Universidade de Brasília – Campus UnB Gama

Disciplina: Desenho de Software

Responsável: André Luiz Peron Martins Lanna

Aluno: Cleiton da Silva Gomes **Matricula: 10/0097022**

Aluna: Vanessa Barbosa Martins **Matricula: 10/0131182**

Atividade Extra Classe 01

1. Defina os conceitos coesão e encapsulamento.

Coesão: Relaciona o grau de responsabilidades existentes de uma determinada classe com seus métodos existentes. Uma classe altamente coesa tem responsabilidades e propósitos claros e bem definidos e logo são fáceis de serem visualizados no projeto. Entretanto, a classe com baixa coesão tem muitas responsabilidades diferentes e pouco relacionadas, gerando ausência de clareza do que a classe realmente faz. Por exemplo, uma classe Cachorro com o método voar() é uma classe com baixa coesão, enquanto uma classe Passaro com o método voar() é uma classe com alta coesão.

Encapsulamento: Agrupamento de idéias afins em uma unidade, conceito esse que pode então ser informado em uma só palavra (Meilir Page-Jones). Em outras palavras, encapsulamento é um mecanismo utilizado para lidar com o aumento de complexidade e consiste em exibir "o que" pode ser feito sem informar "como" é feito. O encapsulamento também permite que a granularidade de abstração do sistema seja alterada, criando estruturas mais abstratas.

2. Quais são os tipos de aclopamentos existentes? Apresente-os em ordem crescente (do menos acoplado para o mais acoplado).

Encapsulamento nível 0: Completa inexistência de encapsulamento
– Linhas de código efetuando todas as ações

Encapsulamento nível 1: Módulos procedimentais
– Procedimentos permitindo a criação de ações complexas

Encapsulamento nível 2: Classes de objetos
– Métodos isolando o acesso às características da classe

Encapsulamento nível 3: Pacotes de classes
– Conjunto de classes agrupadas, permitindo acesso diferenciado entre elas

Encapsulamento nível 4: Componentes
– Interfaces providas e requeridas para fornecer serviços complexos

3. Apresente através de diagramas UML como se dão os tipos de acoplamento entre classes e/ou objetos.

Tipos de aclopamento:

- Acoplamento de dados
- Acoplamento de controle
- Acoplamento de dados globais
- Acoplamento de dados internos

Acoplamento de dados

Situações:

- Saída de um objeto é entrada de outro
- Uso de parâmetros para passar itens entre métodos

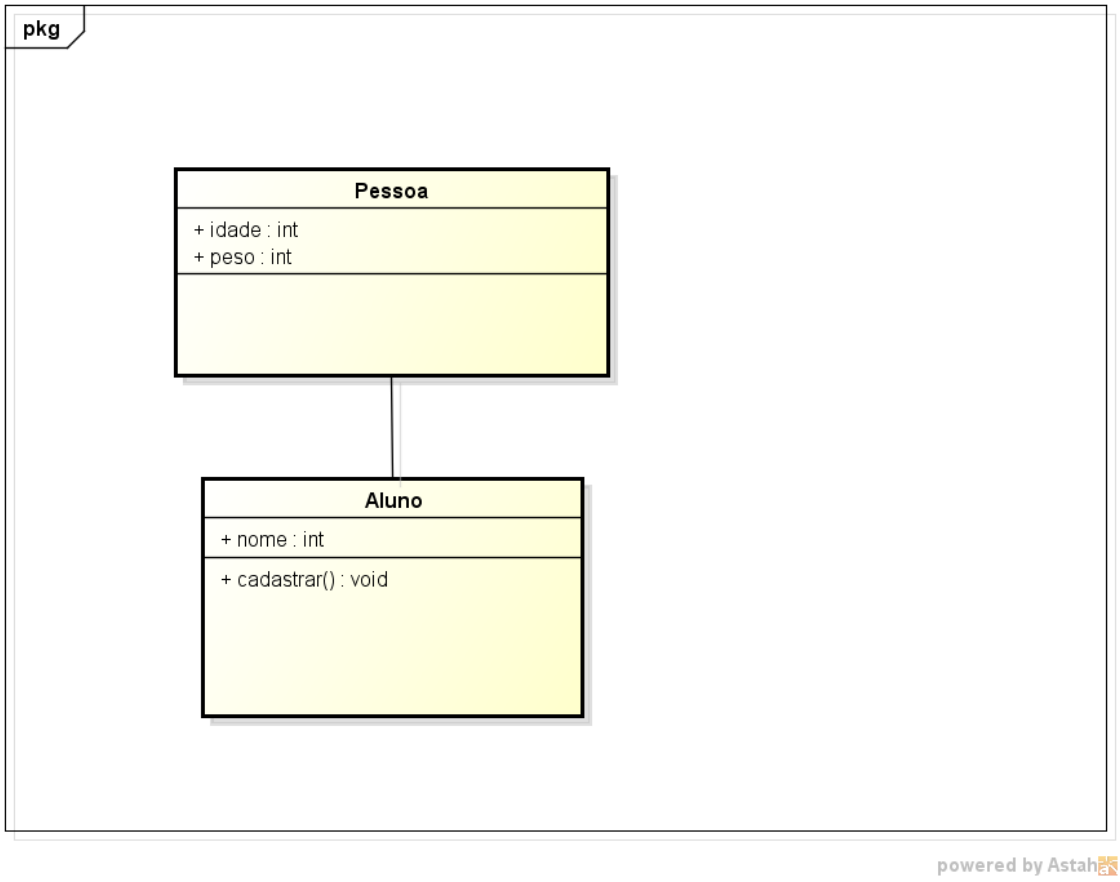
Ocorrência comum:

