

FIA/P GRADUAÇÃO

DISCIPLINA: PROJETO DE SISTEMAS APLICADO AS MELHORES PRÁTICAS EM QUALIDADE DE SOFTWARE E GOVERNANÇA DE TI

AULA:
18 – AVALIAÇÃO DO PROJETO DO SOFTWARE

PROFESSOR:
RENATO JARDIM PARDUCCI

PROFRENATO.PARDUCCI@FIAP.COM.BR

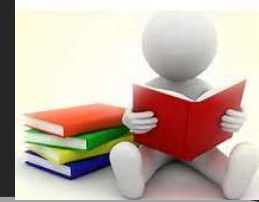
[Renato Parducci - YouTube](#)

AGENDA DA AULA

- ✓ CMMi nível 3 - VER/VAL
- ✓ MPS.br nível D - VER/VAL
- ✓ Métricas de avaliação do desenho do software
- ✓ Métricas de avaliação (CBO, RFC, LCOM, NOC, DIT)

**PRÁTICAS E NÍVEL 3 –TS,
VER/VAL**
**Métricas Avaliativas do Desenho
do Software**

ESTUDO DE CASO SIMULADO



Consuelo, Dilan e Trenilda, em comum acordo, decidiram que daqui por diante, o treinamento em técnicas e ferramentas da qualidade será feito “on the job”, ou seja, serão apresentados diversos casos reais da empresa como “Atividades Práticas”, que serão solucionados com os recursos que Consuelo apresentar.

Com isso, a equipe aprenderá, enquanto resolve problemas reais da software house.

TESTE DE SOFTWARE

ATIVIDADE PRÁTICA



Você é responsável por um grande projeto de software que levará 2 anos para finalizar e que está iniciando agora. Ele envolverá 25 desenvolvedores, sendo que toda a equipe vai utilizar ferramentas e técnicas padronizadas de trabalho.

As programações serão em JAVA, a modelagem funcional será feita com UML, a documentação de processos de trabalho com BPMN, o gerenciamento será com SCRUM, o banco de dados será ORACLE, servidores de aplicação e dados LINUX RED HAT.

Os primeiros programas estão previstos para serem concluídos 8 meses após iniciar o projeto.

*Quando você imagina que poderá testar o software?
Quais seriam os primeiros testes que você faria?*

APLICANDO MÉTRICAS ESTRUTURAIS NA AVALIAÇÃO DE SOFTWARE

No nível de testes de INTEGRAÇÃO do modelo V, aplicam-se métricas que indicam a qualidade do projeto quanto ao funcionamento previsto dos componentes do software, uma vez juntos.

Esses testes são de CAIXA BRANCA, por demandarem a compreensão da especificação técnica do projeto.

Eles também são NÃO FUNCIONAIS por estarem focados na avaliação de estruturas de dados e aplicação que entregam funcionalidades do sistema para atender as regras de negócio.

As avaliações de software podem e devem começar muito antes da programação!

Desenhos de software mal resolvidos irão gerar impactos em diversas partes do produto final:

- **Regras de negócio** podem estar muito **concentradas em determinados componentes** que se tornam pontos de risco – desenvolvidos por um ou dois programadores que acabam definindo grande parte da inteligência do software;
- **As associações entre componentes** e dependências entre eles podem **complicar muito a adaptação/manutenção** – que devem estar bem resolvidas em projetos ágeis que preveem refatoração;
- O desenho inadequado das **comunicações entre componentes do software podem afetar o desempenho** – o sistema se torna não aderente à necessidade de negócio por não possibilitar o uso no dia a dia.

Alguns **testes ESTRUTURAIS** são muito populares na hora de avaliar o modelo de engenharia do sistema e **geram métricas** das quais são destacadas:

- Métodos ponderados por Classe (**WMC – Weight of Methods in a Class**);
- Profundidade da árvore de herança (**DIT – Depth Inside the Tree**);
- Número de filhos (**NOC – Number of Children**);
- Coesão em métodos (**LCOM – Level of Cohesion of Method**);
- Acoplamento entre Classes (**CBO – Coupling Between Objects**);
- Resposta para uma Classe (**RFC – Response For a Class**).

Cada uma dessas métricas a serem calculadas devem constar na lista de Casos de Testes (uma ficha descritiva para cada uma).



CASOS DE TESTE

Detalhes dos Casos de Testes e Métricas de Avaliação Estrutural da Engenharia do Software

