Engenharia de Software III

Modelando classes
http://dl.dropbox.com/u/3025380/ES3/aula9.pdf

(flavio.ceci@unisul.br)
29/09/2011

Agenda

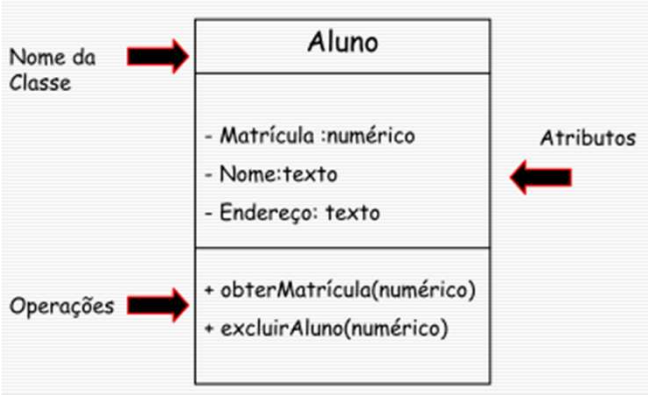
- Modelando diagramas de classe:
  - Classes, instancias, atributos...;
  - Relacionamentos:
    - Associação;
    - Herança;
    - Composição;
    - Agregação;
    - Dependência;
  - Modelo de domínio;
  - Onde paramos no Iconix?

Classes

- A classe pode ser vista como a descrição de um tipo de objeto, com propriedades semelhantes (atributos), o mesmo comportamento (operações), os mesmos relacionamentos com outros objetos e a mesma semântica.
- A classe agrupa objetos com características e comportamentos comuns !

Classes

A classe aluno de uma escola, apresenta um conjunto de alunos que apresentam as mesmas informações:



Instâncias

- A classe por sua vez deve descrever as **propriedades** e **comportamentos** daquele objeto;
- Uma classe **descreve** um grupo de objetos.
- Cada **objeto** do mundo real **pertencente** a uma **classe** é denominado **instância da classe**.

- Fulano é então uma **instância** da classe Aluno.

Atributos

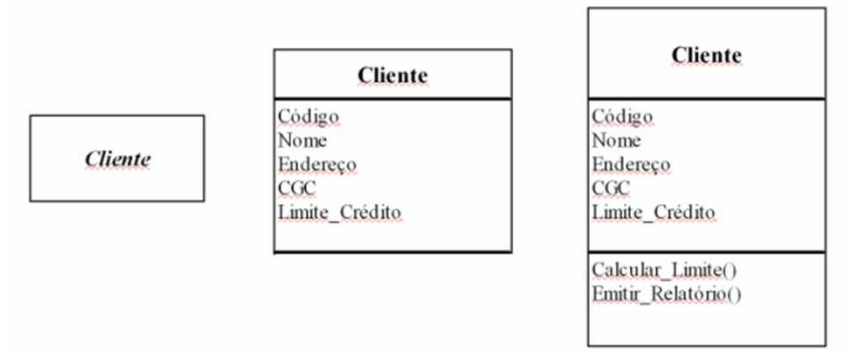
- O atributo é a descrição dos dados armazenados pelos objetos de uma classe.
- O atributo de uma classe está associado a um conjunto de valores que o atributo pode assumir.
- Atributos não tem comportamento. Cada valor de um atributo é particular para um dado objeto.
- Uma classe pode ter qualquer número de atributos ou mesmo nenhum atributo.

Operações

- As operações implementam serviços que podem ser solicitados por algum objeto da classe para modificar o comportamento.
- Todos os objetos da classe vão compartilhar destas operações.
- Para nomear a classe você deve escolher um substantivo, por exemplo: Fornecedor, Produtos, Cliente;

Notação

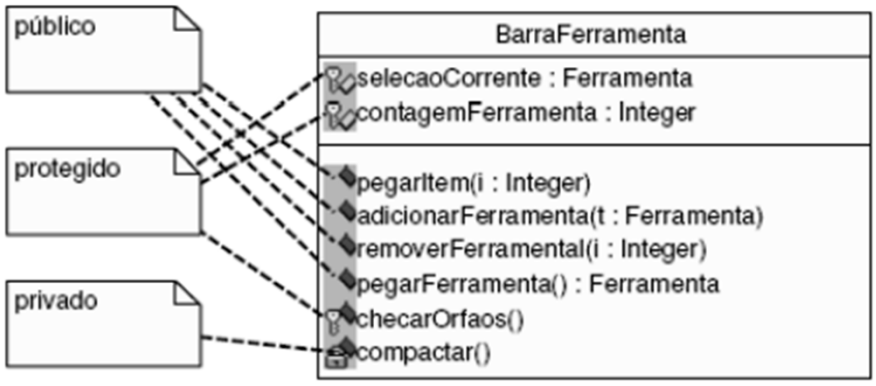
A notação de classes apresenta 3 compartimentos mas as classes podem ser apresentadas em diferentes níveis de abstração como apresentado na figura abaixo:



