05

Getters e Setters de referência

Transcrição

Elevaremos o nível de dificuldade na prática de métodos **getters** e **setters** considerando o tipo `Cliente`.

Neste ponto, já trabalhamos com os atributos `saldo`, `agencia` e `numero`. Lembrando que para o caso de `saldo`, não precisamos utilizar um setter, porque outros métodos como `deposita()` e `saca()` já servem para modificar o atributo.

É possível que haja atributos em que não exista a necessidade tanto de um método `get` quanto de um `set`, como por exemplo, uma conexão para o banco de dados que só possui um uso interno com relação à própria classe `Conta`.

No caso do atributo `titular`, é de nosso interesse que possamos acessar o titular de uma determinada conta e modificá-lo.

Na classe `TestaGetESet`, criaremos um novo `Cliente` e faremos a atribuição do `titular` cujo valor é `null`, para um novo titular identificado como `paulo`.

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

        conta.setNumero(1337);
        System.out.println(conta.getNumero());

        Cliente paulo = new Cliente();
        conta.titular = paulo;
    }
}
```

O código desta forma não compila, pois `titular` é um atributo privado. Precisaremos invocar algum método para finalizar nossa operação.

Poderíamos acionar um método hipotético, como `setTitular()`. Esse código também não compila, pois ainda não criamos seu **setter**.

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

        conta.setNumero(1337)
        System.out.println(conta.getNumero());

        Cliente paulo = new Cliente();
        conta.setTitular(paulo);
    }
}
```

Na classe `Conta`, criamos no novo método `setTitular()`. Feito isso, criaremos, também, o método `getTitular()`.

```
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 this.agencia;
    }

    public void setAgencia(int agencia){
        this.agencia = agencia;
    }

    public void setTitular(Cliente titular) {
        this.titular = titular;

    }

    public Cliente getTitular() {
        return titular;
    }
}
```

Feito isso, veremos que o nosso código na classe `TestaGetESet` já está sendo compilado e podemos fazer alterações no atributo `titular`. Mudaremos o nome do titular para `paulo silveira` e queremos incluir `cpf` e `profissao`.

Poderíamos modificar o atributo `nome` de forma direta:

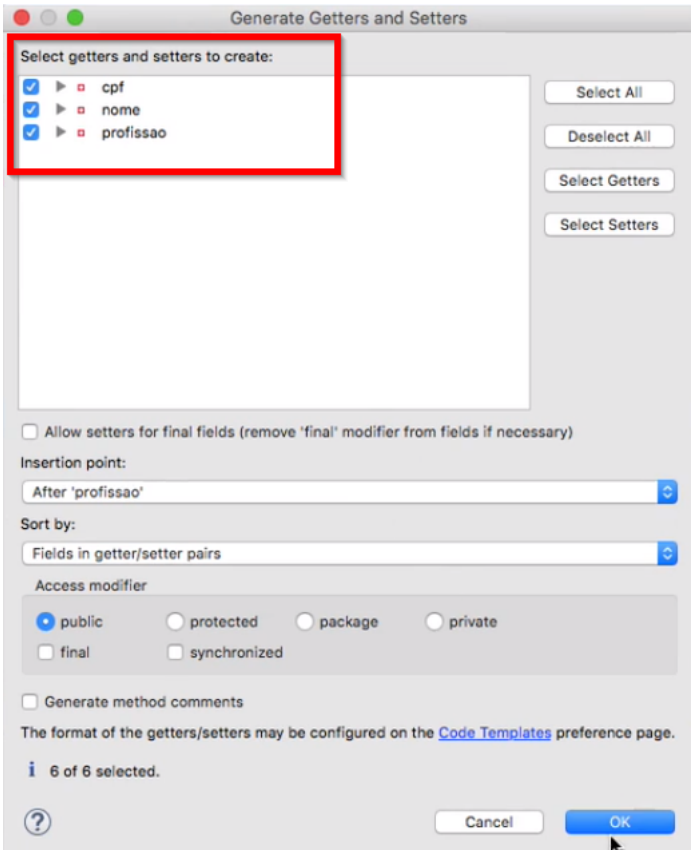
```
public class TestaGetESet {
    public static void main(String[] args) {
        Conta conta = new Conta();

        conta.setNumero(1337)
        System.out.println(conta.getNumero());

        Cliente paulo = new Cliente();
        paulo.nome = "paulo silveira";

        conta.setTitular(paulo);
    }
}
```

