02

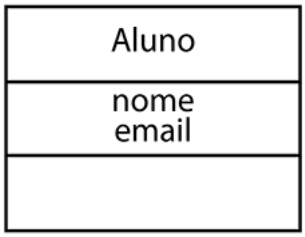
Sobre Diagrama de classes

Toda equipe quando vai começar o software e já tem os requisitos levantados, para e pensa um pouco sobre como vai modelar aquele domínio. Quais classes serão criadas? Quais os comportamentos existentes? Como uma se relaciona com a outra?

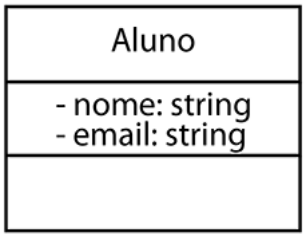
Se fôssemos usar português para isso, gastaríamos muito tempo, afinal precisaríamos de muitas palavras para expressar algo. É por isso que a UML nos provê o chamado **diagrama de classes**.

Vamos começar modelando nosso sistema de e-learning. A primeira classe a ser modelada é a classe `Aluno`. Toda classe é representada por um quadrado, que é dividido em 3 partes:

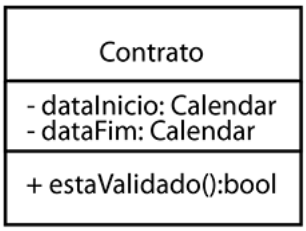
A parte de baixo, é onde colocamos os métodos (comportamentos) daquela classe. Na parte do meio, colocamos os atributos. Um aluno, por exemplo, tem nome, e-mail, e assim por diante.



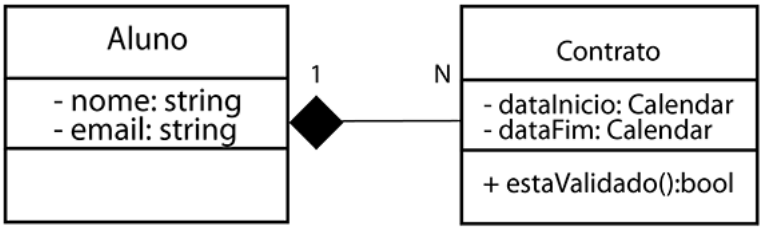
O desenho acima já é um diagrama de classe válido. Mas ainda podemos colocar mais informação nele. Nós programadores sabemos que atributos são geralmente privados. Podemos indicar isso no diagrama, usando o sinal de `-`. Podemos também indicar que nome é do tipo "String", colocando ": String" após o nome do atributo. Veja:



Vamos para a próxima classe, a classe `Contrato`. Um atributo tem data de início e data de fim. Além disso, o contrato tem o comportamento `estaValido()`, que nos diz se o contrato é válido ou não. Por ser um método, o colocamos na parte de baixo do diagrama, e marcamos também seu tipo de retorno. Vamos marcá-lo também como público, usando o sinal de +. Veja:

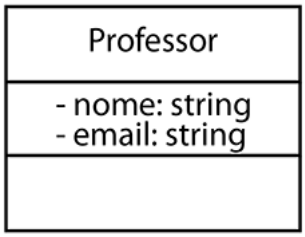


Agora temos duas entidades: Aluno e Contrato. Precisamos agora relacionar uma na outra, afinal um aluno pode ter muitos contratos. Para indicar isso, usamos um losango (diamante). Colocamos o diamante do lado da classe que contém a outra. No nosso caso, como Aluno tem muitos contratos, o losango fica do lado dele:

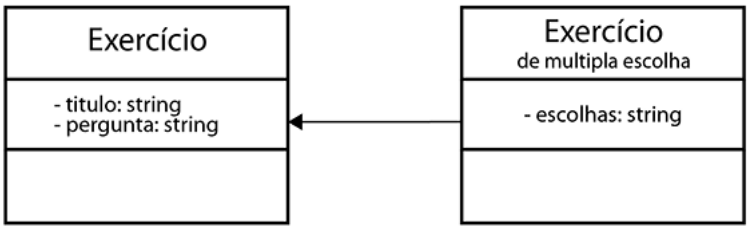


Veja também que aproveitamos o diagrama e indicamos que o relacionamento é "um-para-muitos", por meio do "1" e do "n" no diagrama.

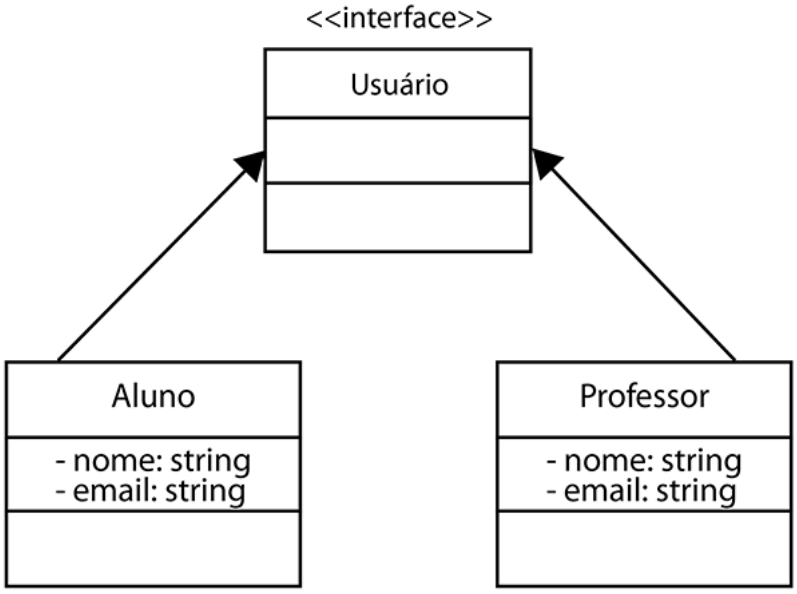
Vamos criar a classe `Professor`, que também é simples:



Vamos para a próxima classe, que é a classe `Exercicio`. O exercício tem um título e uma pergunta, ambos do tipo String. Mas exercícios podem ser de tipos diferentes: múltipla escolha, aberto, etc. Em linguagens OO, reaproveitaríamos bem o código, usando herança. A classe `ExercicioDeMultiplaEscolha` por exemplo herdaria da classe `Exercicio`. Indicamos herança, usando uma seta:



Vamos agora criar uma interface em comum para `Aluno` e `Professor`: a interface `Usuario`. Para indicar que `Usuario` é uma interface, usamos estereótipos. Veja:



Veja só então como o diagrama de classes facilita a vida dos desenvolvedores que precisam modelar classes. É fácil de ler e de escrever. Você pode se aprofundar o quanto quiser nesse diagrama... Ou não! Se você não vê necessidade de colocar atributos, por exemplo, você pode ignorá-los. Lembre-se que o diagrama serve para passar a mensagem para frente, e não para "ser bonito".