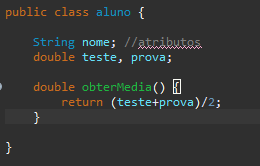
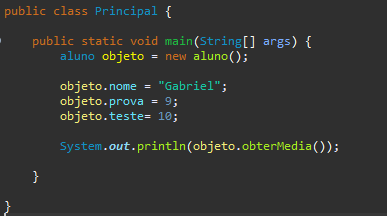
**Classe:**

A planta do objeto!



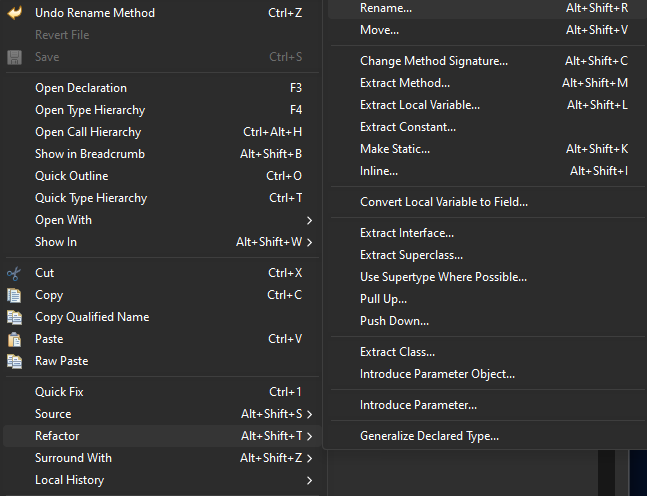
Como posso fazer com que outra classe chame essa classe aluno?



**Objeto:**

Materialização da classe, utiliza new pois não é um tipo primitivo.

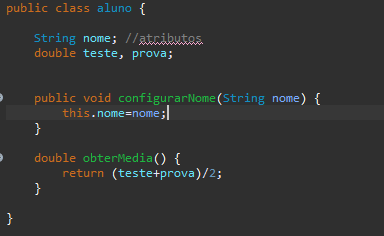
DICA:

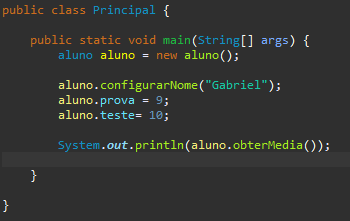


Usando refactor ele muda o nome em todos os lugares! (da palavra selecionada)

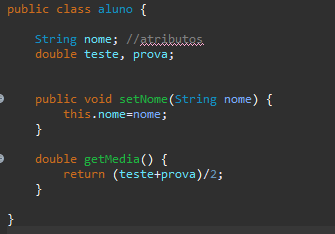
**Encapsulamento:**

É recomendado que toda manipulação dos dados não pode ser dentro da classe! Mas fora. Assim:

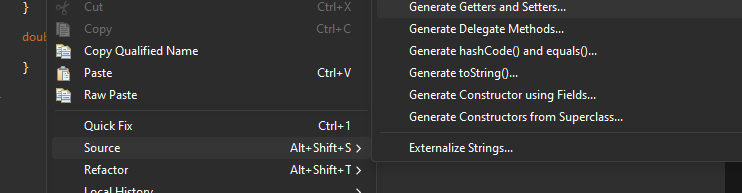




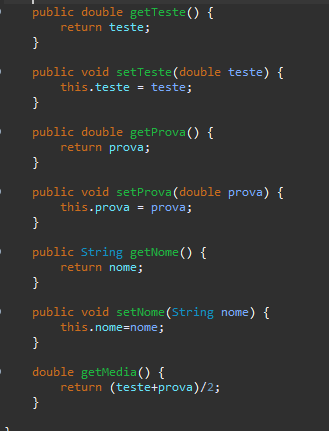
Só que o pessoal não faz assim! tem uma boa pratica para isso.(na nomenclatura):



