



Programação Orientada a Objetos – Aula 12


IFPE – Campus Igarassu
2016.1

Ranieri Valença
06/10/2016



Tópicos de hoje

- Herança
 - Estudo de caso
 - Herança de atributos
 - Herança de métodos
 - Casting
 - Casting de classes



Falando sobre a nossa Quest Lendária (sim, aquela...)

Pokemon

Espécie

Tipo

Apelido

Poder inicial

Experiência atual


Digimon

Nome

Apelido

Poder inicial

Experiência atual



Falando sobre a nossa Quest Lendária (sim, aquela...)

Pokemon

Espécie

Tipo

Apelido

Poder inicial

Experiência atual

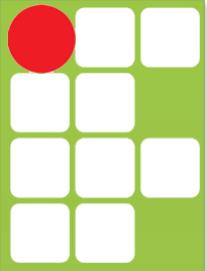
Digimon

Nome

Apelido


Poder inicial

Experiência atual



Falando sobre a nossa Quest Lendária (sim, aquela...)

Ou seja, os atributos **apelido, poder inicial e experiência atual** estão repetidos.




Falando sobre a nossa Quest Lendária (sim, aquela...)

Outra situação: observe o seguinte trecho da descrição da quest:

"

*O poder total de um **Pokemon** ou de um **Digimon** é calculado conforme a fórmula*
*[poder total = ((poder inicial) + (poder inicial) * (level atual) * (level atual) / 5)]*


"



Falando sobre a nossa Quest Lendária (sim, aquela...)

Então, poderíamos pensar nesse cálculo do poder total como o seguinte método:

```
public int poderTotal() {  
    int ini = this.poderInicial;  
    int lvl = this.levelAtual;  
    int total = ini+(ini*lvl*lvl)/5;  
    return total;  
}
```



Falando sobre a nossa Quest Lendária (sim, aquela...)

E esse método serviria para **ambas as classes**. Logo, o mesmo código estaria repetido em duas classes (nesse caso – poderíamos ter mais outras classes com a mesma ideia).



Herança

Orientação a Objetos possui um recurso chamado **herança**, na qual uma classe **compartilha atributos e métodos** de outra classe, como se estivesse "herdando-os".



Herança

Por exemplo, num dado sistema, as entidades **Aluno** e **Professor** podem ambas compartilhar atributos como nome, endereço e telefone.

Mas esses atributos poderiam estar associados a uma outra entidade, que poderíamos chamar de **Pessoa**.

