Herança

Nós já discutimos algumas relações entre objetos: associação, composição e agregação.

Em um jogo de xadrez, um jogador (player) pode ser um humano ou pode ser um robô que possui uma inteligência artificial.

Nesse caso, não é interessante associar o jogador a um humano.

Também não é interessante colocar o robô como parte do objeto jogador.

Nós simplesmente queremos dizer que:

João é um jogador
AND
Robozinho é um jogador

Precisamos utilizar uma relação muito importante em orientação a objetos:
Herança

Herança é como uma árvore genealógica.

Você pode herdar olhos azuis do seu pai ou da sua mãe.

O seu pai herda o último nome do seu avô.

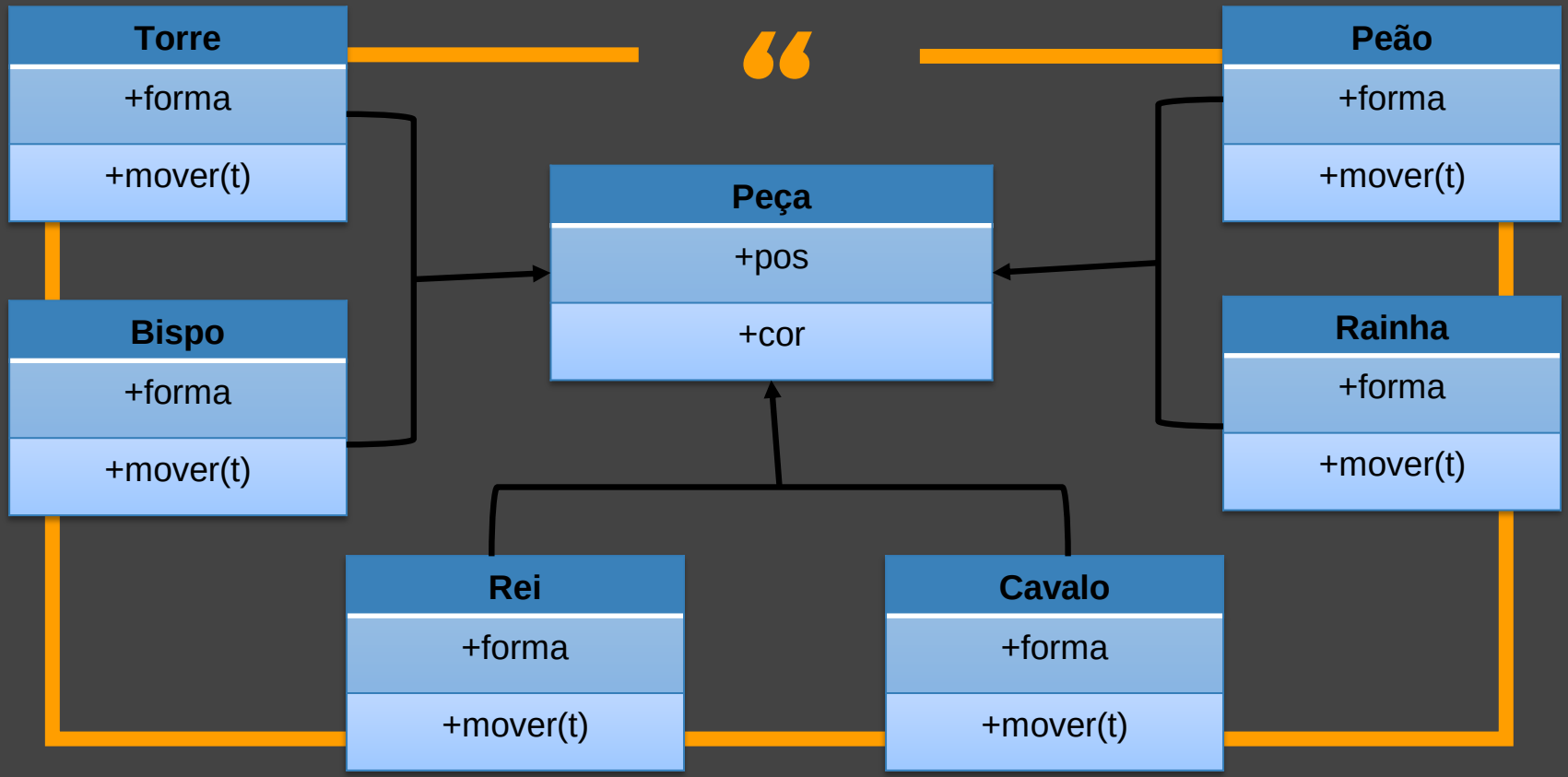
Em orientação a objetos, ao invés de herdar características e comportamentos de uma pessoa, uma classe pode herdar atributos e métodos de outra classe.