Essa modificação direta foi possível, pois na classe `Cliente` todos os atributos são não-privados, diferentemente da classe `Conta`.

Vamos transformar os atributos da classe `Cliente` em *privados*, pois queremos colocar padrões específicos em cada atributo, como não permitir números em `nome`, por exemplo.

```
public class Cliente {
    private String nome;
    private String cpf;
    private String profissao;
}
```

Para que você não precise ficar escrevendo todos os getters e setters nececessários, no cabeçalho do Eclipse acione a opção "Source > Generate Getters and Setters..." Em "Select getters and setters to create" selecionaremos todas as opções de atributos.



Com isso, todos os métodos foram automaticamente gerados.

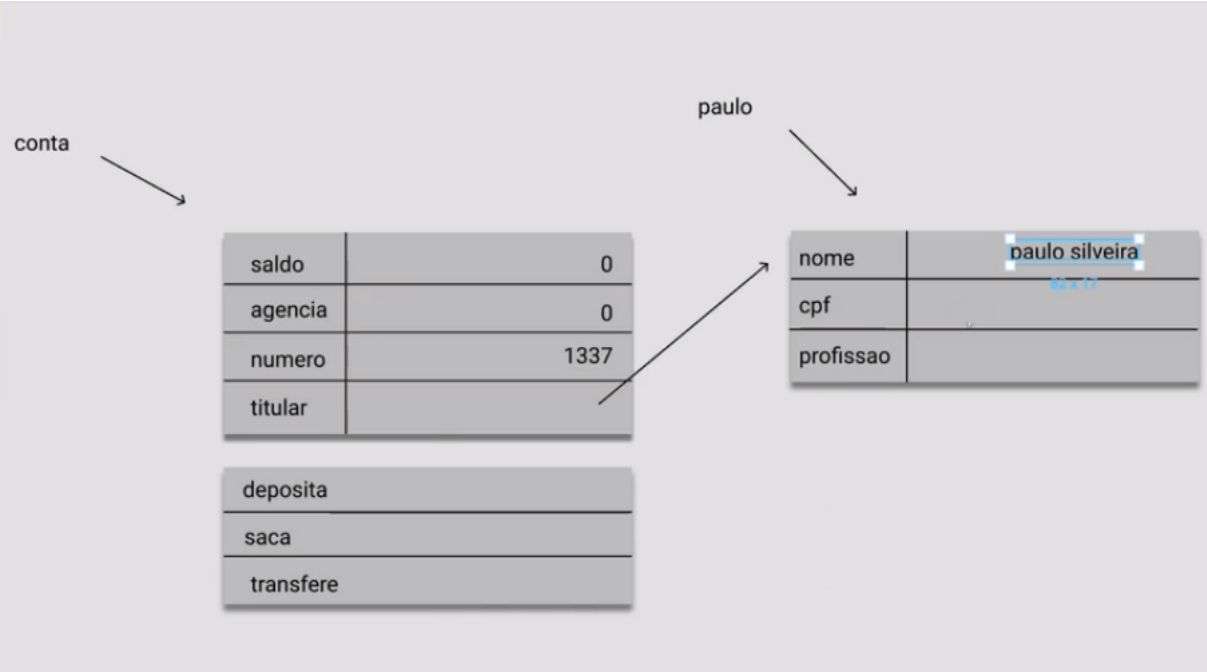
```
public class Cliente {
    private String nome;
    private String cpf;
    private String profissao;

    public String getNome() {
        return nome;
    }

    public String getCpf() {
        return cpf;
    }
```

```
    public String getProfissao() {
        return profissao;
    }

    public void setProfissao(String profissao) {
        this.profissao = profissao
    }
}
```

Com isso, na classe `TestaGetESet`, não podemos mais alterar diretamente os atributos de `Cliente`, precisamos invocar métodos.