Visibilidade

- Publico (+) – Qualquer classificador externo com visibilidade para que determinado classificador seja capaz de usar a característica.
- Protegido (#) – Qualquer descendente do classificador é capaz de usar a característica;
- Privado (-) – Somente o próprio classificador é capaz de usar a característica;
- Pacote (~) – Somente classificadores declarados no mesmo pacote podem usar a característica.

Visibilidade



Persistência

- **Persistent** – O objeto persistente é o salvo em banco de dados ou em alguma outra estrutura de armazenamento.
- **Static** – O Objeto estático é o que permanece na memória até que o programa seja finalizado. Há, no máximo, uma instância de um objeto estático na memória em um dado momento.

Persistência

- **Transient** – O objeto transiente é o que permanece na memória por pouco tempo (até que a lógica no diagrama de seqüência tenha finalizado).

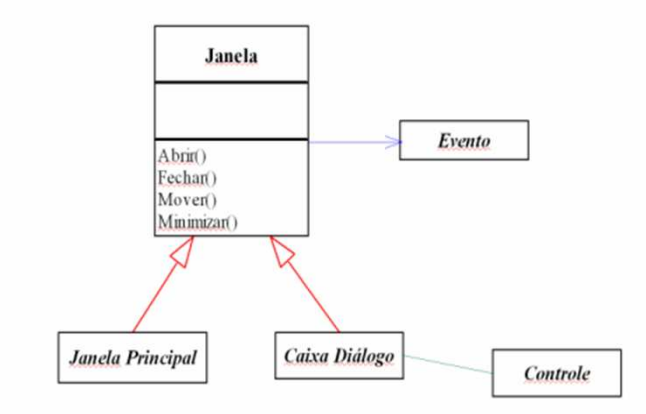
Relacionamento entre Objetos

Relacionamento entre Objetos

- Um relacionamento representa a interação entre as classes e objetos, eles apóiam o refinamento das classes.
- Existem diferentes tipos de relacionamentos possíveis entre as classes identificadas, três deles são os mais importantes as associações, as dependências e as generalizações.

Relacionamento entre Objetos

- dependência
- generalização
- associação

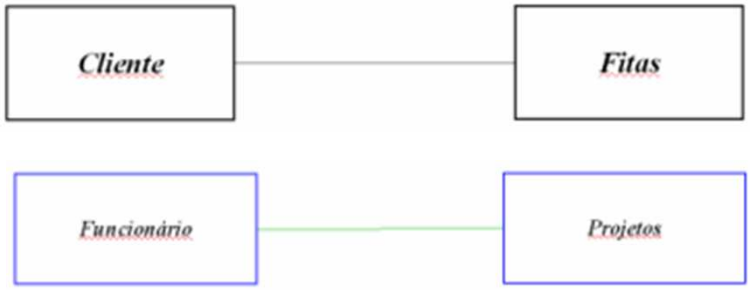


Relacionamento de Associação

- Associação é uma relação que descreve um conjunto de vínculos entre elementos de modelo.
- Quando duas classes ou mesmo uma classe, consigo própria, apresenta interdependência.
- Quando uma determinada instância de uma das classes origina ou se associa a uma ou mais instâncias da outra classe você pode dizer que temos um relacionamento de associação (Furlan, 1992).

Relacionamento de Associação

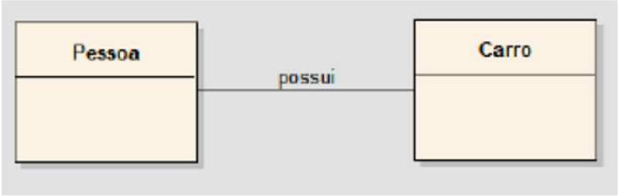
Veja: no domínio de uma vídeo locadora o cliente loca fitas, este é um relacionamento de associação. A associação entre Funcionário e Projetos significa que os **objetos da classe *Funcionário*** estão **conectados** aos **objetos** da classe ***Projetos***.



