codecademy

Métodos de acesso, encapsulamento e estáticos

As palavras-chave públicas e privadas

Em Java, as palavras-chave `public` e `private` definem o acesso de classes, variáveis de instância, construtores e métodos.

`private` restringe o acesso apenas à classe que declarou a estrutura, enquanto `public` permite o acesso de qualquer classe.

Encapsulamento

O encapsulamento é uma técnica usada para manter os detalhes da implementação ocultos de outras classes. Seu objetivo é criar pequenos pacotes de lógica.

A palavra-chave privada

Em Java, as variáveis de instância são encapsuladas usando a palavra- `private` chave. Isso impede que outras classes acessem diretamente essas variáveis.

```
public class CheckingAccount{
  // Three private instance variables
  private String name;
  private int balance;
  private String id;
}
```

Métodos de acesso

Em Java, os métodos de acesso retornam o valor de uma `private` variável. Isso dá a outras classes acesso a esse valor armazenado nessa variável. sem ter acesso direto à própria variável.

Os métodos de acesso não recebem parâmetros e têm um tipo de retorno que corresponde ao tipo da variável que estão acessando.

```
public class CheckingAccount{
  private int balance;

  //An accessor method
  public int getBalance(){
    return this.balance;
  }
}
```

Métodos Mutantes

Em Java, os métodos modificadores redefinem o valor de uma `private` variável. Isso dá a outras classes a capacidade de modificar o valor armazenado nessa variável sem ter acesso direto à própria variável.
Os métodos mutantes recebem um parâmetro cujo tipo corresponde ao tipo da variável que está modificando. Os métodos mutantes geralmente não retornam nada.

```
public class CheckingAccount{
  private int balance;

  //A mutator method
  public void setBalance(int newBalance){
    this.balance = newBalance;
  }
}
```

Variáveis locais

Em Java, as variáveis locais só podem ser usadas dentro do escopo em que foram definidas. Esse escopo geralmente é definido por um conjunto de chaves. As variáveis não podem ser usadas fora desses colchetes.

```
public void exampleMethod(int
exampleVariable){
  // exampleVariable can only be used
inside these curly brackets.
}
```

Esta palavra-chave com variáveis

Em Java, a palavra- `this` chave pode ser usada para designar a diferença entre variáveis de instância e variáveis locais. Variáveis com `this.` referência a uma variável de instância.

```
public class Dog{
  public String name;

  public void speak(String name){
    // Prints the instance variable named
name
    System.out.println(this.name);

    // Prints the local variable named
name
    System.out.println(name);
  }
}
```

Esta palavra-chave com métodos

Em Java, a palavra- `this` chave pode ser usada para chamar métodos ao escrever classes.

```
public class ExampleClass{
  public void exampleMethodOne(){
    System.out.println("Hello");
  }

  public void exampleMethodTwo(){
    //Calling a method using this.
    this.exampleMethodOne();
    System.out.println("There");
  }
}
```

Métodos estáticos

Métodos estáticos são métodos que podem ser chamados dentro de um programa sem criar um objeto da classe.

```
// static method
public static int getTotal(int a, int b)
{
  return a + b;
}

public static void main(String[] args) {
  int x = 3;
  int y = 2;
  System.out.println(getTotal(x,y)); //
Prints: 5
}
```

Chamando um método estático

Métodos estáticos podem ser chamados anexando o operador ponto a um nome de classe seguido pelo nome do método.

```
int largerNumber = Math.max(3, 10); //
Call static method
System.out.println(largerNumber); //
Prints: 10
```

A aula de matemática

A `Math` classe (que faz parte do pacote java.lang) contém uma variedade de métodos estáticos que podem ser usados para realizar cálculos numéricos.

```
System.out.println(Math.abs(-7.0)); //
Prints: 7

System.out.println(Math.pow(5, 3)); //
Prints: 125.0

System.out.println(Math.sqrt(52)); //
Prints: 7.211102550927978
```

A palavra-chave estática

Métodos e variáveis estáticos são declarados como estáticos usando a `static` palavra-chave na declaração.

```
public class ATM{
  // Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;

  // A static method
  public static void averageMoney(){
    System.out.println(totalMoney
/ numATMs);
  }
```

Métodos estáticos e variáveis

Static methods and variables are associated with the class as a whole, not objects of the class. Both are used by using the name of the class followed by the `.` operator.

```
public class ATM{
  // Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;

  // A static method
  public static void averageMoney(){
    System.out.println(totalMoney
/ numATMs);
  }

  public static void main(String[] args){

    //Accessing a static variable
    System.out.println("Total number of
ATMs: " + ATM.numATMs);

    // Calling a static method
    ATM.averageMoney();
  }

}
```

Static Methods with Instance Variables

Static methods cannot access or change the values of instance variables.

```
class ATM{
// Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;

  public int money = 1;

  // A static method
  public static void averageMoney(){
    // Can not use this.money here
because a static method can't access
instance variables
  }

}
```

Methods with Static Variables

Both non-static and static methods can access or change the values of static variables.

```
class ATM{
// Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;
  public int money = 1;

 // A static method interacting with
a static variable
  public static void staticMethod(){
    totalMoney += 1;
   }

  // A non-static method interactingwith
a static variable
  public void nonStaticMethod(){
    totalMoney += 1;
  }
}
```

Static Methods and the this Keyword

Static methods do not have a `this` reference and are therefore unable to use the class's instance variables or call non-static methods.

```
public class DemoClass{

  public int demoVariable = 5;

  public void demoNonStaticMethod(){

  }
  public static void demoStaticMethod(){
    // Can't use "this.demoVariable" or
"this.demoNonStaticMethod()"
  }
}
```