UML – Unified Modeling Language

- A UML é a linguagem gráfica para a visualização, especificação, construção e documentação de sistemas de software.
- A notação (a própria UML) é relativamente trivial
- Muito mais importante: habilidade para modelar com objetos
- Só aprender a notação UML não ajuda
- A UML não é
  - um processo ou metodologia
  - APOO
  - regras de projeto

UML – HISTÓRICO

- Linguagens orientadas a objetos surgiram entre a metade da década de 1970 e 1980
- Entre 1989 e 1994 a quantidade de métodos de modelagem aumentou de 10 para 50
- Havia uma dificuldade inicial, por parte dos desenvolvedores, devido a diversidade de métodos, visto que não atendiam completamente às suas necessidades.
- Neste conexto alguns método se destacaram: Notação de Booch; o OOSE (Object-Oriented Software Engineering) de Ivar Jacobson e o OMT (Object Modeling Technique) de James Rumbaugh

UML – HISTÓRICO

- Todos eram completos, apresentando pontos fortes e fracos
- Por volta da metade da década de 1990 Booch (Rational Software), Jacobson (Objectory) e Rumbaugh(GE) se unem para desenvolver a UML
- Outubro/1995 o esboço da versão 0.8 foi lançada
- Junho/1996 – Lançada a versão 0.9 da UML
- Foi estabelecido um consórcio de UML com a participação das empresas: Rational Software; Digital; Hewlett-Packard; I-Logix; Intelicorp; IBM; Icon Computing; MCI SystemHouse; Microsoft; Oracle; Texas Instruments e Unysis.
- Este consórcio apresenta em Janeiro/1997 a UML 1.0

UML – HISTÓRICO

- A UML 1.0 foi oferecida ao OMG (Object Management Group)
- Em 1997 o consórcio se ampliou
- Em Novembro/1997 o OMG adotou a UML 1.1
- O grupo Revision Task Force (RTF) do OMG assume a manutenção da linguagem e em Junho/1998 a versão 1.2 foi lançada
- Em Dezembro/1998 o RTF lançou a versão 1.3 da UML
- 2001 - Versão 1.5
- 2003 – Lançamento da Especificação 2.0 pelo OMG
- 2005 – UML 2.0
- 2015 – UML 2.5

POR QUE MODELAR?

- Uma empresa bem sucedida é aquela que fornece software de qualidade e é capaz de atender as necessidades dos usuários.
- A modelagem é a parte central de todas as atividades que levam a implantação de um bom software
- Um modelo é uma simplificação da realidade
- Modelos são construídos para:
  - Prover a comunicação entre a estrutura e o comportamento do sistema
  - Visualizar e controlar a arquitetura do sistema
  - Ter uma melhor compreensão do sistema
  - Gerenciar os riscos

POR QUE MODELAR?

- Com a modelagem, quatro objetivos podem ser alcançados:
  - Visualizar o sistema como ele é ou como desejamos que seja
  - Possibilita a especificação da estrutura e o comportamento do sistema
  - Proporciona um guia para construção do sistema
  - Documenta as decisões tomadas.
- Os modelos são usados extensivamente na engenharia. Na construção civil, a construção de um prédio é precedida pela construção de modelos (desenhos)
- Da mesma forma linguagem UML fornece uma maneira para a modelagem de software através de desenhos

PRINCÍPIOS DA MODELAGEM

1. A escolha dos modelos a serem criados tem uma profunda influência sobre a maneira como um determinado problema é atacado e como uma boa solução é definida
2. Cada modelo poderá ser expresso em diferentes níveis de precisão
3. Os melhores modelos estão relacionados à realidade
4. Nenhum modelo é único e suficiente. Qualquer sistema não-trivial será melhor investigado por meio de um pequeno conjunto de modelos quase independentes.

VISÃO GERAL DA UML

- As linguagens fornecem um vocabulário e as regras para combinação de palavras desse vocabulário com a finalidade de comunicar
- Uma linguagem de modelagem é a linguagem que tem seu foco voltado para a representação conceitual e física de um sistema.
- O vocabulário e as regras de uma linguagem como a UML, indicam como criar e ler modelos bem-formados, mas não apontam quais modelos deverão ser criados, nem quando você deverá criá-los.

VISÃO GERAL DA UML

- A UML É UMA LINGUAGEM DESTINADA A:
  - VISUALIZAR…
  - ESPECIFICAR…
  - CONSTRUIR…
  - DOCUMENTAR…

  ...OS ARTEFATOS DE UM SISTEMA DE SOFTWARE.
- Artefato é um conjunto de informações utilizado ou produzido por um processo de desenvolvimento de software

UML – BLOCOS DE CONSTRUÇÃO

- O vocabulário de UML abrange três tipos de blocos de construção:
  - ITENS
  - RELACIONAMENTOS
  - DIAGRAMAS
- Os ITENS são as abstrações; Os RELACIONAMENTOS reúnem esses itens; Os DIAGRAMAS agrupam um conjunto de itens afins.

