SCC0504 - Programação Orientada a Objetos

Herança e Sobrescrita

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

Vocês lembram das nossas classes de tipos de Pokémon?

Classe

```
class WaterPokemon {
  int hp, atk, def;
  String type = "water";
  void takeDamage(amount,
enemy_type){
    if(enemy_type == "fire"){
      hp = hp - (amount/2);
    }
    else{
      hp = hp - amount;
    }
  }
}
```

```
class FirePokemon {
  int hp, atk, def;
  String type = "fire";
  void takeDamage(amount,
enemy_type){
    if(enemy_type == "water"){
      hp = hp - (amount * 2);
    }
    else{
      hp = hp - amount;
    }
  }
}
```

Classe

```
class PokemonWorld{
  void main(){
    WaterPokemon squirtle;
    FirePokemon charmander;
    squirtle.takeDamage(20, "fire");
  }
}
```

Contextualização

- É muito comum encontrarmos classes muito similares entre si, que necessitam apenas de alguns atributos ou métodos diferentes, mas mantém grande parte do resto.
- O mesmo acontece com TADs da programação estruturada.
- Re-escrever todas as partes em comum para uma nova classe/TAD e alterar apenas o que é necessário gera muito código extra, dificulta entendimento e reuso...

Contextualização

- Esse mesmo problema ficou claro na década de 80 pelos desenvolvedores.
- A maior reutilização de software era uma das melhores maneiras de aumentar a produtividade
- Os TADs eram as unidades ideais para reutilizar
- Mas as características e capacidades dos tipos existentes não eram adequadas para novo uso.

Contextualização

- O tipo requeria pelo menos algumas modificações, que podiam ser difíceis. Ou era necessário alterar todos os programas clientes.
- Todas as definições eram independentes e estavam no mesmo nível. Isso dificulta a estruturação do problema.
- Geralmente, os problemas possuem categorias de objetos relacionados, como “irmãos” (similares) e “pais e filhos” (algum tipo de subordinação).

Como Resolver?

Herança!

Herança

➔ Existem muitas semelhanças entre Pokémon de tipos diferentes:

Flareon : FirePokemon
- HP
- Attack
- Defense
- Special Attack
- Special Defense
- Speed

Vaporeon: WaterPokemon
- HP
- Attack
- Defense
- Special Attack
- Special Defense
- Speed

➔ Nesse caso, seria interessante ter uma classe "**Pokemon**".

Herança

Pokemon
- HP, Attack, Defense, Special Attack, Special Defense, Speed
- Tackle()

FirePokemon
- Ember()
- Flamethrower()

WaterPokemon
- Bubble()
- WaterGun()

Herança

Nesse caso, cada pokémon possui os **mesmos atributos**, mas cada tipo tem **habilidades diferentes**! As classes de cada tipo **herdam** da classe principal e se especializam.

Pokemon
- HP, Attack, Defense, Special Attack, Special Defense, Speed
- Tackle()

FirePokemon
- Ember()
- Flamethrower()

WaterPokemon
- Bubble()
- WaterGun()

Herança

A classe *Pokemon* é a chamada **classe base**, **superclasse** ou **classe pai/mãe**.

As classes *FirePokemon* e *WaterPokemon* são chamadas de **classe derivada**, **subclasse**, ou **classe filho/filha**

Pokemon
- HP, Attack, Defense, Special Attack, Special Defense, Speed
- Tackle()

FirePokemon
- Ember()
- Flamethrower()

WaterPokemon
- Bubble()
- WaterGun()

Herança

Quando ocorre a herança, as classes filhas como FirePokemon e WaterPokemon são **capazes de utilizar os atributos e métodos da classe mãe** Pokemon

Pokemon
- HP, Attack, Defense, Special Attack, Special Defense, Speed
- Tackle()

FirePokemon
- Ember()
- Flamethrower()

WaterPokemon
- Bubble()
- WaterGun()

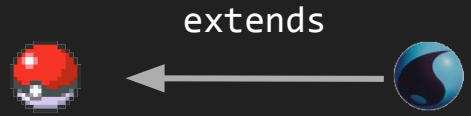
Herança

- *"A herança é uma forma de reutilização de software em que o programador cria uma classe que absorve dados e comportamentos de uma classe existente e os aprimora com novas capacidades. A reusabilidade de software economiza tempo durante o desenvolvimento de programa. Ela também encoraja a reutilização de softwares de alta qualidade já testados e depurados, o que aumenta a probabilidade de um sistema ser eficientemente implementado."*
  - DEITEL, H. M.; DEITEL, P. J. C++ Como Programar: 5 ed. São Paulo: Bookman, 2006. 1208 p.