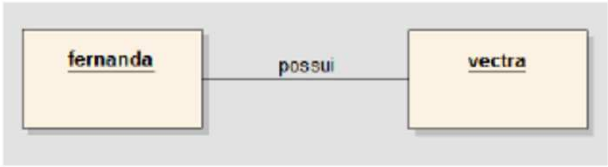
Associações e Links

- Associação:
  - Refere-se a um relacionamento representado por uma linha num Diagrama de Classes.
  - A linha pode ter um nome lógico que descreve o relacionamento.
- Link:
  - Refere-se a um relacionamento entre dois objetos mostrado num diagrama de objetos.

Associações e Links



Uma Associação entre duas classes.



Um Link entre dois objetos.

Multiplicidade

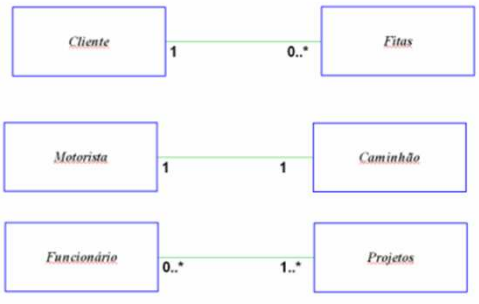
- Quando você fala em associação é possível representar a quantidade de objetos aos quais o outro objeto esta associado.
- Um exemplo prático:
  - Um projeto existe sem que seja alocado um funcionário para este projeto?
  - Quantos projetos podem ser alocados para cada funcionário?
  - Quantos funcionários podem ser alocados para cada projeto?

Multiplicidade

Nome	Simbologia
Apenas um	1
Zero ou muitos	0...*
Um ou muitos	1...*
Zero ou um	0...1
Intervalo específico	$1_i..1_s$

Exemplos de multiplicidade

- Um cliente pode alocar nenhuma ou várias fitas, mas uma fita pode estar locada por apenas um cliente.
- Em uma empresa de transporte um motorista dirige apenas um caminhão, e cada caminhão pode ser dirigido por apenas um motorista.
- No terceiro exemplo um funcionário pode estar alocado a vários projetos, por outro lado um projeto possui vários funcionários.



Nomeando uma associação

- Há várias maneiras de nomear associações, mas você deve escolher o nome pensando na descrição da natureza do relacionamento, prefira o uso de um verbo ou uma frase verbal .
- Além de indicar um nome você pode ainda indicar a direção da leitura da associação inserindo um triangulo de orientação.
- A classe cliente aloca fitas, o triângulo indica a direção que você deve ler a associação: “Cliente aloca Fitas”.

Nomeando papéis

- Além destes recursos você tem que lembrar que a classe ao participar de uma associação tem um papel específico.
- O papel é utilizado em um dos lados de uma associação para indicar o papel com que a classe a seu lado apresenta-se para a classe do lado oposto.
- Um papel define o propósito ou capacidade de uma classe, utilize substantivos para indicar os papéis.

Herança

- Demonstra como atributos e funcionalidades podem ser compartilhadas entre classe de natureza ou propósito similar.
- Um relacionamento de herança indica que objetos do elemento especializado (subclasse) podem substituir os objetos do elemento generalizado (super classe).
- A subclasse tem todos os atributos e operações da superclasse mas pode ter outros atributos e operações.

Herança

A herança trata da classificação e sub-classificação de coisas ou objetos. Considere as seguintes sentenças:

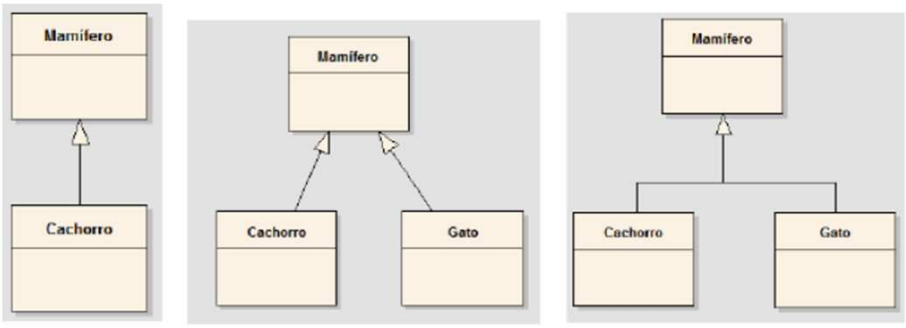
- Um labrador é um cachorro.
- Um cachorro é um mamífero.
- Um mamífero é um animal.
- Um labrador é um mamífero.
- Um labrador é um animal.
- Um labrador NÃO é um vegetal.

