Engenharia de Software

Prof. Alex Mateus Porn, Me.

Organização da Aula

- Referente ao conteúdo abordado nas Aulas 3, 4, 5 e 6 de Engenharia de Software, abordaremos assuntos complementares aos seguintes conteúdos:
  - Modelagem de software com:
    - Diagrama de casos de uso e diagrama de classes;
    - Relacionamento entre casos de uso;
    - Relacionamento entre classes e níveis de visibilidade.
  - Qualidade de Software e Estimativas de Software.

Modelagem de Software: Diagrama de Casos de Uso

Diagrama de Casos de Uso

- Casos de Uso;
- Atores;
- Relacionamentos.

Casos de Uso

- Os casos de uso são uma técnica para captar os requisitos funcionais de um sistema.
- Descrevem as interações típicas entre os usuários de um sistema e o próprio sistema, fornecendo uma narrativa sobre como o sistema é utilizado.

Casos de Uso

- Exemplo de um caso de uso:
  - Receber produtos devolvidos
- Em vez de descrever o caso de uso do início, pode ser tornar mais fácil começar descrevendo os cenários.

Cenários

- Um cenário é uma sequência específica de ações e interações entre atores (usuários) e o sistema;
- Também é chamado de instância de caso de uso;

Cenários

- Supostos cenários do caso de uso “Receber produtos devolvidos”:
  - Cenário de sucesso principal: um cliente chega ao caixa com os itens a serem devolvidos. O caixa usa o sistema PDV para registrar cada item devolvido.

Cenários

- Supostos cenários do caso de uso “Receber produtos devolvidos”:
  - Cenário alternativo 1: Se o cliente pagou com cartão de crédito e a transação de reembolso para estorno em sua conta de crédito é rejeitada, deve-se informar o cliente e o reembolsar com dinheiro.

Cenários

- Supostos cenários do caso de uso “Receber produtos devolvidos”:
  - Cenário alternativo 2: Se o identificador do item não for encontrado no sistema, este notifica o caixa e sugere que entre manualmente com o código do produto (talvez ele esteja corrompido).

Cenários

- Supostos cenários do caso de uso “Receber produtos devolvidos”:
  - Cenário alternativo 3: Se o sistema detecta uma falha para se comunicar com o sistema externo de contabilidade, a falha da operação deve ser registrada em log para recuperação tardia.

Casos de Uso

- Um caso de uso é uma coleção de cenários relacionados de sucesso e “fracasso” (alternativos), que descrevem um ator usando um sistema como meio para atingir um objetivo.

Atores

- Um ator é algo com comportamento, tal como uma pessoa (identificada por seu papel), um sistema de computador ou uma organização.
  - Por exemplo, um vendedor.

Exemplo de um caso de uso

Compra de um Produto

Cenário Principal de Sucesso:

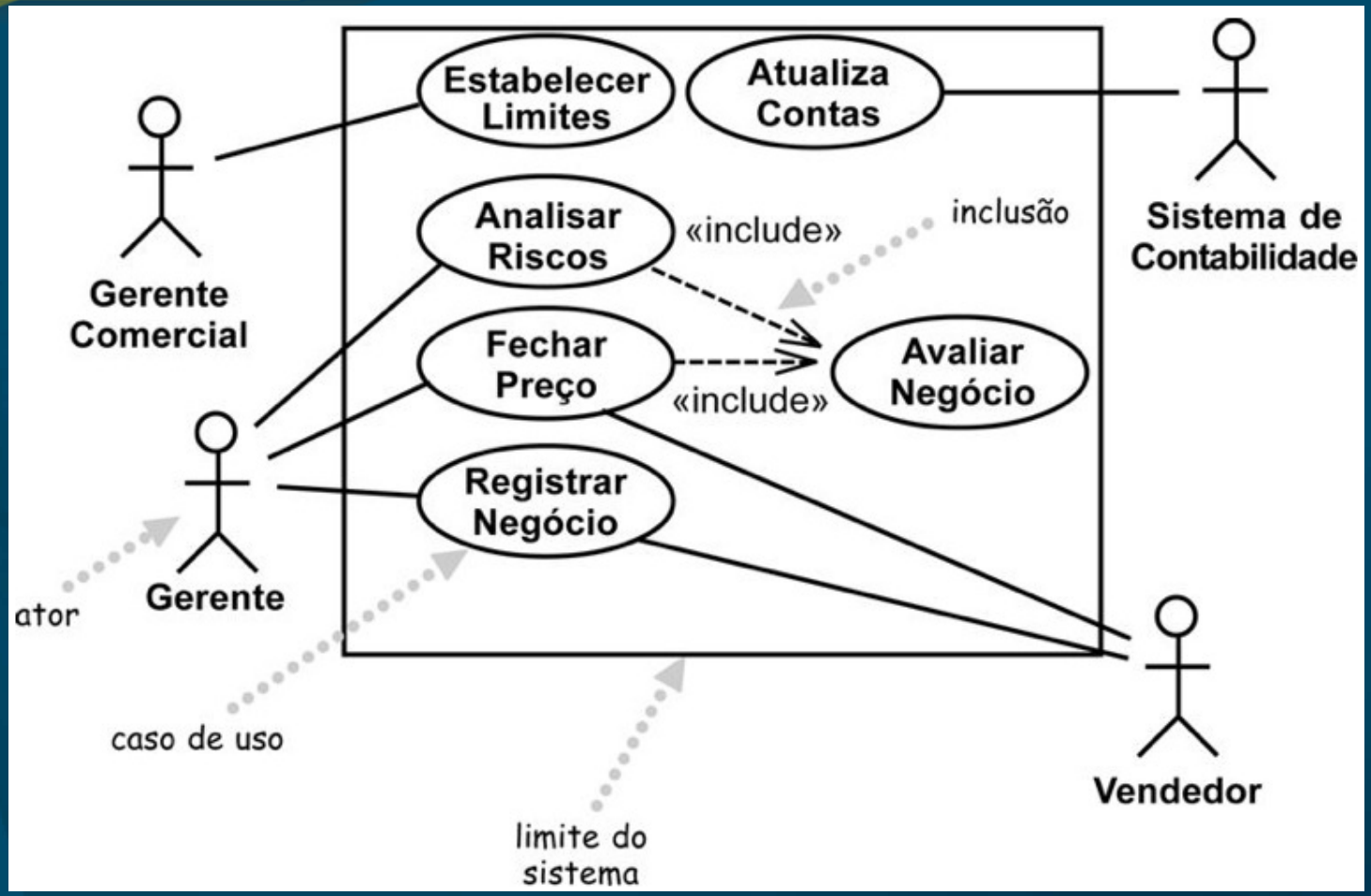
1. O cliente navega pelo catálogo e seleciona itens para comprar
2. O cliente vai para o caixa
3. O cliente preenche o formulário da remessa (endereço de entrega; opção de entrega imediata ou em três dias)
4. O sistema apresenta a informação completa do faturamento, incluindo a remessa
5. O cliente preenche a informação de cartão de crédito
6. O sistema autoriza a compra
7. O sistema confirma imediatamente a venda
8. O sistema envia uma confirmação para o cliente por *e-mail*

Extensões:

3a: Cliente regular
.1: O sistema mostra a informação atual da remessa, a informação de preço e a informação de cobrança
.2: O cliente pode aceitar ou escrever por cima desses padrões, retornando ao CPS, no passo 6

6a. O sistema falha na autorização da compra a crédito
.1: O cliente pode inserir novamente a informação do cartão de crédito ou cancelar