Um pouco de código!

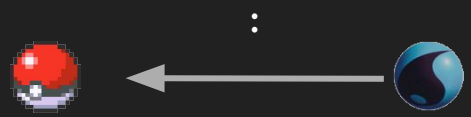
Herança

A palavra-chave extends é utilizada para definir herança em Java.



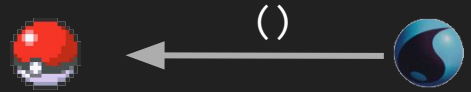
Pokemon WaterPokemon

O caractere “:” é utilizado para definir herança em C++ e C#



Pokemon WaterPokemon

Em Python, a classe mãe é colocada entre parênteses depois do nome da classe.



Pokemon WaterPokemon

Herança

- **Java**

```
class FirePokemon extends Pokemon{
    Métodos e Atributos}
```

- **C++**

```
class FirePokemon : public Pokemon{
    Métodos e Atributos}
```

- **C#**

```
class FirePokemon : Pokemon{
    Métodos e Atributos}
```

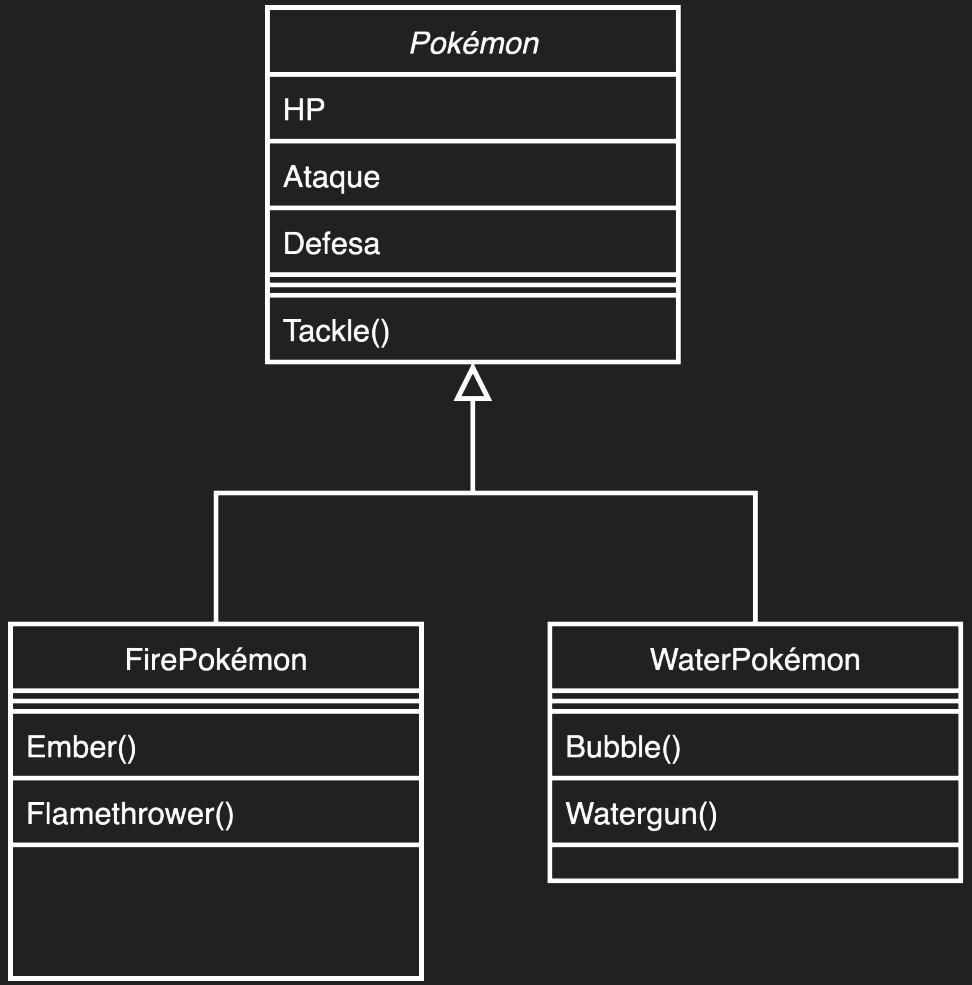
- **Python**

```
class FirePokemon(Pokemon):
    Métodos e Atributos
```