UML – ITENS

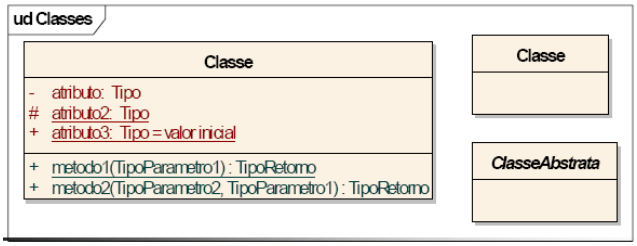
- Existem quatro tipos de itens na UML
  - Itens Estruturais – Substantivos utilizados no modelo; Partes mais estáticas; Representam elementos conceituais ou físicos
  - Itens Comportamentais – São as partes dinâmicas dos modelos UML. São os verbos utilizados no modelo, representando comportamentos no tempo e no espaço
  - Itens de Agrupamentos – São partes organizacionais dos modelos.
  - Itens Anotacionais – São as partes explicativas dos modelos UML

UML – ITENS ESTRUTURAIS

- Substantivos utilizados no modelo;
- São partes mais estáticas e representam elementos conceituais ou físicos
- A linguagem UML define os seguintes itens estruturais:
  - Classes
  - Interfaces
  - Casos de Uso
  - Colaborações
  - Classes Ativas
  - Componentes
  - Nós

UML – ITENS ESTRUTURAIS

- CLASSES – A seguir é mostrado como as classes são representadas em UML
- A UML sugere que o nome de classe comece com uma letra maiúscula e seja representado em negrito e no centro do compartimento superior.
- O compartimento do **meio contém os atributos e o** compartimento **inferior contém os métodos da classe**



UML – ITENS ESTRUTURAIS
Classes

- ATRIBUTOS

  [visibilidade] [/] nome [: tipo] [[multiplicidade]]

  [= valor inicial][{propriedades}]

  - VISIBILIDADE
    - publico (+); protegido (#); privado (-); pacote (~)
  - / - Indica um atributo opcional, normalmente derivado de uma superclasse
  - TIPO
    - classe ou tipo primitivo de uma linguagem
  - MULTIPLICIDADE
    - Indica a quantidade de vezes que o atributo aparece. Pode ser representada da forma [inicio .. final] ou [total]
  - VALOR INICIAL
    - Valor padrão utilizado para o atributo
  - PROPRIEDADES
    - Características associadas aos atributos (“unique”; “ordered”; “readonly”; etc)

UML – ITENS ESTRUTURAIS
Classes

- OPERAÇÕES (Métodos)
  [Visibilidade] nome (listaParametros) : tipoRetorno [{propriedades}]
- Visibilidade - publico (+); protegido (#); privado (-); pacote (~)
- Nome - Nome do método
- listaParametros
  - Parâmetros recebidos e/ou retornados pelo método. Podem ser escritos da seguinte forma:
  - **[direçãoParametro] nome : Tipo [[Multiplicidade]] [=valorPadrão]**
  - direçãoParâmetro
    - in – somente entrada; out – somente saída; inout – entrada e saída
  - valorPadrão – Valor padrão associado ao parâmetro
  - Tipo - classe ou tipo primitivo de uma linguagem
  - Multiplicidade - Indica a quantidade de vezes que o atributo aparece.
- Propriedades - podem representar “pré-condições”; “pós-condições”; “query”; etc.
- tipoRetorno - classe ou tipo primitivo de uma linguagem

UML – ITENS ESTRUTURAIS
Classes - Abstratas

- Algumas classes na hierarquia são tão gerais que nenhum objeto será criado a partir delas. Neste caso a classe é dita ABSTRATA
- Uma classe abstrata não pode ser instanciada ou seja, não é possível criar objetos a partir da mesma
- A classe ABSTRATA é uma classe que está incompleta. Esta classe pode conter métodos abstratos que são aqueles métodos apenas declarados, mas que não foram implementados.
- Os métodos abstratos devem ser obrigatoriamente implementados nas subclasses

UML – ITENS ESTRUTURAIS
Classes - Abstratas

- O método abstrato contém apenas sua assinatura (nome, número e tipo dos seus parâmetros).
- Para a criação de classes e métodos abstratos deve ser utilizado o modificador de tipo que normalmente é a palavra “abstract”
- Classe CONCRETA é aquela a partir da qual objetos serão instanciados. Neste tipo de classe todos seus métodos devem ser, obrigatoriamente, definidos.

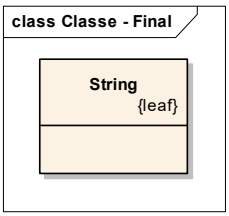
UML – ITENS ESTRUTURAIS
Classes - Modificadores

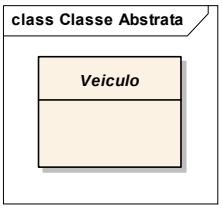
- Nas linguagens orientadas a objeto existem palavras reservadas chamadas Modificadores.
- Estas palavras modificam o comportamento de classes, atributos e métodos
- Por exemplo, na linguagem Java, temos os seguintes modificadores:

CLASSE	ATRIBUTO	MÉTODO
abstract		abstract
final	final	final
	static	static
	transient	
	volatile	
		synchronized
		native

UML – ITENS ESTRUTURAIS
Modificadores de Classe

- Abstract
  - Não é possível criar objetos de uma classe abstrata.
  - Além disso uma classe abstrata pode conter métodos abstratos que são aqueles que não possuem corpo, mas apenas sua assinatura.
- Final
  - Impede que uma determinada classe possa ser especializada, ou seja, impede a herança
  - Neste caso a classe não poderá ter nenhuma classe filha



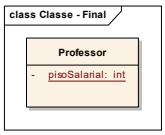


UML – ITENS ESTRUTURAIS
Modificadores de Atributo

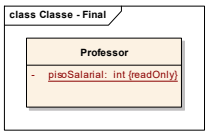
- Static
  - Conhecido como “atributo de classe”, este atributo está associado à classe e não a um objeto da classe.
  - Todos os objetos compartilham o mesmo atributo estático.
  - Uma mudança no atributo é percebida por todos os objetos da classe
- Final
  - Indica que o atributo é um valor constante e cujo valor não pode ser alterado.
  - O valor é associado ao atributo no momento em que o mesmo é inicializado e esta inicialização é feita no código
  - Exemplo: public static final PI = 3.141592653589793;
- Transient (Java)
  - Atributos transientes não pode ser serializados
  - Desta forma ao gravar os dados do objeto, no disco, por exemplo, este tipo de atributo será excluído do processo de serialização do objeto.
- Volatile (Java)
  - Em um código com vários threads (multithreaded) um atributo volátil conterá o mesmo valor em todos os threads.
  - Neste caso quando uma alteração é feita no atributo por um thread o mesmo será atualizado em todas as cópias, existentes em cada thread, daquele atributo existentes

