

POO: Polimorfismo

Eng. Computação

Prof. Victor A P Oliveira



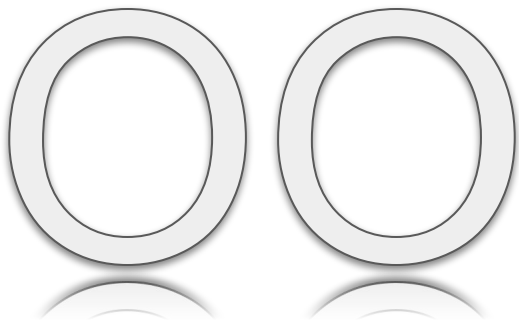
20 is here!

Introdução



Pilares:

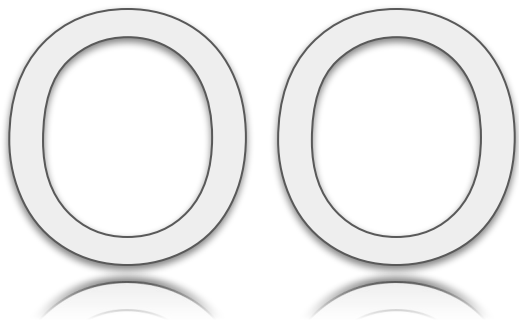
- Abstração
- Encapsulamento
- Herança
- Polimorfismo



Introdução



Pilares:

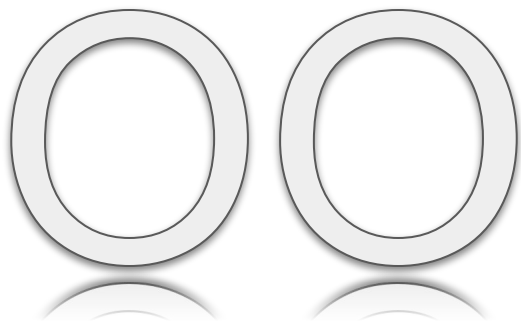


- Abstração 🍏
- Encapsulamento
- Herança
- Polimorfismo

Introdução



Pilares:

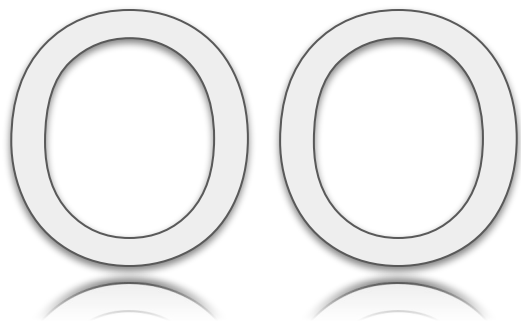


- Abstração 🍏
- Encapsulamento 🍏
- Herança
- Polimorfismo

Introdução



Pilares:

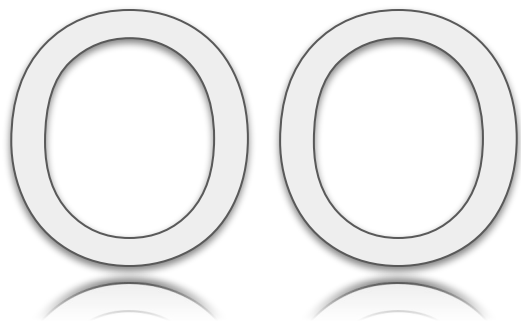


- Abstração 🍏
- Encapsulamento 🍏
- Herança 🍏
- Polimorfismo

Introdução



Pilares:



- Abstração 🍏
- Encapsulamento 🍏
- Herança 🍏
- Polimorfismo 🙌



Na aula passada...

Falamos sobre **Herança**

Relacionamento “**é um**” (herança public)

A classe derivada pode:

- adicionar novos **atributos**
- adicionar novos **métodos**
- sobrescrever **métodos** da classe base

Um **ponteiro** (ou **referência**) de uma classe básica **pode ser usado para qualquer objeto de classes derivadas**

Polimorfismo



“Que ser isso?”

Permite processar objetos de classes da mesma hierarquia como se todos fossem da classe básica.