Herança

➔ **Java**

```
class FirePokemon extends Pokemon{
    “Métodos e Atributos”
    private string type = “Fire”
    public int Flamethrower(Pokémon target){
        private int power = 90;
        private int acc = 100;
        private int burnChance = 10;
        if(AccuracyCheck(acc)){
            target.ApplyDamage(power*atk, type);
            if(BurnCheck(10))
                target.ApplyStatus(“Burn”);
        }
    }
}
```

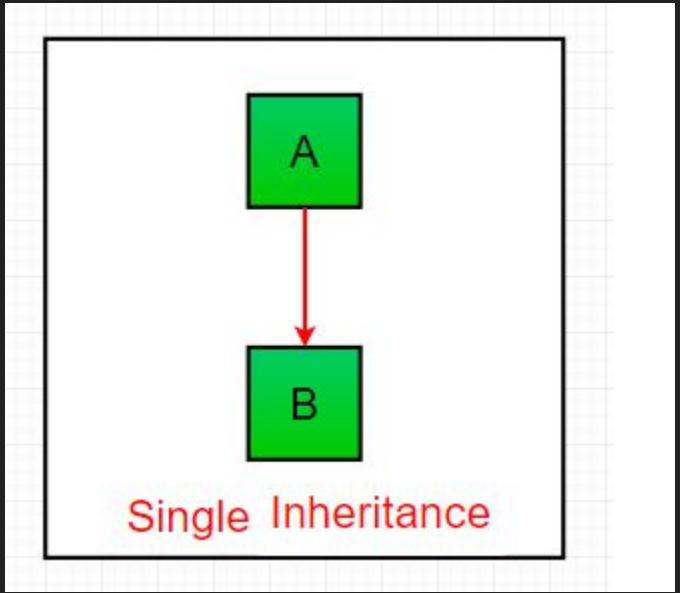

Pokémon
HP
Ataque
Defesa
Tackle()
FirePokémon
Ember()
Flamethrower()
WaterPokémon
Bubble()
Watergun()

Sobre os Modificadores de Acesso

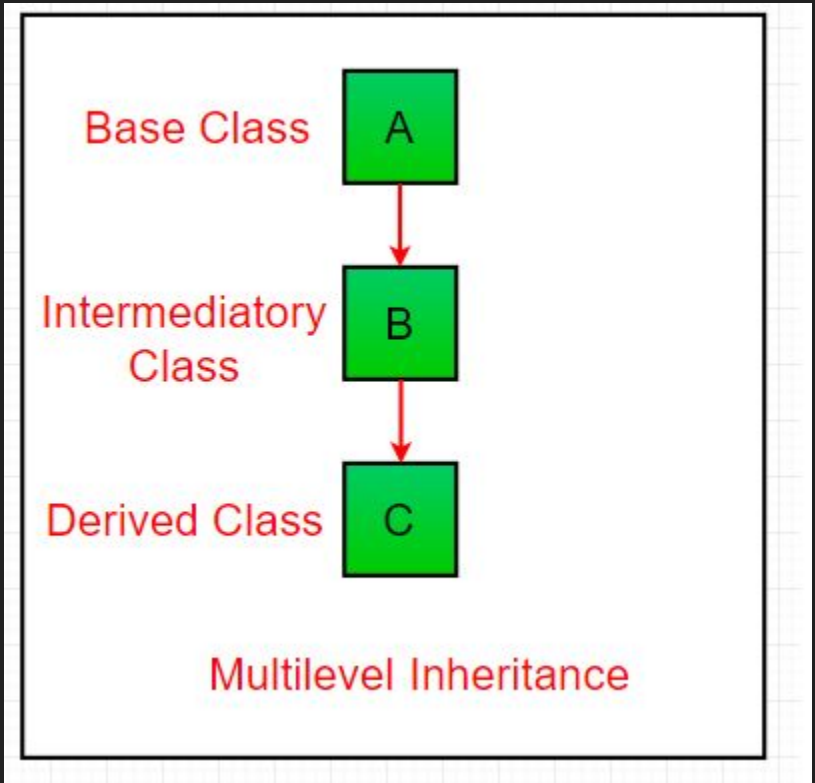
- Os atributos **private** só podem ser acessados na classe que os declarou (as herdeiras não tem acesso!)
- Os atributos **protected** podem ser acessados pela classe que os declarou e qualquer classe filha desta
- Os **public** podem ser acessados por qualquer classe

