UML
Conceito de Classe e Objeto (Aula 2)

José Augusto Fabri

Conceito de Classe

- Classe – Principal primitiva ou elemento de composição de software – Projetos OO.
- Em OO, um software é composto por classe e objetos.
- As classes e objetos colaboram com a execução dos serviços ou usos do sistema (casos de uso).

Conceito de Objeto

- O termo objeto possui duas visões:
  - Conceitual;
  - Implementação.

- Conceitual:
  - O objeto é um elemento componente de um sistema computacional que representa alguma entidade ou “coisa” do mundo real.

Conceito de Objeto – Visão Conceitual

- Exemplo do sistema de controle acadêmico (mundo real):
  - Entidades reais: Secretária, Professores, Alunos, Diários de Classes.
  - Entidades abstratas: Disciplina, Notas e Freqüência.
  - Quando se desenvolve um sistema OO procura-se criar no interior do sw representações das entidades externas na forma de objetos.
  - Assim, poderiam ser criados objetos como:
    - Aluno
    - Diário de classe
    - Disciplina, etc.

Objetivo: representar as entidades Externas dentro do sistema.

Conceito de Objeto – Visão Conceitual

- Intenção da visão conceitual: aproximar representações internas (relação e existência) com as entidades do mundo real.

Conceito de Objeto - Visão Conceitual

- Objetos não representam apenas informações, eles podem implementar processos (algoritmos ou procedimentos).
  - Exemplo: Trabalho manual de composição de um diário de classe (trabalho, processo), possibilidade de se criar um *objeto* responsável por este comportamento.
- Entidades no mundo real sempre se relacionam para realização de tarefa.
  - Exemplo: Dados de alunos (entidade) são utilizados pela secretaria (entidade) para compor (processo) o diário de classe (entidade).

Conceito de Objeto - Visão Conceitual

- Importante: Os objetos que representam as entidades irão se relacionar para juntos cooperarem na execução de um serviço (uso do sistema – caso de uso). Exemplo:
  - Comunicação: um objeto pode trocar informações para comandar outro objeto.
- Características de um objeto:
  - **Identidade**: nome ou identificador que distingue dos demais objetos e que permite que outros objetos o reconheçam e o enderecem.
  - **Atributos**: cada objeto possui informações que são registradas na forma de um conjunto de atributos.
  - **Comportamentos**: cada objeto possui um conjunto de habilidades de processamento que juntas descrevem seu comportamento.

Conceito de Objeto - Visão Conceitual

- Característica de um Objeto (Exemplo)
  - Considere um objeto que represente uma disciplina de matemática.
  - **Identidade**: Calculo numérico
  - **Atributos**: Carga Horária, Pré-requisitos, Ementa, Alunos Matriculados e Notas
  - **Comportamento**: Calcular a Média da Turma, Inscrever um Novo Aluno e Registrar Notas dos Alunos.

Conceito de Objeto - Visão de Implementação

- Objeto pode ser entendido como um pequeno módulo de software.
- Objeto deve possuir um alto grau de modularidade (pequena dependência entre os módulos).
- Linguagens como C e Pascal constroem:
  - Programas como conjuntos de funções e variáveis.
- Linguagens OO constroem:
  - Programas como agrupamento de funções e variáveis.

Conceito de Objeto - Visão de Implementação

- Um objeto pode interagir com outros objetos:
  - Comunicação: realizada através de chamada de *função* ou *procedimento* do objeto para algum outro. Tal função é responsável pela troca de mensagens.
- Execução de Software OO – execução em cadeia ou em concorrência do conjunto de seus objetos que, permanentemente, estariam se comunicando (troca de informações nas funções).
- Execução de Software OO – Colaboração de um conjunto de objetos.

Conceito de Objeto - Visão de Implementação

- Tradução das três características de um objeto para a visão de implementação:
  - **Identidade**: Um objeto que representa uma disciplina de matemática poderia ser nomeado “objMatematica”.
    - Importante: Respeitar as normas para criação de identificadores da linguagem de programação utilizada.
  - **Atributos**: são traduzidos na forma de variáveis internas do objeto. Por exemplo, um objeto representando um ALUNO poderia ter os atributos: “nome” (texto com 30 caracteres), “sexo” (um caractere M ou F), “coeficiente de rendimento” (valor real).
  - Não confundir atributo com valor do atributo. Aqui, “nome” é um atributo cujo valor pode ser “Maria” ou “José”.

Conceito de Objeto - Visão de Implementação

- **Funções** ou **Procedimentos**: definem o comportamento do objeto.
- As funções descrevem os procedimentos ou habilidades do objeto.
- Por exemplo: o objMatematica poderia ter uma função para cálculo da média das notas dos alunos matriculados. objMatematica.calcularMediaNotas().

Conceito de Classe

- Classes:
  - Fundamentais na composição de software orientado a objetos (projeto e implementação).
  - Classe não traduz o conceito de elemento executor, no código de implementação, presente nos objetos.
- Duas visões:
  - Visão conceitual
  - Visão de implementação

Conceito de Classe – Visão Conceitual

- Classes podem ser entendidas com descrições genéricas ou coletivas de entidades do mundo real (aproximação das representações internas e externas)
- As classes são representações genéricas e os objetos representações individualizadas.
- As classes são abstrações coletivas de um mundo real.
- Elas representam um modelo comum para um conjunto de entidades semelhantes.

Conceito de Classe – Visão Conceitual

- Exemplo:
  - Alunos envolvidos no sistema acadêmico:
    - Cada aluno: *Márcia*, *João*, *Cláudia*, etc. É uma entidade individualizada, que poderia ser representada por um **objeto**.
    - Observando os objetos e comparando-os, pode-se constatar que o conjunto de seus **atributos** (nome, sexo, idade) e seus comportamentos são análogos, embora com valores de atributos diferentes.
  - A partir da observação de atributos e comportamentos comum a um conjunto de entidades similares, é possível estabelecer um modelo genérico para este conjunto de entidades.

Conceito de Classe – Visão Conceitual

- Este modelo conteria os atributos e comportamento comuns a este coletivo.
- Desta forma seria possível definir um modelo genérico para todos os alunos, no exemplo anterior.
- Esse modelo genérico é uma abstração dos alunos que denomina-se CLASSE.

O que é uma CLASSE?

"Imagine que eu seja um engenheiro civil. Tive a incumbência de construir um prédio de apartamentos. Todos os apartamentos são rigorosamente iguais, baseados em uma planta que anteriormente já esclarecia o número de dormitórios, portas, louças sanitárias, fiação elétrica e disposição hidráulica".