UML – ITENS ESTRUTURAIS
Modificadores de Atributo

- Static



- Final

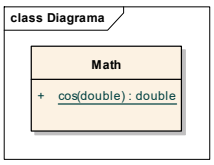


UML – ITENS ESTRUTURAIS
Modificadores de Método

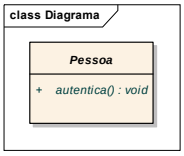
- Static
  - Conhecido como “método de classe”, este método está associado à classe
  - Desta forma é possível acessar o método sem a existência de um objeto da classe
  - Exemplo: p = dt * dU * dI * Math.cos(fi);
- Abstract
  - Neste caso o método somente a sua assinatura e não possui corpo
  - Esta presente em classes abstratas e em interfaces
- Final
  - Indica que o método não pode especializado em qualquer subclasse
  - Este modificador impede o polimorfismo
- Synchronized (Java)
  - Em um código com vários threads (multithreaded) garante que o método será executado de forma individual, por um único thread.
  - Quando um thread executar um método sincronizado, é necessário antes de mais nada, obter o lock do objeto. A partir deste ponto somente um thread poderá executar o código.
  - Ao final da execução o lock é liberado e o método poderá ser executado por outro thread
- Native (Java)
  - Indica que o método é implementando em uma outra linguagem, como C, C++ ou assembly
  - A JNI (Java Native Interface) é uma interface de programação que permite a execução de métodos nativos.

UML – ITENS ESTRUTURAIS
Modificadores de Método

- Static

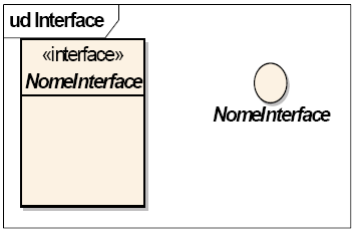


- Abstract



UML – ITENS ESTRUTURAIS
Interfaces

- Coleção de operações (métodos) que especificam os serviços de uma classe ou componente.
- A interface não contém a implementação das operações, mas apenas descreve quais são estas operações
- Através das interface é possível diminuir o acoplamento entre diferentes partes de um sistema
- Uma interface é realizada por uma ou mais classes

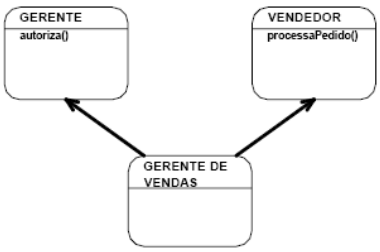


UML – ITENS ESTRUTURAIS
Interfaces

- O conceito de interface é um importante aliado no projeto de software
- Na linguagem Java, este conceito é utilizado para o modelamento da Herança Múltipla

UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla

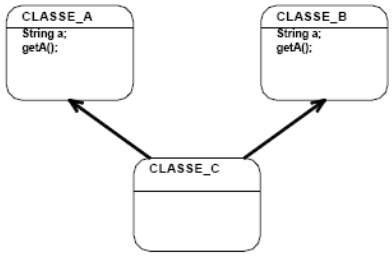
- Existem casos em que uma classe pode herdar o comportamento de mais de uma classe.
- Neste caso temos a herança múltipla, conforme mostrado abaixo



MODELAGEM INCORRETA

UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla (java)

- Como implementar a herança múltipla:



- Como implementar a herança múltipla:
- No exemplo acima, qual cópia do atributo **a a classe CLASSE_C** vai herdar? Qual método **getA() vai utilizar?**
- Java resolve este problema utilizando o conceito de "INTERFACES"

UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla (java)

- Um método possui duas partes: sua assinatura e sua Implementação
- Java não suporta a herança múltipla explicitamente, mas possui meios para que os efeitos da herança múltipla seja realizada de forma indireta utilizando o conceito de INTERFACES
- Através deste conceito uma classe pode herdar as assinaturas dos métodos, mas não a sua implementação.
- A implementação, deve por sua vez, ser definida na subclasse.
- Desta forma uma interface possui apenas um conjunto de métodos

UML – ITENS ESTRUTURAIS
Classes – Herança Múltipla (Java)

- Através da herança múltipla novos métodos,de diferentes classes, podem ser agregados a uma subclasse
- A herança através de interface não possibilita a reutilização do código, visto que o método herdado deve ser implementado para cada subclasse.
- ATRIBUTOS EM UMA INTERFACE
  - Em uma interface os atributos são implicitamente declarados como static e final.
- MÉTODOS EM UMA INTERFACE
  - Todos os métodos são abstratos, não sendo necessário a palavra “abstract”

UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla

- Na interface todos os métodos são abstratos e não possuem implementação apenas sua assinatura.
- A uma classe pode utilizar mais de uma interface em sua definição
- A interface representa um comportamento que a classe que a implementa irá obrigatoriamente possuir.

UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla (Java)

- Uma INTERFACE é definida através da palavra “interface” conforme mostrado a seguir:

  [public] interface B extends A

- Neste caso A deve ser outra interface.

