Atributos e Métodos estáticos

Orientação a Objetos – DCC025

Gleiph Ghiotto Lima de Menezes
gleiph.ghiotto@ufjf.br

Departamento de Ciência da Computação
UFJF

Conteúdo da Aula

- Atributos estáticos
- Métodos estáticos

Exemplo conta bancária

```java
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;

    public void sacar(int valor){
        if(this.saldo >= valor)
            this.saldo -= valor;
    }

    public void depositar(int valor){
        this.saldo += valor;
    }

}
```

Quantidade de contas

- Suponhamos agora que o banco quer controlar a quantidade de contas bancárias existentes no sistema. Qual a forma mais fácil de fazer isso?

Quantidade de contas

```java
package br.ufjf.dcc.oo.desenho.aula;

public class TesteBanco {

    public static void main(String[] args) {
        int totalContas = 0;

        ContaBancaria conta1 = new ContaBancaria();
        totalContas++;

        ContaBancaria conta2 = new ContaBancaria();
        totalContas++;

        System.out.println("Total de contas: " + totalContas);
    }
}
```

Quantidade de contas

- E se esquecermos de incrementar a variável totalContas alguma vez que um objeto da classe ContaBancaria for instanciado?

Quantidade de contas

- Uma outra alternativa para contabilizarmos a quantidade de contas

Novo atributo criado na classe

Somando o número de contas

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    public int totalContas = 0;

    public ContaBancaria() {
        this.totalContas++;
    }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Quantidade de contas

- Desta forma utiliza-se a própria classe para contar as contas

Novo atributo criado na classe

Somando o número de contas

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    public int totalContas = 0;

    public ContaBancaria() {
        this.totalContas++;
    }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Quantidade de contas

- Mas e quando tivermos diversos objetos da classe? Qual será o total de contas em cada um deles?

Novo atributo criado na classe

Somando o número de contas

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    public int totalContas = 0;

    public ContaBancaria() {
        this.totalContas++;
    }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Quantidade de contas

- Mas e quando tivermos diversos objetos da classe? Qual será o total de contas em cada um deles?
  - Apenas 1

Novo atributo criado na classe

Somando o número de contas

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    public int totalContas = 0;

    public ContaBancaria() {
        this.totalContas++;
    }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Atributos e Métodos estáticos

- Seria interessante que essa **informação** fosse única, **compartilhada** por **todos os objetos dessa classe**
  - Quando fosse alterado através de um objeto, os outros enxergariam o mesmo valor
  - Para fazer isso em Java, declaramos o **atributo** como **estático** (**static**).

Quantidade de contas

- Para fazer isso em Java, declaramos o atributo como estático (***static***)

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    private static int totalContas;

    public ContaBancaria() {
        this.totalContas++;
    }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Atributos e métodos estáticos

- Atributos e métodos estáticos são compartilhados por todas as instâncias da classe
- Somente **um valor será armazenado em um atributo estático** e caso o valor seja alterado por qualquer instância da classe, a **modificação será refletida em todas as demais instâncias**
- O modificador **static deve ser declarado antes do tipo de dado do atributo** e pode ser combinado com modificadores como *public, protected e private*

Atributos e métodos estáticos

- Atributos e métodos estáticos também são conhecidos como **atributos ou métodos de classe**
- Atributos e métodos não estáticos são conhecidos como **atributos ou métodos de instância**

Quantidade de contas

- Com atributo estático, o método main() poderia ter as implementações?

```
package br.ufjf.dcc.oo.desenho.aula;

public class TesteBanco {

    public static void main(String[] args) {

        ContaBancaria conta1 = new ContaBancaria();
        ContaBancaria conta2 = new ContaBancaria();

        System.out.println("Total de contas: " + conta1.totalContas);
    }
}
```

Quantidade de contas

- Com atributo estático, o método main() poderia ter as implementações?

```
package br.ufjf.dcc.oo.desenho.aula;

public class TesteBanco {

    public static void main(String[] args) {

        ContaBancaria conta1 = new ContaBancaria();
        ContaBancaria conta2 = new ContaBancaria();

        System.out.println("Total de contas: " + conta2.totalContas);
    }
}
```

```
package br.ufjf.dcc.oo.desenho.aula;

public class TesteBanco {

    public static void main(String[] args) {

        ContaBancaria conta1 = new ContaBancaria();
        ContaBancaria conta2 = new ContaBancaria();

        System.out.println("Total de contas: " + conta1.totalContas);
    }
}
```

Quantidade de contas

- Com atributo estático, o método main() poderia ter as implementações?

