UML - Diagrama de Classes e objetos

Resolvi falar um pouco sobre diagrama de classes e de objetos.. Não é uma tarefa trivial para ser desenvolvida em um pequeno artigo onde deve-se ter a preocupação de ser claro , objetivo sem ser superficial.

Eu poderia começar dando de cara a definição do que é um diagrama de classes , mas creio que preciso falar sobre o conceito de classes , mas para isto preciso falar sobre o que são objetos...

Estou falando para programadores e analistas , certo !!! Então nosso foco e área de atuação será a **Programação Orientada a Objetos**. (POO)

Em POO , os problemas de programação são pensados em termos de objetos , nada de funções , rotinas , nada disto , o assunto são os objetos , propriedades e métodos.

Nota: A preocupação da programação estruturada estava em procurar os processos que envolviam o problema e não os objetos que o compunham.

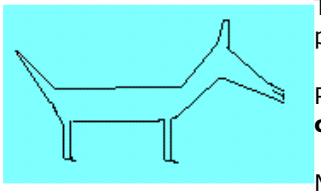
Desta forma quando é colocado o problema de desenvolver um sistema para locadoras , por exemplo , devemos pensar como dividir o problema em objetos. Para este caso podemos ter os seguintes objetos : Clientes , CDs e Fitas , etc..

A melhor maneira de conceituar estes termos é considerar um objeto do mundo real e mostrar como podemos representá-lo em termos conceitos para POO.

Começando com as definições : "Um objeto é um termo que usamos para representar uma entidade do mundo real" (Fazemos isto através de um exercício de abstração.)

Vou usar como exemplo o meu cachorro Bilu. Posso descrever o Bilu em termos de seus atributos físicos: é pequeno , sua cor principal é castanha , olhos pretos , orelhas pequenas e caídas, rabo pequeno , patas brancas.

Posso também descrever algumas ações que ele faz (temos aqui os métodos) : balança o rabo quando chego em casa , foge e se deita se o mando sair debaixo da mesa, late quando ouve um barulho ou vê um cão ou gato, atende e corre quando o chamo pelo seu nome. Temos abaixo a representação do Bilu.



Meu cachorro Bilu

Temos aqui a representação de um objeto , no caso o meu cachorro Bilu , que possui as seguintes propriedades e métodos:

Propriedades : **Cor do corpo : castanha cor dos olhos : preto altura: 18 cm comprimento: 38 cm largura : 24 cm**

Métodos : **balançar o rabo , latir , deitar , sentar**

Em termos de POO para poder tratar os objetos começamos criando classes , neste caso irei criar a classe chamada **Cachorro**.

"Uma classe representa um conjunto de objetos que possuem comportamentos e características comuns".

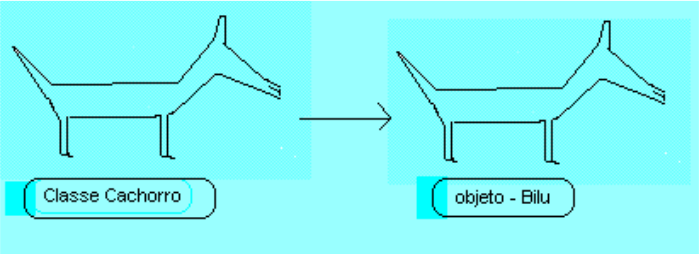
"Na UML o nome de uma classe é um texto contendo letras e dígitos e algumas marcas de pontuação. Na realidade, é melhor guardar os nomes curtos com apenas letras e dígitos. UML sugere capitalizar todas as primeiras letras de cada palavra no nome (ex.: ``Lugar'', ``DataReserva''). É melhor também manter nomes de classe no singular, classes por default ``contem'' mais de um objeto, o plural é implícito.". [Nicolas Anquetil]

Uma classe descreve como certos tipos de objetos se parecem do ponto de vista da programação , pois quando definimos uma classe precisamos definir duas coisas:

1. **Propriedades -** Informações específicas relacionadas a uma classe de objeto. São as características dos objetos que as classes representam. *Ex Cor , altura , tamanho , largura , etc...*
2. **Métodos**: São ações que os objetos de uma classe podem realizar. *Ex: Latir , correr , sentar , comer, etc.*

Você pode pensar em uma classe com um modelo para criar quantos objetos você desejar de um tipo particular. Pense em um carimbo com a imagem de um cachorro , quando você carimba e obtêm um desenho de cachorro

você acabou de criar uma instância da classe e obteve um objeto daquela classe. O novo objeto possuirá todas as características e comportamentos definidos pela classe.
(*As classes especificam a estrutura e o comportamento (operações) dos objetos, que são instâncias das classes*)



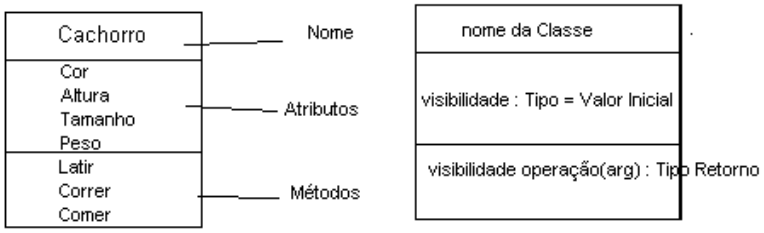
Aqui temos que Bilu é um objeto da classe Cachorro. Em termos de POO acabamos de criar uma instância da classe Cachorro e a chamamos Bilu.