WMC

- **Métodos ponderados por Classe (WMC)**

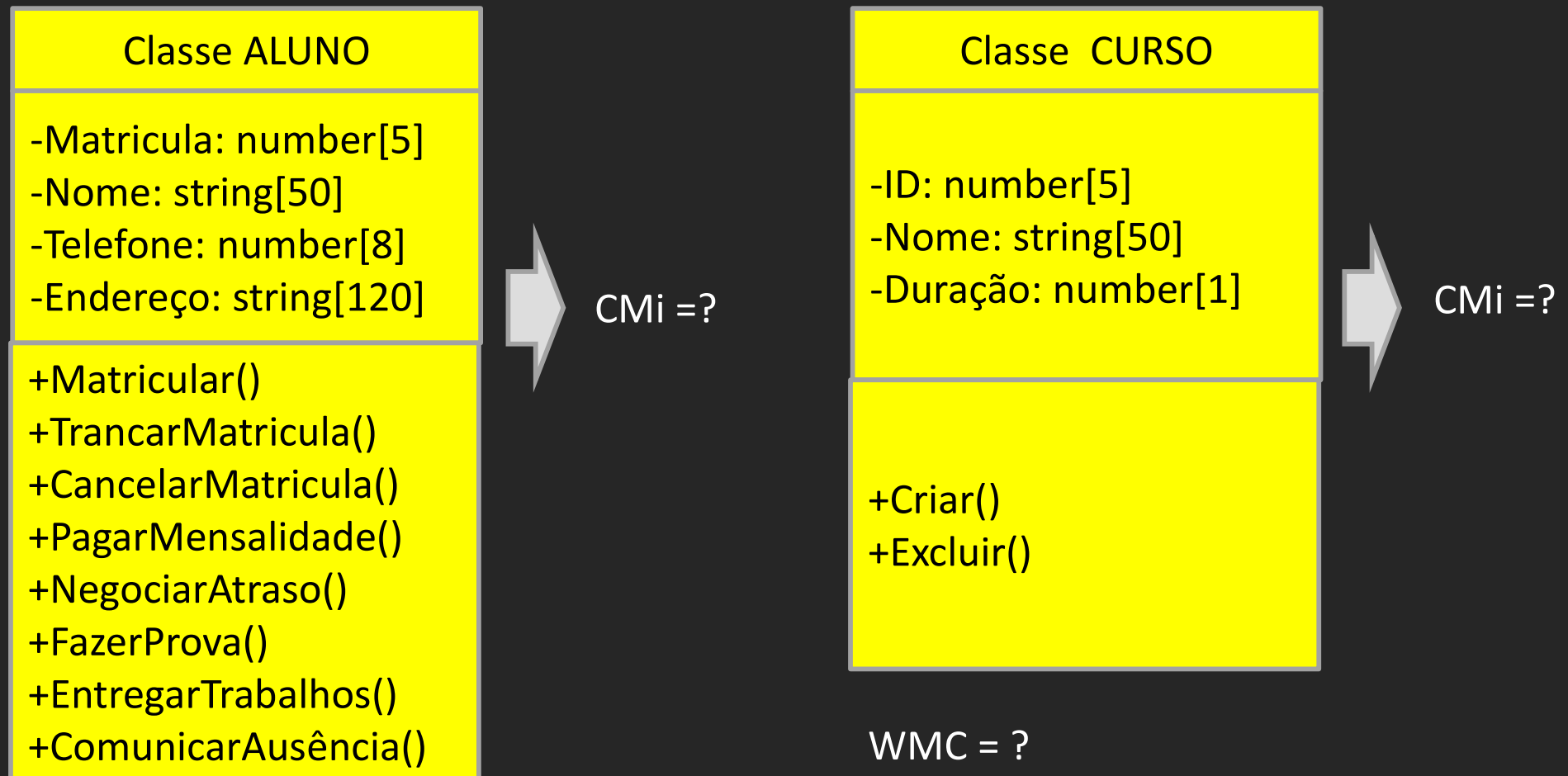


CM_i = número de métodos da Classe i

WMC = número de métodos resultante da soma dos métodos de todas as Classes

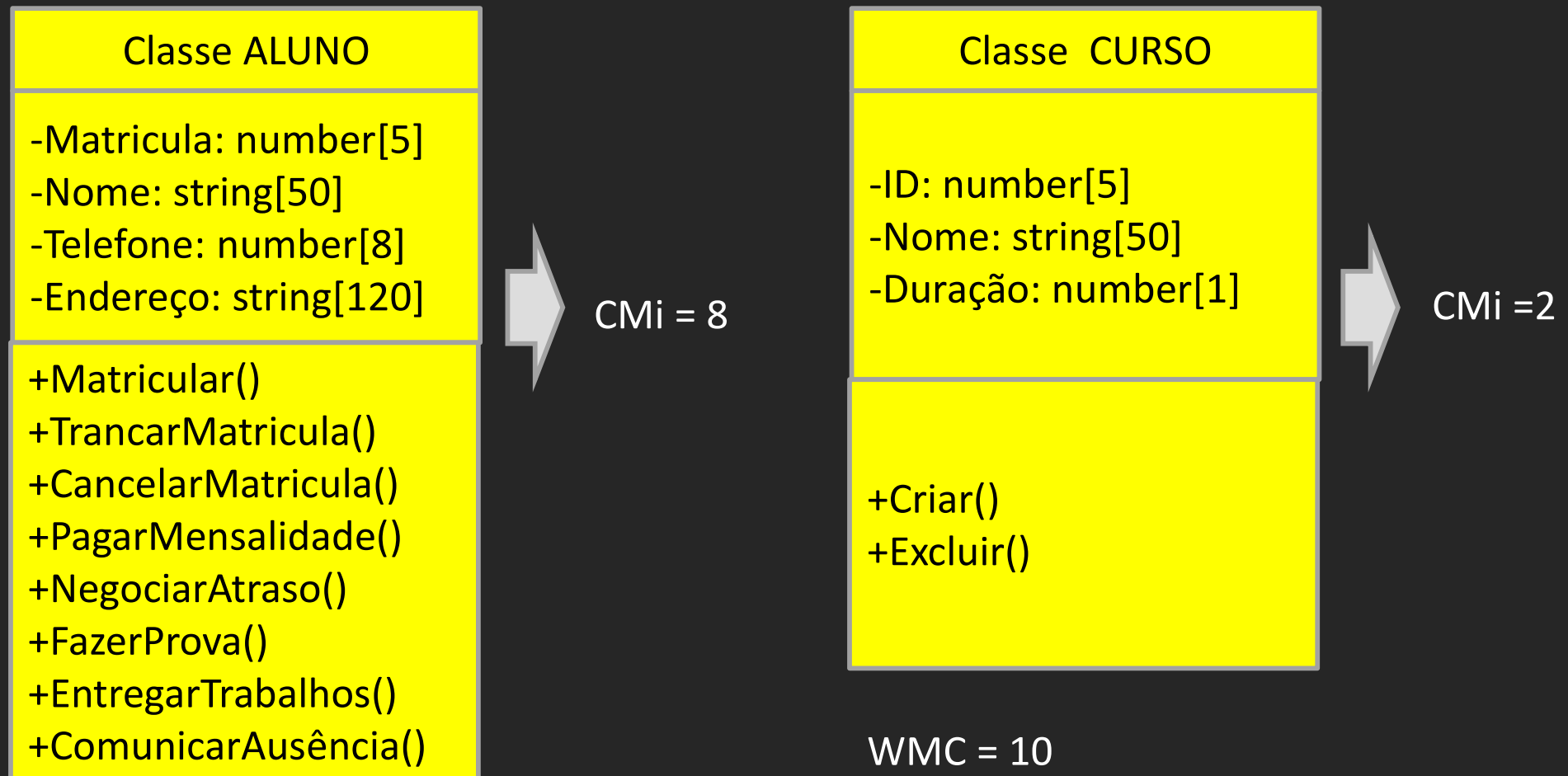
WMC

- Métodos ponderados por Classe (WMC)**

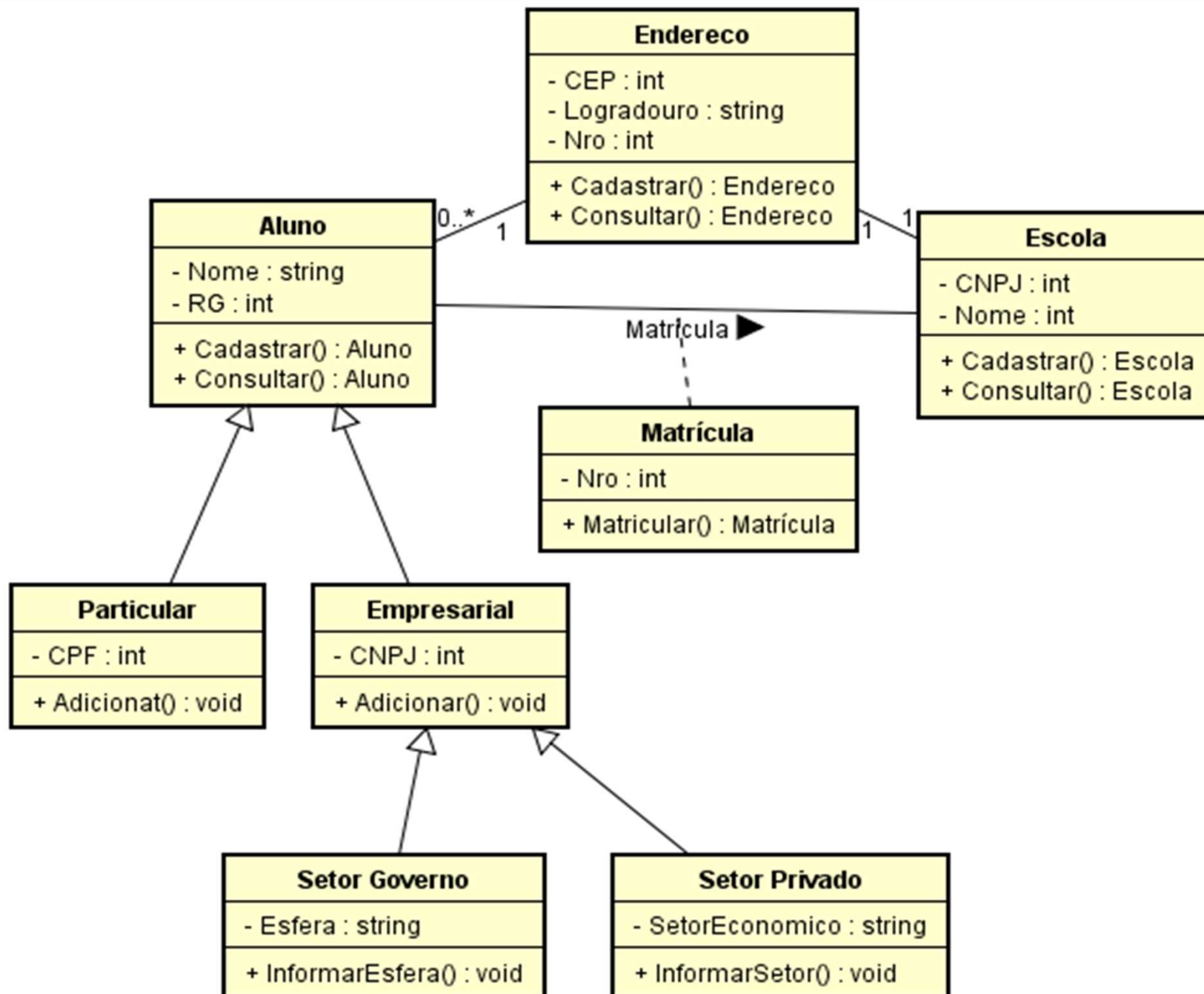


WMC

- Métodos ponderados por Classe (WMC)**



Calcule o WMC do caso ao lado



WMC

- **Métodos ponderados por Classe (WMC)**



Qual decisão você pode tomar com base nessa métrica?

Análise: quanto maior o número de métodos, mais complexa a Classe em termos do seu desenvolvimento e manutenção – deve ser verificada a possibilidade de especialização para Classes com uma quantidade de métodos muito acima do padrão de outras Classes do sistema, ou que representa um grande percentual do total de métodos do projeto de sistema. Poucos métodos podem indicar uma especialização desnecessária de Classe.

