UML 2.0 – Linguagem de Modelação

Engenharia de Software

MIEIC 2006/2007

UML – Modelação da Estrutura

Introdução (I)

- Paradigma da orientação por objectos
  - Identificação das classes e suas relações
  - Objecto:
    - Reflecte uma entidade do mundo real e apresenta um estado e comportamento próprio
    - Interagem entre si por troca de mensagens
  - Classe:
    - Estrutura que permite criar objectos semelhantes
    - "Fábrica de Objectos"
    - Objecto = "instância"
  - Diagramas de Classes e Diagramas de Objectos

3

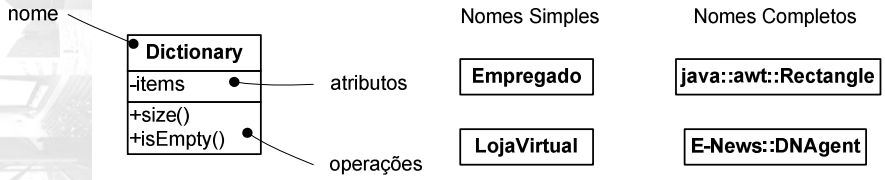
Classes (I)

- Classe
  - Descrição de uma conjunto de objectos que partilham os mesmos atributos, operações, relações e a mesma semântica.
  - Algo tangível ou uma abstracção conceptual no domínio do problema ou da solução.
  - Agrega um conjunto restrito e bem definido de responsabilidades
  - Providencia uma separação clara entre a especificação e a sua implementação.

4

Classes (II)

- 1,2,3 ou 4 secções
  - Nome, atributos, operações ou métodos, e outra informação (opcional)
  - Nome da classe: forma simples ou completa.



Classes (III)

- Atributos
  - Nome, Tipo, Visibilidade, Outras qualificações.
  - Sintaxe completa

[visibility] [/] name [: type] [multiplicity] [= default] [{ property-string }]

  - Visibilidade ([visibility])
    - publica *(public)* (+)
    - protegida *(protected)* (#)
    - privada *(private)* (-)
    - package (~)
  - Atributo derivado ([/])
    - redundante em presença dos restantes (ex: idade)
    - cálculos pesados, desempenho, quase constantes...

Classes (IV)

- Atributos
  - Nome (name)
    - Obrigatório
    - Tão descritivo quanto possível
  - Tipo ([: type])
    - Classificador que determina o tipo de informação que se pode guardar no atributo:
      - Tipo primitivo do UML (Integer, String, ...)
      - Enumerado (Boolean, ...)
      - Tipo específico de um linguagem (float, short, long, ...)
      - Outra classe do sistema.

Classes (V)

- Atributos
  - Multiplicidade ([multiplicity])
    - Nº de valores que se podem associar a um atributo (por omissão 1, exactamente um).
    - Qualquer tipo de multiplicidade
      - Muitos (*), um ou mais (1..*), exactamente um (1), zero ou um (0..1), um determinado número (3), um intervalo (2..6),
      - Multiplicidade mais complexa (e.g., 0..3, 5..7, 10..* para representar “qualquer número de valores excepto 4, 8, ou 9).

Classes (VI)

- Atributos
  - Valor por omissão ([= default])
    - Especificar o valor inicial do atributo, ao momento da sua criação.
  - String de propriedades ([{property-string}])
    - Qualquer informação sobre a definição do atributo não prevista pela definição original do UML.
    - Ex: regras, restrições, especificidades da ling. De programação, ...
  - Atributos de classe ou "estáticos"
    - Atributos acessíveis de dentro classe (e não só do objecto).
    - Todos os objectos da mesma classe partilham esse atributo
    - A sublinhado na representação gráfica

9

Classes (VII)

- Atributos - Exemplo

Membro
+ nome : String # morada : String + email : String # dataNasc : Date {data de nascimento >1950} # / idade : Integer # recebePropostas : Boolean # recebeDiscussoes : Boolean # estado : {activo, suspenso}

10

Classes (VIII)

- Operações
  - Nome, assinatura, visibilidade, outras qualificações.
  - Sintaxe completa

[visibility] name [{ parameter-list }] : [return-type] [{ property-string }]

  - Visibilidade ([visibility])
    - Contexto de visibilidade da operação
    - publica *(public)* (+), protegida *(protected)* (#), privada *(private)* (-), package (~)
  - Nome
    - Determinado comportamento de uma classe de objectos
    - Não tem de ser único, desde que a assinatura seja diferente de outro com o mesmo nome.