Exemplo de diagrama de casos de uso



Relacionamentos de casos de uso

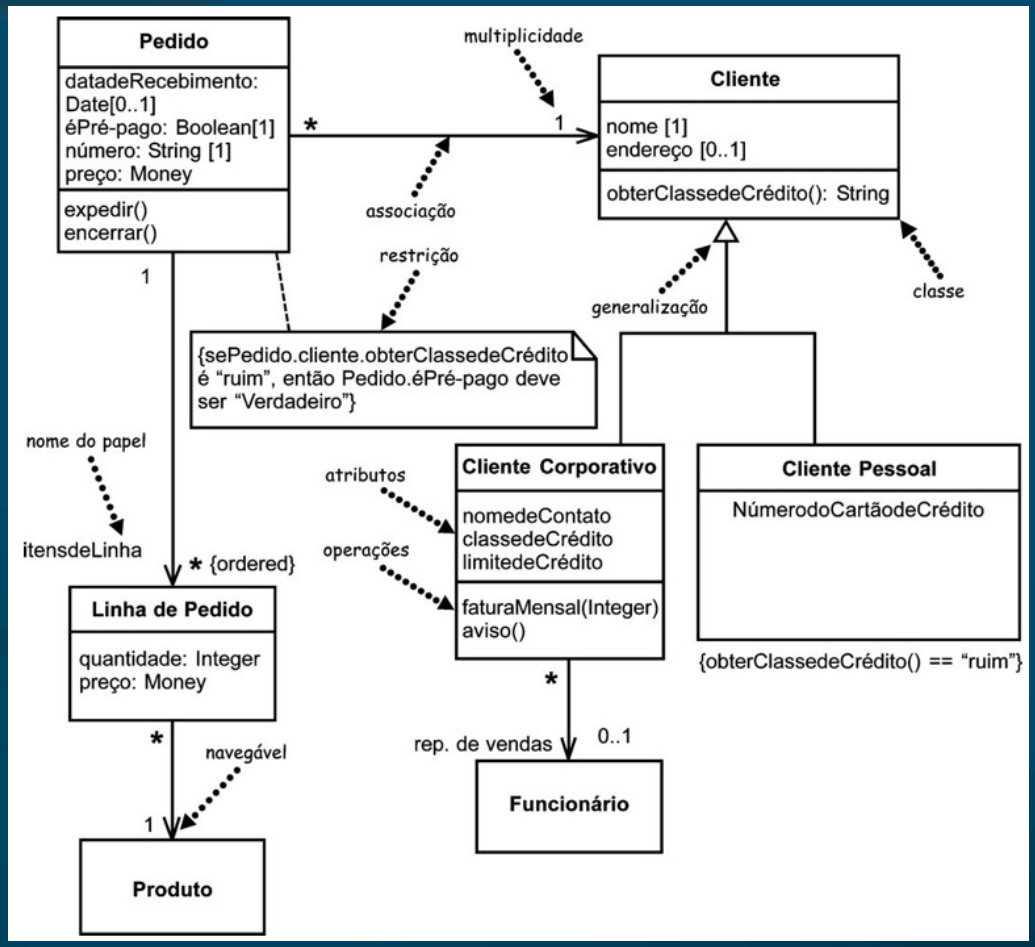
- Generalização;
- Include (Inclusão);
- Extend (Extensão).

Modelagem de Software: Diagrama de Classes

Diagrama de Classes

- Descrevem os tipos de objetos presentes no sistema e os vários tipos de relacionamentos estáticos existentes entre eles.
- Também apresentam:
  - Propriedades (atributos) de um objeto;
  - Operações (métodos) de uma classe;
  - Restrições que se aplicam ao modo como os objetos se conectam.