DIT

- **Profundidade da árvore de herança (DIT)**

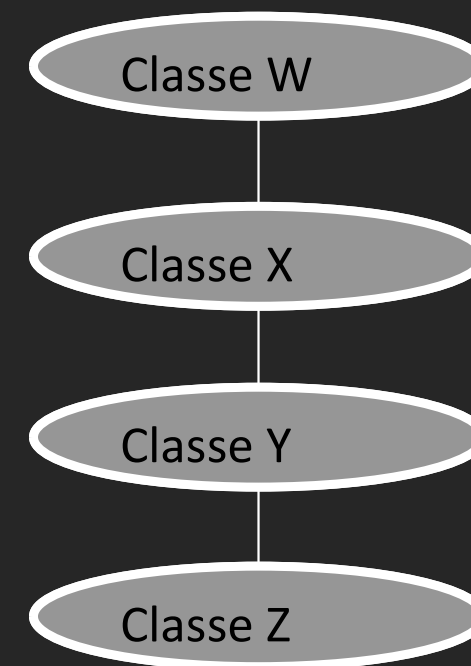
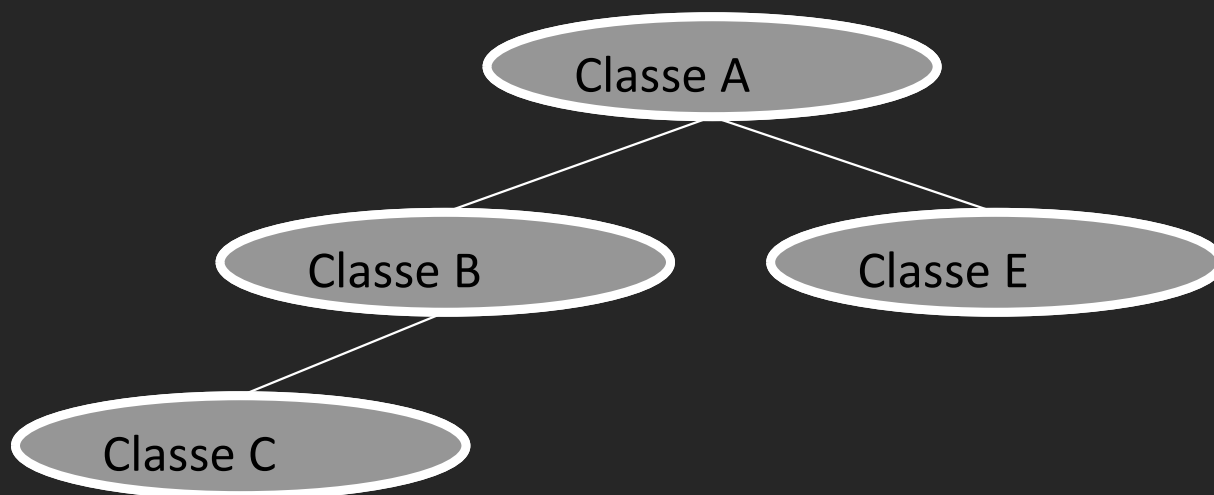


Avalia o número de níveis hierárquicos abaixo de uma Classe.

O indicador é calculado para cada Classe modelada.

DIT

- **Profundidade da árvore de herança (DIT)**



Profundidade da Classe A: 2 (caminho mais longo)

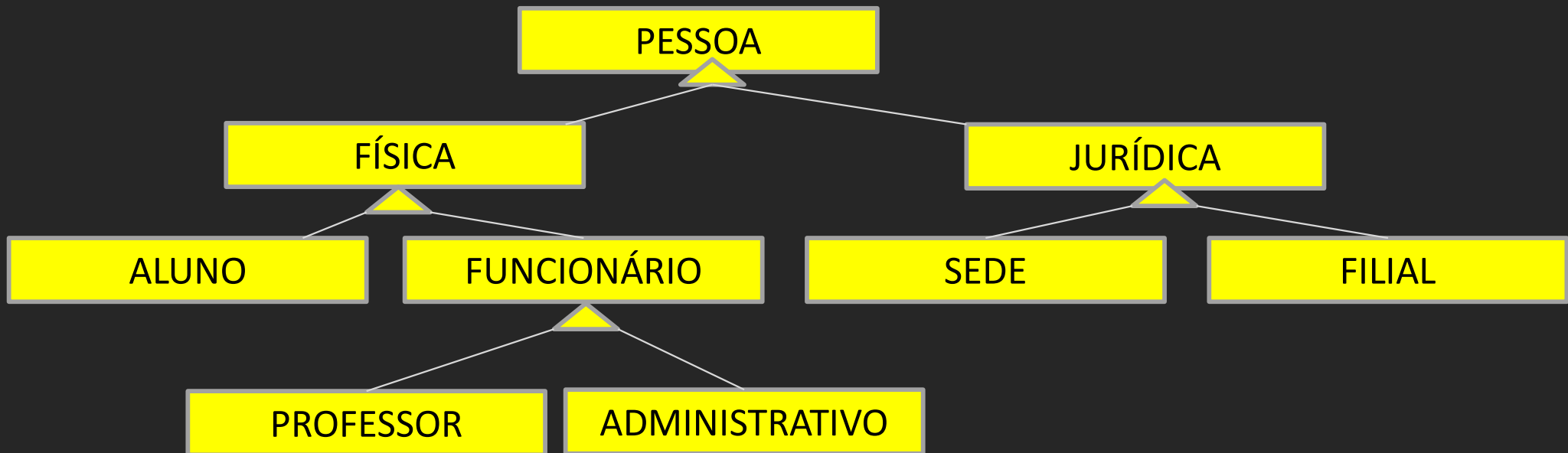
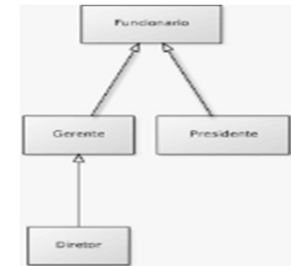
Profundidade da Classe B: 1

Profundidade da Classe E: 0

Profundidade da Classe W: 3

DIT

- **Profundidade da árvore de herança (DIT)**



Profundidade da Classe PESSOA: ?

Classe FÍSICA: ?

Classe JURÍDICA: ?

Classe ALUNO: ?

Classe FUNCIONÁRIO: ?

Classe SEDE: ?