Classes (IX)

- Operações
  - Lista de parâmetros ([parameter-list])
    - Lista ordenada de atributos que definem a entrada (e saída, dependendo da linguagem) de uma operação.
  - Retorno ([return-type])
    - Tipo de resultado da operação, se esta retornar algo.
  - String de propriedades ([{property-string}])
    - Análoga à dos atributos, mas neste caso, referente à operação.
  - Existem igualmente operações de classe ou estáticas (a sublinhado).
  - UML 2.0 ainda não suporta notação para tratamento de excepções.

Classes (X)

- Operações ou métodos - Exemplo

Membro
+ nome : String # morada : String + email : String # dataNasc : Date {data de nascimento >1950} # / idade : Integer # recebePropostas : Boolean # recebeDiscussoes : Boolean # estado : {activo, suspenso}
-Membro() : Membro +Membro(in nome : String, in email : String) : Membro +obterNome() : String +obterMorada() : String +obterEmail() : String +obterInstituição() : String +obterUsername() : String +validaPasswd(in pwd : String) : Boolean +obterRecebePropostas() : Boolean

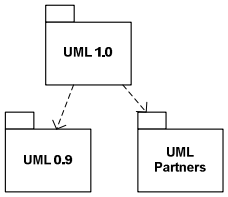
13

Relações (I)

- Relação
  - Ligação entre elementos
  - Representada por um determinado tipo de linha.
  - Três tipos relações mais importantes:
    - Dependências
    - Generalizações
    - Associações

14

Relações (II)



- Relação de Dependência
  - Relação do tipo *cliente-fornecedor*
    - Alteração na especificação do elemento-fornecedor implica alteração do elemento-cliente.
    - Pode ainda representar relação de precedência entre elementos (e.g. workflow, desenho de GUIs).
  - No contexto de classes
    - Usam-se para ilustrar o uso de outras classes como argumentos de métodos ou tipos de atributos
    - Evita-se usar em prol da clareza e simplicidade.
  - Mais usado para ilustrar dependências entre pacotes.
  - Seta a tracejado

Relações (III)

- Tipos de dependência predefinidos
  - Abstracção
  - Ligação
  - Permissão
  - Utilização

Relações (IV)

- Tipos de dependência predefinidos
  - Abstracção
    - «refinement»
      - O cliente representa melhorias, junções, alterações e outros aspectos relativamente ao fornecedor.
    - «derive»
      - O cliente é calculado ou determinado pela utilização do fornecedor (semelhante a atributos derivados).
    - «trace»
      - Representação de relações históricas e dependência de elementos ao longo do tempo. (e.g. requisitos, modelos, implementaçao)

17

Relações (V)

- Tipos de dependência predefinidos
  - Ligação
    - «bind»
      - Ligações entre parâmetros genéricos e parametros efectivos, na criação de elementos não parametrizáveis (e.g., classes-template ou padrões de desenho)
  - Permissão
    - «import»
      - o pacote fornecedor concede ao pacote cliente acesso aos seus elementos públicos, podendo este referi-los.
    - «access»
      - O pacote fornecedor concede ao pacote cliente acesso aos seus elementos, regulado pela respectiva visibilidade de cada um.

18

Relações (VI)

- Tipos de dependência predefinidos
  - Utilização
    - «send», «call», «instantiate»
      - Usados quando o cliente necessita de ter acesso aos membros do fornecedor de forma a funcionar adequadamente.
      - Normalmente usado para explicitar dependências entre objectos e instâncias de componentes.
    - «include», «extend»
      - Usado no contexto dos casos de utilização como já foi visto.

