Professor Marcio Feitosa

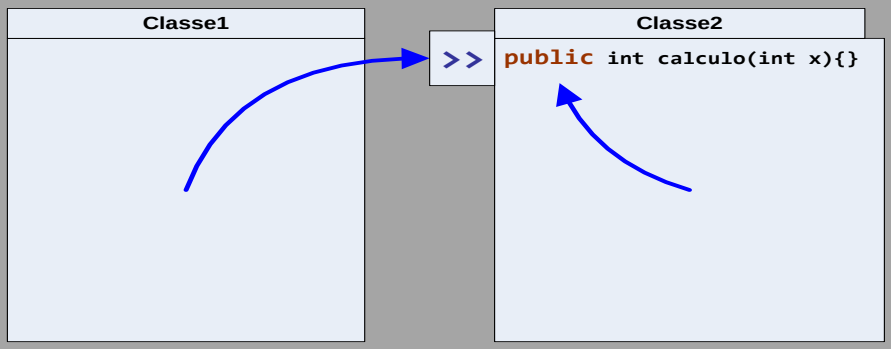
CURSO: Programação Orientada a Objetos com Java e C++

QUALIFICADORES[1] DE ACESSO

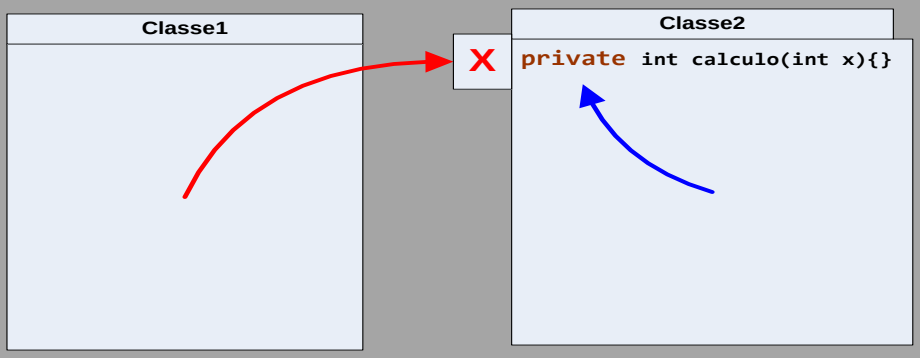
Como já dito, a classe possui atributos e métodos. Eventualmente, pode não ser desejável, ou adequado, que todos os atributos e/ou todos os métodos sejam acessíveis por qualquer outra classe do sistema[2].

São 3 os qualificadores:

public: o atributo ou método qualificado como *public* é acessível por qualquer classe do sistema.

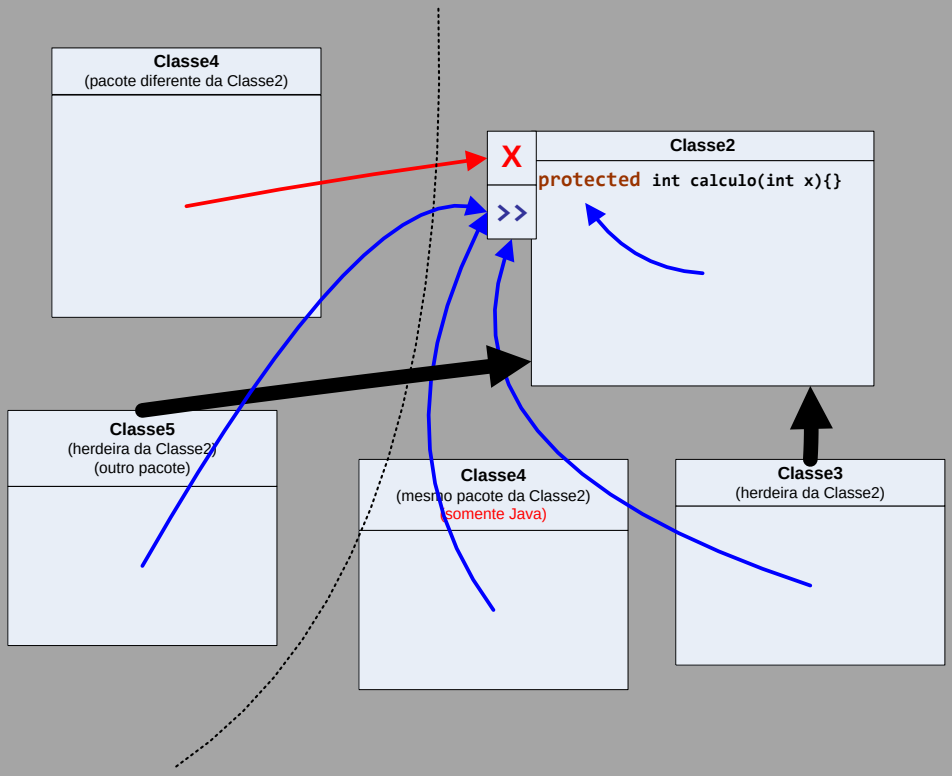


private: neste caso, o atributo ou método só é acessível por elementos internos da própria classe. As classes externas sequer enxergam este atributo/método.