Quando criamos uma nova instância de uma classe dizemos que estamos instanciando a classe.

Geralmente em um sistema de médio porte serão identificados diversas classes que compõem o sistema. Neste contexto a UML surgiu como uma proposta de ser uma linguagem para modelagem de dados que usava diversos artefatos para representar o modelo de negócio ; um destes artefatos é o **diagrama de classes**.

A representação de uma classe usa um retângulo dividido em três partes:

- nome
- atributos
- métodos



Podemos dizer que os diagramas de classes são os principais diagramas estruturais da UML pois ilustram as classes , interfaces e relacionamentos entre elas.

Os diagrama se classes ilustram atributos e operações de uma classe e as restrições como que os objetos podem ser conectados ; descrevem também os tipos de objetos no sistema e os relacionamentos entre estes objetos que podem ser : **associações e abstrações**.

Para poder representar a visibilidade dos atributos e operações em uma classe utiliza-se as seguintes marcas e significados:

- **+ público - visível em qualquer classe**
- **# protegido - qualquer descendente pode usar**
- **- privado - visível somente dentro da classe**

Relacionamento entre classes

Os objetos tem relações entre eles: um professor *ministra* uma disciplina *para* alunos *numa* sala, um cliente *faz* uma reserva *de* alguns lugares *para* uma data, etc. Essas relações são representadas também no diagrama de classe. [Nicolas Anquetil]

A UML reconhece três tipos mais importantes de relações: dependência, associação e generalização (ou herança).

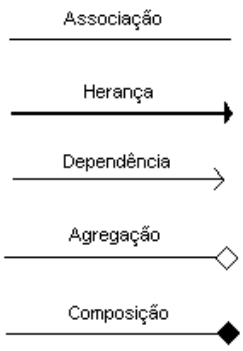
Geralmente as classes não estão sós e se relacionam entre si. O relacionamento e a comunicação entre as classes definem responsabilidades , temos 3 tipos :

1. **Associações : Agregação e composição**
2. **Generalização (herança)**
3. **Dependências**

As representações usam a seguinte notação :

- **Associação** : São relacionamentos estruturais entre instâncias e especificam que objetos de uma classe estão ligados a objetos de outras classes. Podemos ter associação uniária , binária , etc.

  A associação pode existir entre classes ou entre objetos. Uma associação



entre a classe Professor e a classe disciplina (um professor ministra uma disciplina) significa que uma instância de Professor (um professor específico) vai ter uma associação com uma instância de Disciplina. Esta relação significa que as instâncias das classes são conectadas, seja fisicamente ou conceitualmente.[Nicolas Anquetil]

- **Dependência** - São relacionamentos de utilização no qual uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente. A dependência entre classes indica que os objetos de uma classe usam serviços dos objetos de outra classe.

- **Generalização** (herança : simples ou composta) - Relacionamento entre um elemento mais geral e um mais específico. Onde o elemento mais específico herda as propriedades e métodos do elemento mais geral. A relação de generalização também é conhecida como herança no modelo a objetos. Como a relação de dependência, ela existe só entre as classes. Um objeto particular não é um caso geral de um outro objeto, só conceitos (classes no modelo a objetos) são generalização de outros conceitos.

- **Agregação Regular** - tipo de associação (é parte de , todo/parte) onde o objeto parte é um atributo do todo ; onde os objetos partes somente são criados se o todo ao qual estão agregados seja criado. Pedidos é composto por itens de pedidos.

