FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Projeto de Software – Aula 3 – 2º SEMESTRE/2018

Análise, Projeto e Desenvolvimento de Sistemas de Software – Objetos e Classes

---

Objetivos:

- Conhecer a representação do diagrama de classes
- Compreender o conceito de visibilidade
- Compreender conceitos de objetos, classes, ligações e associações

---

Conceitos de Objetos e Classes

Neste texto discutiremos conceitos básicos de modelagem de classes que serão usados ao longo do semestre, definimos cada conceito, apresentando a notação gráfica de representação de objeto e classe correspondente com exemplos, alguns conceitos importantes são objetos, classe, ligação entre objetos e associação entre classes.

Objeto : Uma abstração de alguma coisa em um domínio de problema.

Um objeto é um conceito, abstração ou coisa com identidade que possui significado para uma aplicação. Os objetos normalmente parecem como nomes próprios ou referencias especificas nas descrições de problemas e discussões com os usuários e clientes durante o processo de desenvolvimento de sistemas de software.

Todos os objetos possuem identidade e são distinguíveis. Duas maças com a mesma cor, forma e textura ainda são maças individuais; uma pessoa pode comer uma e depois comer a outra. O termo **identidade** significa que os objetos são inerentemente diferenciáveis por sua existência e não por propriedades descritivas que eles possam ter.

Classe : Uma descrição de um ou mais Objetos por meio de um conjunto uniforme de características e comportamentos.

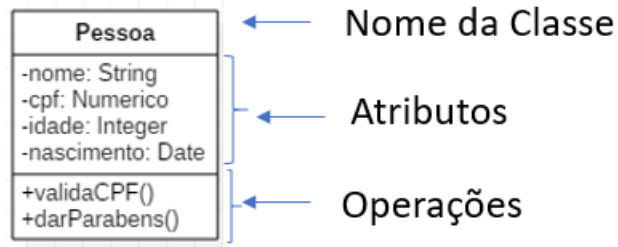
Um objeto é uma *Instância, ou ocorrência* de uma classe. Uma classe descreve um grupo de objetos com as mesmas características ou propriedades (atributos), comportamento (operações), tipos de relacionamento e semântica.
Pessoa, empresa, cargo, são todos classes. Cada pessoa possui nome e data de nascimento pode trabalhar em um cargo de uma empresa. João da Silva é vendedor da Loja de Material de

Construção Campeão S.A. e trabalha junto com outros dois vendedores, Rafaela Gonçalves e Marcos Santos. O Gerente da Loja de Material de Construção Campeão S.A. é José da Silva. Jpão da Silva, Rafaela Gonçalves, Marcos Santos e José da Silva são objetos que pertencem a mesma classe Pessoa. Já Vendedor e Gerentes são objetos que pertencem a classe Cargo, e finalmente, Campeão S.A. é objeto da classe Empresa. As classes normalmente aparecem como nomes comuns e frases nominais nas descrições do problema e em discussões com usuários.

Os objetos de uma classe possuem os mesmos atributos e comportamentos, a individualidade pode ser alcançadas a partir das diferenças em seus valores de atributos e relacionamentos específicos com os outros objetos, compartilhando uma mesma finalidade semântica acima e além do requisito de terem atributos e comportamento comuns. Os objetos da classe Pessoa possuem características em comum, como **nome, sexo, cpf, nascimento** e **idade**.

Cada objeto "conhece" sua classe. A maioria das linguagens de programação pode determinar a classe de um objeto durante a execução. A classe de um objeto é uma propriedade implícita do objeto.

Representação de uma Classe

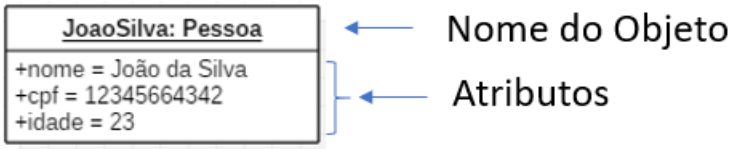


Diagramas de Classes

Oferecem uma notação gráfica para modelar classes e seus relacionamentos, descrevendo assim possíveis objetos. Os diagramas de classes são úteis para a modelagem abstrata e para o projeto de programas reais. Eles são concisos, fáceis de entender e funcionam bem na prática, usamos diagramas de classes para representar a estrutura das aplicações.

Ocasionalmente, também usaremos diagramas de objetos. Um ***diagrama de objetos*** mostra objetos individuais e seus relacionamentos, são úteis para documentar casos de teste e discutir exemplos, um diagrama de classes corresponde a um conjunto infinito de diagramas de objetos.