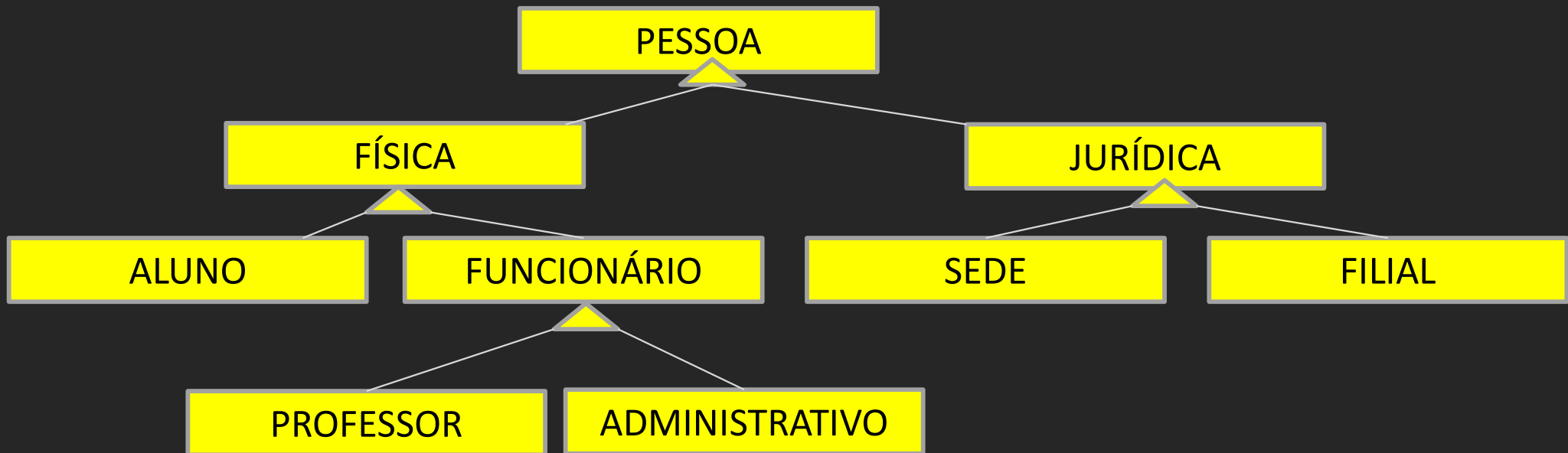
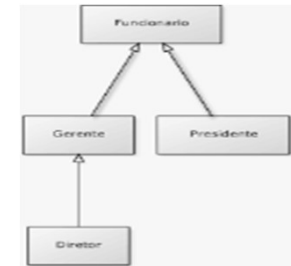
Classe FILIAL: ?

Classe PROFESSOR: ?

Classe ADMINISTRATIVO: ?