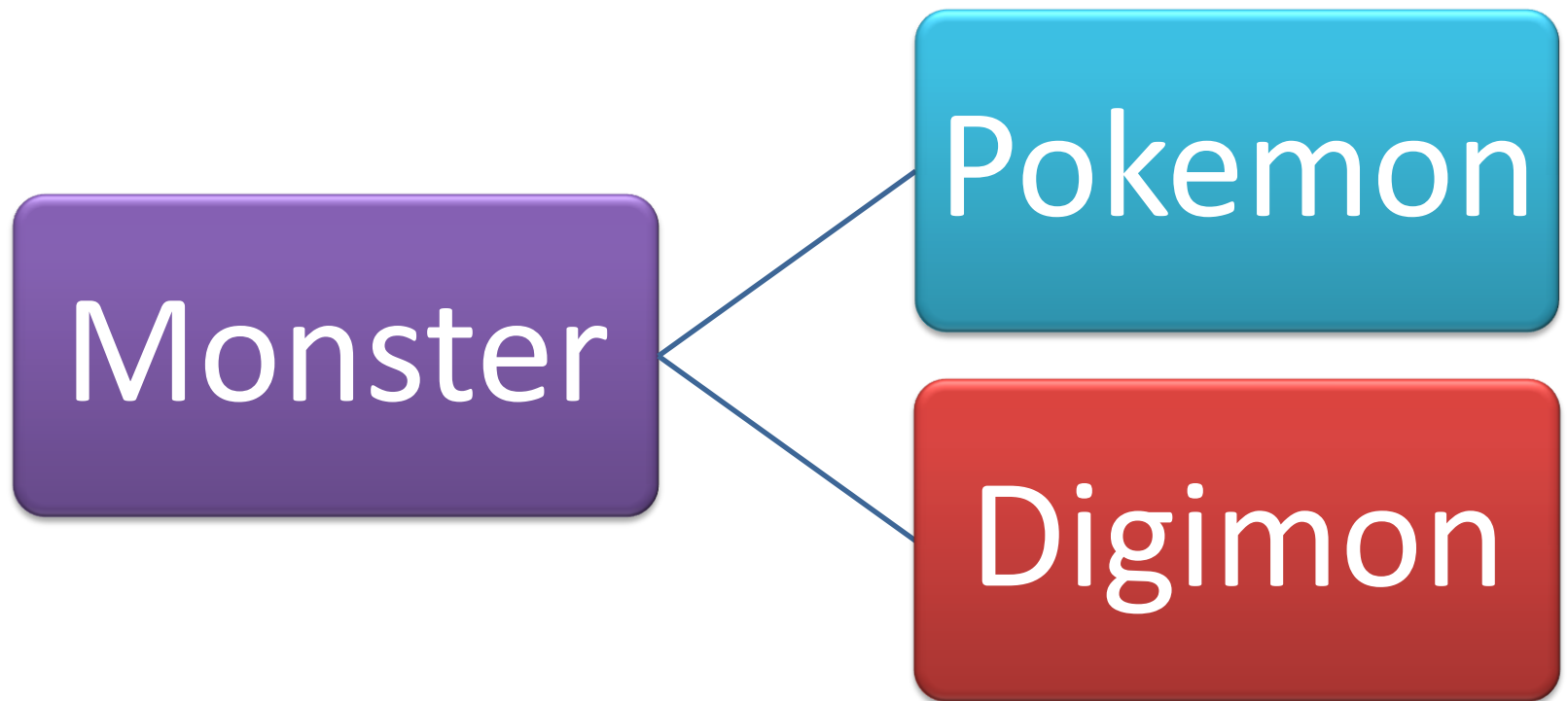
Assim, **toda Pessoa teria nome, endereço e telefone**, e Aluno seria *um tipo de Pessoa* (o que de fato é ~~(às vezes)~~), tendo apenas os atributos específicos de um Aluno. O mesmo aconteceria para a entidade Professor.



