05

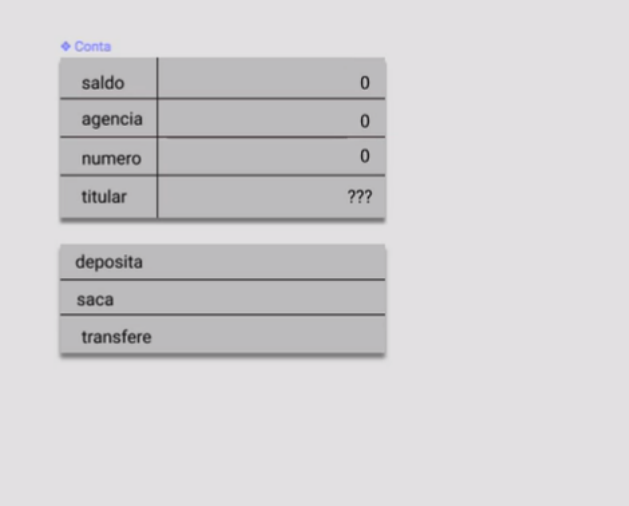
Métodos com retorno

Transcrição

Trabalharemos em um método mais elaborado. O `deposita()` que criamos, não devolvia nenhum tipo de informação para quem o invocou.

```
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 observarmos o diagrama, notamos que existe um método chamando `saca()`, que tem por função *retirar dinheiro de uma conta específica*.



Criaremos o novo método `saca()` na classe `Conta`. Para realizarmos um saque, é preciso atribuir um `valor`, - uma variável do tipo `double` - ao saque.

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

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

    saca(double valor);
}
```

Não precisamos ter a informação de qual conta iremos sacar neste ponto, e essa é uma diferença da forma de programar orientada a métodos e a orientada a funções. Em métodos sempre há um "sujeito" do código à esquerda (no exemplo, `contaDoPaulo`), de forma que sabemos o direcionamento de determinado comando.

```
contaDoPaulo.saldo = 100
```

Iremos fazer com o que o método `saca()` nos retorne um `boolean` : `true` caso o saque seja efetivado, `false` caso não.

Para isso, escreveremos o `public` e `boolean` na linha do método e fechamos as chaves `{}` . Reparem que o método `deposita()` está em um bloco e método `saca()` em outro, não existe método dentro de método.

```
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) {

    }
}
```

Da forma como escrevemos o código ele não será compilado, pois acionamos um `boolean` sem escrever qual seria o retorno dado pelo método. Para prosseguirmos com o nosso código, acionaremos a palavra-chave `if` . Usaremos , também, o `this` , que aciona a referência para a conta que está acionando o método. Se o `saldo` de `this` for `>=` ao `valor` de saque, o novo saldo será o `saldo` de `this - valor` de saque. O método, então, retornará o valor `true` . Caso o contrário, (`else`) será retornado o valor `false` .

```
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 = this.saldo - valor;
            return true;
        } else {
            return false;
        }
    }
}
```

Iremos na classe `TestaMetodo` para analisarmos se o nosso código está funcional. Iremos sacar `20` reais de `contaDoPaulo`:

```
public class TestaMetodo {
    public static void main (String[] args) {
        Conta contaDoPaulo = new Conta();
        contaDoPaulo.saldo = 100;
        contaDoPaulo.deposita(50);
        System.out.println(contaDoPaulo.saldo);
        contaDoPaulo.saca(20);
        System.out.println(contaDoPaulo.saldo);
    }
}
```

Ao rodarmos a aplicação, veremos que o resultado será `130`. Ou seja, o nosso comando de saque foi efetivo. A respeito do `true` e `false` do `boolean`, quando passamos o mouse sobre o código vemos o retorno do método, podemos ou não utilizar esse valor.

Podemos guardar esse valor dentro de uma variável que chamaremos de `conseguiuRetirar`, e acionaremos o `sysout`.

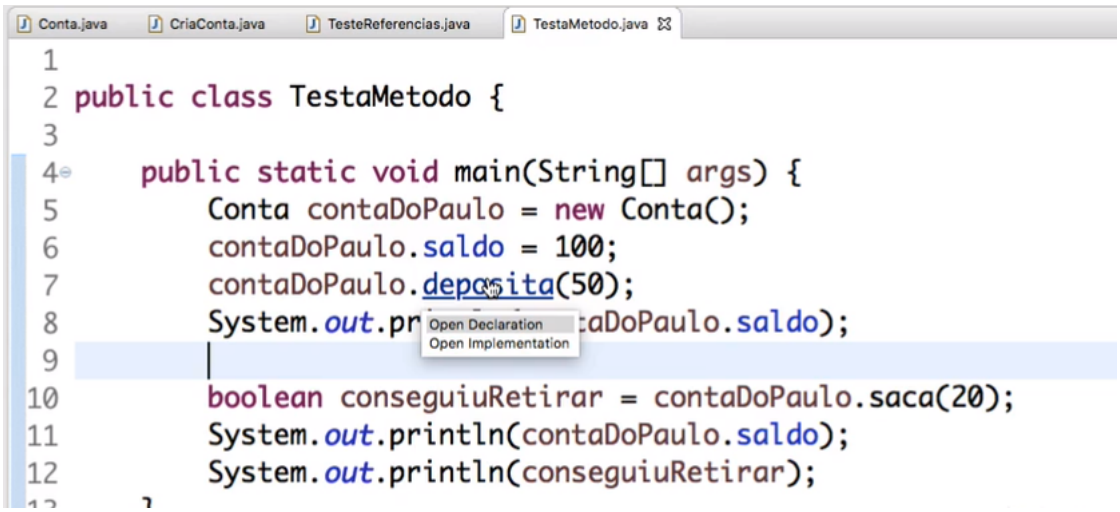
```
public class TestaMetodo {
    public static void main (String[] args) {
        Conta contaDoPaulo = new Conta();
        contaDoPaulo.saldo = 100;
        contaDoPaulo.deposita(50);
        System.out.println(contaDoPaulo.saldo);

        boolean conseguiuRetirar = contaDoPaulo.saca(20);
        System.out.println(contaDoPaulo.saldo);
        System.out.println(conseguiuRetirar);
    }
}
```

Ao executarmos o programa, veremos que o resultado será `true`.

Abordaremos algumas melhorias de navegação que podemos executar no Eclipse. Normalmente, deixamos muitas abas abertas no editor, uma forma mais rápida e fácil de transitar no projeto que está sendo desenvolvido é manter

pressionado atalho "Ctrl", com isso, o Eclipse irá transformar muitos elementos em links.

Ao clicarmos em um dos links, seremos transportados para o código referente ao método ou classe que estamos acionando.



Outra melhoria que podemos fazer quanto a sua formatação, é alterar a linha `this.saldo = this.saldo + valor` utilizando `+=` , e na linha `this.saldo = this.saldo - valor` fazer uso de `-=` .

Essas alterações deixarão o código mais enxuto, e além de ser mais comum na comunidade de programadores Java este tipo de formato, o resultado final será o mesmo.

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

public boolean saca(double valor) {
    if(this.saldo >= valor) {
        this.saldo -= valor;
        return true;
    } else {
        return false;
    }
}
```