DIT

- **Profundidade da árvore de herança (DIT)**



Profundidade da Classe PESSOA: 3

Classe FÍSICA: 2

Classe JURÍDICA: 1

Classe ALUNO: 0

Classe FUNCIONÁRIO: 1

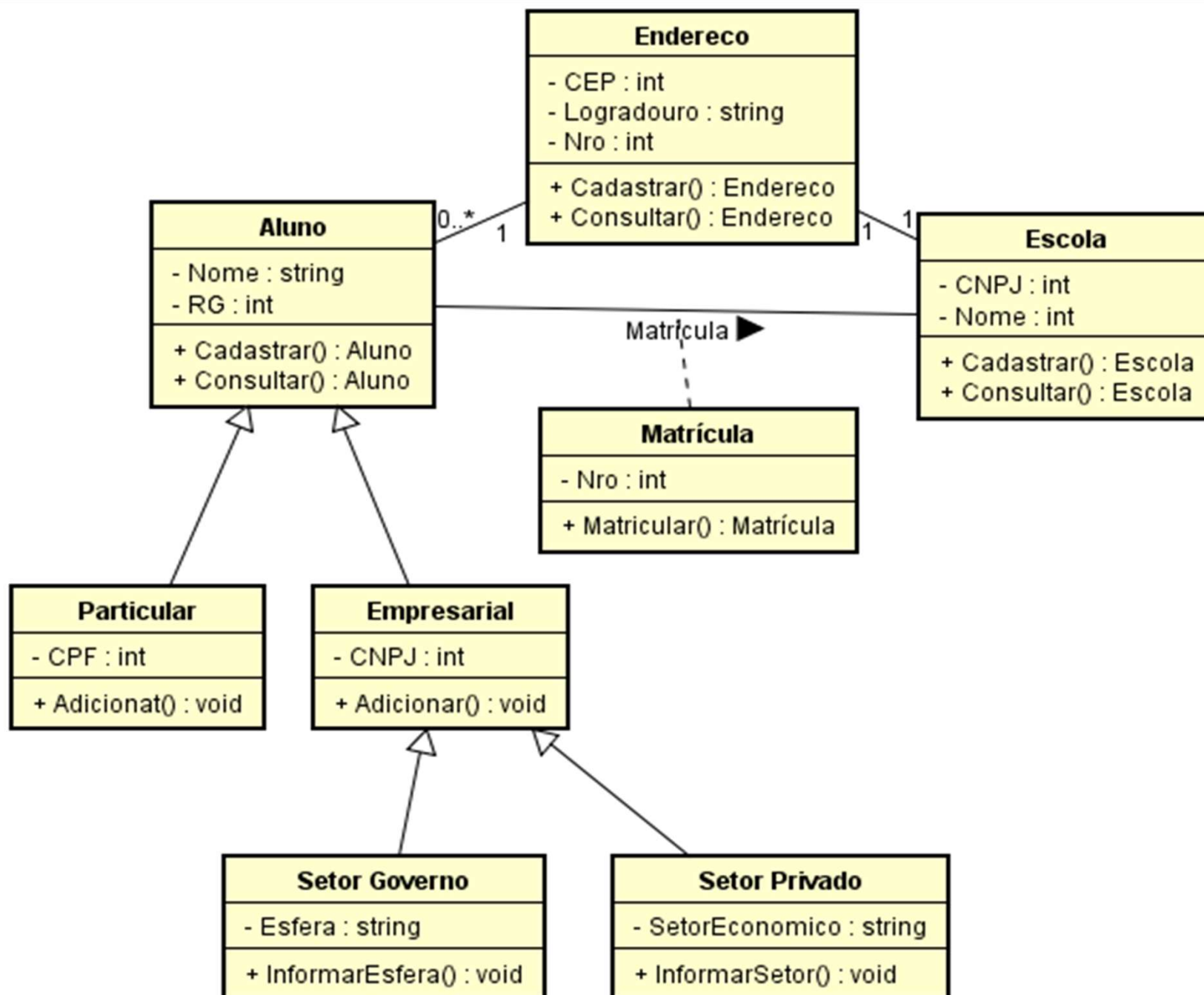
Classe SEDE: 0

Classe FILIAL: 0

Classe PROFESSOR: 0

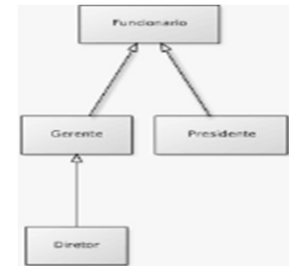
Classe ADMINISTRATIVO: 0

Calcule o
DIT



DIT

- **Profundidade da árvore de herança (DIT)**



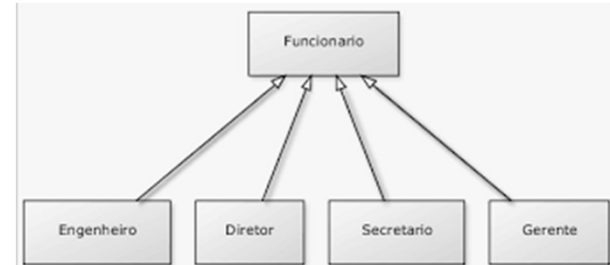
Qual decisão você pode tomar com base nessa métrica?

Análise: se o sistema tiver quase todas as Classes com profundidade 0, a Herança (reuso) está comprometida.

Profundidade muito grande pode apontar excesso de especialização de Classes – o número de Classes pode ser reduzido e os objetos tipificados .

NOC

- **Número de filhos (NOC)**

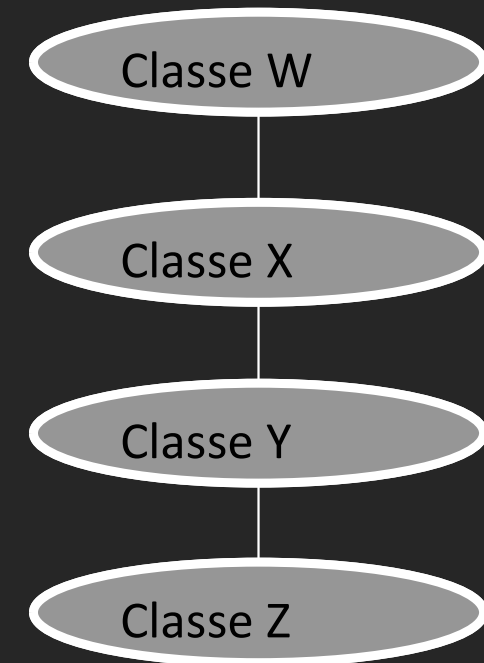
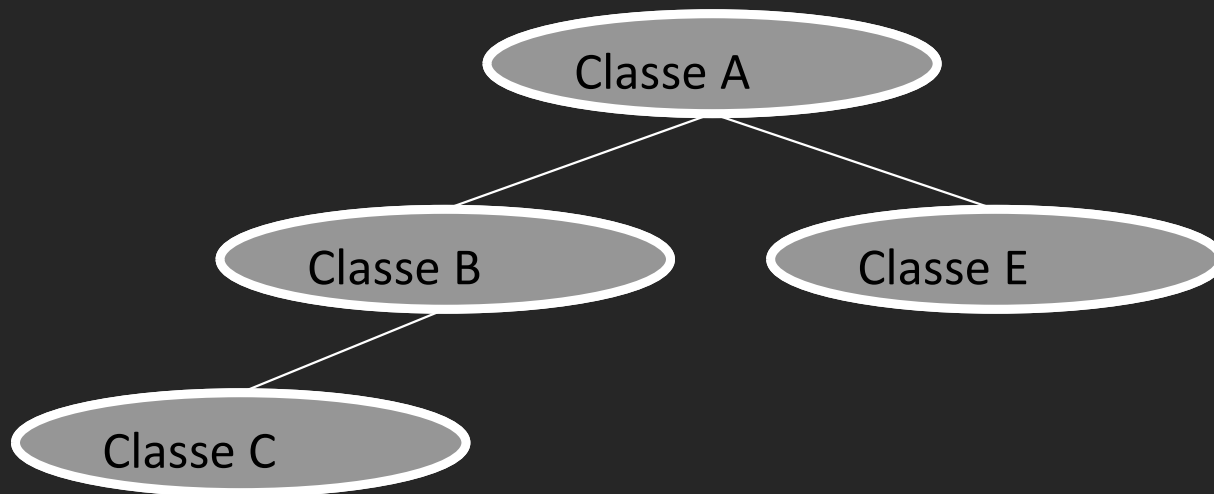


Avalia o número de Classes que são herdeiras diretas (filhos diretos) de uma Classe.