Herança

- Herança simples: a subclasse herda estrutura e ou comportamento de uma única superclasse.
- Herança múltipla: a subclasse herda a estrutura e o comportamento de mais de uma superclasse.

Notação UML para Herança

Seta com ponta triangular cheia apontando em direção á classe sendo estendida (super classe)

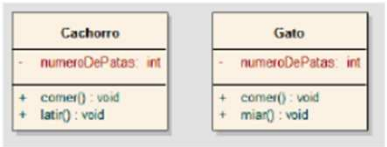


Analisando a Herança

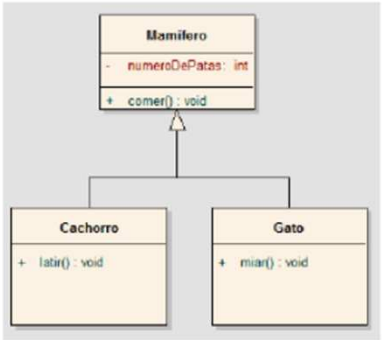
- Existem duas maneira pelas quais a herança é adicionada a um modelo:
  - Generalização;
  - Especialização.

Generalização

- Quando várias classes do Diagrama de Classes exibem funcionalidade, estrutura e protótipo comuns, pode-se aplicar o conceito de generalização.



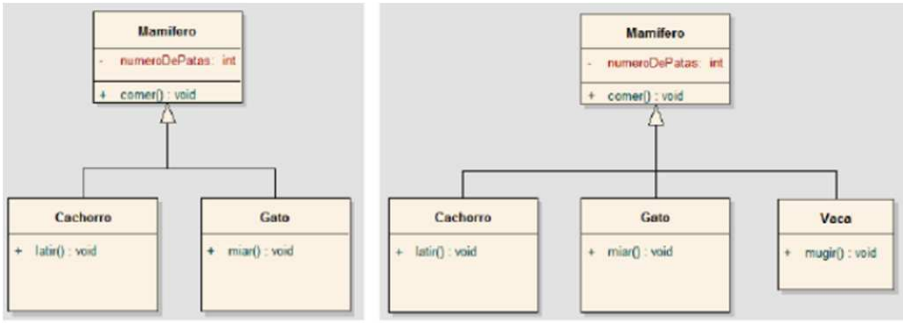
Antes



Depois

Especialização

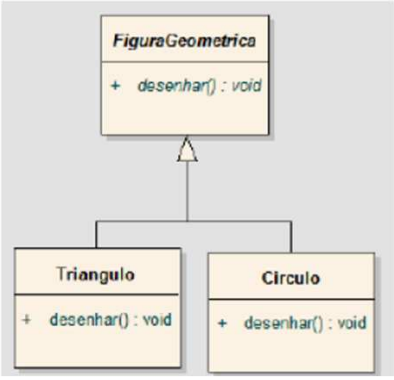
Ocorre quando uma nova classe tem toda a funcionalidade, estrutura e propósito de uma classe existente, mas requer novo código ou atributos:



Classes Abstratas

- São classes que contêm funcionalidades incompletas e não podem ser instanciadas dentro de um sistema.
- Uma classe que estende uma classe abstrata herda todos os seus métodos (incluindo os abstratos)
- Qualquer classe que herda um método abstrato, também é considerado abstrata, a menos que o método seja reescrito dentro da subclasse e totalmente implementado.
- Quando uma subclasse reescreve um método que ela herdou, é dito que ela sobrescreveu o método.

Classes Abstratas



- Os nomes das classes abstratas são escritos em *itálico*
- Quaisquer métodos abstratos daquela classe também são escritos em *itálico*
- Ou a palavra "abstract" é colocada entre chaves após o nome do método (isto é {abstract})
- Chaves após o nome do método são chamadas de lista de propriedades do método.