A planta dos apartamentos é a CLASSE. Os apartamentos são os objetos baseados naquela CLASSE.

Conceito de Classe – Visão Conceitual

- Classe é um modelo genérico porque estabelece um formato padrão para a representação, na forma de objetos, de todas as entidades externas associadas.
- Classe estabelece comportamento comum a um conjunto de entidades similares:
  - É o estabelecimento de um modelo genérico coletivo para tais entidades.
- No exemplo anterior temos:
  - A Classe Aluno.
- A definição das classes de um sistema passa, necessariamente, pela observação das entidades externas.
  - Objetivo: Buscar entidades semelhantes (coletivos) que possam ser abstraídas em uma representação genérica.

Conceito de Classe – Visão de Implementação

- Implementação através da criação de tipos - Semelhança aos tipo de dados.
- Exemplo de Tipos de Dados:
  - Inteiro
  - Real
  - Booleano.
- Os tipos de dados permitem que o programador crie novas estruturas, por exemplo registros (struct).

Conceito de Classe – Visão de Implementação

```
/* definição de um tipo de dado*/
struct Aluno{
    int codigo;
    char nome[30];
    char sexo;
}
int valor;
aluno a1;
```

Conceito de Classe – Visão de Implementação

- Classes são tipos e não podem ser considerados elementos executáveis ou de armazenamento de dados dentro de um programa.
- Classes são modelos para criação de variáveis.
- A partir da criação de uma classe é possível criar instâncias ou variáveis desta classe.
- As variáveis criadas a partir de uma classe são chamadas "**objetos**".
- *Não esquecer* que na visão conceitual - Classes são consideradas modelos para os objetos semelhantes.
  - Em termos de implementação, uma classe, é um modelo (ou tipo usado para criação de objetos.

Conceito de Classe – Visão de Implementação

- Na OO
  - Todo objeto deve ser criado a partir de uma classe.
- Declaração da classe deve ocorrer antes da criação do objeto.
- Exemplo:

```
Classe cAluno{
        char nome[30];
        char sexo;
public:
        int codigo;
        void definir (char *n, char s, int cod);
}
```

Conceito de Classe – Visão de Implementação

```
void cAluno::definir(char *n, char s, int cod){
      strcpy(nome, n)
      sexo = s;
      codigo = cod;
}

/* definição de objetos */
cAluno a1;
cAluno estudante;
```

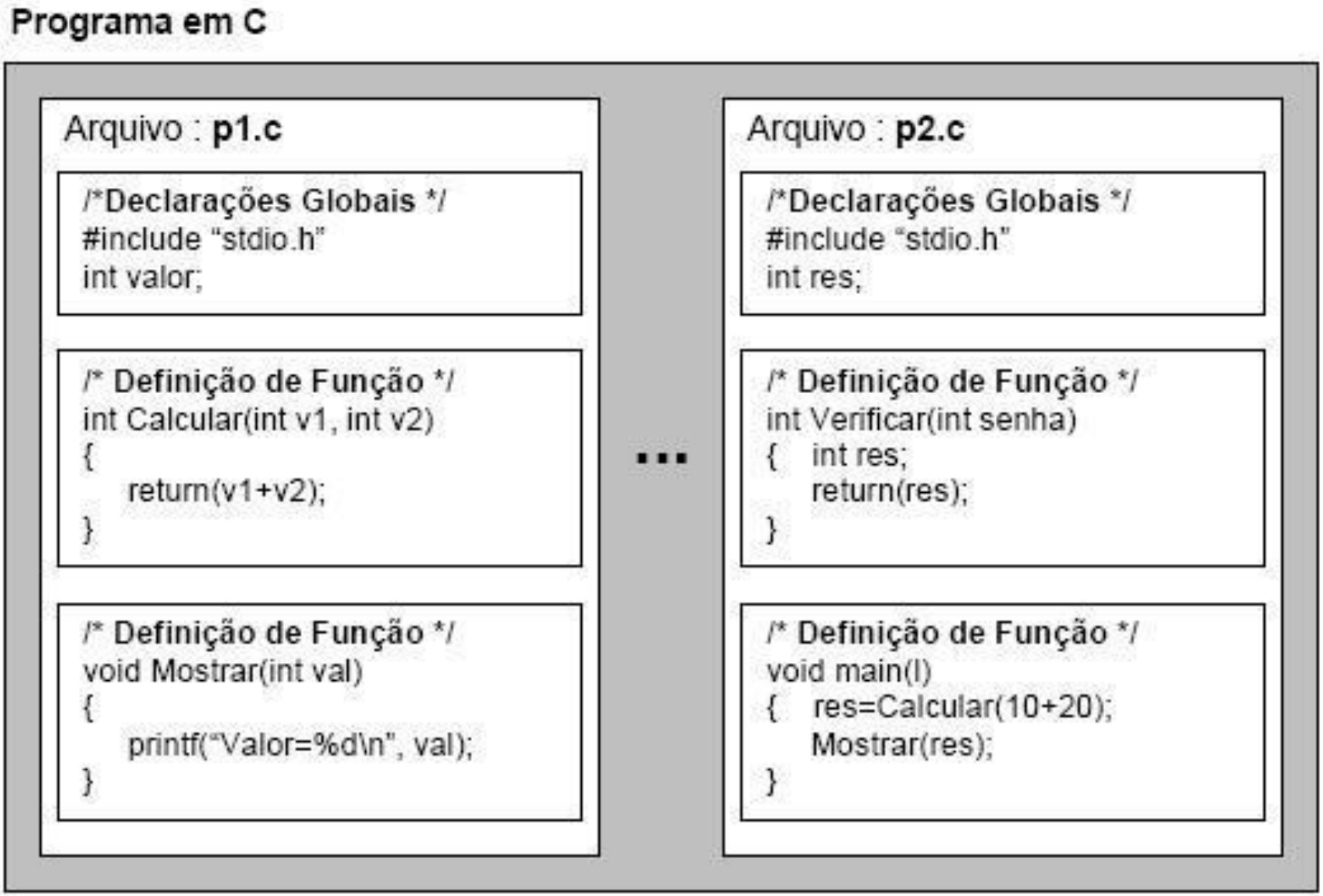
Conceito de Classe – Visão de Implementação

- O programa ilustra a declaração de uma classe
  - (cAluno)
- E a definição de dois objetos:
  - a1
  - estudante
- Observa-se que a classe cAluno possui 03 atributos
  - Nome
  - Sexo
  - Código
- E uma função:
  - definir

Conceito de Classe – Visão de Implementação

- Em termos de implementação pode-se dizer que OO introduziu um nível adicional de estruturação nos programas.
- Programação estruturada – 3 níveis:
  - Instrução
  - procedimentos (funções)
  - arquivos (.c, .h, .pas, etc);
- O fonte é distribuído em arquivos, que contém conjunto de funções, que finalmente são descritas por instruções.

