01

Nosso primeiro método

Transcrição

Estamos prontos para evoluir o nosso modelo de classes, bem como os objetos que estamos construindo a partir delas.

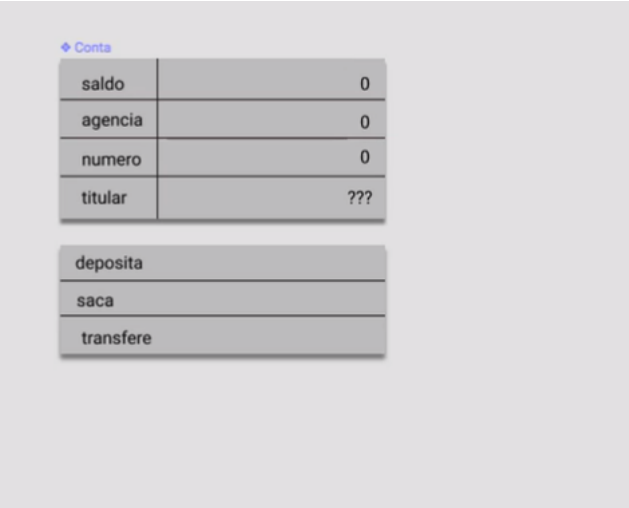
Em nossa classe `Conta` nós temos quatro atributos.

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

Sabemos que no momento em que acionamos a palavra-chave `new` , objetos são criados e os atributos referentes a esses objetos são zerados, inclusive os de tipo `String` .

Além de modificarmos os atributos de um determinado objeto, criaremos algumas funcionalidades. No caso de uma conta bancária é estranho que simplesmente surja `300` reais de saldo do nada. O interessante é que haja uma funcionalidade de *depósito* que inclua valores ao atributo `saldo` . Portanto, criaremos certos comportamentos para o objeto "conta bancária".

Podemos ordenar para a nossa conta funções como *sacar*, *depositar* e *transferir*.



Nossa conta, então, possuirá quatro atributos e três funções. Quando formos transferir essas informações para o Java, chamaremos essas funções de **métodos**, ou seja, uma maneira de fazer algo. Veremos posteriormente quais são as diferenças entre o termo "função" utilizado em outras linguagens e "método" no Java.

Escreveremos o método `deposita()` em nossa classe `Conta` . Atenção para a sintaxe utilizada: no parênteses `()` adicionaremos o que está sendo recebido pelo método, ou seja, um **parâmetro**. No caso de uma conta bancária, precisamos adicionar um `valor` a ser depositado. Em Java não se pode declarar uma variável sem especificar seu tipo, portanto, especificaremos a variável `valor` como sendo do tipo `double` .

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

    deposita(double valor)
}
```

Depois que depositamos um valor em uma determinada conta bancária, poderemos receber uma mensagem, um número, uma espécie de comprovante ou algo do gênero. No caso do nosso projeto **ByteBank**, não há qualquer tipo de retorno à ação de depósito. Quando não existe qualquer tipo de retorno ao acionarmos um método, utilizamos a palavra-chave `void` . Feito isso, fecharemos o bloco utilizando as chaves `{}`

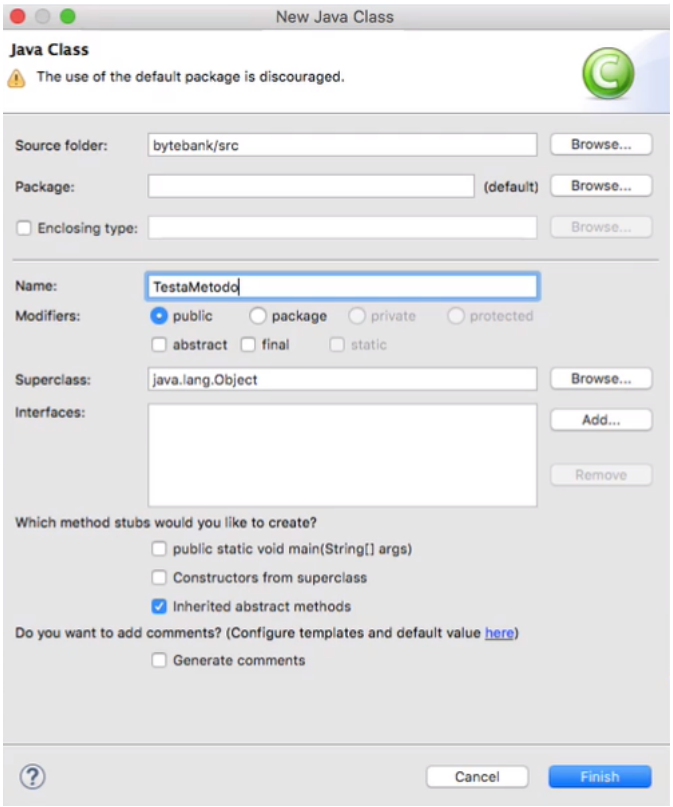
```
public class Conta {
    double saldo;
    int agencia;
    int numero;
    String titular;