Relacionamento de Agregação

Agregação

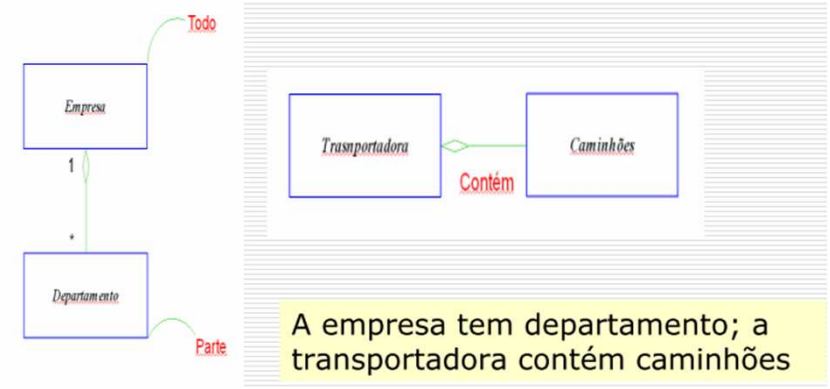
- A agregação é um caso particular da associação.
- A agregação indica que uma das classes do relacionamento **é uma parte**, ou está **contida** em outra classe.
- Mas as duas classes estão no **mesmo nível**, ou seja não existe uma classe mais importante do que a outra

Agregação

- As palavras chaves usadas para identificar uma agregação são: "***consiste em***", "***contém***" ou "***é parte de***".
- Outra dica importante é que as partes não morrem obrigatoriamente com o todo e uma mesma parte pode estar em mais de um "todo".

Agregação

- Graficamente você vai representar a associação de agregação por uma linha e um diamante aberto na extremidade.



Relacionamento de Composição

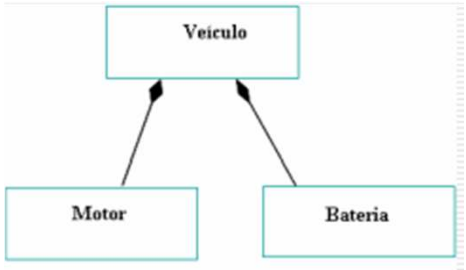
Composição

- É um tipo especial de agregação onde a multiplicidade do lado "***todo***" **é sempre 1**.
- As partes **vivem** e **morrem** obrigatoriamente **com o todo**.
- Uma mesma parte não pode estar em mais de um "todo".
- Os objetos da classe parte não existem de forma independente da classe todo.

Composição

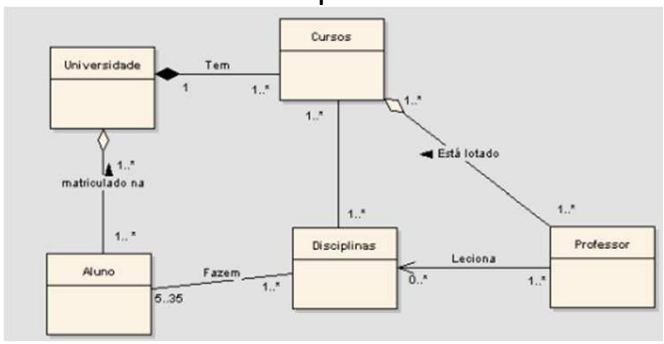
- A composição é um tipo forte de associação, onde um objeto agregado é composto de vários objetos componentes (Rumbaugh,1994).

A classe "motor" e "bateria" neste exemplo não existem de forma independente da classe veículo. Elas fazem parte do todo "Veículo".



Composição

- Na representação das classes a associação de composição exprime que o “item de pedido” não existe sem o “Pedido”, ou seja , o “item de pedido” não existe de forma independente no sistema.



Revisão sobre relacionamentos

Associação

Pode incluir relacionamentos mais fortes como Agregação e Composição

- O termo associação tem sido usado até agora para descrever o relacionamento entre duas classes num diagrama de classes.
- Uma associação, na verdade, representa um tipo de relacionamento entre duas classes e descreve a extensão pelas quais elas dependem uma da outra.

Frases de validação (relacionamentos)

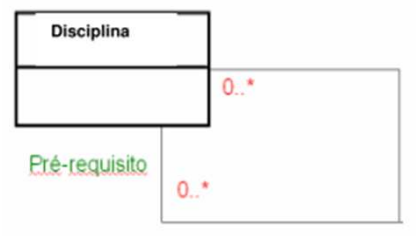
- Herança: Um labrador *é um* cão.
- Associação: Um gerente *supervisiona* um contratado.
- Agregação: Um carro *tem um* rádio.
- Composição: Um carro *sempre contém um* motor.