19

Relações : Generalização (I)

- Generalização
  - Relação entre um elemento geral (e.g., superclasse, super-caso-utilização, super-pacote) e um elemento mais específico (e.g., subclasse, sub-caso-utilização, sub-pacote).
  - "is-a", "is-a-kind-of"
  - Linha dirigida a cheio com um triangulo a branco numa das extremidades.
  - Conceito de "Herança" do paradigma de orientação por objectos.

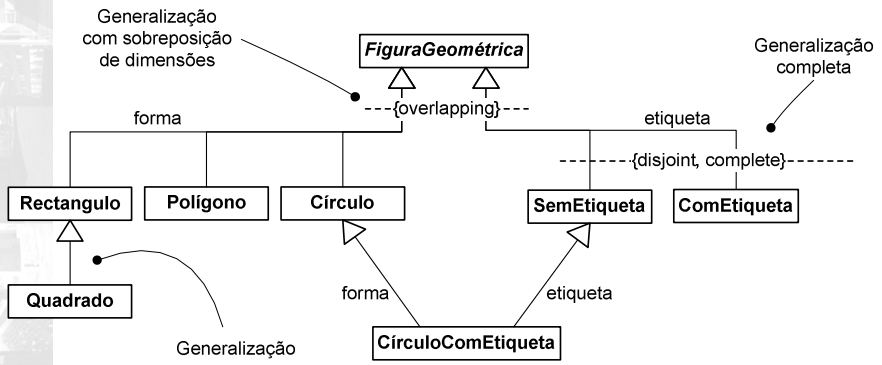
20

Relações : Generalização (II)

- Restrições nas generalizações
  - Generalização **disjunta** *{disjoint}*
    - Uma classe descendente de X pode apenas descendente de uma das subclasses de X. Tipo por omissão.
  - Generalização **sobreposta** *{overlapping}*
    - Uma classe descendente de X pertence ao produto cartesiana das subclasses de X.
  - Generalização **completa** *{complete}* vs. **Incompleta** *{incompleta}*
    - Generalização completamente especificada.

Relações : Generalização (III)

- Exemplo



Relações : Associação (I)

- Associação
  - Relação semântica entre dois ou mais elementos de um modelo.
  - Regras que estabelecem e garantem a integridade da relação:
    - Um modo de identificar univocamente a associação.
    - O número de objectos que podem participar.
    - As restrições sobre os objectos que podem participar.
    - O papel que cada tipo de objecto desempenha.
  - Linha a cheio complementada por um conjunto de adornos que especificam diferentes informações:
    - Nome, indicador direccional, papel de cada participante, multiplicidade e tipo de agregação.

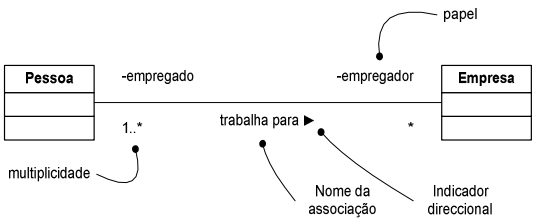
Relações : Associação (II)

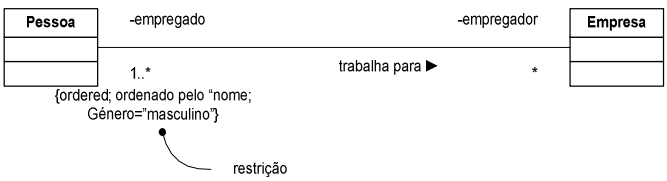
- Nome da associação e Indicador Direccional
  - Semanticamente relevante.
  - Forma recomendada de nomear uma associação: verbo ou expressão verbal
    - Ex: “Trabalha para”, “Pertence a”, “Dirige”, …
  - Esclarecer a direcção da leitura
    - Complementar com o indicador direccional (►)

Relações : Associação (III)

- Adornos Comuns dos Extremos de uma Associação
  - O papel da associação
    - Informa semanticamente como um objecto participa nessa relação.
    - Na geração de código, pode ser usado para nomear referencias.
  - A multiplicidade
    - Traduz o número de instâncias de uma classe que se relacionam com um única instância da outra classe participante na associação.
    - Tipos de multiplicidade semelhantes aos atributos de uma classe.
  - A ordem e outras restrições
    - De que forma, para multiplicidades com mais do que um elemento, se ordenam esses elementos e outras restrições se aplicam a esse extremo da associação.