Calculado Classe a Classe

NOC

- **Número de filhos (NOC)**



Filhos da Classe A: 2

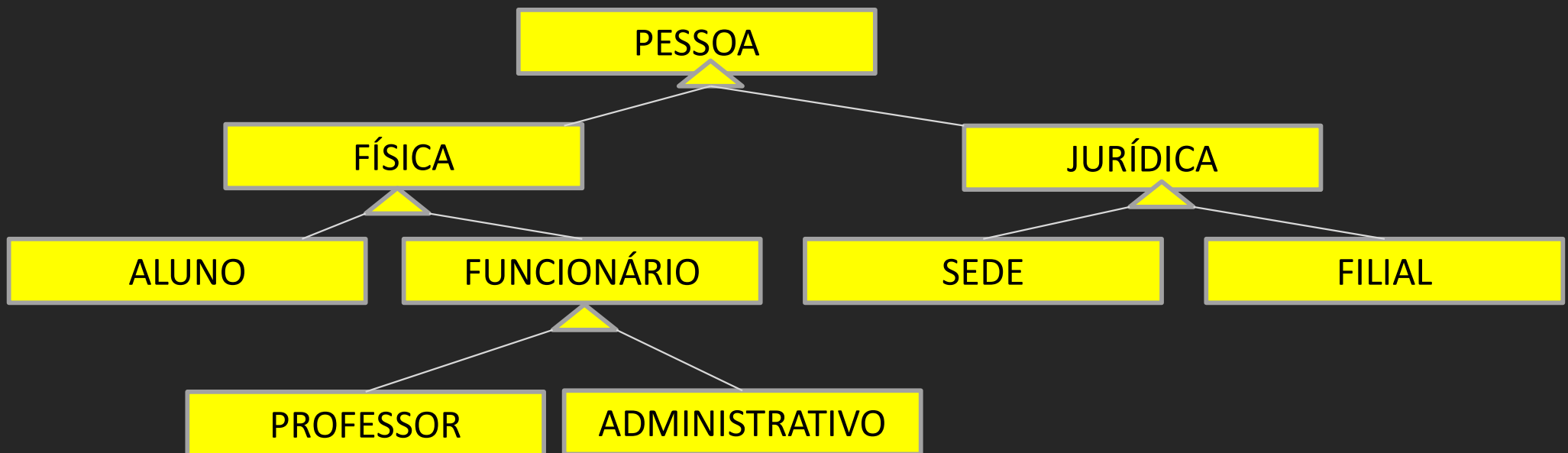
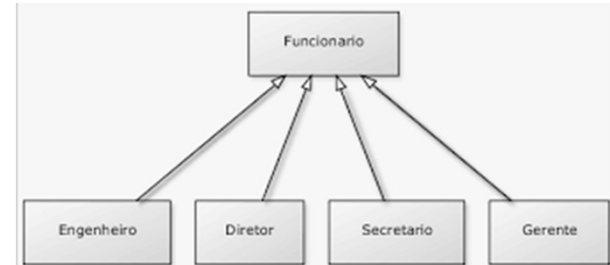
Filhos da Classe B: 1

Filhos da Classe E: 0

Filhos da Classe W: 1

NOC

- **Número de filhos (NOC)**



Número de Filhos de PESSOA: ?

Classe FÍSICA: ?

Classe JURÍDICA: ?

Classe ALUNO: ?

Classe FUNCIONÁRIO: ?

Classe SEDE: ?

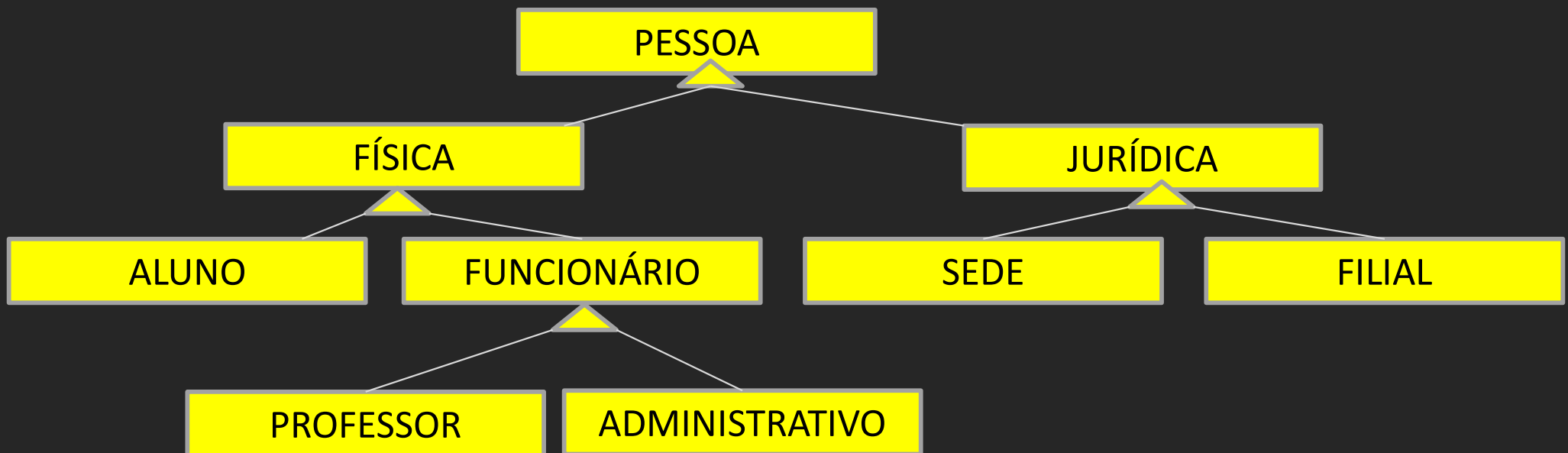
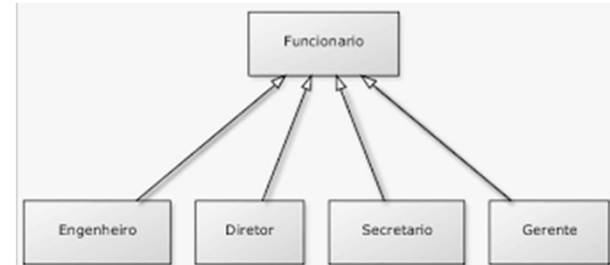
Classe FILIAL: ?

Classe PROFESSOR: ?

Classe ADMINISTRATIVO: ?