Representação de Objeto

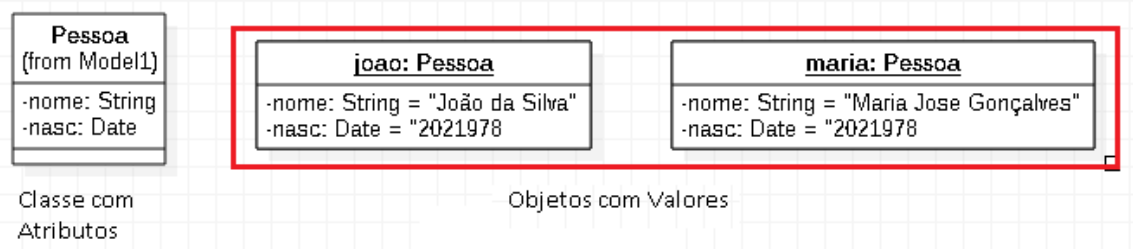


Valores e Atributos

Atributo: é um dado (informação de estado) para o qual cada Objeto em uma Classe tem seu próprio valor.

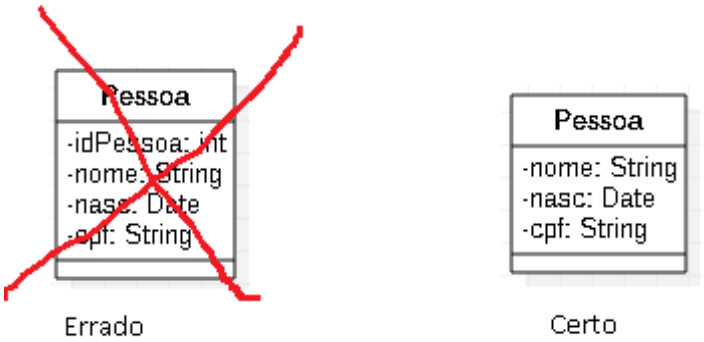
Um ***valor*** é um elemento dos dados. Você pode achar valores através do exame da documentação do problema. Um ***atributo*** é uma propriedade nomeada de uma classe, que descreve um valor retido por cada objeto da classe. Pode-se encontrar atributos procurando adjetivos ou abstraindo valores típicos.

Analogia: objeto está para classe assim como valor está para atributo.



Exemplo de atributos e valores. Atributos elaboram as classes.

Alguns meios de implementação exigem que um objeto tenha um identificador exclusivo. Esses identificadores são implícitos em um modelo de classes – você não precisa e não deve listá-los explicitamente.



Identificadores de objeto (idPessoa). Não liste identificadores pois eles são implícitos nos modelos.

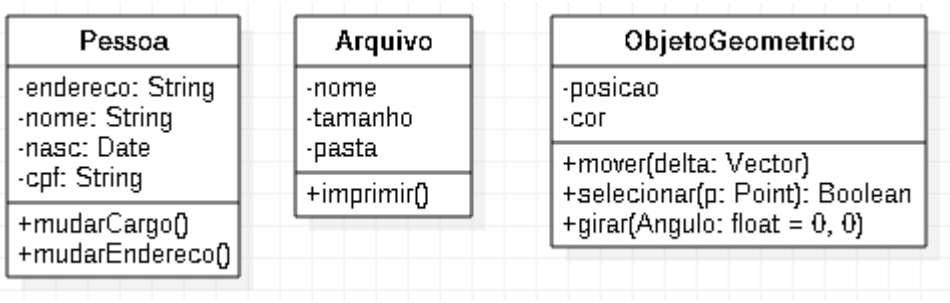
Os identificadores são um artefato computacional e não possuem significado intrínseco, os identificadores internos são uma conveniência de implementação, ao contrário do número do contribuinte, número da placa e número do telefone, pois possuem significado no mundo real.

Operações e Métodos

Uma ***operação*** é uma função ou um procedimento que pode ser aplicado a/ou por objetos em uma classe. *Contratar, demitir e pagarDividendos* são operações da classe *Empresa* por exemplo. Cada operação possui um objeto destino como um argumento implícito. O comportamento da operação depende da classe de seu destino. Um objeto "conhece" sua classe e portanto, a implementação correta da operação.