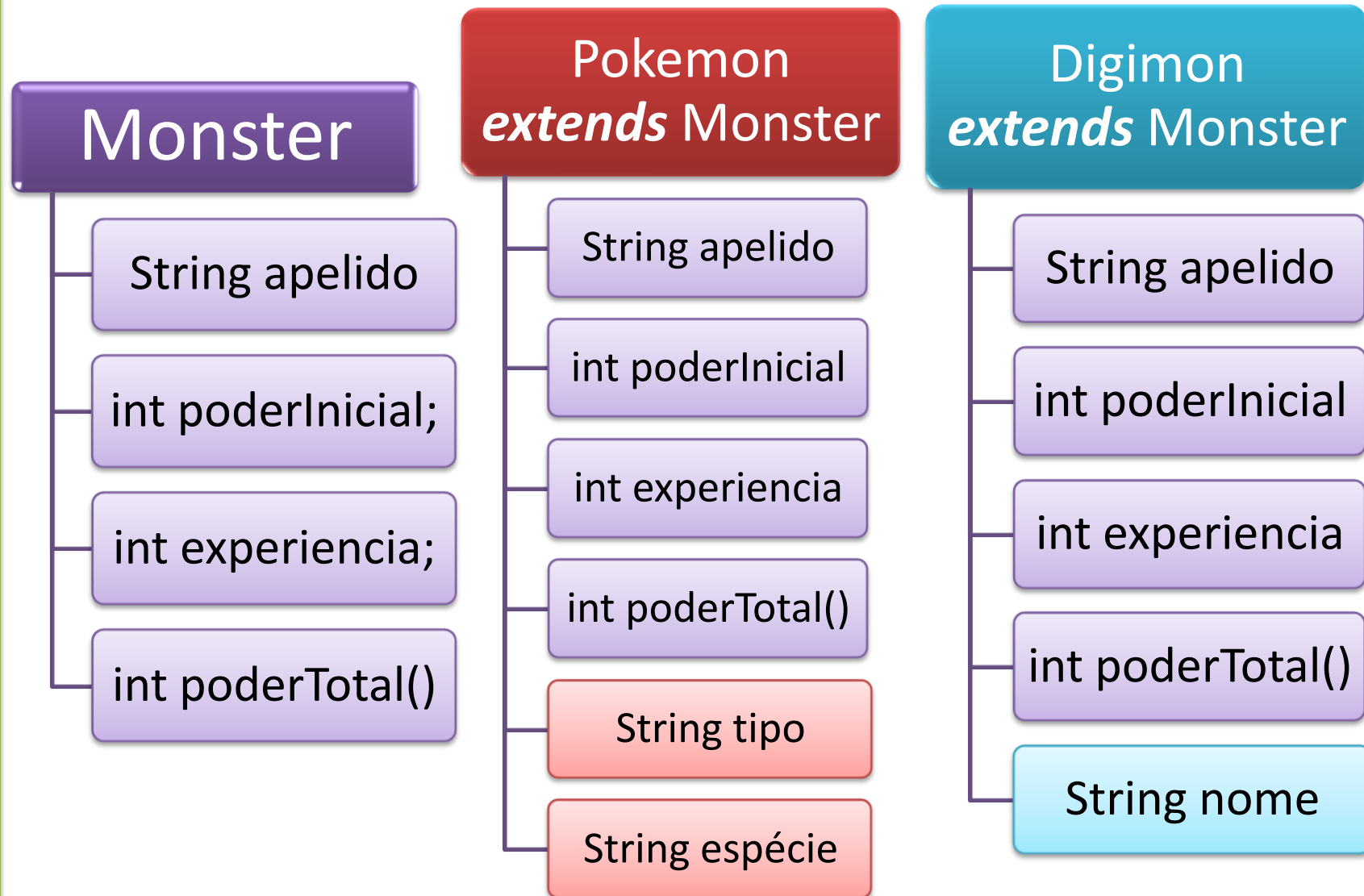
Herança

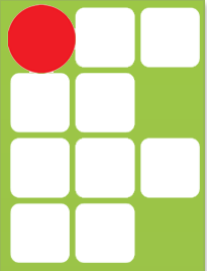
Pensando dessa forma, como poderíamos melhorar o código que fizemos ~~(ou deveríamos ter feito)~~ para nossa quest lendária?

Herança na Quest Lendária



Herança na Quest Lendária





Herança na Quest Lendária

Sintaxe em Java



Herança

A classe que é herdada é também conhecida como **classe pai** ou **superclasse**.

A classe que herda é conhecida como **classe filha** ou **subclasse**.



Herança

E os construtores?


Também são herdados!

O construtor da superclasse pode ser usado através da palavra chave **super**.



Herança

Exemplo



Herança na Quest Lendária

Exercício:

Implemente as classes
TreinadorPokemon e
TreinadorDigimon utilizando o
conceito de **Herança**

(note que pode ser preciso criar uma outra
classe)



Casting

Linguagens de programação tipadas (como é o caso de Java) possuem um mecanismo de conversão de tipos embutido.

Já fazemos isso naturalmente, tratando tipos primitivos como Strings:

```
int a = 10;
```

```
String s = "O número é " + a;
```



Casting

Nesse caso, a conversão foi feita de forma **implícita** – a variável "a" é do tipo int, mas está sendo concatenada com uma String, logo ela *precisa* ser tratada como uma String.



Casting

Essa conversão é chamada de casting (ou moldagem).

Podemos fazer um casting explícito, colocando o tipo desejado entre parênteses, antes da variável:

```
double pi = 3.1415;
```

```
int i = (int)pi;
```



Casting entre objetos

Exemplos (casting)

Exemplos (declaração usando
superclasses)



Exercício

Quest lendária 1

([re]escreva as classes básicas da última quest lendária utilizando herança onde for possível)