- Objeto a passa objeto x para objeto b
- Objeto x e b estão acoplados
  - Uma mudança na interface de x pode implicar em mudanças a b

- Exemplo:
-

```
class Servidor {
  public void mensagem(MeuTipo x) {
    // código aqui
    x.façaAlgo(Object dados); // dados e x estão acoplados
                        // (se interface de dados mudar x terá que mudar)
    // mais código
  }
}
```

A seguir segue o diagrama de acoplamento de dados:



Acoplamento de controle

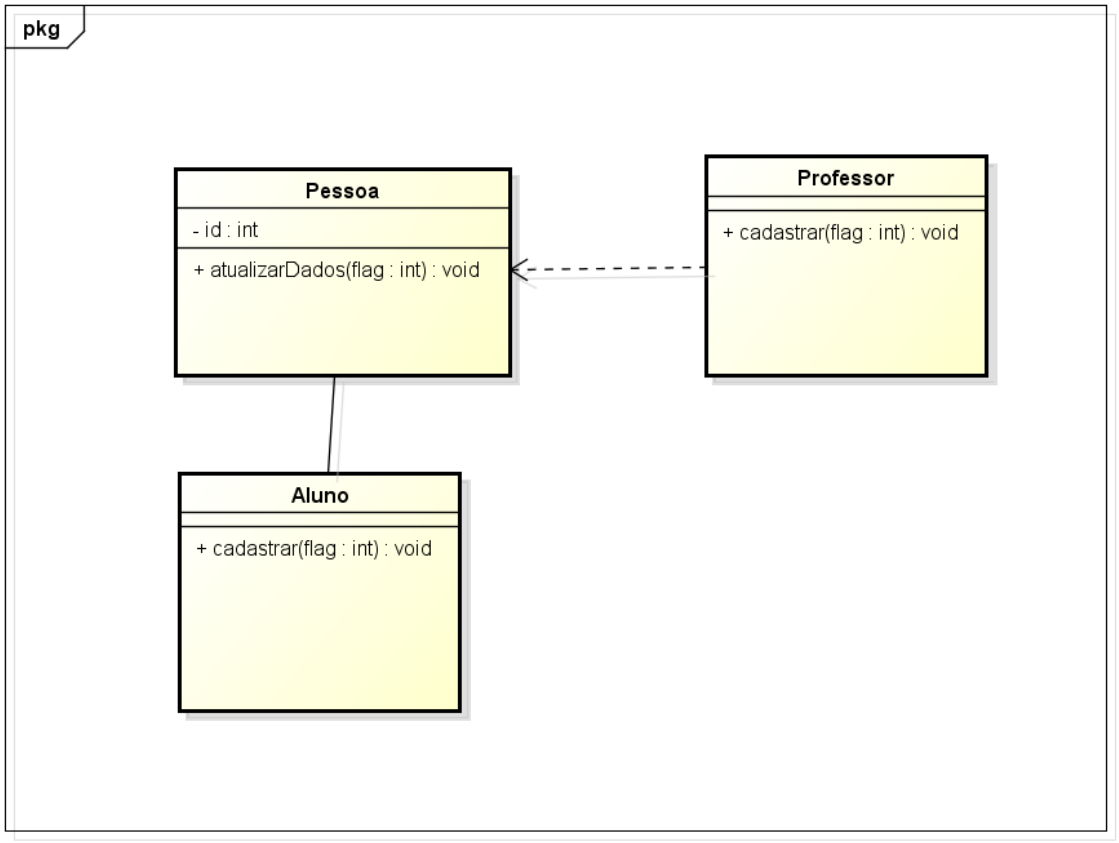
- Passar flags de controle entre objetos de forma que um objeto controle as etapas de processamento de outro objeto
- Ocorrência comum:
  - Objeto a manda uma mensagem para objeto b
  - b usa um parâmetro da mensagem para decidir o que fazer

```
class Lampada {
 public final static int ON = 0;

 public void setLampada(int valor) {
  if(valor == ON) {
   // liga lampada
  } else if(valor == 1) {
   // desliga lampada
  } else if(valor == 2) {
   // pisca
  }
 }
}

Lampada lampapa = new Lampada();
lampada.setLampada(Lampada.ON);
lampada.setLampada(2);
```