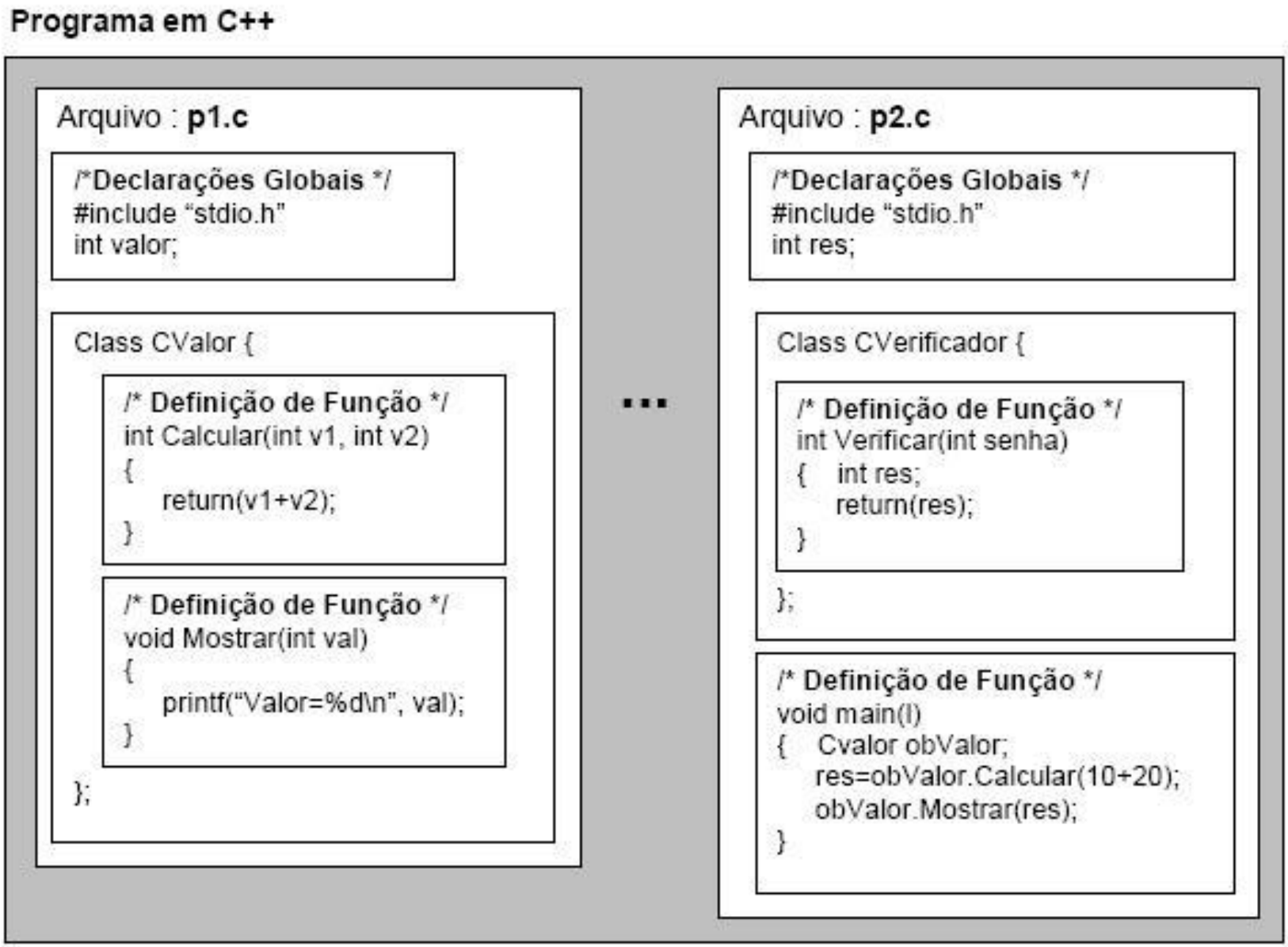
Conceito de Classe – Visão de Implementação



Conceito de Classe – Visão de Implementação

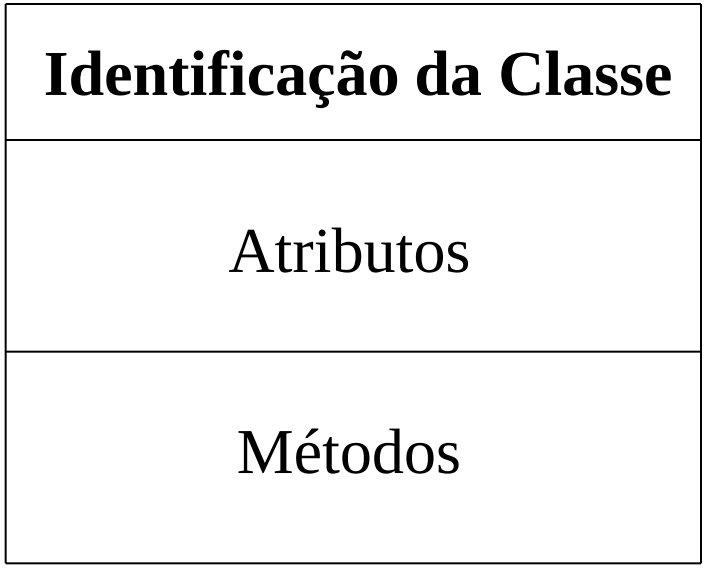
- Programação orientada a objetos:
  - Os programas são organizados em Arquivos;
  - Arquivos são organizados em classes;
  - Classes incluem funções que, por sua vez, incluem instruções.