A mesma operação pode ser aplicada a muitas classes diferentes. Tal operação e ***polimórfica***, ou seja, a mesma operação assume diferentes formas em diferentes classes. Um ***método*** é a

implementação de uma operação na classe. Uma operação pode ter argumentos além do seu objeto destino, esses argumentos podem ser valores ou outros objetos. A escolha de um método depende inteiramente da classe do objeto destino e não de quaisquer argumentos objeto que uma operação possa ter.

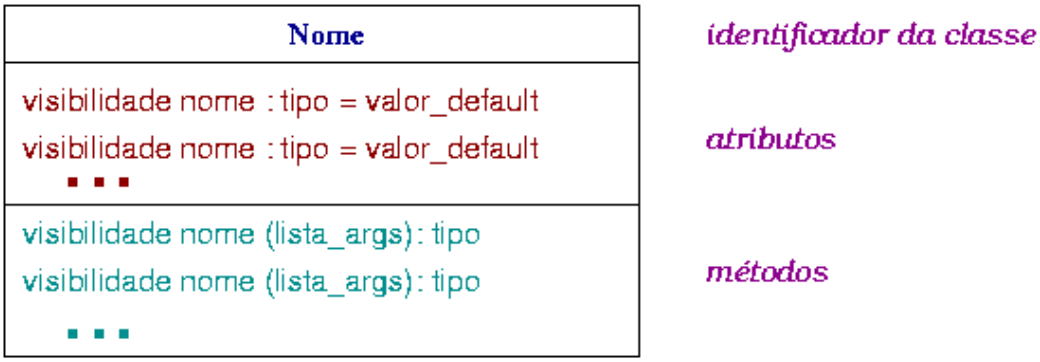


Na figura acima a classe *Pessoa* possui os atributos *nome* e *dataNascimento* e as operações *mudarCargo* e *mudarEndereço* que são características de *Pessoa.* ***Característica*** é uma palavra genérica para um atributo ou operação.

A notação UML lista operações no terceiro compartimento da caixa de classe. Nossa convenção é listar o nome da operação em fonte regular, alinhado à esquerda na caixa e com a primeira letra maiúscula. Os detalhes opcionais como uma lista de argumentos e tipo de resultado, podem seguir cada nome de operação. Parênteses delimitam uma lista de argumentos, separados por vírgulas, um sinal de dois-pontos precede o tipo do resultado. Uma lista de argumentos vazia entre parênteses mostra explicitamente que não existem argumentos; caso contrário, você não poderá concluir nada.

Resumo da Notação de Classes

Uma caixa representa uma classe e pode ter até três compartimento, de cima para baixo: nome de classe, lista de atributos e lista de operações, detalhes opcionais como tipo e valor-padrão podem seguir cada nome de atributo e na lista de operações, a lista de argumentos e tipos de resultado pode vir acompanhando cada nome de operação.



- Aprender sobre visibilidade e associações entre classes
- Aprender dois novos tipos de associações: agregação e composição
- Praticar um método para construir diagrama de classes

Visibilidade

A visibilidade refere-se a capacidade de um método referenciar uma característica de outra classe, e tem os seguintes valões possíveis: público, protegido, privado e pacote.

É importante ressaltar que, o significado exato depende da linguagem de programação.

Por exemplo:

Qualificadores	Java	C++
private	O método ou atributo pode ser acessado somente dentro da própria classe.	O método ou atributo pode ser acessado somente dentro da própria classe.
public	O método ou atributo pode ser acessado externamente por outro código.	O método ou atributo pode ser acessado externamente por outro código.
protected	O método ou atributo pode ser acessado pela própria classe ou por classes-filhas (aquelas que herdam desta classe).	O método ou atributo pode ser acessado pela própria classe ou por classes-filhas (aquelas que herdam desta classe).
package (sem modificador)	O método ou atributo pode ser acessado pela própria classe ou por classes que participem do mesmo pacote.	Não existe em C++.

Visibilidade	Sintaxe Java	Sintaxe UML
public	public	+
protected	protected	#
package		~
private	private	-

Devem-se considerar várias questões quando se escolhe a visibilidade:

- Compreensão: Você precisa entender todas as características públicas para entender as capacidades de uma classe, ao

contrário, você pode ignorar características privadas, protegidas e de pacote, uma vez que elas são simplesmente uma conveniência de implementação.

- Capacidade de extensão: muitas classes podem depender de métodos públicos, de modo que pode ser altamente prejudicial mudar sua assinatura (número de argumentos, tipos de argumentos, tipo de valor de retorno). Com menos classes dependem de métodos privados, protegidos e de pacotes, há mais espaço para alterações.
- Contexto: Métodos privados, protegidos e de pacote podem se basear em precondições ou informações de estado criadas por outros métodos na classe. Aplicado fora do contexto, um método privado pode calcular resultados incorretos ou causar falha do objeto.