Código na associação

```
public class Carro {

  private Pessoa pessoa;
  //outros atributos

  public setPessoa(Pessoa pessoa)
  {
    this.pessoa = pessoa;
    //mais codigo
  }

}

// em outro lugar do código

Carro carro = new Carro(2000,6,"Ford","Scorpio");
Pessoa pessoa = new Pessoa("Homer","Simpson");
carro.setPessoa(pessoa);

//fim do código
```

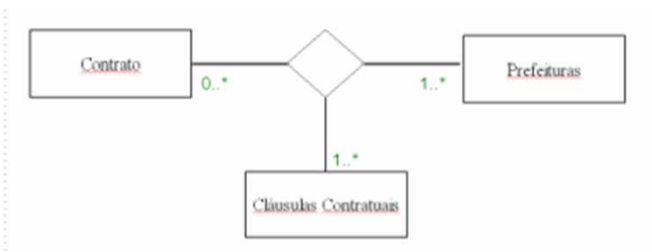
Associação Recursiva

- A associação recursiva ocorre quando objetos da própria classe estão se relacionando.
- É possível conectar uma classe a ela mesma através de uma associação representando semanticamente a conexão entre dois objetos onde os objetos conectados são da mesma classe.



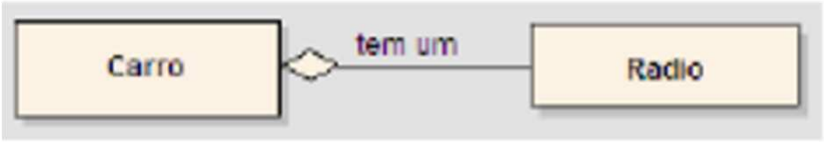
Associação Ternária

- A associação ternária associa três classes, então a associação ternária é a forma de relacionar mais do que 2 classes ao mesmo tempo.
- Ela é mostrada como um losango (diamante) e uma associação de classe ligada a ela.



Relembrando agregação

- Forma de associação
- Ênfase mais forte em como os dois objetos relacionarão dentro do sistema;
- Caracterizado por um relacionamento “tem um”;
- Conceito de todo/parte;
- Um carro tem um rádio para ser completo.

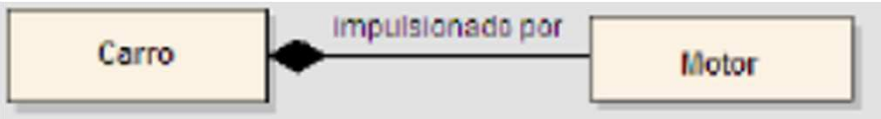


Código na agregação

```
public class Carro {

  private Motor motor;
  //outros atributos

  public Carro(Motor motor, String marca, String modelo)
  {
    this.motor = motor;
    //mais codigo
  }

}

// em outro lugar do código
Motor motor = new Motor(2000,6);
Carro carro = new Carro(motor,"Ford","Scorpio");
//fim do código
```

Relembrando composição

- A linha da associação na extremidade do “composto” é marcada por uma losango preto.
- Um carro sempre contém um motor.
- Usar composição ao invés de associação ou agregação depende de quão forte é o relacionamento.



Código composição

```
public class Carro {

  private Motor motor;

  //outros atributos

  public Carro(int tamanhoMotor, int cilindros, String marca, String modelo)
  {
    motor = new Motor(tamanhoMotor, cilindros);
    //mais codigo
  }

}

// em outro lugar do código
Carro carro = new Carro(2000,6,"Ford","Scorpio");
//fim do código
```

Relacionamento de Dependência

Relacionamento de Dependência

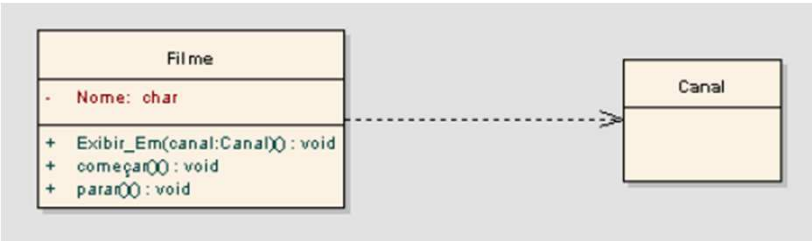
- A dependência indica a ocorrência de um relacionamento entre dois ou mais elementos onde uma classe cliente é dependente de algum serviço da classe fornecedora.
- Bezerra (2000) indica situações que levam a um relacionamento de dependência:
  - Dependência por atributo – onde a classe A possui um atributo cujo tipo é B.
  - Dependência por variável global – a classe A utiliza uma variável global cujo tipo é B.