Diagrama de Classes



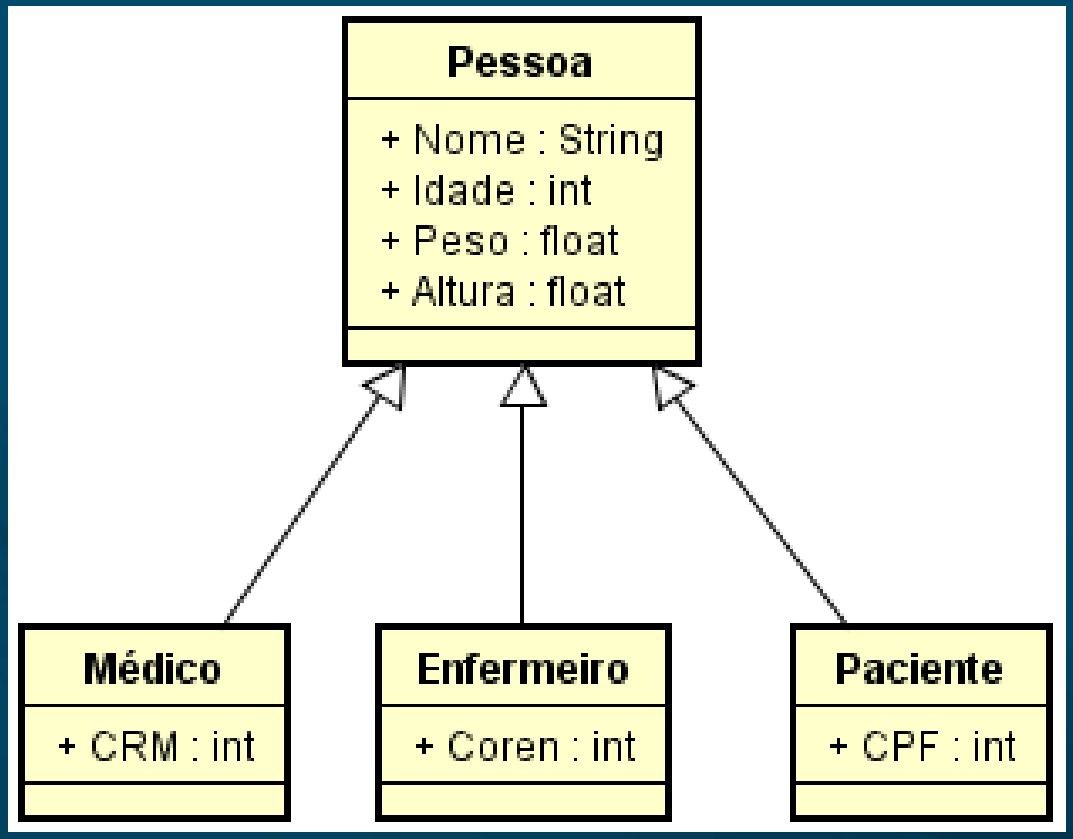
Relacionamento entre Classes

- Basicamente são três os principais relacionamentos entre classes:

- Generalização ou herança;
- Composição;
- Agregação.