As duas alternativas de implementação ainda apresentam problema porque o atributo totalContas está privado na classe ContaBancaria.

```
package br.ufjf.dcc.oo.desenho.aula;

public class TesteBanco {

    public static void main(String[] args) {

        ContaBancaria conta1 = new ContaBancaria();
        ContaBancaria conta2 = new ContaBancaria();

        System.out.println("Total de contas: " + conta1.totalContas);
    }
}
```

```
package br.ufjf.dcc.oo.desenho.aula;

public class TesteBanco {

    public static void main(String[] args) {

        ContaBancaria conta1 = new ContaBancaria();
        ContaBancaria conta2 = new ContaBancaria();

        System.out.println("Total de contas: " + conta2.totalContas);
    }
}
```

Quantidade de contas

- Poderíamos alterar a classe ContaBancaria como sugerido entre as figuras abaixo. Com isso o método main() poderá ser mantido como uma das duas implementações anteriores

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    private static int totalContas;

    public ContaBancaria() {...3 lines }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    public  static int totalContas;

    public ContaBancaria() {...3 lines }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Atributos e métodos estáticos

- Quando um **atributo** é declarado como **estático** ele não é mais de cada objeto e sim da **classe**
- A informação fica guardada pela classe
- Para invocar atributos ou métodos estáticos de uma classe não é necessário ter objetos da classe instanciados
- Podemos invocar itens estáticos através da própria classe (utilizando seu nome)
- Neste caso, não usaremos mais o termo this quando tivermos de acessar atributos ou métodos estáticos

Quantidade de contas

- Da forma como a classe ContaBancaria está implementada até agora, o usuário poderia alterar o total de contas fora do construtor (como sugerimos até o momento)

```
package br.ufjf.dcc.oo.desenho.aula;

public class TesteBanco {

    public static void main(String[] args) {

        ContaBancaria conta1 = new ContaBancaria();
        ContaBancaria conta2 = new ContaBancaria();

        System.out.println("Total de contas: " + conta2.totalContas);

        ContaBancaria.totalContas++;

        System.out.println("Total de contas: " + conta2.totalContas);
    }
}
```

Quantidade de contas

- Utilizando a ideia de encapsulamento, poderíamos ter a classe implementada na seguinte forma

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    private  static int totalContas;

    public static int getTotalConta(){
        return ContaBancaria.totalContas;
    }

    public ContaBancaria() {...3 lines }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Quantidade de contas

- Desta forma o usuário perderia o acesso direto ao atributo totalContas e seria obrigado a utilizar métodos da classe até mesmo para consultar o valor armazenado no mesmo

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    private  static int totalContas;

    public static int getTotalConta(){
        return ContaBancaria.totalContas;
    }

    public ContaBancaria() {...3 lines }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Quantidade de contas

- E se tirássemos o termo static da assinatura do método getTotalContas()?

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    private  static int totalContas;

    public static int getTotalConta(){
        return ContaBancaria.totalContas;
    }

    public ContaBancaria() {...3 lines }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Quantidade de contas

- E se tirássemos o termo static da assinatura do método getTotalContas()?
  - Seríamos obrigados a ter objetos instanciados da classe para invocarmos este método. Modo incorreto para trabalhar com itens estáticos.

```
package br.ufjf.dcc.oo.desenho.aula;

public class ContaBancaria {

    private String agencia;
    private String numero;
    private String cliente;
    private float saldo;
    private  static int totalContas;

    public static int getTotalConta(){
        return ContaBancaria.totalContas;
    }

    public ContaBancaria() {...3 lines }

    public void sacar(int valor){...4 lines }

    public void depositar(int valor){...3 lines }

}
```

Atributos e métodos estáticos

- Métodos e atributos estáticos só podem acessar outros métodos e atributos estáticos na mesma classe
  - Um atributo ou um método estático é chamado através da classe, e não de um objeto

Atributos e métodos estáticos

- Maiores utilidades de atributos estáticos:
  - Manter uma única informação ou estado para todas as instâncias de uma classe que possa ser modificada ou acessada por qualquer das instâncias
  - Armazenar valores que não serão modificados (variados) por instâncias
    - Valores constantes - modificador final

Atributos e métodos estáticos

- Maior utilidade de métodos estáticos:
  - Executar uma ação sem a necessidade de instanciar um objeto da classe inteira
  - Métodos que produzam o mesmo resultado independente de qual instância da classe
  - Implementar métodos que não dependam de dados da classe
  - Métodos que só dependam dos dados passados por parâmetro
  - O uso mais frequente de métodos estáticos é a criação de bibliotecas de métodos

Atributos e Métodos estáticos

Orientação a Objetos – DCC025

Gleiph Ghiotto Lima de Menezes

gleiph.ghiotto@ufjf.br