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

        conta.setNumero(1337)
        System.out.println(conta.getNumero());

        Cliente paulo = new Cliente();
        paulo.setNome("paulo silveira");

        conta.setTitular(paulo);
    }
}
```

Nosso código no diagrama está representado da seguinte maneira: temos uma variável `conta` que faz referência a um objeto `Conta`. O atributo `titular` da conta bancária para o atributo `nome` do objeto `Cliente`, referenciado pela variável `paulo`.



Tentamos imprimir apenas o `titular da conta` para averiguarmos como o código se comporta.

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

        conta.setNumero(1337)
        System.out.println(conta.getNumero());

        Cliente paulo = new Cliente();
        paulo.setNome("paulo silveira");

        conta.setTitular(paulo);

        System.out.println(conta.getTitular());
    }
}
```

Ao executarmos a aplicação, veremos que o foi impresso o valor de `Cliente@15db9742` . Não era esse o resultado que estávamos querendo, pois esse é o valor da referência ao objeto, e nós queremos imprimir o atributo `nome` do objeto.

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

        conta.setNumero(1337)
        System.out.println(conta.getNumero());

        Cliente paulo = new Cliente();
        paulo.setNome("paulo silveira");

        conta.setTitular(paulo);

        System.out.println(conta.getTitular());
    }
}
```

Reconfiguraremos a linha, utilizando os métodos getters para chegarmos à informação do atributo `nome` do objeto.

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

        Cliente paulo = new Cliente();
        paulo.setNome("paulo silveira");

        conta.setTitular(paulo);

        System.out.println(conta.getTitular().getNome());
    }
}
```

Ao rodarmos o programa, veremos que o resultado impresso é `paulo silveira` , como o esperado.

Caso seja do nosso interesse alterar alguma informação do titular, o percurso é parecido. Iremos incluir a profissão programador: acionaremos `getTitular()` e depois o `setProfissao()`.

```
System.out.println(conta.getTitular().getNome());
conta.getTitular().setProfissao("programador");
```

Podemos quebrar o código em duas linhas.

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

        conta.setNumero(1337)
        System.out.println(conta.getNumero());

        Cliente paulo = new Cliente();
        paulo.setNome("paulo silveira");

        conta.setTitular(paulo);

        System.out.println(conta.getTitular().getNome());

        conta.getTitular().setProfissao("programador");
        //agora em duas linhas:
        Cliente titularDaConta = conta.getTitular();
        titularDaConta.setProfissao("programador");
    }
}
```

O resultado é mesmo, a única diferença é que criamos uma variável temporária. Podemos quebrar a linha do código para facilitar a leitura nos momentos em que ela estiver muito grande.

A variável `titularDaConta` possui o mesmo endereço da variável `paulo`. como podemos verificar, realizando o `sysout`.

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

    conta.setNumero(1337)
    System.out.println(conta.getNumero());

    Cliente paulo = new Cliente();
    paulo.setNome("paulo silveira");

    conta.setTitular(paulo);

    System.out.println(conta.getTitular().getNome());

        conta.getTitular().setProfissao("programador");
        //agora em duas linhas:
        Cliente titularDaConta = conta.getTitular();
        titularDaConta.setProfissao("programador");
```

```
        System.out.println(titularDaConta);
        System.out.println(paulo);
        System.out.println(conta.getTitular());


    }
}
```

Ao executarmos a aplicação, veremos que todas as referências são para o mesmo endereço de memória `Cliente@15db9742` .