Tipos de Herança

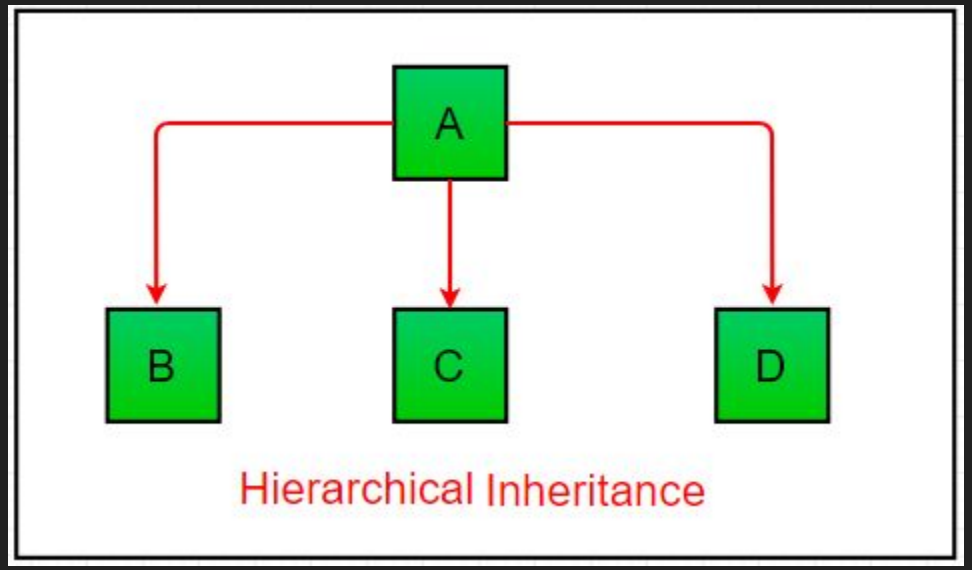
- As heranças podem ter vários tipos dependendo da quantidade e nível da herança



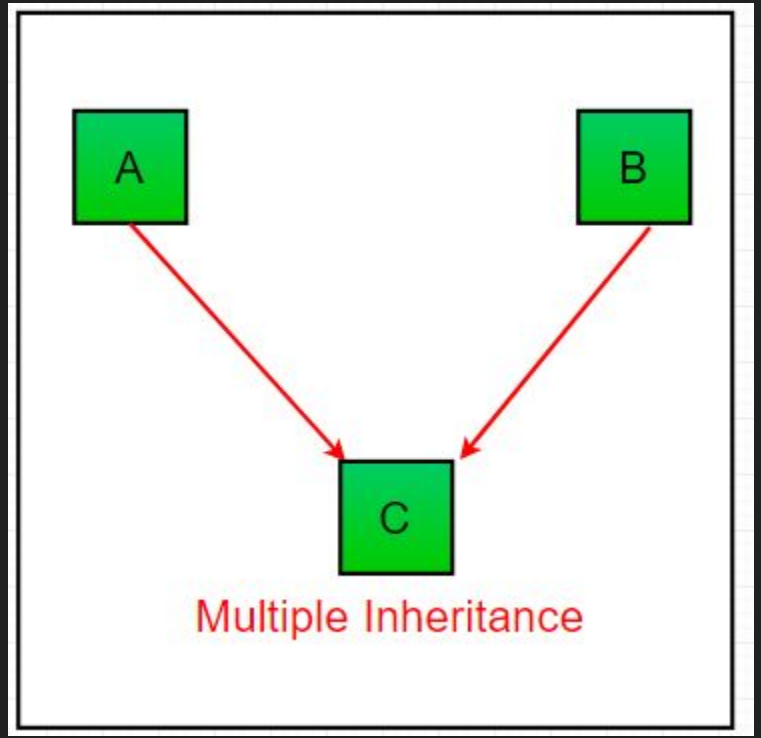
Fonte: https://www.geeksforgeeks.org/inheritance-in-java/