Conceito de Classe – Visão de Implementação

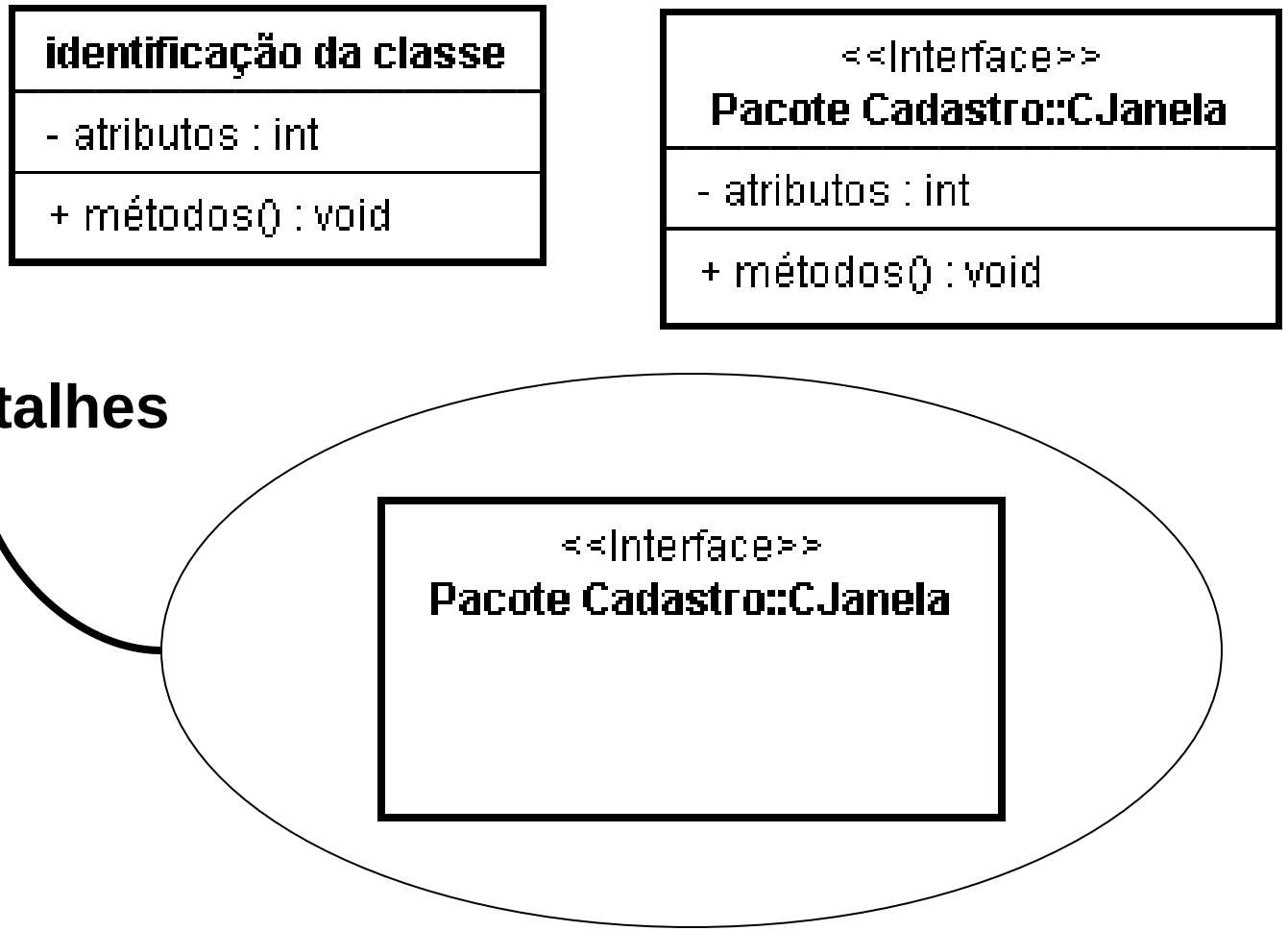


Notação UML para Classes

- Retângulo com três compartimentos:
  - Identificação;
  - Definição de Atributos;
  - Definição de Métodos.



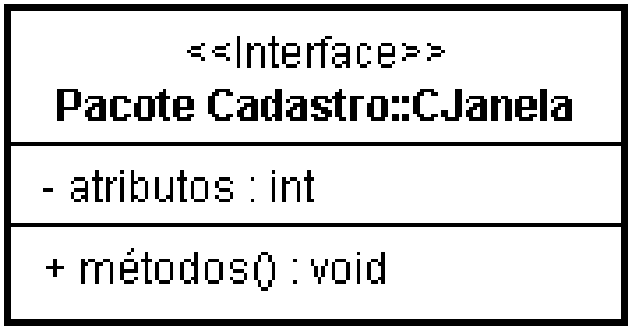
Notação UML para Classes



Notação UML para Classes

Compartimento de *Identificação*:

- **Nome da Classe**:
  - Exemplo:
    - cAluno
    - cTurmaEspecial
    - cJanelaDeInterfaceComUsuário.

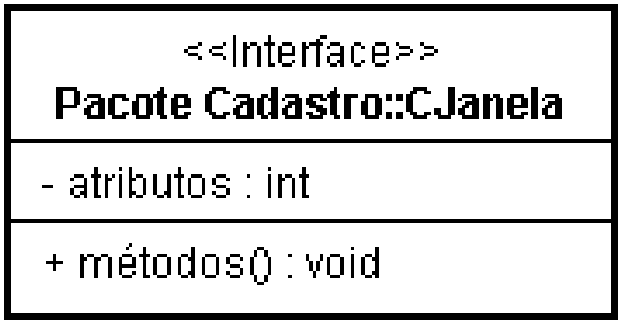


- **Estereótipo** (opcional) – são utilizados para indicar tipos de classes. Pode-se por exemplo criar classe do tipo:
  - **Interface** ou
  - **Controle**.
  - Identifica-se o **estereótipo** entre os sinais **<< >>**
- **Encapsulamento**: Pacote dentro do qual a Classe está declarada:
  - A especificação do pacote é incluída abaixo do nome da classe e escrita em *itálico* na forma "*de nome-do-pacote*"

Notação UML para Classes

Compartimento de definição de *Atributos*:

- Define os atributos da classe.
- Atributos são **variáveis membro da classe**.
- Os atributos podem ser usados por todos os métodos da classe - Leitura e Escrita.
- Notação UML - similar a linguagem de programação.



Notação UML para Classes

Compartimento de definição de *Atributos*:

- Notação:

  [Visibilidade] Nome-Atributo [Multiplicidade]:[Tipo]=[Valor]
  [{Propriedades}]

- A visibilidade indica se o atributo pode ou não ser acessado do exterior da classe por funções que não sejam membro da classe. Existem 3 especificadores de Visibilidade:
- Visibilidade:

  + Indica visibilidade pública. O atributo é visível no exterior da classe. Tanto funções (Métodos) membros da classe quanto funções externas podem acessar o a atributo.

Notação UML para Classes

Compartimento de definição de *Atributos*:

- Visibilidade (Continuação)

  - Indica visibilidade privada. O atributo não é visível no exterior da classe. Somente funções dentro da classe podem acessar o atributo.