  Exemplo – Definição da INTERFACE IGerente

```
public interface IGerente{
    public void demitir(Empregado e);
    public boolean concedeAumento();
    public boolean contratar(Empregado e);
}
```

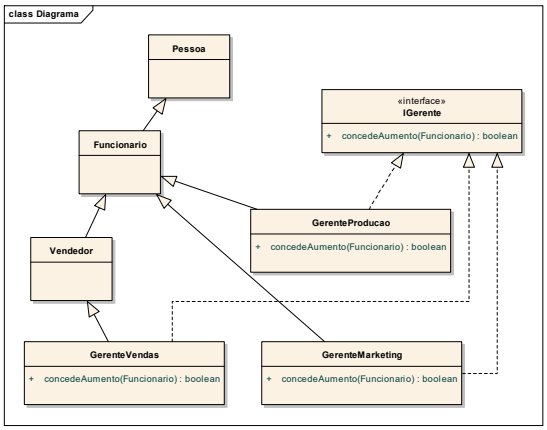
  - O gerente de vendas, bem como qualquer outro tipo de gerente, realiza as operações demitir e contratar, porém cada um a seu modo

- A indicação da herança múltipla é feita da seguinte forma:

  [public] [modTipo] class B [extends A] implements C,[D,E,F..]

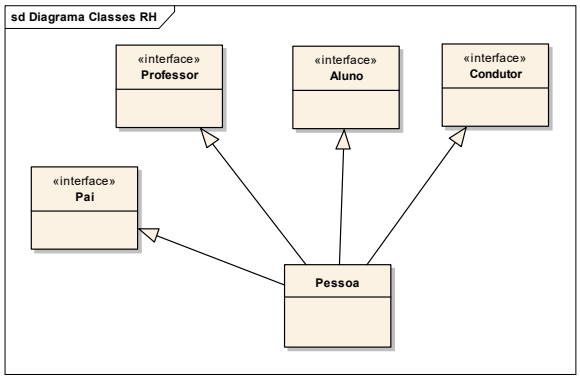
UML – ITENS ESTRUTURAIS
Interfaces – Comportamento Padrão

- Exemplo de Uso de Interface
- A interface garante que um conjunto de classes contenha um comportamento padrão (método), porém com as particularidades necessárias
- Cada tipo de Gerente, pode “concederAumento” porém segundo seus critérios particulares
- A interface garante que todos contenham o método



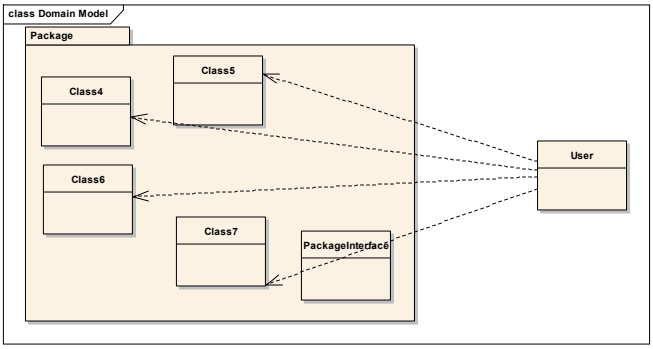
UML – ITENS ESTRUTURAIS
Interfaces – Comportamento Padrão

- Exemplo de Uso de Interface
- A classe Pessoa pode representar diversos papéis. Cada papel associa um conjunto de comportamentos (métodos)
- O que garante a associação é a interface



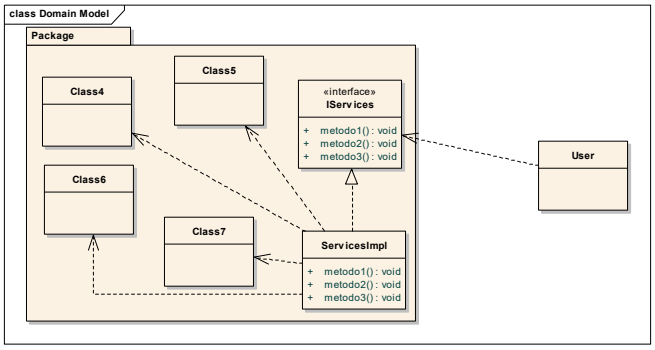
UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla

- Redução do Acoplamento
  - O diagrama abaixo mostra uma classe que utiliza outras classes
  - Neste caso existe um alto acoplamento entre as mesmas



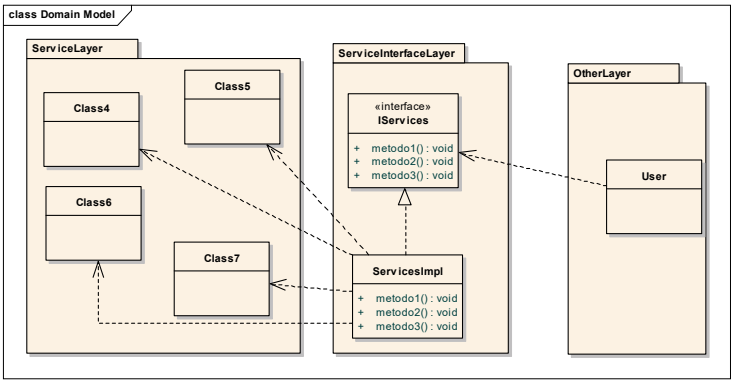
UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla

- Redução do Acoplamento
  - A partir do uso da uma Interface é possível reduzir o acoplamento
  - O único ponto de contato entre os módulos neste exemplo é a interface
  - Caso a interface seja mantida (assinatura dos métodos) os módulos são independentes entre si.