Fonte: https://www.geeksforgeeks.org/inheritance-in-java/



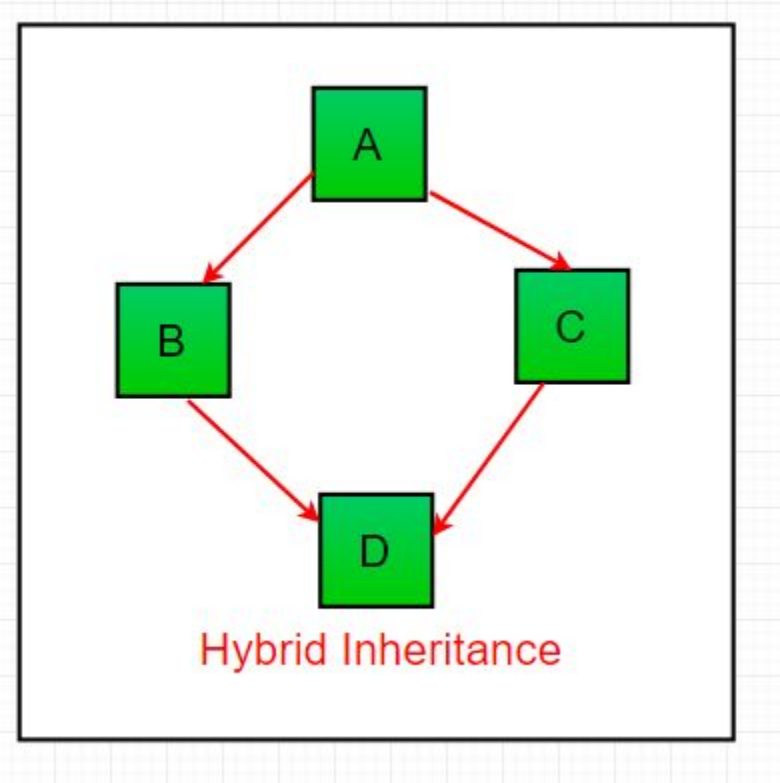
Fonte: https://www.geeksforgeeks.org/inheritance-in-java/



Fonte: https://www.geeksforgeeks.org/inheritance-in-java/

Herança Múltipla

- ➔ Não pode ser feita diretamente em Java e C#
- ➔ Mas pode ser alcançada por outros meios
- ➔ Veremos com mais detalhes nas aulas seguintes



Fonte: https://www.geeksforgeeks.org/inheritance-in-java/

Herança Híbrida

- ➔ Também não pode ser feita diretamente em Java e C#
- ➔ Mas pode ser feita através de interfaces
- ➔ Também vamos ver com mais detalhes em aulas futuras

Sobrescrita

Sobrescrita

- É a habilidade que uma classe filha tem de modificar um método que uma de suas classes superiores possui
- É possível re-escrever o método para aquela classe filha em específico (e suas filhas)
- Você pode impedir que um método seja sobrescrito declarando ele como *final*

Sobrescrita

```
class Pokemon{
  int hp, atk, def;
  String type = "normal";
  void takeDamage(amount, enemy_type){
    if(enemy_type == "fighting"){
      hp = hp - (amount*2);
    }
    else{
      hp = hp - amount;
    }
  }
}
```

Sobrescrita

```
class WaterPokemon extends Pokemon{
  type = "water";
  void takeDamage(amount, enemy_type){
    if(enemy_type == "fire"){
      hp = hp - (amount/2);
    }
    else{
      hp = hp - amount;
    }
  }
}
```

Referências

- https://www.geeksforgeeks.org/inheritance-in-java/
- https://stackify.com/oop-concept-inheritance/