[1] Chamados também de **especificadores** ou ainda de **modificadores**.
[2] Conjunto de elementos (classes, no caso) que operam de forma colaborativa.

protected: apenas as classes herdeiras[3], e a própria classe, podem acessar o atributo/método. No caso da linguagem Java, também acessam aquelas que estiverem no mesmo pacote, mesmo não sendo herdeiras.

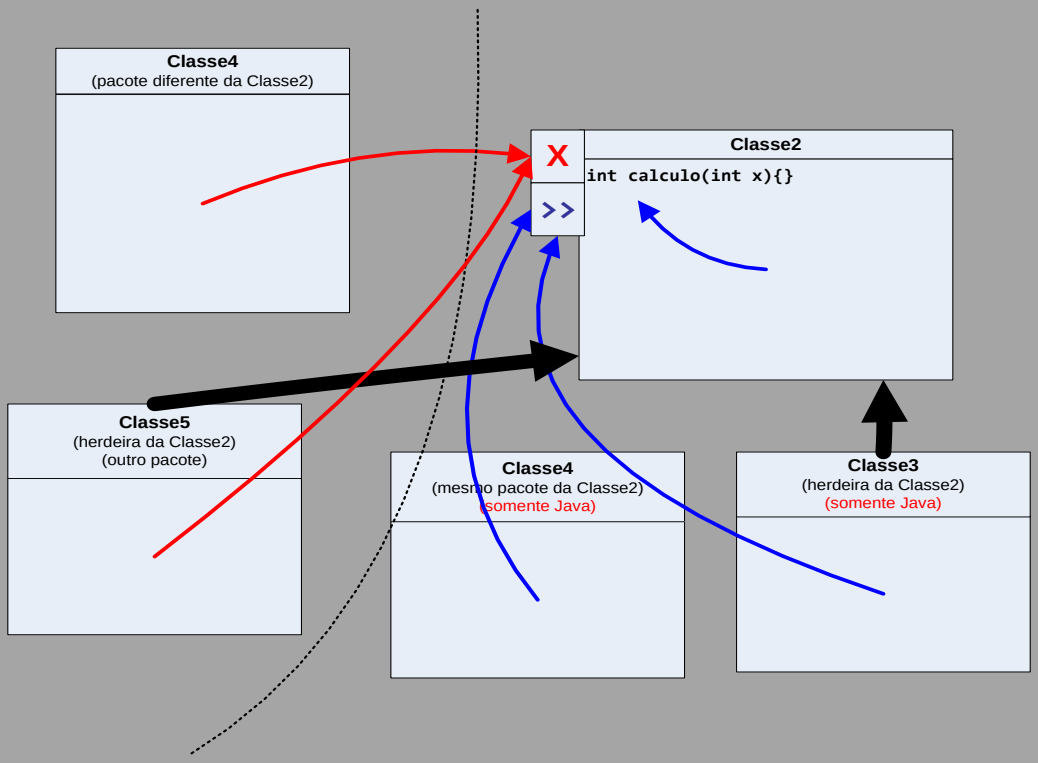


sem qualificador: atributos/métodos não qualificados têm comportamentos diferentes em Java e C++.

> **Java**: apenas a própria classe e as classes do mesmo pacote conseguem acessar. Classes herdeiras que estejam em outro pacote, não acessam.
>
> **C++**: atributos/métodos não qualificados são *private*.

[3] Veremos o mecanismo da herança mais adiante.

Professor
Marcio Feitosa



ENCAPSULAMENTO

A ideia fundamental do encapsulamento é não permitir que um atributo seja acessado diretamente e sim através de um método, chamado de **método de acesso**.

Para que isso seja possível, o atributo deve ser qualificado como *private* e seu método de acesso ser *public*[4].

Exemplo (em Java):

```
private int x;

public int getX(){
   return x;
}

public void setX(int x){
   this.x = x;
}
```

[4] Há alguns casos mais raros, dependendo da intenção do projeto, em que o método pode ser *protected* ou mesmo, no caso da linguagem Java, sem qualificador.

