03

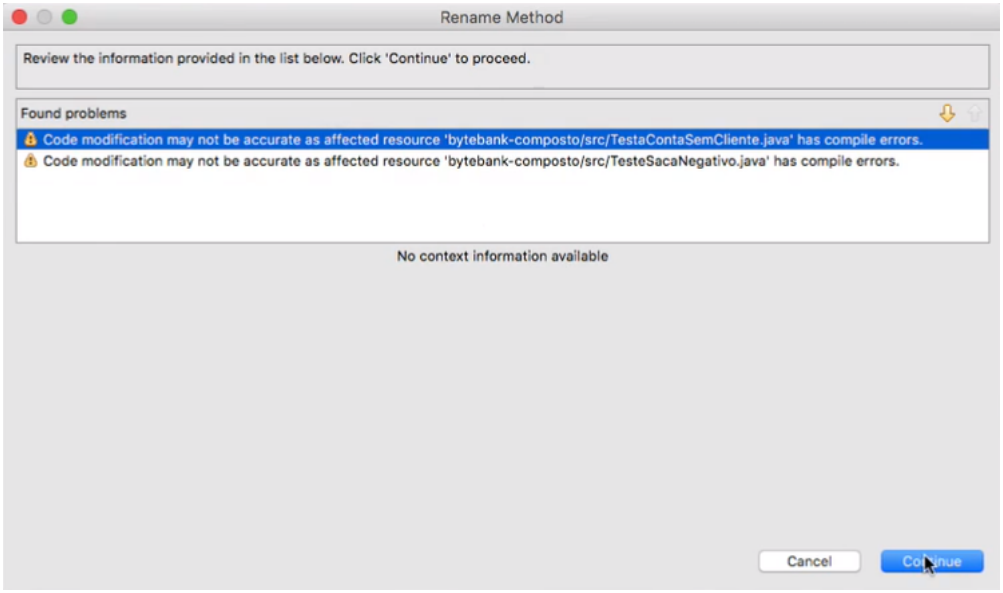
Getters e Setters

Transcrição

O nome do método `pegaSaldo()` que criamos, poderia ter qualquer outro nome. Por uma questão de convenção, alteraremos o nome para `getSaldo()`.

Devido à essa alteração, os outros arquivos não serão compilados, porque estarão com o nome antigo `pegaSaldo()`. Para resolvermos esse problema, na classe `Conta`, selecionaremos o método e daremos um clique duplo sobre ele. Feito isso, escolheremos ao opção "Refactor > Rename".

Surgirá uma caixa de diálogo e pressionamos "Continue". Com isso, o nome do método foi alterado em todos os arquivos.



O nome desse tipo de método que simplesmente *exibe uma informação* é **getter**. Não se trata de um elemento que compõe a sintaxe do Java, **get** não é uma palavra-chave do Java.

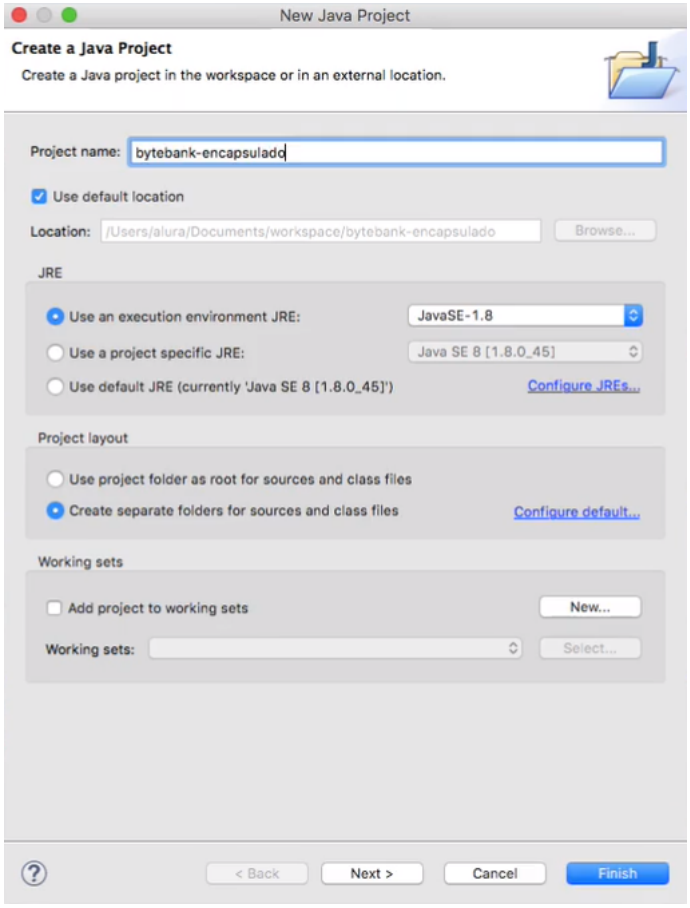
Dizemos que temos um método *getter* para `saldo`, pois este atributo é privado e sentimos a necessidade ao longo do projeto de acessar a informação contida em `saldo`.