Quando quiser ter acesso para consultar ou alterar usar a palavra: get ou set



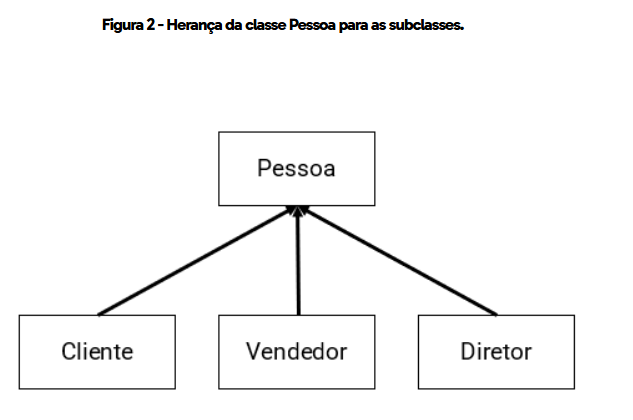
Ele faz automaticamente de todas as variaveis:



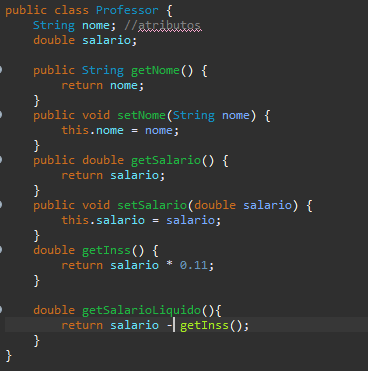
**Herança e classe abstrata:**

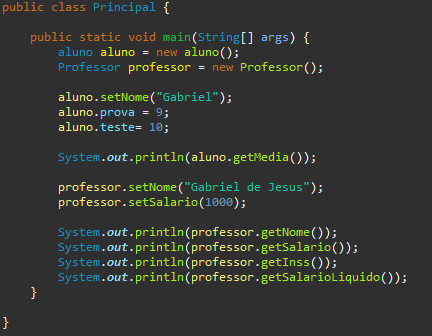
Para que repetir código? Bora aproveitar código já escrito.

herança é um mecanismo disponível nas linguagens de programação que se apoiam no paradigma de objetos que possibilita que características comuns a diversas classes sejam reunidas em uma classe base. A partir desta base, as demais classes podem ser especificadas de modo que cada classe derivada (ou subclasse) apresente características e comportamentos herdados da classe base e outras que foram definidas exclusivamente para elas. Na prática, isso significa reunir todas as características comuns entre várias entidades e a partir disso criar uma base em que todas elas se baseiam. Em resumo, as características e comportamentos de uma classe (base) são herdadas a outras classes (subclasses).



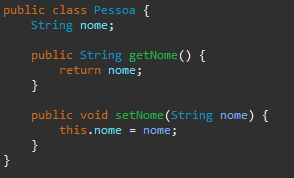
Herança é sempre utilizada em Java, mesmo que não explicitamente. Quando uma classe é criada e não há nenhuma referência à sua classe base, implicitamente a classe criada é derivada diretamente da classe Object. É por esse motivo que todos os objetos podem invocar os métodos da classe Object, tais como equals() e toString().



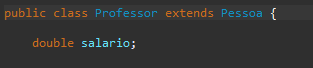
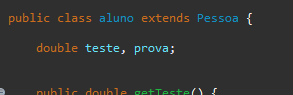


A herança é para que o que é em comum entre a classe, eu não precise escrever 2 vezes! Mas como posso fazer isso?

Nos 2 (aluno e professor) não tem o nome? São pessoas no caso? Então:



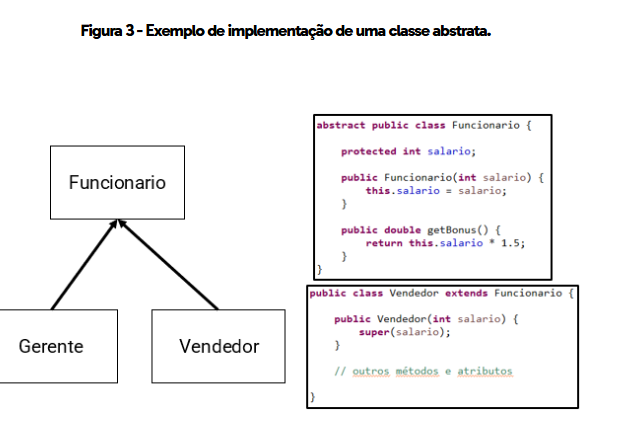
E como passa isso para os alunos e professores? É colocando na classe do aluno/professor o nome extends Pessoa.