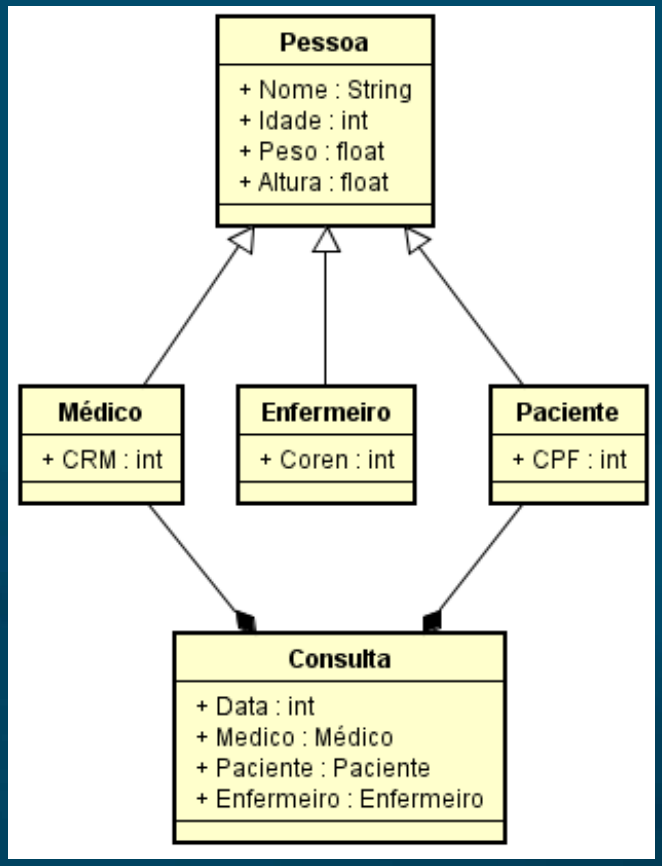
Relacionamento entre Classes

- Generalização ou herança



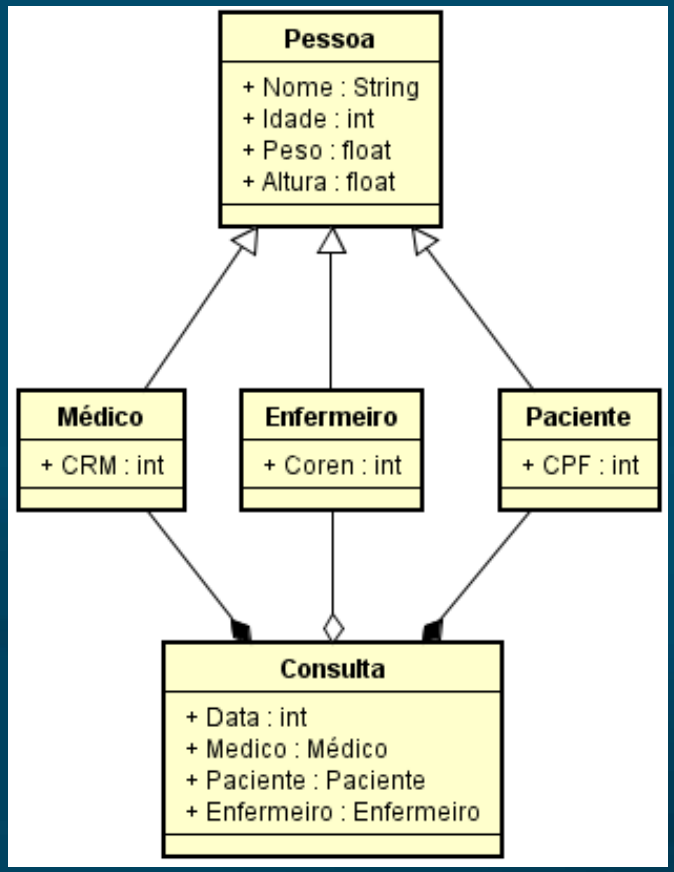
Relacionamento entre Classes

- Composição



Relacionamento entre Classes

- Agregação



Visibilidade de atributos e métodos

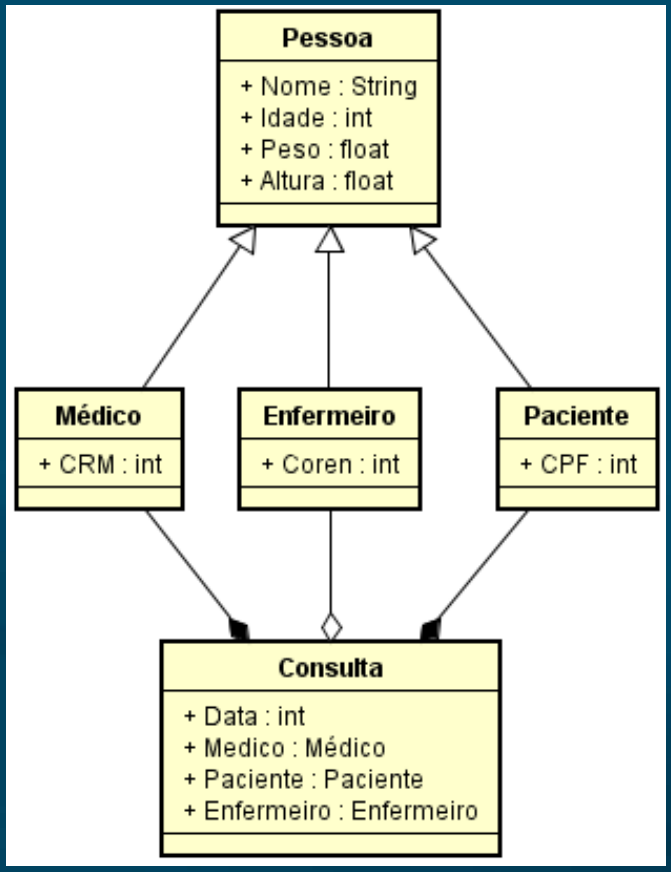
- Public (+)
- Private (-)
- Protected (#)

Visibilidade de atributos e métodos

- Public (+)
  - Atributos de uma classe definidos como public são acessíveis onde quer que o programa tenha uma referência a um objeto dessa classe ou uma de suas subclasses.