UML – ITENS ESTRUTURAIS
Interfaces – Herança Múltipla

- Redução do Acoplamento
  - A mesma informação, porém organizada de uma diferente forma



UML – ITENS ESTRUTURAIS
Casos de Uso

- Descrevem um conjunto de ações realizadas pelo sistema que proporciona resultados observáveis de valor para alguma entidade que está de fora deste sistema.
- Um caso de uso descreve um comportamento do sistema
- Um caso de uso é realizado por uma colaboração

UML – ITENS ESTRUTURAIS
Colaborações

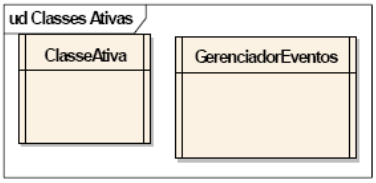
- É uma sociedade de classes,interfaces e outros elementos que trabalham em conjunto para fornecer algum comportamento cooperativo maior que a some de todas as suas partes.
- Uma determinada classe pode fazer parte de várias colaborações

UML – ITENS ESTRUTURAIS
Colaborações

- A colaboração possui dois aspectos:
  - Parte estrutural (classes; interfaces; componentes e nós)
  - Parte comportamental (como os elementos da parte estrutural se interagem)
- A colaboração não possui nenhum de seus elementos estruturais mas apenas faz referência a cada um deles, sendo portanto um grupo conceitual de itens.
- A colaboração pode ser utilizada para indicar a “realização” ou seja, a implementação do caso de uso
- Outro uso das colaboração é indicar a “realização” de uma operação

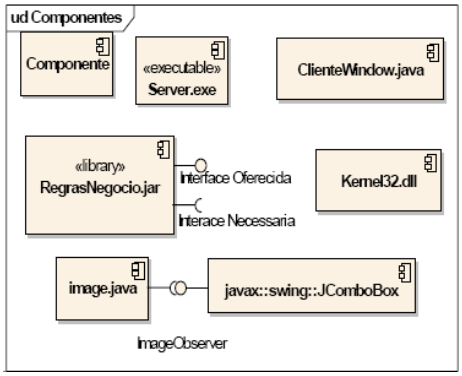
UML – ITENS ESTRUTURAIS
Classes Ativas

- Classes cujos objetos possuem um ou mais processos ou threads associados.
- Desta forma realizam alguma atividade de controle.
- São representadas conforme as classes, porém com duas linhas laterais
- Os objetos de classes ativas podem realizar operações concorrentes



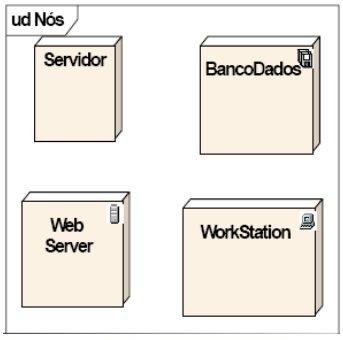
UML – ITENS ESTRUTURAIS
Componentes

- São partes físicas e substituíveis de um sistema que proporcionam a realização de um conjunto de interfaces
- Representam um pacote físico de classes; interfaces e colaborações



UML – ITENS ESTRUTURAIS
Nós

- Elemento físico que representa um recurso computacional, geralmente com alguma memória e capacidade de processamento.
- Um conjunto de componentes poderá estar em um nó.

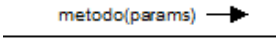


UML – Itens Comportamentais

- São as partes dinâmicas dos modelos UML.
- São os verbos utilizados no modelo, representando comportamentos no tempo e no espaço
- Existem dois tipos de itens comportamentais
  - Interação
  - Máquina de estado
- Os itens comportamentais costumam estar ligados a vários elementos estruturais

UML – Itens Comportamentais Interação

- Um comportamento que abrange um conjunto de mensagens trocadas entre objetos, em determinado contexto, para a realização de um propósito
- As interações podem ser encontradas em qualquer parte em que os objetos estejam vinculados entre si.
- Existem no contexto de sistema ou subsistema; no contexto de uma operação e no contexto de uma classe.
- A modelagem dos aspectos dinâmicos é feita pela utilização de interações
- A interação é representada da seguinte forma:



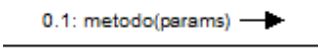
UML – Itens Comportamentais Interação

- Normalmente a linha cheia com a seta contém também o nome da mensagem que está sendo enviada.
- Vínculo
  - Um vínculo é uma conexão semântica entre objetos. Em geral um vínculo é uma instância da "associação".
- Sempre que existir uma associação de uma classe com outra poderá haver um vínculo entre elas e havendo o vínculo poderá haver a troca de mensagens.
- O vínculo especifica o caminho pelo qual o objeto pode enviar uma mensagem

UML – Itens Comportamentais Interação

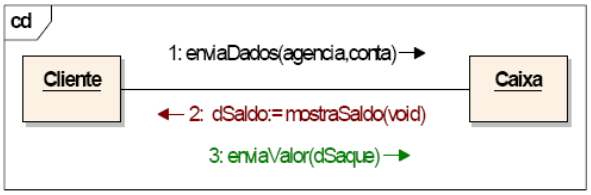
- Além da operação a interação pode conter outras informações:
- SEQÜENCIAMENTO
  - A fim de se obter a ordem em que as iterações acontecem em um contexto é possível associar as mesmas um número de seqüência.
  - O número de seqüência poderá ser da seguinte forma: i.j.k
  - No exemplo acima existem 3 níveis de chamadas
  - No caso da existência de múltiplos fluxos de controle (threads) pode ser utilizado o nome o thread que está controlando a interação:

    th2 : operation()



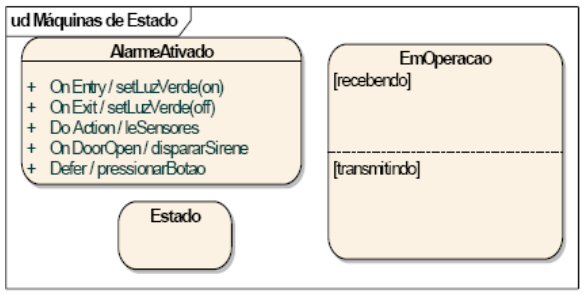
UML – Itens Comportamentais Interação

- Além disso a interação pode conter os argumentos que a operação utiliza e também valores de retorno.
  - 1.3.2 : p : find("Joao");