A visibilidade é um conceito muito importante. As mensagens entre objetos só podem ocorrer se o receptor for visível ao remetente.
Há quatro formas básicas de visibilidade

- Visibilidade de atributos: Para que o Objeto A possa enviar uma mensagem a Objeto B, usa-se um atributo em Objeto A.
- Visibilidade de parâmetros: Um objeto tem visibilidade para outro se este for recebido como parâmetro.
- Visibilidade declarada localmente: Um objeto é visível a partir de uma variável local de um método. Pode inicializar a variável local com um novo objeto, receber como valor de retorno de alguma chamada, etc.
- Visibilidade global: Um objeto global é visível a todos, não uma boa forma de ter visibilidade.

A visibilidade em UML é apresentada através do uso de Stereotypes, normalmente, não é usada em diagramas UML.

Ligação e Associação

Conceito de Ligação e Associação

São o meio de estabelecer relacionamento entre objetos e classes.

Ligações e Associações

Uma ***ligação*** é uma conexão física ou conceitual entre objetos. Por exemplo, João Silva *TrabalhPara* a empresa Simplex.

Uma associação é uma descrição de um grupo de ligações com estrutura e semântica comuns. Por exemplo, uma pessoa *TrabalhaPara* uma empresa. As ligações de uma associação conectam objetos das mesmas classes. Uma associação descreve um conjunto de ligações em potencial da mesma maneira que uma classe descreve um conjunto de objetos em potencial. Ligações e associações normalmente aparecem como verbos em relatos de problemas.

Diagrama de Classes

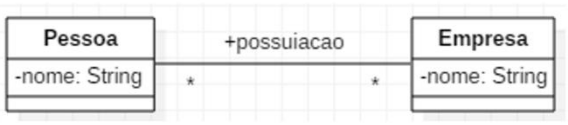
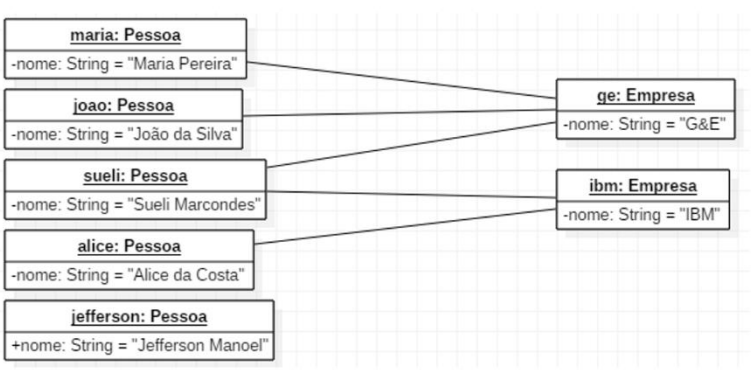


Diagrama de Objetos



Associação muitos-para-muitos. Uma associação descreve um conjunto de ligações em potencial da mesma maneira que uma classe descreve um conjunto de objetos em potencial.

A notação UML para uma ligação é uma linha entre os objetos; uma linha pode consistir em vários segmentos de linha. Se a ligação tiver um nome, ele estará sublinhado. Por exemplo, João possui ações da empresa GE. Uma associação conecta as classes relacionadas e também é indicada por uma linha. Por exemplo, as pessoas possuem ações das empresas, nossa convenção é mostrar os nomes das ligações e associações em itálico e confinar os segmentos de linha a uma grade retilínea. É bom arrumar as classes em uma associação para se ler da esquerda para a direita. O nome da associação é opcional, se o modelo não é ambíguo.

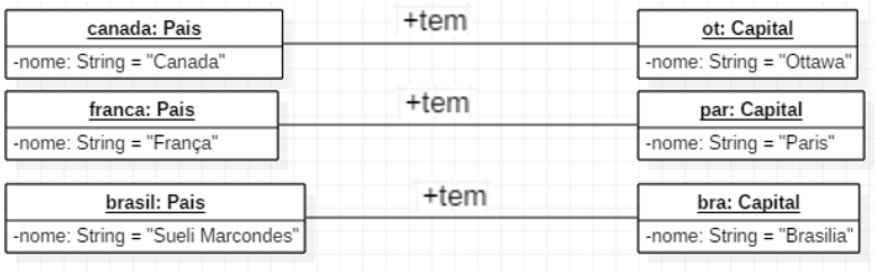
Os desenvolvedores normalmente implementam associações em linguagens de programação como referências de um objeto para outro. Uma referência é um atributo em um objeto que se refere a outro objeto. Por exemplo uma estrutura de dados para ***Pessoa*** poderia conter um atributo ***empregador***, que se refere a um objeto ***Empresa*** e um objeto ***Empresa*** poderia conter um atributo ***empregados***, que se refere a um conjunto de objetos ***Pessoa.***