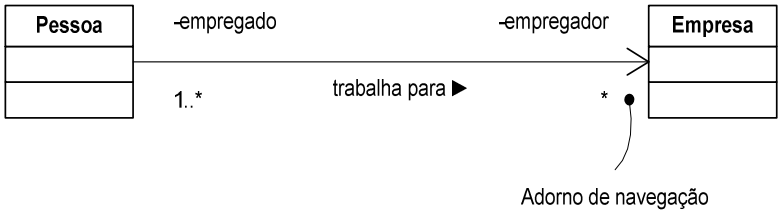
Relações : Associação (IV)





Relações : Associação (V)

- Navegação
  - Forma como, a partir de uma instância de uma classe de pode aceder a uma ou mais instâncias de outra classe relacionada.
  - Por omissão, a navegação numa associação é bidireccional.
  - Há situações em que é conveniente representar as associações com direccionalidade.

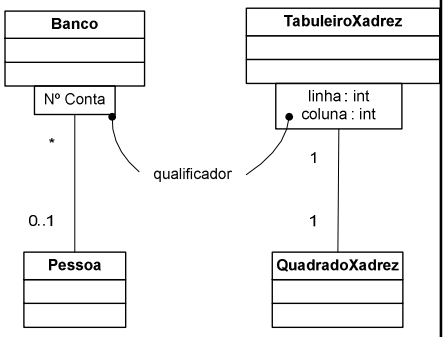


Relações : Associação (VI)

- Associações Qualificadas
  - *Qualificador*
    - Atributo ou lista de atributos (da associação), cujos valores servem para "partir" o conjunto de instâncias associadas a uma instância base ao longo de uma associação.
    - Graficamente representado por um pequeno rectângulo junto de um extremo da associação.
    - Juntamente com a classe de origem, permite seleccionar univocamente um subconjunto das instâncias da classe de destino (o outro extremo).

Relações : Associação (VII)

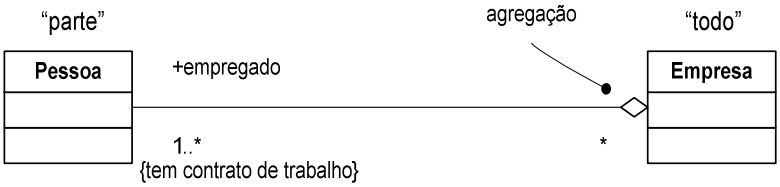
- Associações Qualificadas
  - Multiplicidade afecta extremo destino, com base no par: instância de origem e valor do qualificador.
    - "0..1" : um único valor pode ou não ser seleccionado.
    - "1" : um único valor tem de ser seleccionado.
    - "*" : valor do qualificador é o índice que agrega as instâncias em diferentes subconjuntos.



29

Relações : Associação (VIII)

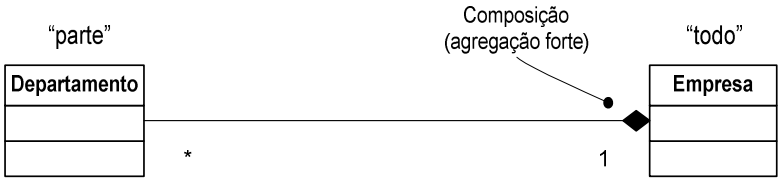
- Agregação (Simples)
  - Relação do tipo *"is-part-of"* ou *"has-a"*.
  - Determinada instância possuir ou ser composta por várias instâncias de outra classe.
    - Gerir configurações de objectos controlados apenas por outro.
    - Objecto agregador serve de "interface" sobre os outros (*bridge*).
  - Gráfico: losango junto da classe agregadora.