Relacionamento de Dependência

- Bezerra (2000) indica situações que levam a um relacionamento de dependência:
  - Dependência por variável local – A possui alguma operação cuja implementação utiliza uma variável local do tipo B.
  - Dependência por parâmetro – A possui pelo menos uma operação que possui pelo menos um parâmetro cujo tipo é B.

Relacionamento de Dependência

- Para que as operações da classe Filme sejam executadas a existência da classe Canal é fundamental, pois existe uma dependência de parâmetros entre as classes.

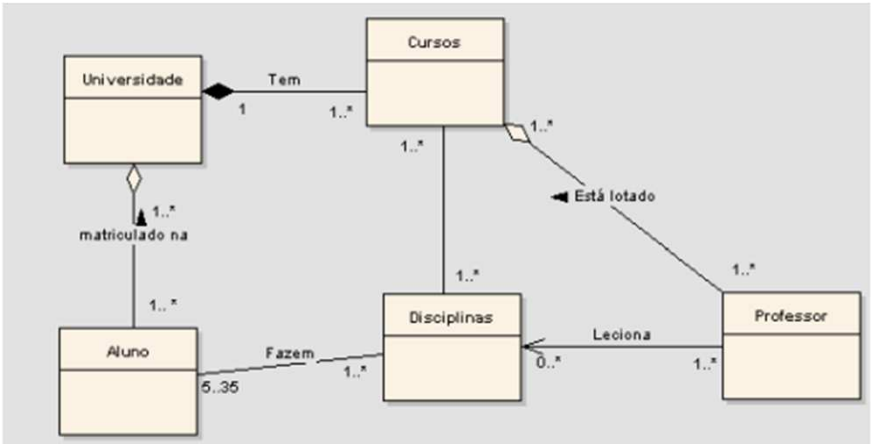


Modelo de domínio

Modelo de domínio

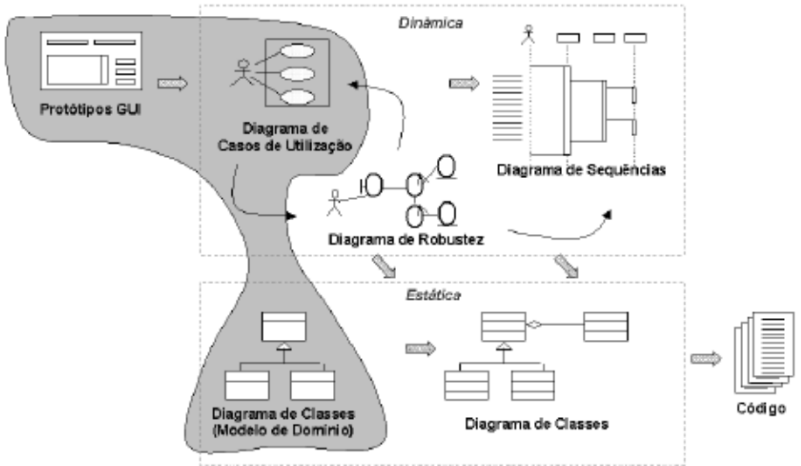
- O **modelo de domínio** representa o cenário do sistema a ser desenvolvido utilizando elementos do diagrama de classe.
- Utiliza conceitos da **orientação a objeto** para modelar o domínio em questão.
- É composto por:
  - Classes;
  - Relacionamentos;

Modelo de domínio



Onde paramos no Iconix?

- Tarefa de Análise de requisitos:



Próxima aula prova!

- Estudar exercício da aula 2;
- Estudar a teoria sobre o Iconix [aula 6];
- Estudar todos os elementos utilizados para a concepção do trabalho final até hoje;
- Rever todas as aulas.

Atividade de hoje

- A partir do cenário proposto para a utilização no trabalho final:
  - Criar um diagrama de modelo de domínio que reflita o cenário levantado;
  - Utilize a notação do diagrama de classe vista na aula de hoje:
    - Classes;
    - Cardinalidade;
    - e Relacionamentos.