A seguir segue o diagrama do acoplamento de dados de controle:

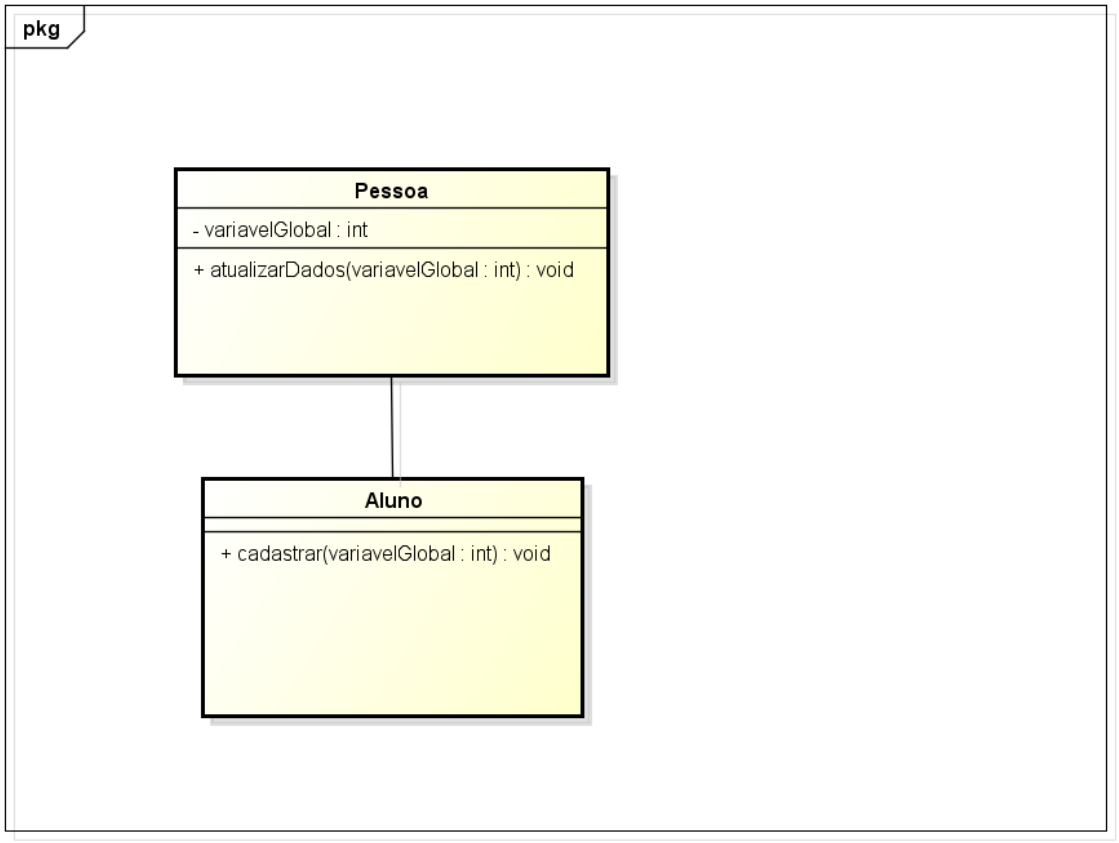


Acoplamento de dados globais

- Dois ou mais objetos compartilham estruturas de dados globais
- É um acoplamento muito ruim pois está escondido
    - Uma chamada de método pode mudar um valor global e o código não deixa isso aparente

Um tipo de acoplamento muito ruim

A seguir segue o diagrama do acoplamento de dados globais:

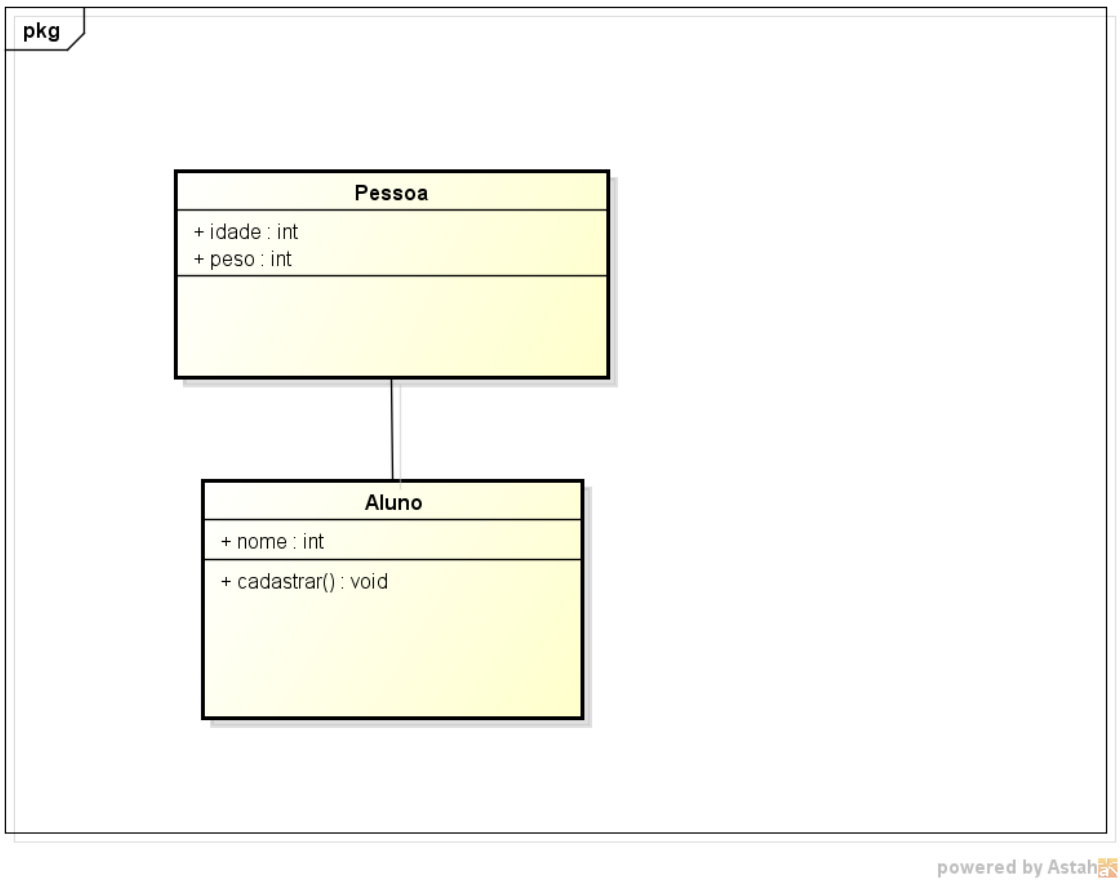


Acoplamento de dados internos

- Um objeto altera os dados locais de um outro objeto
- Ocorrência comum:
  - Friends em C++
  - Dados públicos, package visibility ou mesmo protected em java

  Use com cuidado!

A seguir segue o diagrama UML do aclopamento de dados internos:



4. Quais são os tipos de coesão existentes? Apresente-os em ordem crescente (do menos coeso para o mais coeso).

Melhor Manutenibilidade
. Funcional: função com responsabilidade bem definida.
. Seqüencial: seqüência de funções onde a saída de uma será entrada de outra.
. Comunicacional: funções que usam a mesma entrada ou possuem a mesma saída.
. Procedural: módulos compostos por funções pouco relacionadas entre si.
. Temporal: atividades que estão relacionadas no tempo.
. Lógica: módulos que possuem a mesma característica geral.
. Coincidental: módulos cujos elementos contribuem para atividades sem relacionamento.
Pior Manutenibilidade

5. O que é a independência funcional em projetos de software?

Segundo Pressman (1995), um software monolítico ou um grande programa composto de um único módulo, é difícil de ser entendido pelo leitor. Módulos devem ser especificados e projetados de tal forma que suas informações sejam inacessíveis a outros módulos que não necessitem dela. Essa ocultação define e reforça as restrições de acesso, trazendo benefícios quando modificações são exigidas. Isso evita que essas

modificações se propaguem a outros locais do sistema (PRESSMAN, 1995; DENNIS, 2005; STAA, 2000).

Sendo assim, Independência funcional em projetos de software é a vertente que trabalha com um módulo altamente coeso e menos aclopado possível.