30

Relações : Associação (IX)

- Composição (Agregação Composta)
  - Agregação "forte"
  - Variante da Agregação simples, adicionando-se a seguinte semântica:
    - Forte pertença ao "todo" em relação à "parte".
    - Tempo delimitado (as "partes" não podem existir sem o "todo")
  - O "todo" é responsável pela disposição das suas "partes" (criação e destruição).
  - Gráfico : losango a cheio junto do "todo".



31

Relações : Associação (X)

- Associação vs agregação vs composição

Associação
Os objectos têm consciência uns dos outros de modo a colaborarem entre si

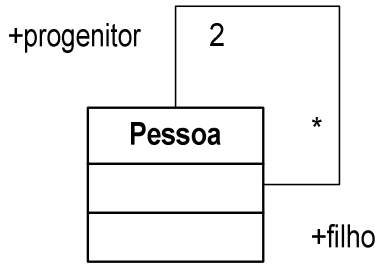
Agregação
Proteger a integridade de uma configuração de objectos
Dar a ilusão de funcionar como um objecto único
Controlo mediado por um único objecto (o "todo")

Composição
A "parte" pode ser membro de apenas uma única configuração
A "parte" não pode existir fora da configuração

32

Relações : Associação (XI)

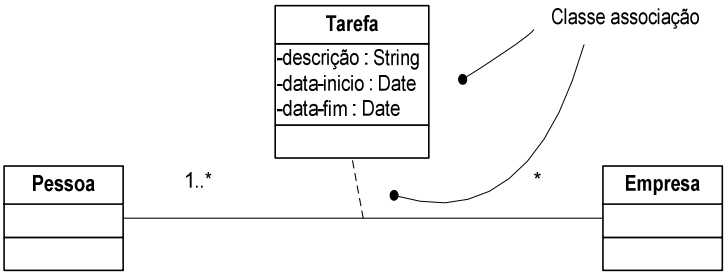
- Associações Reflexivas
  - Relação de uma classe consigo própria.
  - Classe com objectos com diferentes papeis.



33

Relações : Associação (XII)

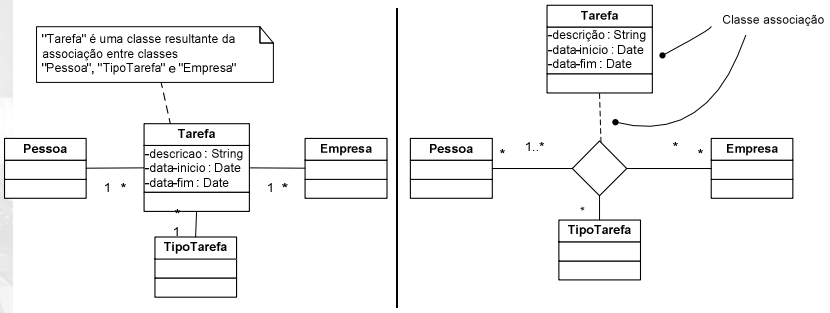
- Classes-Associação
  - Associação com atributos e, logo, deve ser tratada ele própria como uma classe.



34

Relações : Associação (XIII)

- Associações N-Árias (N >= 3)
  - Aridade maior ou igual a 3.
  - Pouco comuns, mas vantajosas em termos de clareza de modelo.
  - Gráfico: losango com linhas a ligar todas as classes participantes.
  - Pode ser transformada em várias associações binárias, mas tal deve ser devidamente documentado.



35

Exercício (I)

- Um consultório médico presta consultas a pacientes.
- Um paciente é identificado pelo seu BI, nome, morada, idade e sexo.
- Um médico é identificado por nome, especialidade e data de entrada ao serviço. Cada médico pode ter até um máximo de 4 pacientes e dar um máximo de 10 consultas por dia.
- Cada consulta tem uma duração e apenas um paciente. Um paciente pode ter quantas consultas quiser, com os médicos que quiser.
- O médico pode prescrever fármacos por cada consulta que dá. Cada fármaco é discriminado por princípio activo, dosagem e formato.

