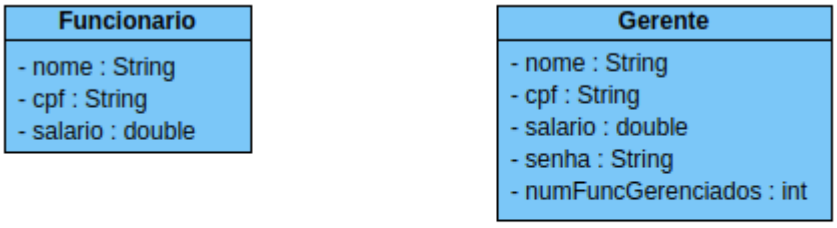
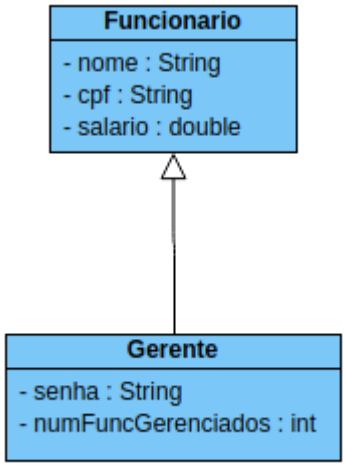
Herança

Herança é um relacionamento entre classes onde novas classes são criadas a partir de outras já existentes, absorvendo atributos e comportamentos e adicionando os seus próprios. A herança também pode ser chamada de generalização. De modo simplificado, herança é "uma classe (classe filha) que tem os mesmos atributos de outra (classe pai), mais alguns atributos distintos". Para facilitar o entendimento sobre herança vamos imaginar as seguintes classes, Funcionario e Gerente, onde o diagrama de classes das duas classes serão os seguintes:



Analisando o diagrama, podemos ver que ambas as classes tem os atributos nome, cpf e salario. Se pararmos para pensar podemos concluir que ambas tem alguns atributos iguais, porque o gerente também é um funcionário, mas com mais informações que são específicas do cargo. Nesse caso, se tivéssemos um outro tipo de funcionário, por exemplo o tipo diretor, que tem características diferentes do funcionário comum, precisaríamos criar uma outra classe e copiar os atributos nome, cpf e salario, para essa nova classe. Além disso, se um dia precisarmos adicionar uma nova informação para todos os funcionários, precisaremos passar por todas as classes de funcionários e adicionar esse atributo. Para resolvermos esses problemas que podem ocorrer, podemos utilizar a **herança**, onde o gerente irá herdar todas as características do funcionário e ao mesmo tempo terá suas próprias especificidades. A herança em UML é representada por uma flecha que sai da classe filha até a classe pai. O diagrama de classes ficaria da seguinte forma,após adicionarmos a herança:



Com isso, em todo momento que criarmos um objeto do tipo Gerente, este objeto possuirá também os atributos e métodos definidos na classe Funcionario, pois um Gerente **é um** Funcionario. Dizemos que a classe Gerente **herda** todos os atributos e métodos da classe pai, no nosso caso, a Funcionario. Porém, os atributos e métodos privados que são herdados, não podem ser acessados diretamente. Para acessar um membro privado na classe filha indiretamente, seria necessário que o pai expusesse um outro método visível que invocasse esse atributo ou método privado.

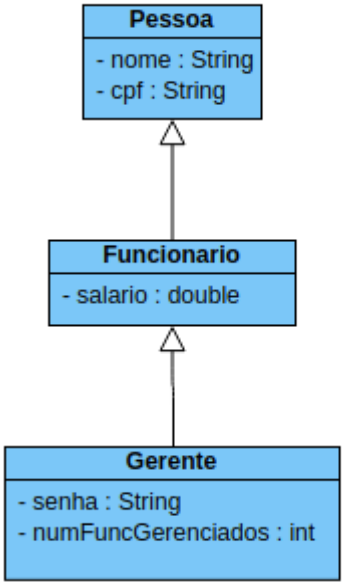
E se precisamos acessar os atributos que herdamos? Não gostaríamos de deixar os atributos de Funcionario, public, pois dessa maneira qualquer um poderia alterar os atributos dos objetos deste tipo. Existe um outro modificador de acesso, o **protected**, que fica entre o private e o public. Um atributo **protected** só pode ser acessado diretamente pela própria classe, por suas classes filhas, e pelas classes que se encontram no mesmo pacote. Na UML um atributo protected é representado por "# <nome do atributo>".

Propriedades da Herança

A herança tem duas propriedades são elas:

- **Transitividade**: uma classe em uma hierarquia herda propriedades e relacionamentos de todos os seus ancestrais, ou seja, a herança pode ser aplicada em vários níveis, dando origem a **hierarquia de generalização**.
- **Assimetria**: dadas duas classes A e B, se A for uma filha de B, então B não pode ser uma filha de A, ou seja, não pode haver ciclos em uma hierarquia de generalização.

Um exemplo das propriedades da herança pode ser o seguinte:



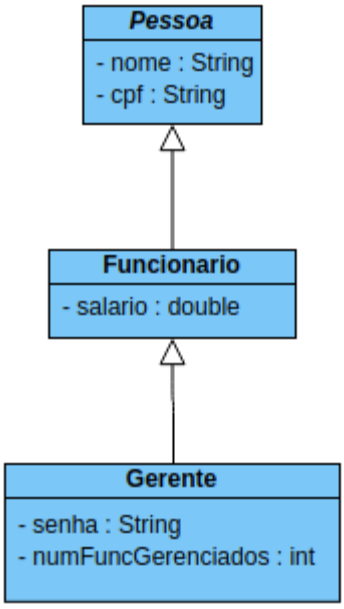
No exemplo, a classe Gerente herda a classe Funcionario que por sua vez herda a classe Pessoa, ou seja, indiretamente a classe Gerente também irá herdar a classe Pessoa

(**Transitividade**). Outra coisa que pode ser observado no exemplo, é que como a classe Gerente herda a classe Funcionario, e por isso a classe Funcionario não pode herdar a classe Gerente, isso também vale para a classe Pessoa, pois a mesma está em uma hierarquia mais alta que a classe Gerente (**Assimetria**).

Classes Abstratas

Usualmente, a existência de uma classe se justifica pelo fato de haver a possibilidade de criar objetos/instâncias da mesma, essas são as classes concretas. No entanto, podem existir classes que não geram instâncias diretas, essas são as **classes abstratas**.

Classes abstratas são utilizadas para organizar e simplificar uma hierarquia de generalização. Subclasses de uma classe abstrata também podem ser abstratas, mas a hierarquia deve terminar em uma ou mais classes concretas. Na UML, uma classe abstrata é representada com o seu nome em itálico. No exemplo a seguir, Pessoa é uma classe abstrata.



No caso do exemplo, como a classe Pessoa é uma classe abstrata, não poderá ser instanciado nenhum objeto deste tipo. Uma classe abstrata pode ter atributos e métodos. mas em relação aos métodos, uma classe abstrata pode possuir tanto **métodos abstratos** (métodos que necessitam de implementação nas classes filhas), quanto **métodos concretos** (ou seja, métodos que possuem implementação). Assim como a classe abstrata, os métodos abstratos são representados na UML com o nome em itálico.