Percebam que não há a necessidade de criarmos novo método equivalente - algo como "`setSaldo`" - para *modificar o atributo* `saldo`, essas modificações serão feitas pelos já conhecidos `saca()`, `deposita()` e `transfere()`. Queremos que as alterações de saldo em nossas contas do **ByteBank** sejam sempre feitas através de **saques**, **depósitos** ou **transferências**.

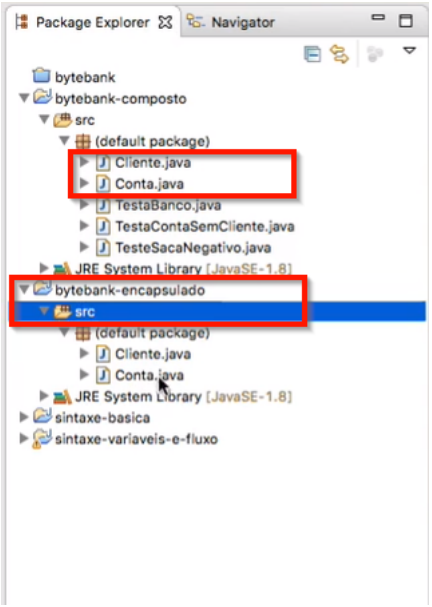
A ideia é diferente para `agencia`, `numero` e `titular`. Não temos métodos para alterar esses atributos, e por enquanto, só conseguimos fazer modificações diretas, o que quer dizer que podemos inserir números negativos como valores, por exemplo.

Com o tempo, entenderemos que o ideal é que todos os atributos sejam privados. Existem casos raros de atributos públicos, mas são realmente exceções.

Criaremos um novo projeto Java que chamaremos de `bytebank-encapsulado`. "Encapsulado" é o termo utilizado para elementos ocultos, escondidos.



Na área *Package Explorer*, copiaremos as classes `Cliente` e `Conta` e as colaremos no `default package` do novo projeto `bytebank-encapsulado`.



Utilizando o atalho "Ctrl + W", fecharemos todas as abas do Eclipse. Também fecharemos o projeto `bytebank-composto` para não confundirmos as classes dos dois projetos, tornado o ambiente de trabalho mais claro e limpo.

Na classe `Conta`, transformaremos todos os atributos em `private`.

```
public class Conta {
    private double saldo;
    private int agencia;
    private int numero;
```

```
    private Cliente titular;

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

O fato de tornarmos um atributo privado facilita a modificação e atualização do código. Com uma classe sendo responsável por seus próprios atributos, a manutenção do sistema se torna localizada, por conseguinte, mais simples.

Para que os atributos sejam acessados fora da classe, utilizaremos os *getters*. Na classe `Conta` criaremos um método público denominado `getNumero()` que devolve um `int` e retorna `numero`.

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

    public void deposita(double valor) {...}

    public boolean saca (double valor) {...}

    public boolean transfere(double valor, Conta destino, Conta origem) {...}

    public double pegaSaldo() {...}

    public int getNumero() {
        return this.numero;
    }
}
```

Além de termos um método que devolve qual é o `numero` de uma conta, queremos também um método que *altere* esse mesmo `numero`. Esse tipo de método é chamado de **set**.

Diferente do **get**, há um parâmetro a ser passado para o método, afinal queremos modificar o número e precisamos informar qual será essa modificação. Usaremos uma variável `novoNumero` e o método não retornará nada.