UML – Itens Comportamentais Máquinas de Estado

- Uma máquina de estado é um comportamento que especifica as seqüências de estados pelas quais objetos ou iterações passam durante sua existência em resposta a eventos, bem como suas respostas e estes eventos.
- As Máquinas de Estado podem ser compostas por um ou vários estados



UML – Itens Comportamentais Máquinas de Estado

- Conceitos Associados
  - EVENTO – Ocorrência significativa que tem uma localização no espaço e no tempo, capaz de provocar uma transição entre estados
  - ESTADO - condição ou situação da vida do objeto em que ele satisfaz alguma condição, realiza alguma atividade ou aguarda algum evento.
  - TRANSIÇÃO – Relacionamento entre dois estados

UML – Itens Comportamentais Máquinas de Estado

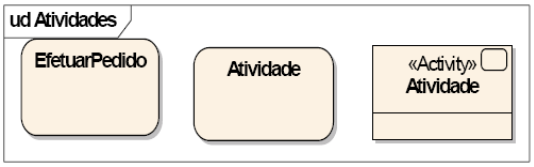
- As máquinas de estado podem representar ESTADOS DE AÇÃO ou ESTADOS DE ATIVIDADE.
  - ATIVIDADE – Execução **não atômica** em andamento em uma máquina de estados. Exemplo: EfetuarPedido; BuscarRegistros; etc.
  - AÇÃO – Execução **atômica** que resulta na alteração de estado ou no retorno de um valor.

UML – Itens Comportamentais Máquinas de Estado

- ESTADOS DE AÇÃO
  - Consiste de máquinas de estado que realizam ações.
  - Seu tempo de execução é considerado insignificante.
  - Um estado de ação não pode ser decomposto.
  - Exemplos:
    - Criação de um objeto;
    - Destruição de um objeto;
    - Enviar um sinal;
    - Receber um sinal

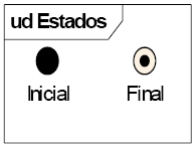
UML – Itens Comportamentais Máquinas de Estado

- ESTADOS DE ATIVIDADE
  - São estados que podem ser decompostos, sendo que suas atividades podem ser representadas por DIAGRAMAS DE ATIVIDADE.
  - Seu tempo de execução é não é considerado insignificante e além disso sua execução pode ser interrompida pois não são atômicos.



UML – Itens Comportamentais Máquinas de Estado

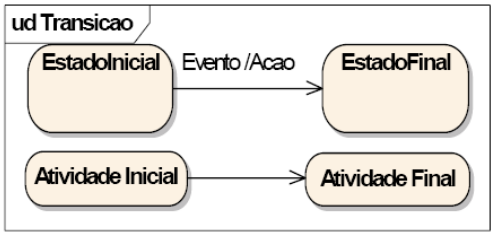
- Além disso existem dois estados especiais que possuem uma representação diferenciada.
- O ESTADO INICIAL e o ESTADO FINAL de uma máquina de estados.
- Um estado pode possuir:
  - Nome
  - Ações Entrada (“Entry") e Saída (“Exit")
  - Transições Internas
  - Atividade(“Do")
  - Evento Adiado (“Defer”)



UML – Itens Comportamentais Máquinas de Estado

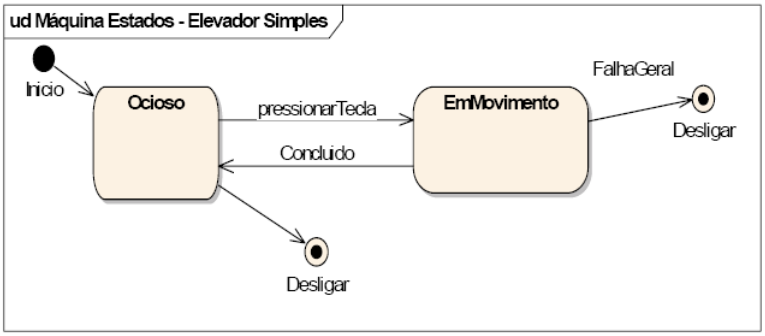
- TRANSIÇÃO
  - Relacionamento entre dois estados, indicando que um objeto no primeiro estado realizará certas ações e entrará em um segundo estado quando um EVENTO especificado ocorrer.
  - Uma transição é a ligação entre dois estados
  - A transição pode conter a seguinte nomenclatura:

evento / ação



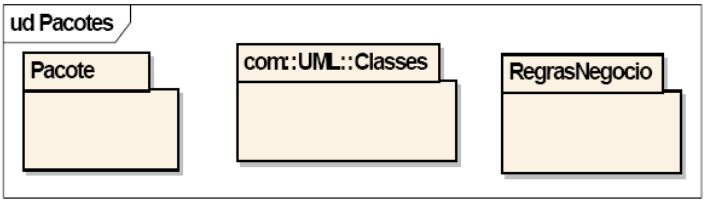
UML – Itens Comportamentais Máquinas de Estado

- O exemplo abaixo mostra uma máquina de estados utilizando todos os conceitos vistos anteriormente