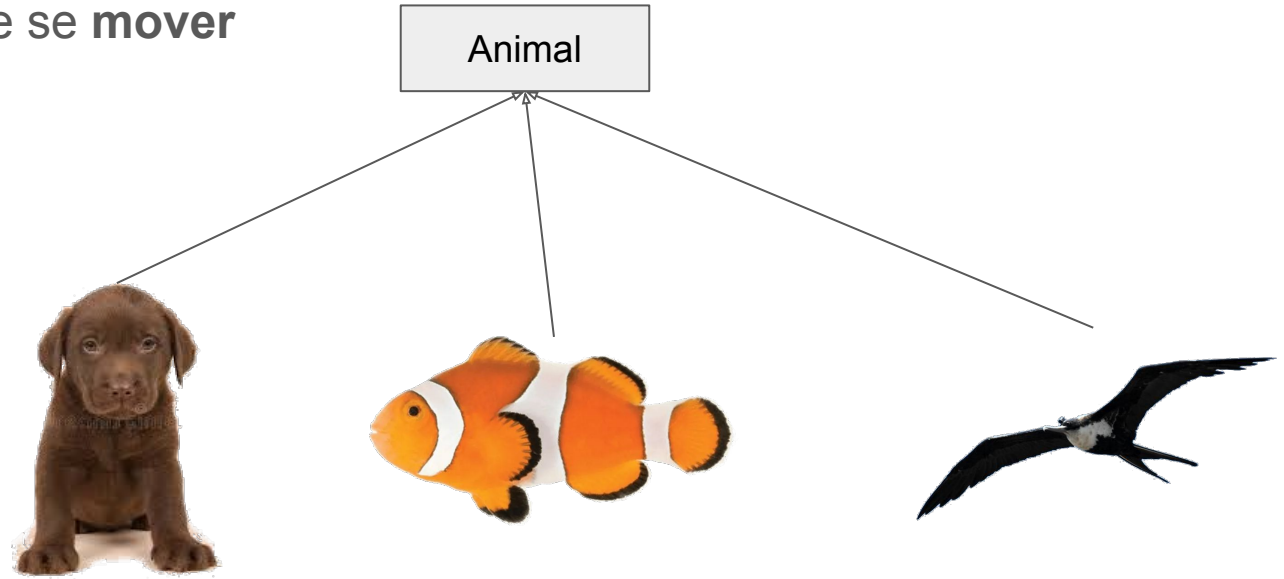
Programação é **focada no geral** ao invés de no específico.

Polimorfismo



Intuitivamente...

Um animal pode se **mover**



Polimorfismo



Intuitivamente...

Um animal pode se **mover**

O polimorfismo permite uma ação diferente a depende do tipo do objeto

Poli + **morfo** = Muitas formas de se comportar

Polimorfismo



Mecanismos básicos para o polimorfismo em C++:

- Herança +
- Sobrescrita +
- Métodos virtuais +
- Ponteiros ou referências

Polimorfismo



Mecanismos básicos para o polimorfismo em C++:

- Herança +
- Sobrescrita +
- Métodos virtuais +
- Ponteiros ou referências





Polimorfismo

Mecanismos básicos para o polimorfismo em C++:

- Herança +
- Sobrescrita +
- Métodos virtuais +
- Ponteiros ou referências

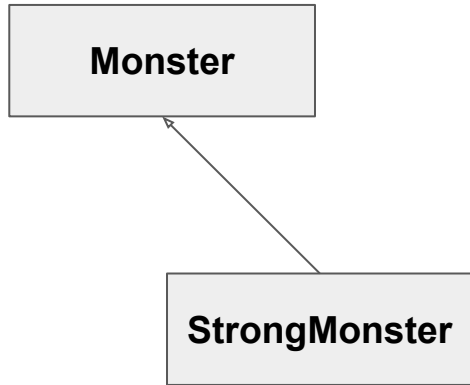
Como ele acontece na prática?!