36

Exercício (II)

- Uma empresa de imobiliário vende imóveis. Para tal detém uma carteira de clientes e uma carteira de imóveis que tenta conciliar.
- Os clientes são descritos pelo seu nome e número, morada e telefone.
- Os imóveis são descritos pelo seu código, tipologia, zona, ano de construção e preço.
- Os clientes tem um conjunto de interesses que descrevem as suas preferências por tipologia, zona e preço.
- Os clientes podem firmar acordos de promessa de compra/venda com a empresa para um determinado imóvel, sendo este acordo firmado numa determinada data. Um cliente pode comprar mais do que um imóvel, mas para cada compra terá de ter um acordo distinto.

37

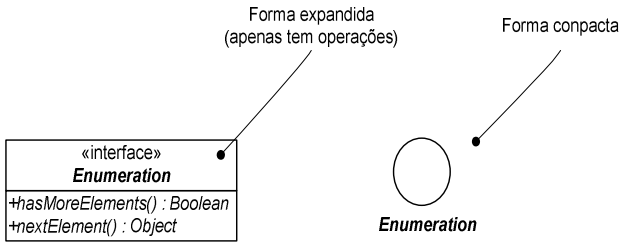
Interfaces (I)

- Interface
  - "Contrato" que estabelece o compromisso de uma classe de fornecer um conjunto de operações.
  - Benefícios:
    - Captura das semelhanças entre classes não relacionadas sem forçar a criação de relações artificiais entre elas.
    - Declaração de métodos que uma ou mais classes esperam implementar.
    - Revelar a interface de programação de um objecto sem revelar a sua implementação. Ou seja, um objecto pode ser visto de diferentes perspectivas consoante diferentes situações.

38

Interfaces (II)

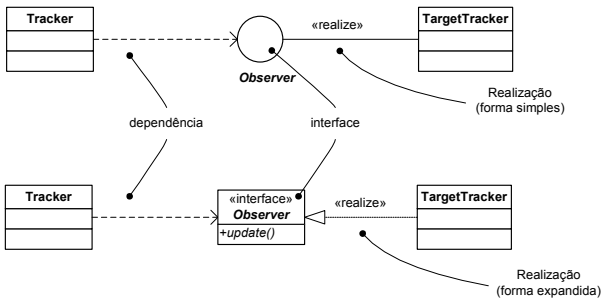
- Representação Gráfica
  - Classe com o estereótipo «interface»
  - Alternativamente, um circulo.



39

Interfaces, Classes e Componentes (I)

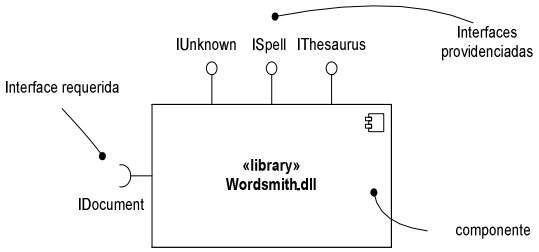
- Interfaces podem participar em relações
  - Generalização, Associação e Dependência
  - ***Relação de Realização***
    - Relação entre uma interface e classe ou componente.
    - Duas formas de representação (compacta ou expandida)



40

Interfaces, Classes e Componentes (II)

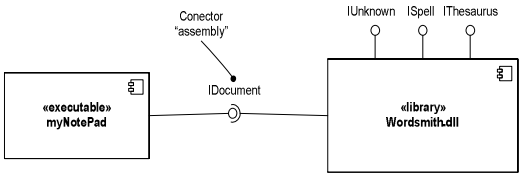
- Exemplo de relação componentes-interfaces
  - Wordsmith.dll
    - *IUnknown* : interface comum a todos os componentes ActiveX
    - *ISpell* : interface com funcionalidades para correcção ortográfica de documentos de texto
    - *IThesaurus* : interface com funcionalidades de *thesaurus* linguístico.
    - *IDocument* : interface requerida para o funcionamento do componente.

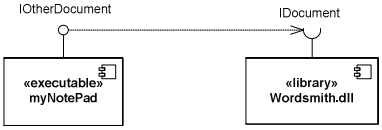


41

Interfaces, Classes e Componentes (III)

- Conector “assembly”
  - Relações claras e directas entre componentes mediadas por pares interface-providenciada e interface-requerida.