Professor Marcio Feitosa

Neste exemplo, temos um atributo de nome **x** qualificado como *private*. Consequentemente não é acessível por classes externas.

No entanto, os métodos abaixo, **getX** e **setX**, ambos *public*, possibilitam fazer o que seria feito diretamente com **x** se este fosse *public*, só que através desses métodos. O primeiro, **getX**, permite que seja o lido o valor de **x** e o segundo, **setX**, permite que o valor de **x** seja modificado.

Por convenção, o método de leitura tem o nome de *get* + o nome da variável, e o de escrita, *set* + o nome da variável.

Aliás, aproveitando o ensejo, a convenção da nomenclatura básica é a seguinte:

- nomes de classes iniciam com letra maiúscula, sendo o restante em letras minúsculas. Exemplo: **Classe1**.
- nomes de atributos e métodos são em minúsculas. Exemplos: **nome**, **calcular**();
- se o nome, tanto para classes como para atributos e métodos, for composto por duas ou mais palavras, todas as primeiras letras vão em maiúsculo (exceto a primeira letra de atributos e métodos que sempre é minúscula).
    - Exemplos:
        - **ClientePessoaJuridica** (classe),
        - **nomeCliente**,
        - **calculoTotalPagar**().
- constantes tem os nomes com todas as letras em maiúsculo. Exemplo: **ALIQUOTA_DO_IMPOSTO**.

Mas, voltando aos métodos **get** e **set**, a pergunta que pode ser feita é: se o acesso, no final das contas, é o mesmo, por que então a burocracia de se passar por um método se o acesso direto é computacionalmente mais econômico?

As respostas são basicamente duas:

1. o método só pode ser lido, mas não modificado, pois seu valor é modificado internamente sem a interferência direta do usuário. Neste caso só se implementa o método **get**.

2. o atributo está sujeito a alguma(s) regra(s), não podendo receber qualquer valor. Neste caso o método de acesso **set** poderá fazer esta verificação antes de atribuir o novo valor.

No caso 2, suponhamos que x só pode receber valores no intervalo de 0 a 10. Uma nota escolar, por exemplo. Sendo assim, o método pode ser escrito da seguinte forma:

```
public void setX(int x){
  if (x > 10) x = 10;
  else if (x < 0) x = 0;
   this.x = x;
}
```

ou então:

```
public boolean setX(int x){
  if (x < 0 || x > 10) return false;
   this.x = x;
  return true;
}
```

E mesmo que não haja, em primeiro momento, restrições quanto a uma faixa específica de valores, uma futura restrição poderá surgir devido a novas necessidades. Se isto ocorrer, a chamada do método já estará implementada, de forma que os pontos de chamada ao método, em sua maioria, não serão impactados de forma significativa, ao passo que o acesso direto ao atributo obrigará a alteração do código em **todos** os pontos de chamada (de acesso ao atributo para chamada ao método). Imagine o problemão que seria se a classe fosse utilizada por diversos softwares. Então, em já havendo o método, minimizam-se, ou até anulam-se, problemas futuros.

O QUALIFICADOR *static*

Como já dito, na videoaula anterior, o qualificador static exclui o atributo ou o método do paradigma POO. Este atributo ou método passará a ser fixado no chassi do programa, não podendo ser replicado em inúmeras instâncias.

Professor Marcio Feitosa

Os atributo/métodos *static* são chamados de:

- atributos/métodos de classe,

Ao passo que os não-*static*, podemos chamar da categoria POO, são chamados de:

- atributos/métodos de instância.

Os elementos *static* não precisam da instância de um objeto para serem acessados. Acessa-se diretamente da classe. Por exemplo, o método **static int calculo(int x)** na **Classe1**.

Se esse método não fosse static, precisaríamos instanciar um objeto para poder utilizá-lo:

```
Classe1 cl1 = new Classe1();
int z = 25;
int resultado = cl1.calculo(z);
```

Mas, sendo *static*, podemos fazer:

```
int z = 25;
int resultado = Classe1.calculo(z);
```

Um elemento *static* pode ser utilizado por uma instância da classe, mas ele será único e comum a todas as instâncias. Um atributo, por exemplo, se modificado por uma instância, esta modificação será comum a todas as outras.

A PALAVRA "INSTÂNCIA"

Instância, do verbo instanciar, não é uma palavra oficial da língua portuguesa. É um neologismo utilizado na computação que vem da língua inglesa, onde *instance* significa "caso". Ou seja, pode-se ter o caso 1, o caso 2, o caso 3, etc (objetos da mesma classe).

--- 10101010101010 ---