Exemplo: existem 32 peças no jogo de xadrez, mas somente seis tipos diferentes de peças onde cada uma se comporta de forma diferente quando é movida.

Todas essas classes de peças tem propriedades, mas cada uma tem a sua forma para ser desenhada na tela e faz movimentos diferentes.

Você tem seis tipos de peças que herdam de uma classe Peça.

Uma imagem vale mais que mil palavras, vamos para uma ilustração...


Torre
+forma
+mover(t)
Peão
+forma
+mover(t)
Peça
+pos
+cor
Bispo
+forma
+mover(t)
Rainha
+forma
+mover(t)
Rei
+forma
+mover(t)
Cavalo
+forma
+mover(t)

Peça é a classe base e Torre, Bispo e o restante são subclasses. Todos os subtipos possuem o atributo “pos” e “cor” herdados da classe base.

Cada peça por exemplo tem o seu método de movimento para poder se mover para uma nova posição no tabuleiro.

Por que cada peça tem um método mover?

Se não tivéssemos, então ficaria confuso ao tentar mover uma peça.

Cada peça tem um tipo de movimento. Exemplo: os peões se movem somente para frente, uma casa por vez.

Nós podemos também criar um método modelo para se mover na classe Peça.

Então as subclasses podem sobrepor (override) esse método com uma implementação mais específica.

A implementação padrão pode por exemplo mostrar uma mensagem de erro "Essa peça não pode ser movida".

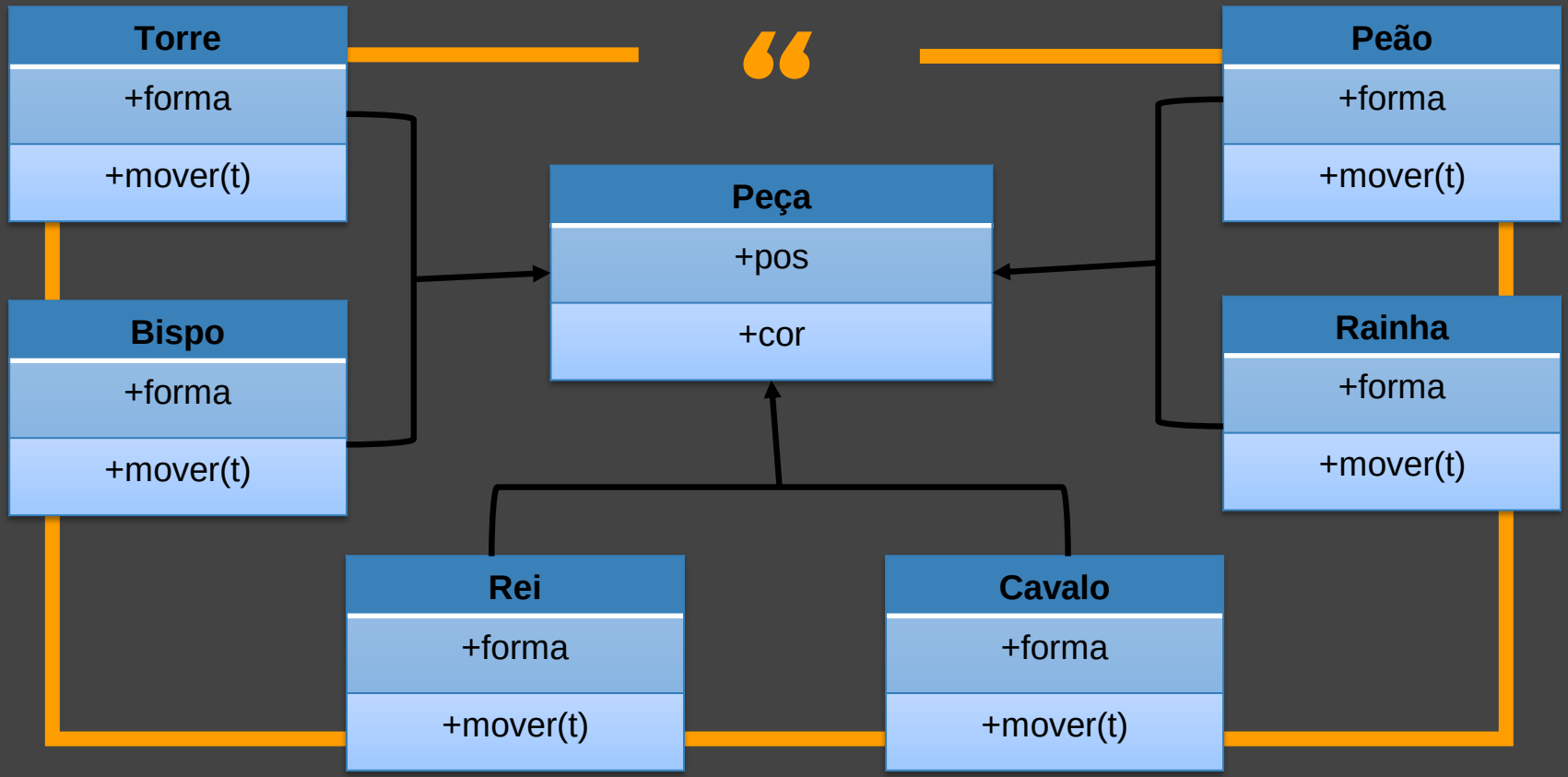
Hora do café, continuamos na próxima aula