O polimorfismo ocorre quando se invoca um método virtual por meio de um ponteiro (ou referência) de uma classe básica (direta ou indireta)

Analizando ponteiros x hierarquia



Ponteiros (ou referências) para objetos em uma hierarquia de Classes



Analizando ponteiros x hierarquia



Ponteiros (ou referências) para objetos em uma hierarquia de Classes

O relacionamento “é um” se dá da classe derivada para a classe básica



Analizando ponteiros x hierarquia



Ponteiros (ou referências) para objetos em uma hierarquia de Classes

*Sem **método virtual**, o ponteiro (ou referência) de uma classe básica para um objeto de classe derivada invoca as funcionalidades da classe básica*



Analizando ponteiros x hierarquia



Resumo da ópera

- Apontar um ponteiro de classe básica para um objeto de classe básica -> **OK**
- Apontar um ponteiro de classe derivada para um objeto da classe derivada -> **OK**
- Apontar um ponteiro de classe básica para um objeto de classe derivada -> **OK**
- Apontar um ponteiro de classe derivada para um objeto de classe básica -> **ERRO**

Estudo de caso

Os Monsters contra-atacam ;)



Métodos virtuais:



Estudo de caso



Métodos virtuais

Com métodos **virtual**, o tipo do objeto apontado, não o tipo do ponteiro (ou referência), determina qual versão do método invocar

A versão é invocada **dinamicamente** (em tempo de execução)

Estudo de caso



Métodos virtuais

Uma vez que um método é definido como virtual, ele permanece virtual (as sobrescritas) por toda a cadeia de herança.

Contudo, é importante acrescentar a palavra **override** no final do método nas classes derivadas, pois o compilador verifica se, de fato, está acontecendo uma sobrescrita, e avisa ao programador em caso de falha.



Estudo de caso



Atenção

Um ponteiro de classe básica que aponta para um objeto de classe derivada só vai “enxergar” os membros da classe básica!!!!





Classes **Abstratas**

Classe Abstrata é uma classe com um nível de abstração tão elevado (muito genéricas) que não se pode (ou não faz sentido) instanciar objetos. São classes criadas apenas para serem herdadas; para constituírem níveis superiores de uma hierarquia.

Em contraste, **Classes Concretas** são classes em que se pode instanciar objetos.



Classes Abstratas

Temos uma **classe abstrata** quando pelo menos um dos métodos é um **método virtual puro**

Método virtual puro: **virtual tipo** metodo(params) = 0;

Considere:

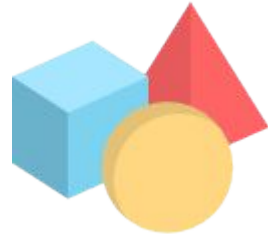
- Um método virtual puro não fornece uma implementação
- Se uma classe derivada é concreta, esta deve sobrescrever todos os métodos virtuais puros

Classes Abstratas



Temos uma **classe abstrata** quando pelo menos um dos método é um **método virtual puro**

Estudo de caso **Formas Geométricas**:





Classes Abstratas

Temos uma **classe abstrata** quando pelo menos um dos método é um **método virtual puro**

Atenção!!!

- Um método virtual tem uma implementação e dá a classe derivada a opção de sobrescrever
- Um método virtual puro não tem implementação e obriga a classe derivada a sobrescrever o método (senão a classe derivada continua abstrata)



Classes Abstratas

Temos uma **classe abstrata** quando pelo menos um dos método é um **método virtual puro**

Aplicação!!!

- Métodos virtuais puros são utilizados quando não faz sentido a classe básica ter uma implementação, mas o programador quer que todas as classes derivadas concretas implementem tais métodos

Destrutores virtuais

Se existir a possibilidade de objetos de classes derivadas serem destruídos a partir de ponteiros de classe básica então o destrutor da classe básica deve ser declarado como **virtual**

Veja:



Dica master top: Se uma classe tiver métodos virtuais, forneça um destrutor virtual, mesmo que ele não seja requerido.

Estudo de caso

A volta dos **Monsters** que não foram...



Outros pontos...

Downcast com `dynamic_cast`

Replit: <https://replit.com/@VictorOliver/Aula11Ex6>

operador `typeid`

Replit: <https://replit.com/@VictorOliver/Aula11Ex7#main.cpp>