    void deposita(double valor) {

    }
}
```

O método `deposita()` não produz nenhum resultado por enquanto, mas sua sintaxe é válida.

Agora que já conhecemos a sintaxe, aprenderemos como acionar e utilizar os métodos em Java. Criaremos uma nova classe intitulada `TestaMetodo`



Para invocarmos o método `deposita()`, é necessário nos referenciar à uma conta específica, neste caso, usaremos uma variável chamada de `contaDoPaulo`.

> Lembrem-se: é comum o nome de uma variável ser igual ao da classe, sendo que a variável por convenção é escrita com letra minúscula.

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

    }
}
```

O `saldo` de `contaDoPaulo` terá um valor de `100`. Para invocarmos o método `deposita()` utilizaremos o caractere ponto `.` seguindo dos parênteses que contém o valor que queremos depositar, que no caso será `50`.

```
public class TestaMetodo {
  public static void main(String[] args) {
    Conta contaDoPaulo = new Conta();
    contaDoPaulo.saldo = 100;
    contaDoPaulo.deposita(50);

  }
}
```

Nosso método é válido, e mesmo não gerando nenhum resultado, poderá ser executado sem nenhum erro. Podemos realizar um teste adicionando o `Sysout` a nossa classe `TestaMetodo`:

```
public class TestaMetodo {
  public static void main(String[] args) {
    Conta contaDoPaulo = new Conta();
    contaDoPaulo.saldo = 100;
    contaDoPaulo.deposita(50);
    System.out.println(contaDoPaulo.saldo);
  }
}
```

Ao executarmos a aplicação veremos que o resultado impresso será `100`, ou seja, não foi depositado nenhum novo valor à `contaDoPaulo`. Nada ocorreu, pois ao evocarmos o método `deposita` estamos nos referindo ao código escrito na classe `Conta`, e este código não executa nada, como já sabemos.

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

    void deposita(double valor) {
```

```
    }
}
```

Queremos que o método `deposita()` adicione valores à uma determinada conta. Existem muitas maneiras de escrevermos esse código, mas faremos da seguinte forma:

> Adicionamos o `public` ao método `deposita()`, não se atente para isso neste momento do curso, discutiremos essa questão posteriormente

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

    public void deposita(double valor) {
        saldo  = saldo + valor;
    }
}
```

Reparem que há uma diferença de cores entre `saldo` e `valor`, pois o primeiro é um atributo do objeto `Conta`, enquanto o segundo é uma variável.

Feitas estas alterações, retornaremos à classe `TestaMetodo` e executaremos a aplicação.

```
public class TestaMetodo {
  public static void main(String[] args) {
    Conta contaDoPaulo = new Conta();
    contaDoPaulo.saldo = 100;
    contaDoPaulo.deposita(50);
    System.out.println(contaDoPaulo.saldo);
  }
}
```

Teremos, dessa vez, um valor impresso de `150`. O método `deposita()` alterou o o valor do atributo `saldo`, que é uma característica de conta.

Para deixarmos mais clara essa noção, usaremos outra palavra-chave do Java que é opcional neste caso específico, o `this`. Ao observarmos classe `Conta`, veremos que no bloco do método `deposita()` há uma referência de `saldo`, mas qual saldo? De que conta estamos falando?

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

    public void deposita(double valor) {
        saldo  = saldo + valor;
```

```
    }
}
```

Podemos escrever nosso código da seguinte forma:

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

    public void deposita(double valor) {
        contaDoPaulo = saldo + valor;
    }
}
```

Não conseguiremos compilar nosso código desta forma, pois `contaDoPaulo` não é uma variável presente neste escopo.Queremos que `saldo` seja relacionado à conta que está evocando o método `deposita()`, para isso, faremos uso da palavra-chave `this`.

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

    public void deposita(double valor) {
        this.saldo = this.saldo + valor;

    }
}
```

Como o método está sendo invocado pela `contaDoPaulo`, o `saldo` é referente a esta conta. Não incluímos a palavra-chave `this` junto à variável `valor`, pois ela **não** é um atributo de um objeto.

Criamos, assim, nosso primeiro método. Escrevemos `deposita()`, parâmetros e qual será a devolução gerada pelo método.