Visibilidade de atributos e métodos

- Public (+)

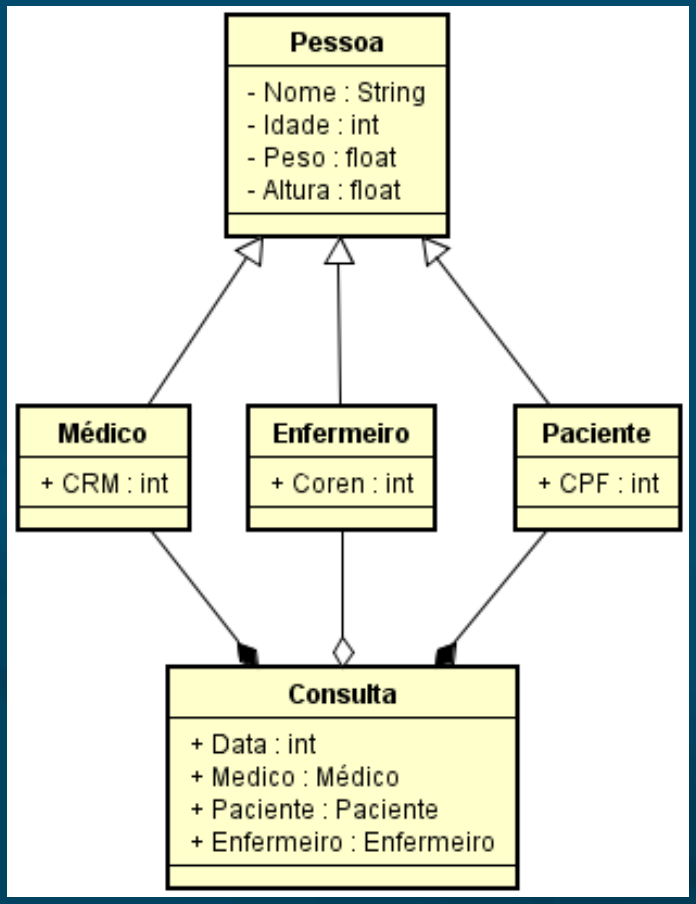


Visibilidade de atributos e métodos

- Private (-)
  - Atributos de uma classe definidos como private somente são acessíveis dentro da própria classe.

Visibilidade de atributos e métodos

- Private (-)