Multiplicidade

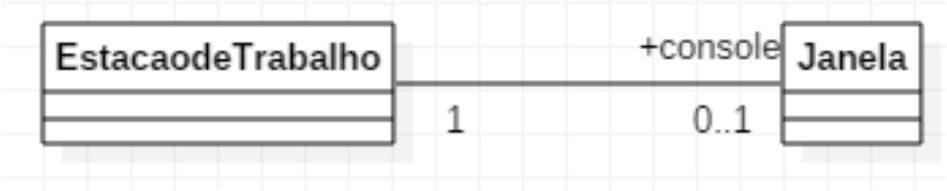
A *Multiplicidade* especifica o número de instâncias de uma classe que podem se relacionar a uma única instância de uma classe associada. A multiplicidade restringe o número de objetos relacionados. A literatura normalmente descreve a multiplicidade como sendo "um" ou "muitos", mas geralmente ela é um subconjunto de inteiros não negativos finitos.

Diagrama de Classes

Diagrama de Objetos



A figura acima mostra uma associação de um-para-um e algumas ligações correspondentes. Cada país possui uma capital. Uma Capital administra um País. A multiplicidade especifica o número de instâncias de uma classe que podem se relacionar a uma única instância de uma classe associada.



Multiplicidade zero-ou-um. Uma estação de trabalho pode ter uma de suas janelas designadas como console para receber mensagens de erro gerais. Porém é possível que não exista uma janela de controle. (A palavra "console" no diagrama é um nome de extremidade da associação, que veremos mais tarde).

Não confunda “multiplicidade” com “cardinalidade”, a primeira é um restrição no tamanho da coleção, a segunda é uma contagem de elementos que estão realmente em uma coleção. Portanto multiplicidade é uma restrição sobre a cardinalidade.

Uma multiplicidade “muitos” especifica que um objeto pode estar associado a vários objetos. Porém para cada associação, existe no máximo uma ligação entre determinado par de objetos, outras formas veremos nas próximas aulas.

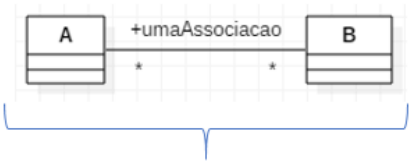


Diagrama de Classes

Diagrama de Objetos

Associação versus ligação. Um par de objetos pode ser instanciado no Maximo uma vez por associação *(exceto para bags e sequencias, que veremos mais a frente)*

A multiplicidade depende das suposições e de como você define os limites de um problema. Requisitos vagos normalmente tronam a multiplicidade incerta. Não se preocupe excessivamente com a multiplicidade cedo no desenvolvimento do software. Em primeiro lugar determine classes e associações, depois decida sobre a multiplicidade. Se for omitida será considerada não especificada.

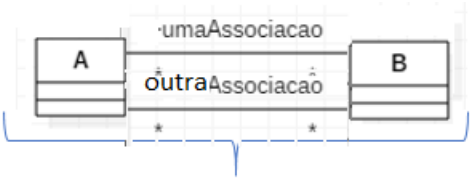


Diagrama de Classes

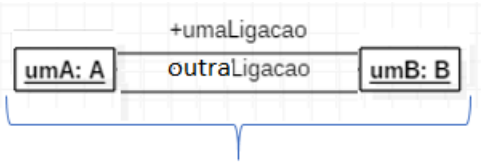


Diagrama de Objetos

Associação versus ligação. Podemos usar múltiplas associações para modelar múltiplas ligações entre os mesmos objetos.

A multiplicidade normalmente expões suposições ocultas, embutidas em um modelo. Por exemplo, a associação TrabalhoPara entre Pessoa e Empresa é um-para-muitos ou muito-para-muitos? Depende do contexto. Uma aplicação de declaração de imposto permitiria que uma pessoa trabalhasse para várias empresas. Por outro lado, os registros de membro para um sindicato de trabalhadores da indústria automotiva podem considerar o segundo emprego irrelevante. Os diagramas de classe ajudam a trazer “a tona essa suposições ocultas, tornando-as visíveis e sujeitas a escrutínio.