  # Indica visibilidade protegida. Nesse caso o atributo também é privado mas funções membro de classes derivadas também têm acesso a atributo.

- A definição de visibilidade do atributo é opcional. Entretanto, se ela não for especificada, assume-se por padrão visibilidade privada.

Notação UML para Classes

Compartimento de definição de *Atributos*:

- Nome do atributo: Cadeia de caracteres que identificam, exclusivamente, um atributo dentro da classe, exemplo:
  - valorPagamento;
  - nomeDoAluno.
- Multiplicidade: (opcional) - Trata o conceito de vetores.
  - vetor[5];
  - matriz[10,20].

[Visibilidade] Nome-Atributo [Multiplicidade]:[Tipo]=[Valor] [{Propriedades}]

Notação UML para Classes

Compartimento de definição de *Atributos*:

- Tipo: Classificador que indica o formato (tipo de dado), geralmente, ligados aos tipos da linguagem de programação.
  - resultado : int;
  - salario : float;
  - sexo : char

[Visibilidade] Nome-Atributo [Multiplicidade]:[Tipo]=[Valor] [{Propriedades}]

Notação UML para Classes

Compartimento de definição de *Atributos*:

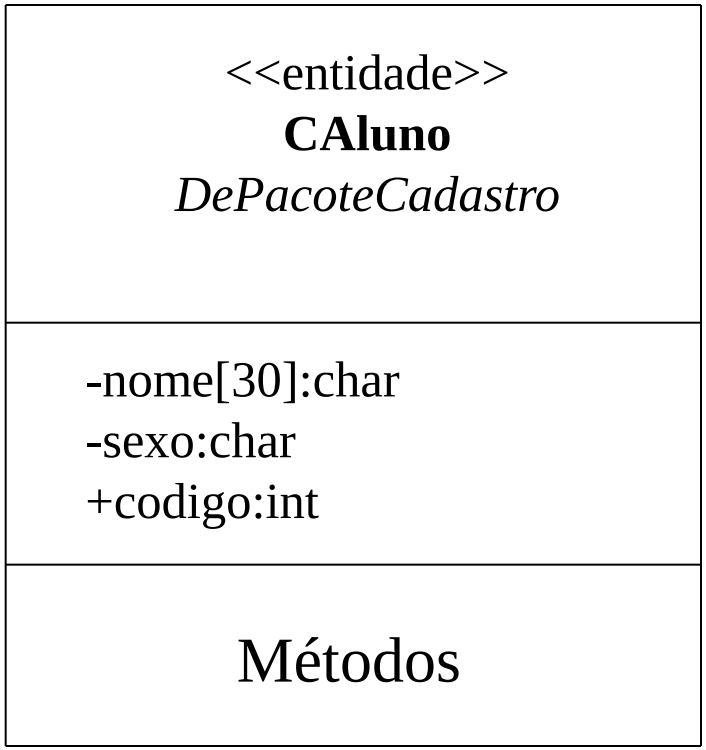
- Valor inicial: Indica o valor do atributo, imediatamente, após sua criação.
  - resultado : int = 10
- Propriedades: Podem ser utilizadas para incluir comentários ou outras indicações sobre o atributo, por exemplo se seu valor é constante.

[Visibilidade] Nome-Atributo [Multiplicidade]:[Tipo]=[Valor] [{Propriedades}]

- sNome [30] : char = ‘Heloísa’ {blá-blá-blá}

Notação UML para Classes

Exemplo de definição de atributos de uma classe:



Notação UML para Classes

Compartimento de definição de *Métodos*:

- Neste compartimento são declarados os métodos (funções) que a classe possui.
- Notação para definição dos atributos:

[Visibilidade] Nome-Método (parâmetros) : [Valor-de-Retorno] [{propriedades}]

- **Visibilidade**: indica se o método pode ou não ser chamado do exterior da classe por funções que não sejam membro da classe.

  + visibilidade pública: O método é visível no exterior da classe. Funções da classe e funções externas podem acessar o método.

  - visibilidade privada: Método não é visível no exterior da classe. Somente funções membro da classe podem acessar ao método.

  # visibilidade protegida: Funções membro de classe derivada também tem acesso ao método.

A visibilidade é opcional. Entretanto, se ela não for especificada, assume-se **visibilidade privada.**

Notação UML para Classes

Compartimento de definição de *Métodos*:

- Nome do método:
  - Cadeia de caracteres que identificam, exclusivamente, o método dentro da classe. Exemplo:
    - CalculaValor
    - ArmazenarDados.
- Parâmetros:
  - Variáveis definidas junto aos métodos. São utilizados pelos métodos para o recebimento de valores no momento da ativação.
  - Os parâmetros podem também ser utilizados para retorno de valores após a execução do método.

Notação UML para Classes

Compartimento de definição de *Métodos*:

- Parâmetros:
  - Cada parâmetro é especificado usando uma notação:

    Nome-do-Parâmetro:Tipo=Valor-Padrão

    - Nome-do-Parâmetro: Cadeia de caracteres que identifica o parâmetro dentro do método.
    - Tipo: especificador de dado padrão da linguagem de programação.
    - Valor-Padrão: Constante cujo o valor é atribuído ao parâmetro se em uma chamada ao método não for especificado um valor para o parâmetro.Exemplo:

    CalcularValor(val1:int, val2:float = 10.0)

    ArmazenarDados(nome:char[30], salario:float=0.0)

[Visibilidade] Nome-Método (parâmetros) : [Valor-de-Retorno] [{propriedades}]

Notação UML para Classes

Compartimento de definição de *Métodos*:

- Valor-de-Retorno
  - Indica se o método retorna algum valor ao término de sua execução.
  - Qual o tipo de dado retornado.
  - Exemplo:

CalcularValor():int //retorna um valor inteiro

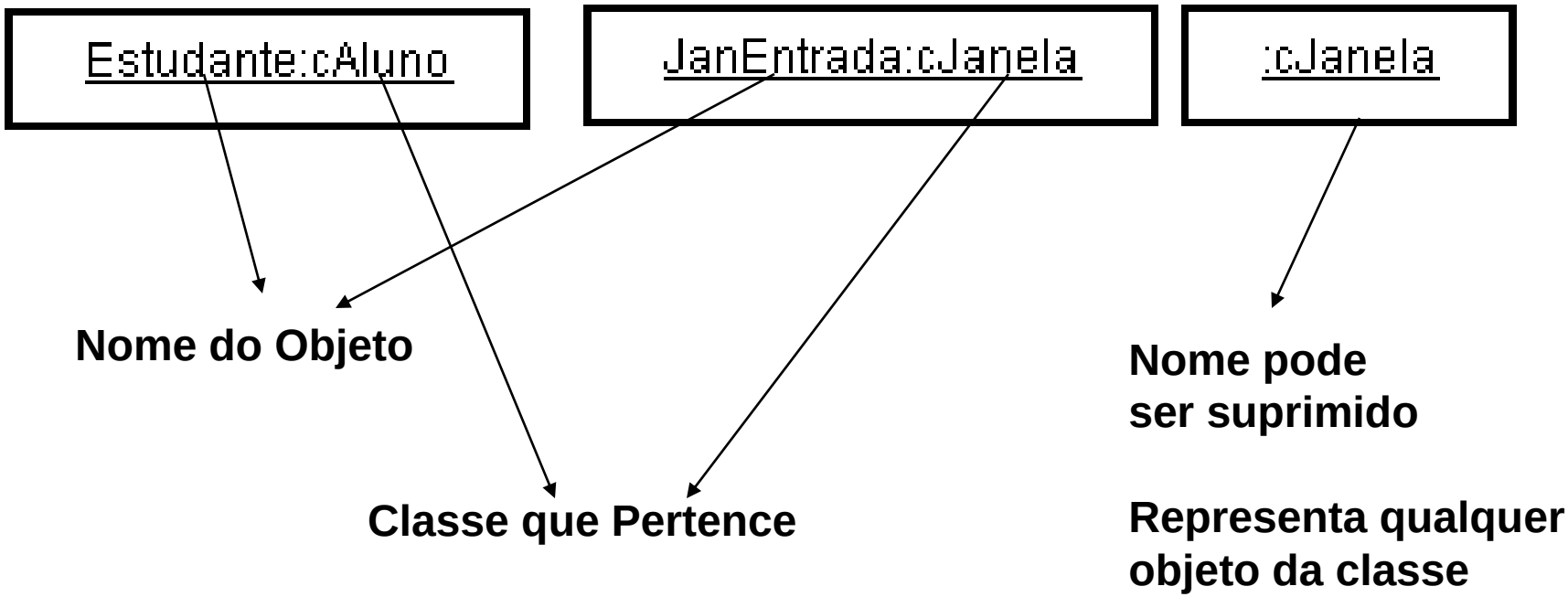
ArmazenarDados(nome:char[30]):boolean //retorna verdadeiro ou falso

+ ArmazenarDados(Nome : String, Código : int) : boolean

[Visibilidade] Nome-Método (parâmetros) : [Valor-de-Retorno] [{propriedades}]

Notação UML para Objetos

- Retângulo com a identificação do objeto em seu interior.
- Identificação do objeto, seguida de : e a indicação da classe.
- Esta identificação deve ser sublinhada.
- O nome do objeto é opcional.
- Quando o nome não for indicado entende-se que se está fazendo referência a um objeto qualquer de determinada classe.



Levantamento de Classes

- Uma vez identificados os atores e casos de uso do sistema, pode-se iniciar, efetivamente, o projeto do software.
- Do ponto de vista estrutural, o projeto deve definir os componentes do sistema e as relações necessárias entre eles.
- Em um sistema OO, os componentes estruturais são as classes.

Levantamento de Classes

- A definição das classes para a realização de todas as funcionalidades do produto envolve um processo de criação.
- Não existe um algoritmo o técnica para o estabelecimento das classes.
- Cabe ao projetista, baseado em suas experiências, determinar as classes que irão compor o produto (existe um conjunto de técnicas).
- A definição das classe exige um estudo detalhado dos requisitos e das possibilidade de separação dos dados e processos em classes.

Levantamento de Classes

- Técnicas:
  - Definir as classes por partes (por caso de uso);
  - Proceder refinamentos;
  - Utilizar estereótipos.

1. Definição das classes por caso de uso.
   - Divisão do sistema em módulos ou subsistemas.
   - Utilizar cada caso de uso para definição das classes.
   - Cuidado: não se deve considerar os casos de usos como subsistema. Uma classe pode ser empregada em mais de um caso de uso.

Levantamento de Classes

2. Definição das classes por refinamentos.
   - Definir, inicialmente, as grandes classes (chamadas de "classes de análise").
   - Em seguida retorna-se para decompor estas grandes classes em classes mais detalhadas e menores.
   - Abordagem *top-down*:
     - *Parte-se de classes mais abstratas em direção a classes mais refinadas ou detalhadas.*

Levantamento de Classes

3. Estereótipo para classes.

UML define 3 tipos estereótipos padrões para classes de análises:

- Entidade: identifica classes cujo papel principal é armazenar dados que juntos possuem uma identidade. Este tipo de classe representa entidades do mundo real:
  - Aluno, Professor, Disciplina.
- Controle: identifica classes cujo papel é controlar a execução de processos. Estas classes contém o fluxo de execução de todo ou de parte de casos de uso e comandam outras classes na execução de procedimentos.

Levantamento de Classes

- Fronteira: identifica classes cujo papel é realizar o "interfaceamento" com entidades externas (atores). Classes que contém um protocolo para comunicação com atores.
  - Exemplo: Impressora, monitor, teclado, disco, porta serial, modem, etc.

Levantamento de Classes

- Regras utilizadas no levantamento de classes:
  - Regras para a definição de classes para um caso de uso:
    - Definir uma classe do tipo fronteira para cada ator que participa do caso de uso.
      - Ator: entidade externa.
      - Cada entidade externa pode exigir um protocolo próprio para comunicação.
      - Criação de uma classe específica para tratar a comunicação de cada ator com o sistema.
      - Exemplo: Classe para interface com o usuário; Classe para interface com a impressora; Classe para interface com a porta serial, etc.

Levantamento de Classes

- Continuação
  - Definir, pelo menos, uma classe de controle para cada caso de uso.
    - O algoritmo ou conhecimento do processo a ser realizado deverá estar definido em alguma parte do sistema.
    - Este conhecimento está distribuído em vários componentes do sistema, ou centralizado em um ou em poucos componentes.
    - A vantagem de centralizar o processo em um ou poucos componentes é a maior facilidade de entendimento do processo e sua modificação futura.
    - Assim pode-se criar uma classe de controle para cada caso de uso de forma que ela contenha a descrição e o comando do processo associado ao caso de uso.
  - Definir classes auxiliares.
    - Processo nas classes de controle pode ser longo ou conter sub-processos.
    - Pode-se extrair partes do processo de controle da classe principal é atribuí-los a classes de controle auxiliares.