☺

Sobreposição de métodos

Vamos voltar ao nosso xadrez...

Nós podemos também criar um método mover na classe Peça.


Torre
+forma
+mover(t)
Peão
+forma
+mover(t)
Peça
+pos
+cor
Bispo
+forma
+mover(t)
Rainha
+forma
+mover(t)
Rei
+forma
+mover(t)
Cavalo
+forma
+mover(t)

Subclasses podem sobrepor (override) esse método com uma implementação mais específica.

Nós podemos ter uma classe jogador que faz uso de inteligência artificial que possui um método que calcula o movimento e decide para onde a peça deve se mover no tabuleiro.

Poderíamos também ter outra classe mais básica que escolhe a posição de forma randômica.

Ou seja, temos uma para jogadores mais experientes (usa inteligência artificial) e outra para jogadores iniciantes (calcula de forma randômica).

O importante aqui é que podemos sobrescrever métodos na classe que faz algo específico: usar inteligência artificial ou mover de forma randômica.

Sobreposição de métodos (override) é um recurso em orientação a objetos que permite que a subclasse reescreva uma implementação específica de um método que já está previsto em uma superclasse.

Lembra de herança? Se uma classe apenas herda os métodos de outra classe (sem modificação), então é herança.

Mas tem alguém que caminha lado a lado com a herança: polimorfismo.

Se herda métodos e os modifica, ou seja, o mesmo método se comporta de formas diferentes em classes diferentes, então isso é chamado de polimorfismo.

Tem-se dois tipos de polimorfismo: sobrecarga (overload) e sobreposição (override).

A sobrecarga (overload) permite mais de um método com o mesmo nome dentro da mesma classe se diferenciando nos argumentos.

Já a sobreposição (override) permite reescrever um método em uma subclasse que possua um comportamento diferente do método de mesma assinatura na superclasse.

Na sobreposição eu posso reescrever na subclasse os métodos implementados na superclasse.

A subclasse pode redefinir métodos herdados.

Não é permitida a sobrecarga de métodos em Python!

E se quisermos fazer com que a implementação do método mover seja necessário nas subclasses?

Então teríamos que tornar a classe Peça uma classe abstrata.

Continuamos na próxima aula, te aguardo lá! ☺