A notação usada para a representação para os indicadores de multiplicidade, é:

Muitos	*
Apenas Um	1
Zero ou Muitos	0....*
Um ou Muitos	1....*
Zero ou Um	0....1

Intervalo Específico 2....4

Exercícios

1. Antônia deseja escrever uma aplicação de controle de tarefas para disponibilizar em sites de aplicativos mobile. Ela deseja que sua aplicação seja capaz de armazenar as informações em um servidor na Web assim os dados ficariam armazenados em um banco de dados centralizado. O aplicativo deve funcionar em diferentes dispositivos móveis, como por exemplo: smartphone IPhones, Android, BlackBerry, ou seja, deve disponibilizar diferentes visões (views). Essa aplicação deveria oferecer os seguintes recursos:

   a) Controlee de Aces so
   b) O cadastro de cada tarefa contém o número da prioridade, representado por um valor real. Isso permite entrar com intervalos intermediários. Além da prioridade, o cadastro deve conter: o nome da tarefa, a data limite de execução (se houver), o percentual já concluído e o detalhamento da tarefa.
   c) Para cada tarefa há uma lista de itens que descrevem sua execução. Para cada item de execução, cadastram-se:
   d) o percentual correspondente
   e) a descrição da execução
   f) a data da execução (quando for concluída)
   g) Quando uma tarefa receber 100% de execução, ela deve ser movida automaticamente para a lista de tarefas concluídas, podendo ser apagada, se for o caso. Veja o exemplo desse controle em papel.

TAREFA 1.1. – ANIVERSÁRIO DO FABIO	
Data limite = 06/08/2005	
Percentual já concluído = 65%	
Detalhamento = planejamento dos preparativos para a festa de aniversário do Fabio, no sábado, dia 6 de agosto.	
Lista de Itens para serem executados:	
[20%] Aluguel do salão e da animação	- 01/03/2005
[20%] Encomenda do bolo, salgados e doces	- 15/07/2005
[05%] Compra das bebidas	- _____
[25%] Compra dos itens para a decoração	- 01/07/2005
[30%] Arrumação do salão	- _____

Pede-se: Construa o diagrama de classes para esse problema.

2. Construa um diagrama de classes inicial para a seguinte situação: Pacotes são enviados de uma localidade para outra. Pacotes têm um peso específico. Localidades são caracterizadas pelas facilidades de transporte (por exemplo, rodoviárias, aeroportos e auto-estradas). Algumas localidades são vizinhas, isto é, existe uma rota direta de

transporte entre tais localidades. A rota de transporte entre as localidades tem um certo comprimento (a distância entre as localidades). Trens, aviões e caminhões são usados para transporte de pacotes. Cada um destes meios de transporte pode suportar uma carga máxima de peso. A cada momento, durante seu transporte, é necessário saber a posição (localidade) de cada pacote. Também é necessário manter o controle de que meio de transporte está sendo utilizado em cada parte da rota para um certo pacote.

3. Construa o diagrama de classes para um sistema de informação para controlar o campeonato de Fórmula 1.

4. Partidas de Tênis: Desenhe o diagrama de classes.

   "Num torneio de tênis, cada partida é jogada entre dois jogadores. Pretende-se manter informação sobre o nome e idade dos jogadores; data partida e atribuição dos pontos jogadores às partidas. O máximo de partidas que um jogador pode realizar é 6 e o mínimo 1.

5. Desenhe um diagrama de classes com relacionamentos, nomes de papéis e multiplicidades para as seguintes situações:

- Uma pessoa pode ser casada com outra pessoa;
- Uma disciplina é pré-requisito para outra disciplina;
- Uma peça pode ser composta de diversas outras peças.

6. Identifique as classes e/ou os relacionamentos a partir das seguintes regras do negócio:
   a. Pedidos são compostos de vários itens de pedido;
   b. Um item de pedido diz respeito a um e exatamente um produto;
   c. Um pedido deve conter pelo menos 1 e pode conter até 20 itens.

Referências Bibliográficas

BLAHA, M., RUMBAUGH, J. Modelagem e projetos baseados em objetos com UML 2. Rio de Janeiro:Elsevier-Campus, 2006.

BOOCH, G. Object-oriented analysis and design with applications. 3ª.ed. Addison-Wesley, 2007.