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

    public void deposita(double valor) {...}


    public boolean saca(double valor) {...}

    public boolean transfere(double valor, Conta destino, Conta origem) {...}

    public double pegaSaldo() {...}
```

```
    public int getNumero() {
        return this.numero;
    }

    public void setNumero(int novoNumero) {
        this.numero = novoNumero;
    }
}
```

Criaremos numa nova classe para utilizar tanto o *getter* como o *setter*. Essa nova classe será chamada de `TestaGetESet`. Na nova classe, criaremos uma nova conta chamada `conta` e a instanciaremos através da palavra-chave `new`. Feito isso, escreveremos o `numero` desta nova `conta` como sendo `1337`.

```
public class TestaGetESet {
    public static void main(String[] args) {
        Conta conta = new Conta();
        conta.numero = 1337
    }
}
```

Da forma como o nosso código está escrito ele não será compilado, afinal, `numero` é um atributo privado e *precisa* ser acessado através de um método. Utilizaremos o método `setNumero()`

```
public class TestaGetESet {
    public static void main(String[] args) {
        Conta conta = new Conta();
        conta.setNumero(1337);
    }
}
```

Uma das vantagens de utilizar os métodos, é que dentro do próprio método `setNumero()` podemos adicionar `if`s, gerando especificações, mensagens de erro, e assim por diante.

Para imprimirmos o valor de `numero`, utilizamos o `sysout` e o método `getNumero()`. Ao executarmos a aplicação, teremos o resultado de `1337`.

```
public class TestaGetESet {
    public static void main(String[] args) {
        Conta conta = new Conta();
        conta.setNumero(1337);
        System.out.println(conta.getNumero());
    }
}
```

Como já foi dito, no caso do atributo `saldo`, não criaremos um método setter, pois trabalharemos com os métodos `deposita()`, `saca()` e `transfere()` para modificar seu valor.

Criaremos métodos **get** e **set** para o atributo `agencia`, mas dessa vez, de uma maneira mais simples.

Antes, voltemos ao método `setNumero()` na classe `Conta`.

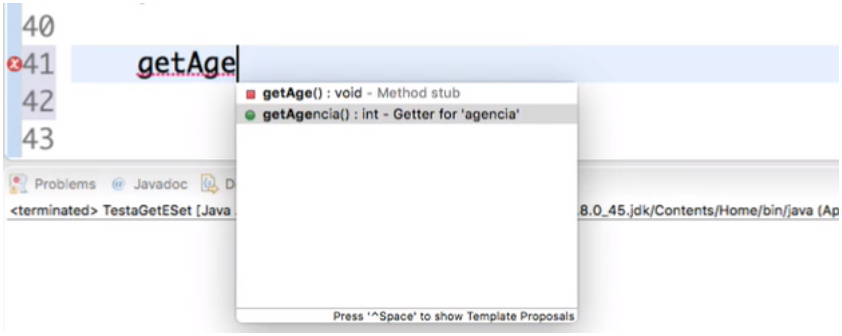
```
public void setNumero(int novoNumero) {
    this.numero = novoNumero;
}
```

Em muitos casos, em vez de escrever o nome da variável `novoNumero`, utiliza-se o mesmo nome do atributo, no caso, `numero`.

A princípio, podemos nos confundir, mas basta nos atentarmos para o uso da palavra-chave `this`, ao lado esquerdo do código, que marca a referência ao atributo.

```
public void setNumero(int numero) {
    this.numero = numero;
}
```

Escreveremos o método get para o atributo `agencia`. Percebam que basta iniciarmos a escrever o código e pressionarmos o atalho "Ctrl + Space", será sugerido a criação automática de `getAgencia()`. Escolheremos esta opção e pressionaremos "Enter".



Feito isso, o código será automaticamente escrito na classe `Conta`.

```
public class Conta {
    // atributos
    // método deposita
    // método saca
    // método transfere
    // método pegaSaldo

    public int getNumero() {
        return this.numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    public int getAgencia() {
        return agencia;
    }
}
```

Adicionaremos o `this` para marcarmos bem o que é atributo e variável temporária.

```
public int getAgencia() {
    return this.agencia;
}
```

Podemos realizar o mesmo procedimento com o método **set** do atributo `agencia`. O Eclipse possui teclas de atalho que facilitam muito a construção do código.

No cabeçalho de ferramentas existe opção de gerar *getters* e *setters*, basta selecionarmos em "Source > Generate Getters and Setters", mas não há necessidade de acessá-los desta forma, já que o "Ctrl + Space" disponibiliza a mesma função, neste caso.