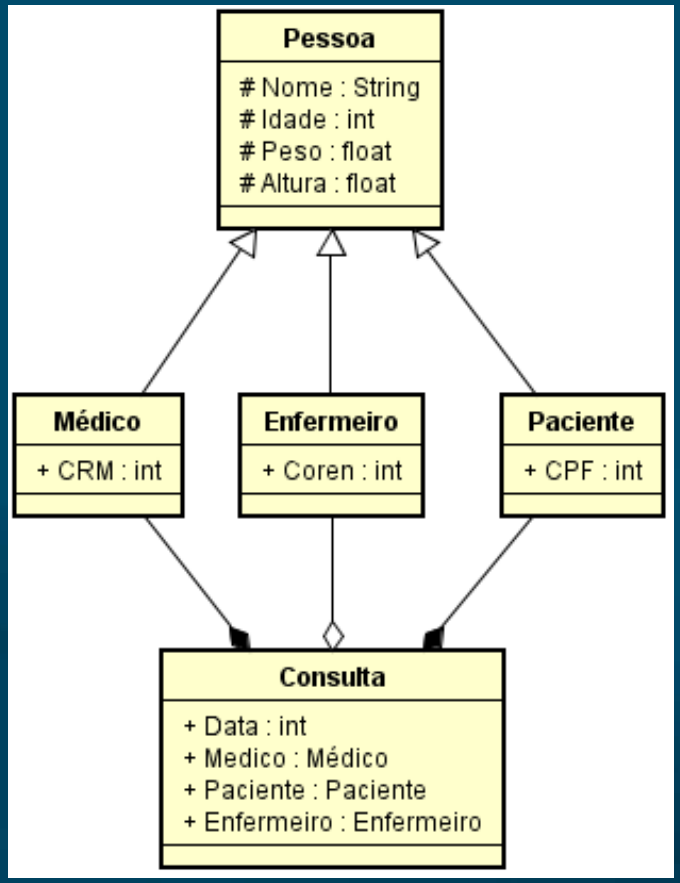


Visibilidade de atributos e métodos

- Protected (#)
  - Oferece um nível intermediário de acesso entre public e private;
  - Atributos de uma classe definidos como protected são acessíveis dentro da própria classe ou por suas subclasses.

Visibilidade de atributos e métodos

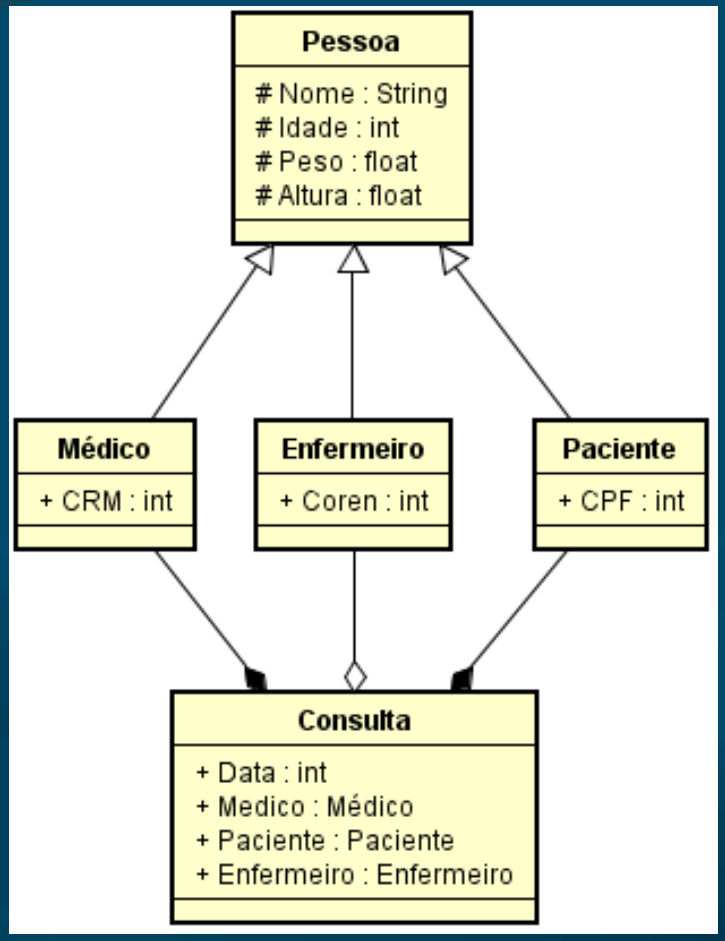
- Protected (#)



Implementação Prática

Implementação prática

- Java
- IDE Eclipse



Qualidade de Software

Qualidade de Software

- Qualidade de desempenho;
- Qualidade de recursos;
- Qualidade de confiabilidade;
- Qualidade de conformidade;
- Qualidade de durabilidade;
- Facilidade de manutenção;
- Qualidade da estética;
- Qualidade de percepção.

Estimativas de Software

Estimativas de Software

- Linhas de Código;
- Análise de Pontos por Função;
- Cocomo II.

Referências

- FOWLER, Martin. UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos. 3ª ed. São Paulo: Pearson, 2004.
- KRUCHTEN, Philippe, Utilizando UML e padrões: Uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3ª ed. Porto Alegre: Bookman, 2007.
- DEITEL, Harvey M. Java como Programar, 8ª ed. São Paulo: Pearson, 2010