42

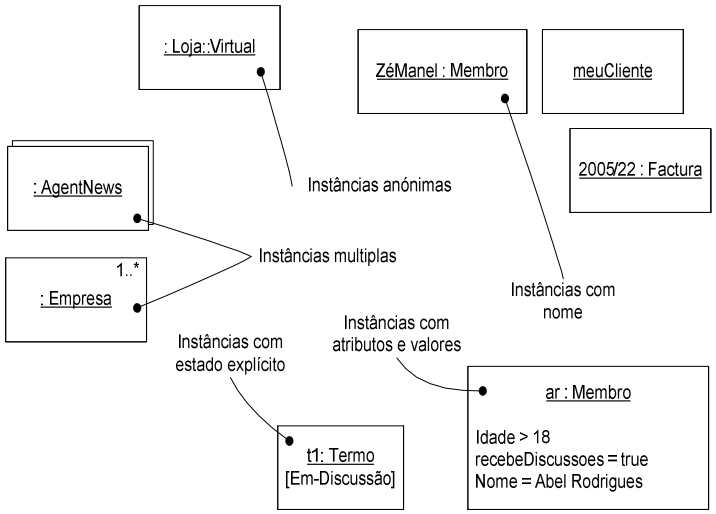
Instâncias e Objectos (I)

- Instância
  - Manifestação concreta de uma abstracção, à qual um conjunto de operações pode ser aplicado, e que tem um estado que regista os efeitos das operações realizadas.
- Objecto
  - Instância de uma classe, herdando, por conseguinte, todos os atributos e métodos definidos na própria classe.
    - Representação de execução própria : Estado
    - Identidade única no contexto da sua execução.
  - Representado por um rectangulo.
    - Nome-do-objecto : Nome-da-classe

Instâncias e Objectos (II)

- Operações
  - Objectos podem efectuar as operações definidas nas suas respectivas classes.
  - Por ser redundante, não se apresentam as operações na sua representação.
- Estado
  - Dado pelos valores assumidos pelo conjunto de atributos de um objecto.
  - É, naturalmente, um facto dinâmico, variando no tempo e no espaço, à medida que este interactua com outros objectos.

Instâncias e Objectos (III)



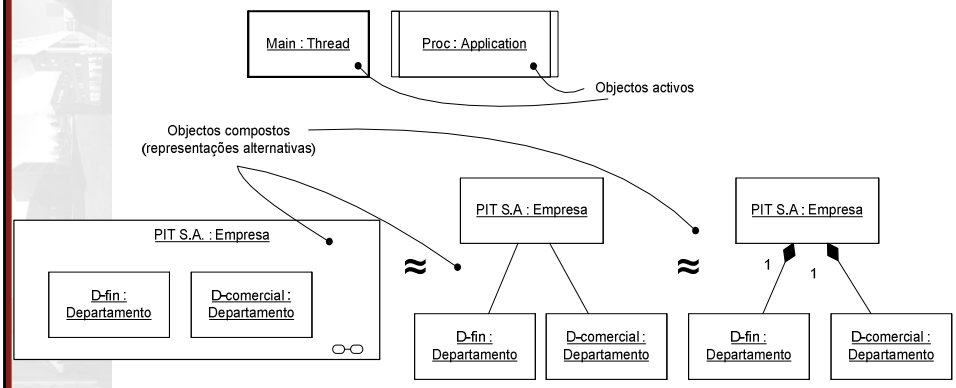
45

Instâncias e Objectos (IV)

- Objectos Activos
  - Processos, fluxos de execução, agentes de software e aplicações.
  - Apresentam uma visão dinâmica de um sistema.
  - Tem controlo sobre o seu próprio fluxo de execução
  - Gráfico: Dupla linha lateral ou rebordo com maior espessura.
- Objectos Compostos
  - Constituído por outros (sub)objectos.
  - Reflectem relações de agregação/composição entre as suas classes.

46

Instâncias e Objectos (V)



47

Dúvidas, Questões, Comentários, Opiniões?

48