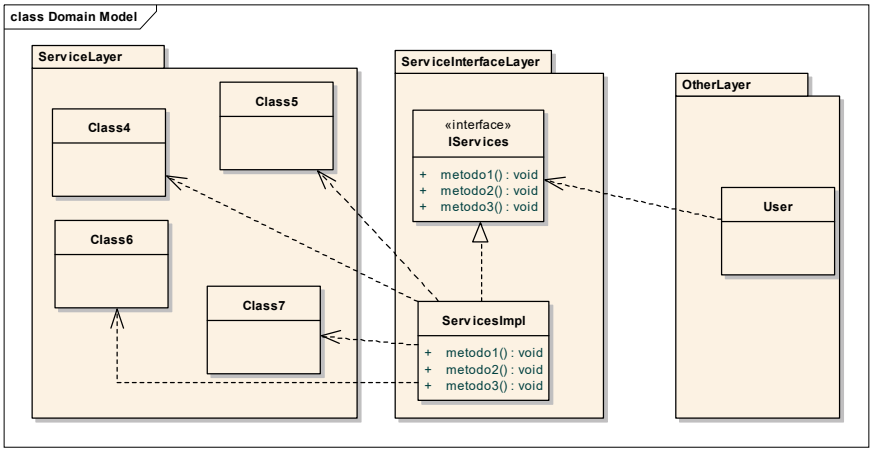
UML – Itens de Agrupamento

- São partes organizacionais dos modelos
- Existe um único tipo de item de agrupamento
  - Pacotes
- Os itens estruturais, comportamentais e até outros itens de agrupamento podem ser colocados em pacotes.
- Os pacotes são puramente conceituais e sua função é organizar os modelos UML



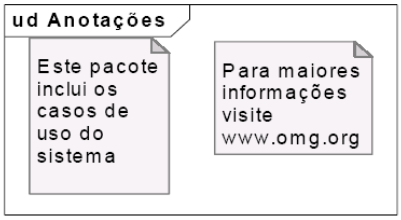
UML – Itens de Agrupamento Exemplo

- Exemplo de uso de pacotes na organização



UML – Itens de Anotacionais

- Existe um único tipo de item anotacional
  - Notas
- São comentários incluídos para descrever, esclarecer e fazer alguma observação sobre qualquer elemento do modelo.

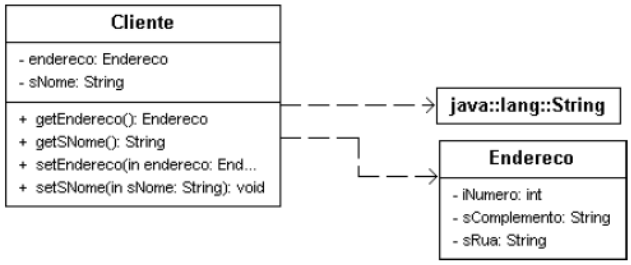


UML – Relacionamentos

- Existem quatro tipos de relacionamentos na UML
  - Dependência
  - Associação
  - Generalização
  - Realização

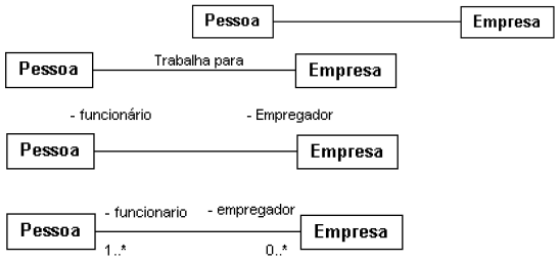
UML – Relacionamentos Dependência

- Relacionamento de utilização entre itens.
- Neste caso as modificações na especificação de um item (dependente) podem afetar o outro item que o utilize.
- Exemplo: A classe Cliente usa a classe String
- No contexto de classes a dependência mostra que uma classe utiliza a outra como argumento na assinatura de uma operação.



UML – Relacionamentos Associação

- Relacionamento estrutural que especifica um item conectado a outro item.
- Na associação as classes são pares umas das outras e realizam algum trabalho em conjunto, ou seja, uma colabora com a outra
- Exemplo: Salas são formadas por Paredes; Tubos estão inseridos em paredes

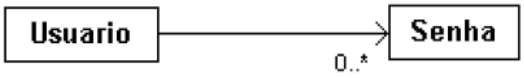


UML – Relacionamentos Associação

- Adornos básicos de uma associação:
  - Nome – O nome descreve a natureza do relacionamento
  - Papel – Indica um papel específico para a classe neste relacionamento
  - Multiplicidade – Indica a quantidade de objetos de uma classe que podem estar conectados à outra.
    - O valor padrão é muitos (0..*) mas podem haver valores como: um ou mais (1..*); exatamente 1 (1);
  - Navegação – Indica sentido de busca
  - Agregação / Composição – Consiste de um diamante aberto ou fechado que é colocado em uma das extremidades da associação

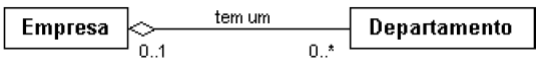
UML – Relacionamentos Associação

- Navegação
  - Adorno utilizado em uma associação
  - Considerando uma associação simples é possível navegar entre objetos de um tipo para outro tipo.
  - Em certos casos porém esta navegação deve ser limitada.
  - Exemplo: Na associação entre as classes Usuário e Senha, é possível encontrar a senha, a partir de um usuário mas o contrário não é possível. Desta forma a navegação é colocada apenas em uma direção, neste caso do usuário para a senha



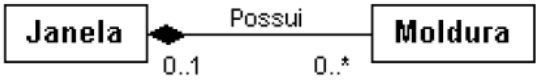
UML – Relacionamentos Associação

- AGREGAÇÃO
  - Neste tipo de associação, as classes estão em um mesmo nível, porém neste caso deve ser feito a modelamento “todo/parte”.
  - O item maior (o todo) é formado por itens menores (as partes).
  - Este é um relacionamento do tipo: A “tem um” B.
  - Para representar o todo é colocado um diamante aberto na extremidade do todo.



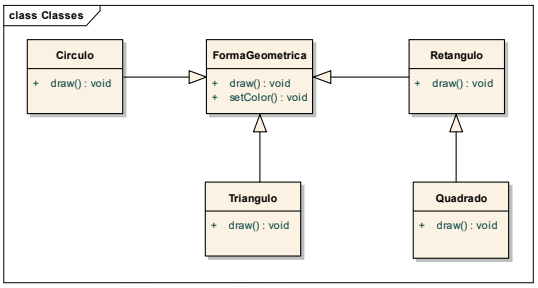
UML – Relacionamentos Associação

- COMPOSIÇÃO
  - A composição é uma forma de AGREGAÇÃO, com propriedade bem definida e tempo de vida coincidente como parte do todo.
  - Neste tipo de associação o "todo" é responsável pela criação e destruição das "partes".
  - Para representar o todo é colocado um diamante fechado na extremidade do todo.