Levantamento de Classes

- Continuação
  - Definir uma classe do tipo entidade para cada grupo de dados.
    - Para cada caso de uso fazer uma análise de quais dados são manipulados pelos mesmos, identificando grupos.
    - Os dados a serem manipulados pelos casos de uso devem estar instanciados em memória.
    - Grupo de dados armazenados em arquivos ou bancos de dados não precisam possuir classes que os representem na totalidade.
    - Classes do tipo entidade estão associadas, unicamente, a dados em memória.

    - **<<classes são diferentes de tabelas>>**

Exemplo de Levantamento de Classes

- Levantamento de Classes para dois casos de uso:
  - Cadastrar aluno;
  - Emitir diário de classes.
- Classes para o caso de uso cadastrar aluno.

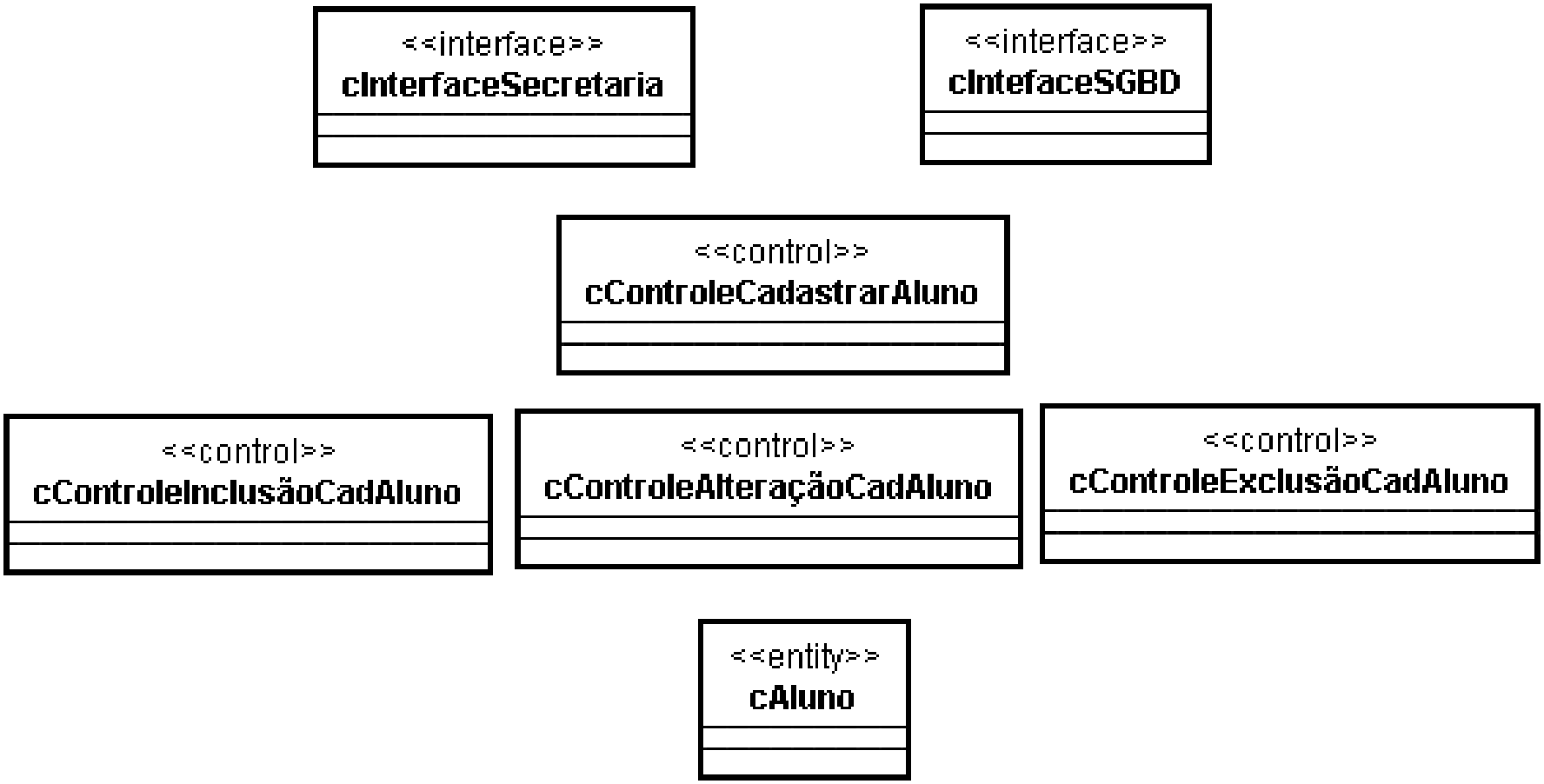
Exemplo de Levantamento de Classes

- Envolvimento de dois atores: Secretaria e SGBD.
  - Duas classes do tipo fronteira para implementar o interfaceamento com estes dois atores :
    - cInterfaceSecretaria e cInterfaceSGBD
  - Para descrever o processo do caso de uso e comandar as demais classes será definida uma classe de controle denominada:
    - cControleCadastrarAluno.
  - Com esse caso de uso existem 3 subprocessos, possibilidade de criação de classes auxiliares:
    - cControleInclusãoCadAluno
    - cControleAlteraçãoCadAluno e
    - cControleExclusãoCadAluno

Exemplo de Levantamento de Classes

- Por fim, analisando os dados manipulados pelo caso de uso, percebe-se a existência de apenas um grupo de dados:
  - Informações do aluno que está sendo incluído, alterado ou excluído.
  - Desta forma apenas uma classe do tipo entidade será definida: cAluno

