else {

return false;

}

}

public boolean transfere(double valor, Conta destino) {

if(this.saldo >= valor) {

this.saldo -= valor;

destino.deposita(valor);

return true;

}

return false;

}

}

Próxima Aula

Comportamento e Métodos

Nessa aula falamos sobre o comportamento que são os métodos.

Vimos:

* como definir métodos com parâmetros e retorno
* como retornar algo usando a palavra chave return
* como usar a referência this para acessar um atributo
* que podemos passar uma referência como parâmetro do método
* métodos são chamadas a partir da referência usando o operador .