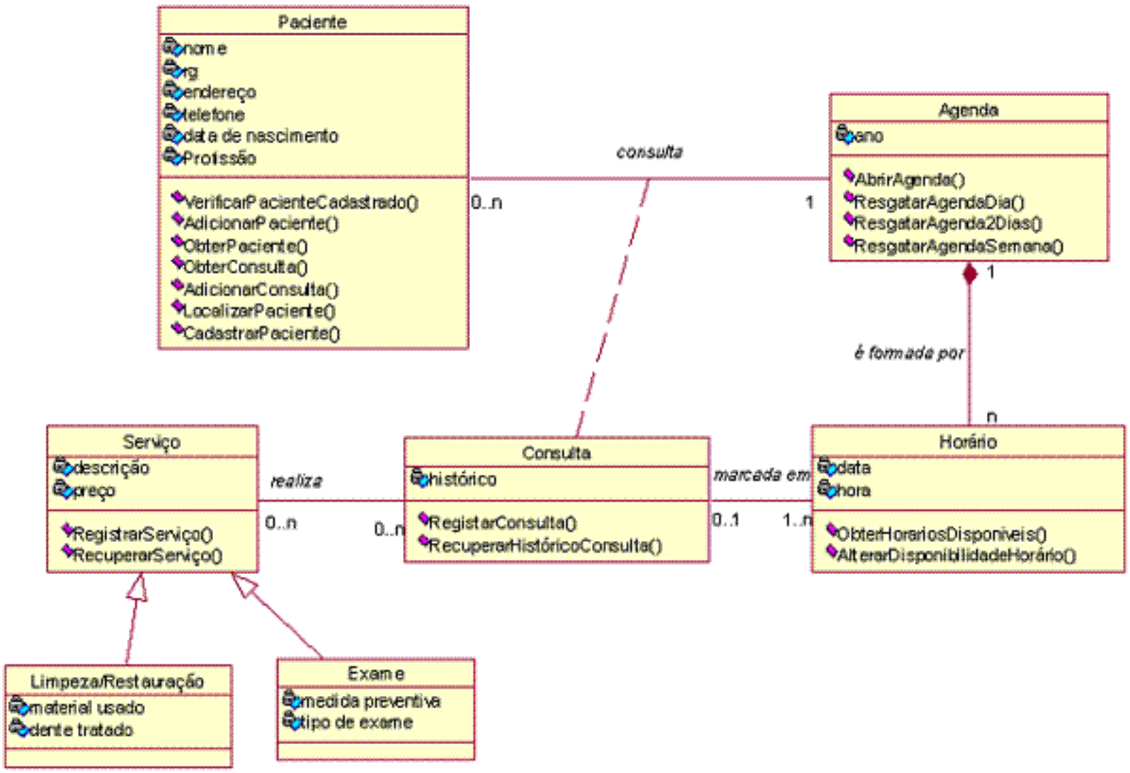
- **Composição** - Relacionamento entre um elemento (o todo) e outros elementos (as partes) onde as parte só podem pertencer ao todo e são criadas e destruídas com ele.

O diagrama de de classes lista todos os conceitos do domínio que serão implementados no sistema e as relações entre os conceitos. Ele é muito importante pois define a estrutura do sistema a desenvolver.

O diagrama de classes não surge do nada ele é consequência do prévio levantamento de requisitos , definição de casos de usos e classes. Como exemplo vamos supor que você tivesse que desenvolver um sistema para automatizar um consultório dentário. As etapas básicas envolvidas seriam:

- Levantamento e análise de requisitos do sistema a ser desenvolvido. Entrevista com o dentista(s) e com as pessoas que trabalham no consultório
- Definição dos objetos do sistema : Paciente , agenda , dentista , serviço , contrato , consulta , pagamento , etc..
- Definição dos atores do sistema : paciente, dentista , secretária
- Definição e detalhamento dos casos de uso: marcar consulta , confirmar consulta , cadastrar paciente , cadastrar serviços , etc.
- Definição das classes : paciente , dentista , exame , agenda , serviço
- Definir os atributos e métodos das classes :

Após toda esta análise você chega no diagrama de classes do sistema (representado abaixo a título de exemplo ilustrativo)



Atributo

Um atributo representa uma propriedade que todos os objetos da classe têm (por exemplo, todos os cachorros tem pelo , orelhas , altura, ,etc. Mas cada objeto terá valores particulares para seus atributos (alguns cachorros são mais baixos , outros são maiores, etc.).

Uma classe pode ter qualquer número de atributos. Na UML, o nome de um atributo é um texto contendo letras e dígitos e algumas marcas de pontuação. UML sugere de capitalizar todas as primeiras letras de cada palavra no nome menos a primeira palavra (ex.: "nome'', "nomeCachorro'').

Num modelo, os atributos devem ser de um tipo simples (inteiro, texto, talvez data), não podem conter outros objetos.

Métodos

Métodos são ações que implementam uma operação. Uma classe pode ter qualquer número de métodos e dois métodos em duas classes podem ter o mesmo nome.

Todos os métodos que vão implementar a operação tem que respeitar exatamente a assinatura dela (mesmo nome, mesmo número de atributo, com os mesmo tipos e o mesmo ordem). Um método não pode acrescentar ou cortar um parâmetro. Isso seria um violação do polimorfismo. Para mandar a mensagem corretamente, teríamos que saber qual é a classe do objeto (cada classe tendo método com assinatura diferente). O que é possível, no caso de cortar um parâmetro, é simplesmente ignorá-lo na implementação. *[Nicolas Anquetil]*

Voltarei falar sobre UML nos próximos artigos onde iremos abordar os seguintes diagramas UML:

Seqüência : Mostra interações entre vários componentes do sistema enfocando a sequência.
Colaboração : Mostra interações entre vários componentes do sistema enfocando a colaboração entre as classes.
Atividade : Mostra como um sub-sistema ou um objeto realizam uma operação.
Estados : Mostra como um sub-sistema ou um objeto realizam uma operação.
Componentes : Mostra a organização dos componentes. Trata da implementação do sistema.
Implantação : Mostra a configuração física sobre qual o sistema será instalado.

E estamos conversados....

Veja também a seção de UML-XML em :	

Para saber mais sobre o assunto veja também os seguintes artigos:

UML - Apresentando os principais diagramas da linguagem
UML - Modelando sistemas - Casos de Uso
UML - Conceitos Básicos
UML - Unified Modeling Language e Visual Modeler.

José Carlos Macoratti