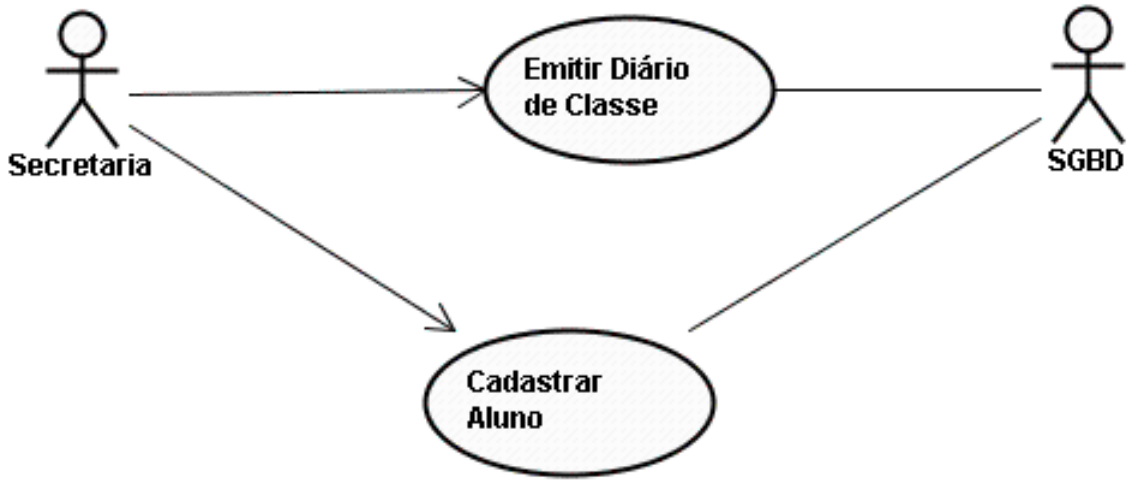
Exemplo de Levantamento de Classes



Classes para o caso de uso Cadastrar Aluno

Exemplo de Levantamento de Classes

- Classes para o Caso de Uso Emitir Diário de Classe



Exemplo de Levantamento de Classes

- Emitir Diário de Classe envolve a comunicação de dois atores:
  - Secretária e SGBD.
  - Duas classes do tipo fronteira:
    - cInterfaceSecretaria e cInterfaceSGBD
- Lembre-se que estas 2 Classes já foram definidas.
- Possibilidade de utilizar as mesmas classes ou;
- Definir novas classes para o interfaceamento com os dois atores.
- Decisão: Depende da dimensão que alcançariam estas classes se agrupassem o interfaceamento de ambos casos de uso.
- Sugestão: Solução unificada, e na seqüência do projeto, se necessário, decompor estas classes em classes mais especializadas.

Exemplo de Levantamento de Classes

- Para descrever o processo do caso de uso e comandar as demais classes será definida uma classe de controle denominada cControleEmitirDiário.
- Não se observa, neste momento, subprocessos para neste caso de uso.
- Tipo de entidade: Quais os dados que devem ser impressos no documento?

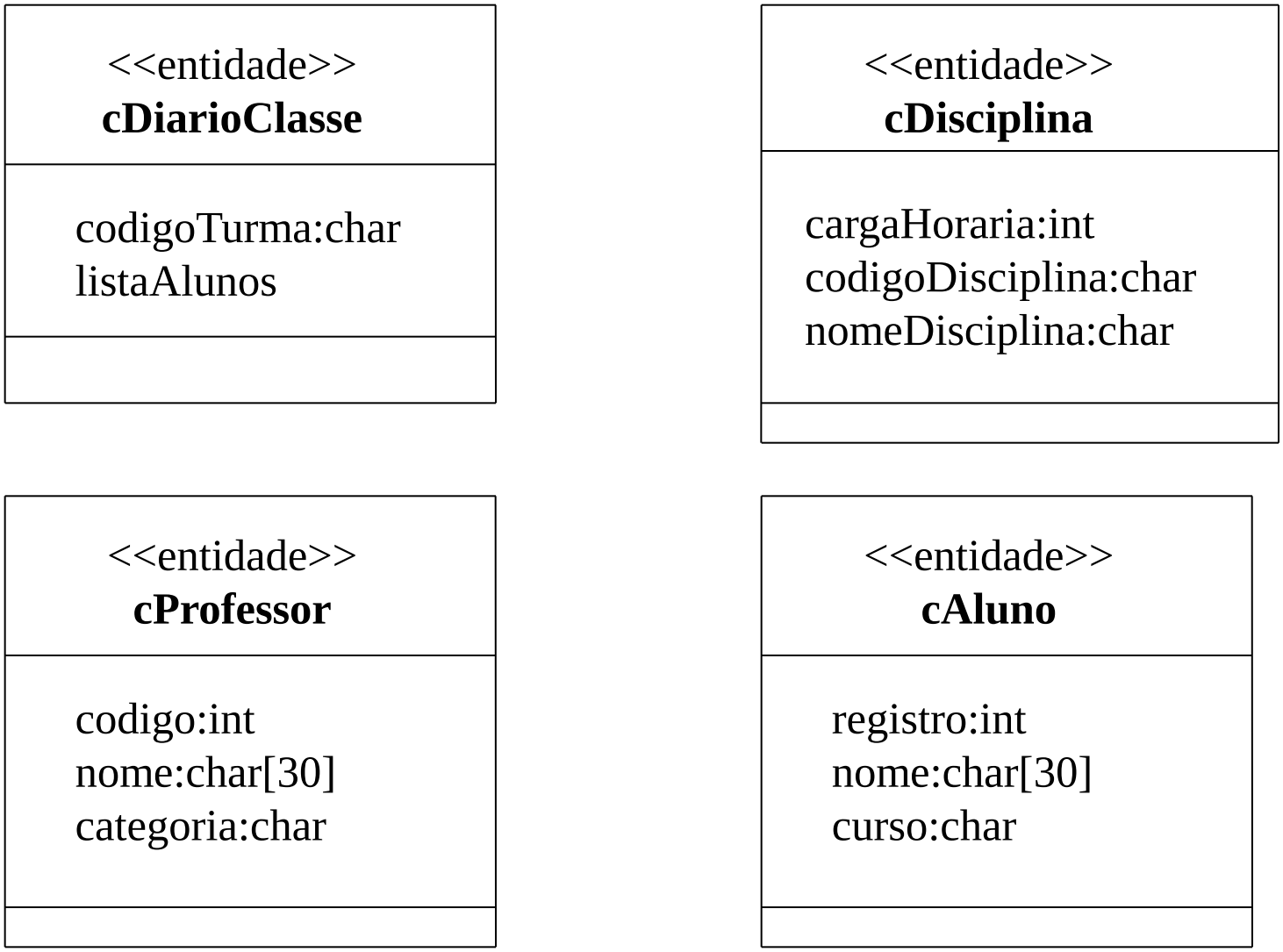
Exemplo de Levantamento de Classes

- Um diário de classe contém:
  - código e nome da disciplina;
  - código e nome do professor;
  - carga horária e a;
  - lista de alunos com o nome e número de registro.

Exemplo de Levantamento de Classes

- Uma alternativa para a representação dos dados do caso de uso Emitir Diário de Classe seria a criação em uma única classe (cDiárioClasse) contendo os dados necessários para a composição do documento impresso.
- Outra alternativa seria extrair da classe (cDiárioClasse) os dados sobre disciplina, professor, e alunos, e colocá-lo em três outras classes:

Exemplo de Levantamento de Classes




MAURICIO