UML – Relacionamentos Generalização

- Relacionamento de especialização/realização nos quais os objetos dos elementos especializados (filhos) são substituíveis por objetos do elemento generalizado (pais).
- Filhos compartilham estrutura e o comportamento dos pais
- Pode-se dizer o filho "**é um tipo do**" pai ou seja: Um retângulo é um tipo de forma geométrica
- Este relacionamento representa a herança entre classes



UML – Relacionamentos Generalização

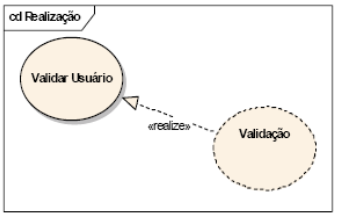
- A generalização significa que os objetos da classe filha podem ser utilizados em qualquer local em que a classe pai ocorra, mas não vice-versa
  - Um método que recebe como parâmetro uma FiguraGeometrica poderá receber qualquer objeto das classes: Circulo; Triangulo ou Quadrado
- Caso a classe filha possua uma operação que tenha a mesma assinatura de uma operação da classe pai, esta operação prevalecerá
  - No exemplo anterior ao chamar o método draw() na classe Quadrado, este será executado a não o método draw() da classe Retangulo
  - Por sua vez, o método setColor() está disponível para a classe Quadrado e poderá ser utilizado por um objeto desta classe
  - O método draw() é um exemplo de Polimorfismo
- Caso uma classe filha possua mais de uma classe pai, diz-se que é um caso de herança múltipla
- A classe filha que não possuem outras classes filhas é chamada de folha

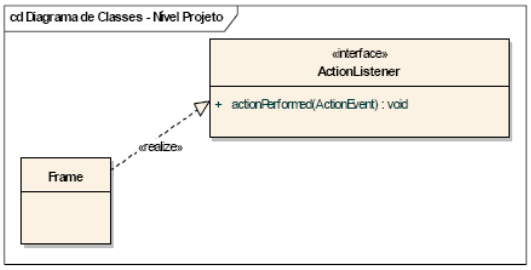
UML – Relacionamentos Realização

- Relacionamento semântico entre classificadores, onde um classificador especifica um contrato que outro classificador garante executar.
- Classificador é um item que pode apresentar instâncias e que possui características estruturais e comportamentais.
  - Incluem classes; Interfaces; Tipos de Dados; Sinais; Componentes; Nós; Casos de Uso e Subsistemas (pacotes)
- A realização é utilizada no contexto das interfaces e colaborações

UML – Relacionamentos Realização

- Exemplos





UML – Regras

- Existem regras para a combinação dos blocos de construção.
  - Nomes – Quais nomes podem ser atribuídos
  - Escopo – Contexto que determina significado específico para cada nome
  - Visibilidade – Como os nomes podem ser vistos e utilizados por outros
  - Integridade – Como os itens se relacionam entre si de forma adequada e consistente
  - Execução – O que significa executar ou simular um modelo dinâmico

UML – Regras da Linguagem

- Como toda linguagem existem regras associadas à linguagem UML
- Os blocos de construção da linguagem (itens; relacionamentos e diagramas) não podem ser combinados de forma aleatória
- As regras da linguagem especificam o que será um modelo bem formado
- A UML possui regras semânticas para:
  - Nomes – Quais nomes podem ser atribuídos
  - Escopo – Contexto que determina significado específico para cada nome
  - Visibilidade – Como os nomes podem ser vistos e utilizados por outros
  - Integridade – Como os itens se relacionam entre si de forma adequada e consistente
  - Execução – O que significa executar ou simular um modelo dinâmico

UML – Regras da Linguagem Modelos

- Os modelos podem se apresentar da seguinte forma:
  - Modelos bem formados(Consistentes) – São modelos corretos e escritos conforme todas as regras da UML
  - Parciais – Certos elementos ficam ocultos para simplificar a visão do modelo
  - Incompletos – Certos elementos podem ser omitidos
  - Inconsistentes – A integridade do modelo não é garantida
- Desta forma a linguagem permite não apenas a presença de modelos “bem formados” mas aceita simplificações ou omissões

UML – Mecanismos da Linguagem

- Mecanismos básicos da linguagem:
  - ESPECIFICAÇÕES – Por trás de cada bloco existem uma série de especificações que permitem sua construção de forma incremental, além de possibilitar sua visão de diferentes formas.
  - ADORNOS – Permitem acrescentar outros detalhes a um um bloco de construção. Exemplo:

    Classe - Visibilidade; Classe Abstrata

    Associação - Multiplicidade; Papel
  - DIVISÕES COMUNS – Blocos podem ser divididos em: Classes e Objetos
  - MECANISMOS DE EXTENSÃO – Permitem uma ampliação controlada da linguagem

UML – Mecanismos da Linguagem Extensão

- Entre os mecanismos de extensão na UML temos:
  - ESTEREÓTIPOS – Ampliam o vocabulário da UML, permitindo a criação de novos blocos. Ex.:<<Interface>>; <<exception>>
  - VALORES ATRIBUIDOS – Estende as propriedades dos blocos.

    Ex: {versao = 1.3}{autor = FOS}
  - RESTRIÇÕES – Amplia a semântica permitindo acrescentar regras ou modificar as já existentes.

    Ex: {ordered}; {velocidade > 10 Mbps}