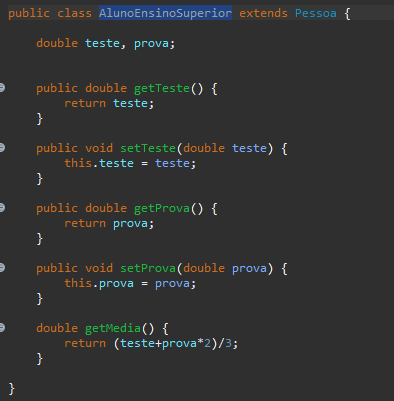
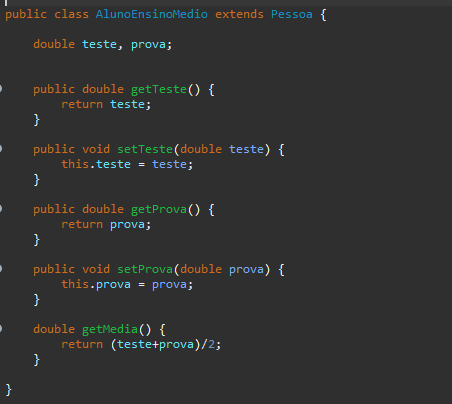
A herança tem que ser para o mesmo tipo: ex: pessoa = aluno e professor. Se fosse cachorro não daria certo por exemplo.

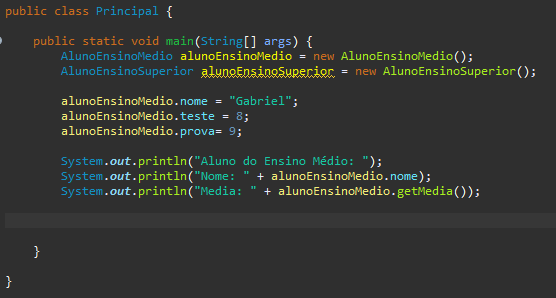
Já a **Classe abstrata** tem métodos que obriga que suas filhas implementarem esses métodos.

As abstratas são classes que estruturam um tipo mas que não podem ser instanciadas. É quase como se fosse uma interface, a diferença é que elas podem implementar comportamentos e características, mas nunca serão usadas diretamente, somente as classes que a herdarem poderão usá-la.



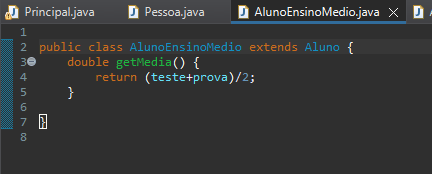
Vamos colocar aluno ensino médio e aluno ensino superior (esse com a prova com peso 2).

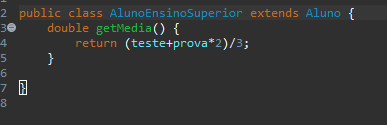


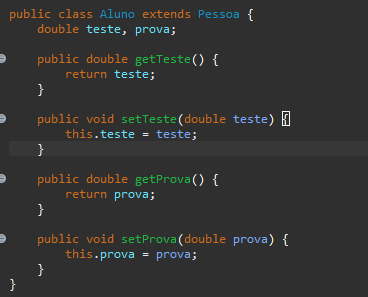


E agora se eu quiser copiar para o ensino superior?

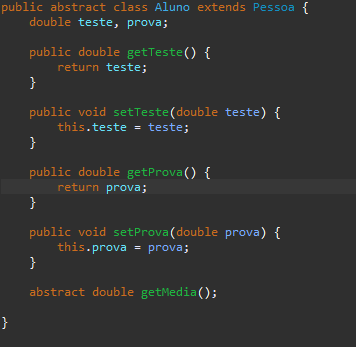
É melhor criar uma classe aluno e o que tiver igual jogar la:







Mas e esse getmedia? Como posso fazer para que ele vá para classe aluno?



Fazendo isso, todos os filhos tem que ter o getMedia();

**Interface:**

Uma classe totalmente abstrata, todos os métodos ser abstrato.

A sua função é fazer com que todas as classes associadas a elas sejam “obrigadas” a cumprir um "acordo’'. Este acordo será possuir as características e comportamentos definidos por ela. Se ela diz que todos devem ter um método pagar(), por exemplo, todas as classes associadas a esta interface terão que, obrigatoriamente, implementar este método.