04

Static

Transcrição

Mais um desafio: precisaremos saber quantas contas foram abertas no sistema. Na linguagem Java, saberemos quantas contas foram instanciadas.

Na classe `TestaValores`, criaremos uma variável chamada `total`, que começa com o valor `0`.

> Lembre-se que variável local precisa ser zerada.

Feito isso, escreveremos que a cada vez que surgir uma nova conta através do `new`, isso seja contabilizado na variável `total++`.

```
public class TestaValores {
    public static void main(String[] args) {
        int total = 0;
        Conta conta = new Conta(1337, 24226);
        total++;

        System.out.println(conta.getAgencia());

        conta.setAgencia(1232123);
    }
}
```

Essa saída é funcional, mas apresenta problemas de ordem prática, por exemplo, toda a vez que uma nova conta for aberta, o desenvolvedor deve escrever a variável `total++`.

Uma forma melhor de contabilizar as contas que foram criadas no nosso banco é acionar o **construtor**. Na classe `Conta`, adicionaremos no construtor a requisição da variável `total++`.

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

    public Conta(int agencia, int numero) {
        total ++;
        this.agencia = agencia;
        this.numero = numero;
        System.out.println("estou criando uma conta " + this.numero);
    }
```

Para que o nosso código compile, iremos declarar `total` como um atributo do tipo `int` privado. Feito isso, iremos imprimir o total de contas adicionando o texto `o total de contas é` mais a variável.

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

    public Conta(int agencia, int numero) {
        total ++;
        System.out.println("o total de contas é " + total);
        this.agencia = agencia;
        this.numero = numero;
        System.out.println("estou criando uma conta " + this.numero);
    }
```

A variável `total++` que escrevemos na classe `TestaValores` não tem qualquer ligação com o atributo `total` que criamos. Portanto, podemos excluí-la, já que descobrimos uma forma mais prática de contabilizar o total de contas.

```
public class TestaValores {
    public static void main(String[] args) {

        Conta conta = new Conta(1337, 24226);
        System.out.println(conta.getAgencia());
        conta.setAgencia(1232123);
    }
}
```

Ao executarmos nosso programa, veremos que o valor impresso será `o total de contas é 1`. O resultado, portanto, está correto. Adicionaremos mais duas novas contas para testarmos se o programa está operando corretamente.

```
public class TestaValores {
    public static void main(String[] args) {
        Conta conta = new Conta(1337, 24226);
        System.out.println(conta.getAgencia());
        conta.setAgencia(1232123);

        Conta conta2 = new Conta(1337, 16549);
        Conta conta3 = new Conta(2112, 14660);
    }
}
```

Veremos que o resultado do programa é `o total de contas é 1`. Ou seja, não houve a atualização quanto ao número de contas. Percebemos que houve um erro na criação do atributo `total` na classe `Conta`. É como se cada objeto conta tivesse um `saldo`, `agencia`, `numero`, `titular` e `total`.

O que queremos é que `total` fosse uma variável que não ficasse em cada instância, mas em algum lugar da classe `Conta`, algo como um atributo compartilhado e não de um objeto especificamente.

Para isso, existe a palavra-chave `static`. O `static` faz com que o atributo seja da classe, e não mais do objeto. Com isso, todo o objeto conta possui acesso a um único `total`.

Ao lado esquerdo da variável `total++` podemos adicionar a classe `Conta`. Afinal, não estamos mais fazendo referência à uma conta - como o `this` fazia - e sim à classe `Conta`.

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

    public Conta(int agencia, int numero) {
        Conta.total ++;
        System.out.println("o total de contas é " + Conta.total);
        this.agencia = agencia;
        this.numero = numero;
        System.out.println("estou criando uma conta " + this.numero);
    }
```

Ao executarmos a aplicação, veremos que o valor impresso será `o total de contas é 3`.

Caso retirássemos os `sysout` da classe `Conta`, o programa não imprimiria mais o total de contas.

```
    public Conta(int agencia,int numero) {
        Conta.total ++;
        //System.out.println("o total de contas é " + Conta.total);
        this.agencia = agencia;
        this.numero = numero;
        //System.out.println("estou criando uma conta " + this.numero);
    }
```

Tendo isso em vista, poderíamos imprimir o total de contas na classe `TestaValor` fazendo um `sysout` no atributo `total`. Isso funcionaria se o atributo não fosse privado. A questão é que não há mais necessidade de marcar `conta2` ou `conta3`, pois o `total` é compartilhado entre as instâncias.

```
public class TestaValores {
    public static void main(String[] args) {
        Conta conta = new Conta(1337, 24226);
        System.out.println(conta.getAgencia());
        conta.setAgencia(1232123);

        Conta conta2 = new Conta(1337, 16549);
        Conta conta3 = new Conta(2112, 14660);

        System.out.println(Conta.total);
    }
}
```

Como já vimos, é interessante que este atributo seja privado para preservarmos características essenciais segundo a regra de negócios que rege o banco. Portanto, deveremos criar um getter para o atributo `total` na classe `Conta`.

```
public int getTotal() {
    return Conta.Total;
}
```

Feito isso, acionaremos o getter na classe `TestaValores` usando o `System.out`. Não precisamos especificar `conta1` ou `conta2`, porque estamos nos referindo à classe `Conta`.

```
public class TestaValores {
    public static void main(String[] args) {
        Conta conta = new Conta(1337, 24226);
        System.out.println(conta.getAgencia());
        conta.setAgencia(1232123);

        Conta conta2 = new Conta(1337, 16549);
        Conta conta3 = new Conta(2112, 14660);

        System.out.println(Conta.getTotal());
    }
}
```

Veremos que o código não está sendo compilado, vamos entender o porquê: todos os métodos declarados eram da instância `conta`, ou seja, de uma conta específica. Neste caso, estamos nos comunicando com um atributo `total`. Para isso, precisaremos declarar que o método é `static`.

```
public static int getTotal() {
    return Conta.Total;
}
```

O método `static` possui usos muito interessantes, mas possui suas limitações. Embora não seja do interesse do nosso projeto, não poderíamos acessar um atributo de instância via `this`, como `saldo`.

```
public static int getTotal() {
    System.out.println(this.saldo);
    return Conta.total;
```

Os métodos estáticos acessam apenas atributos estáticos.