NOC

- **Número de filhos (NOC)**



Número de Filhos de PESSOA: 2

Classe FÍSICA: 2

Classe JURÍDICA: 2

Classe ALUNO: 0

Classe FUNCIONÁRIO: 2

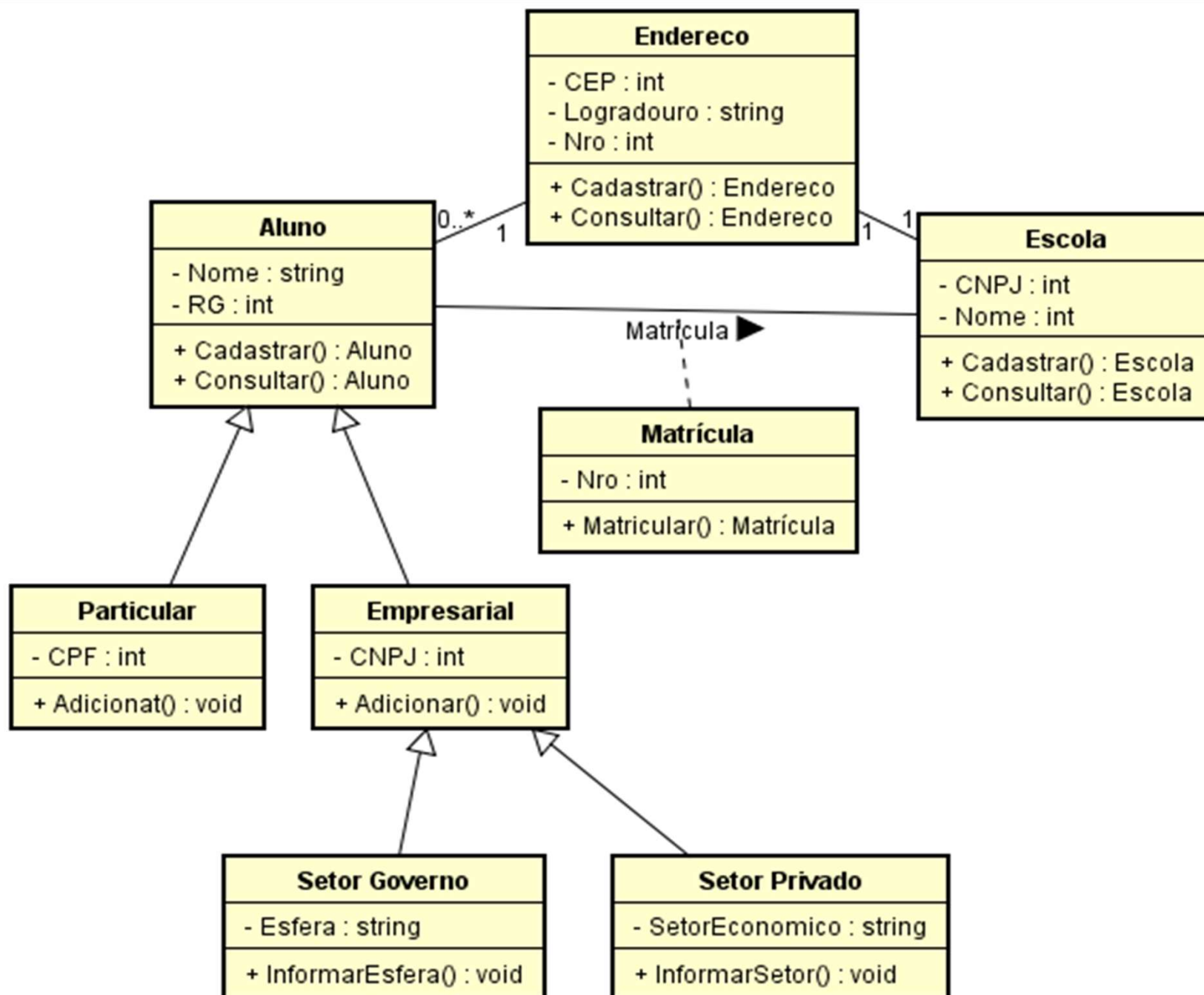
Classe SEDE: 0

Classe FILIAL: 0

Classe PROFESSOR: 0

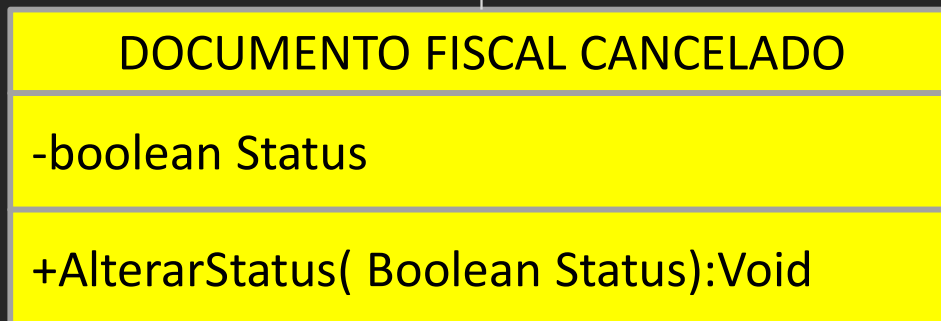
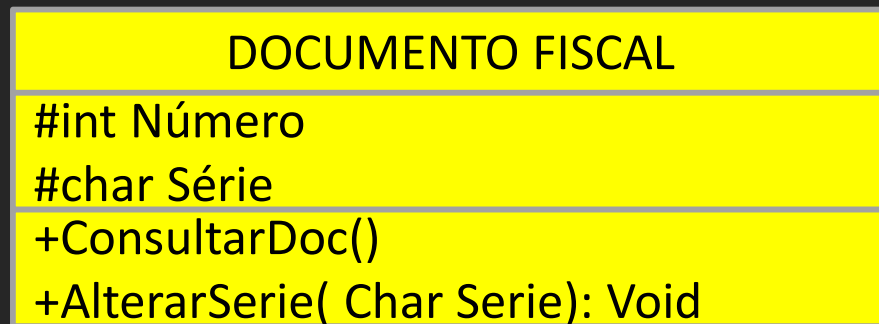
Classe ADMINISTRATIVO: 0

Calcule o
NOC



NOC

- **Número de filhos (NOC)**



NOC

- **Número de filhos (NOC)**



DOCUMENTO FISCAL

-int Número
-char Série
-boolean Status

+ConsultarDoc()
+AlterarDados(Char Serie; Boolean Status): Void

NOC

- **Número de filhos (NOC)**



Qual decisão você pode tomar com base nessa métrica?

Análise: Classes com apenas 1 filho indicam que temos uma especialização de Classes excessiva, gerando complexidade desnecessária para o sistema.

LCOM

- **Coesão em métodos (LCOM)**

```
1 public class Pedido {  
2  
3     public void processar() {  
4         DataServer2 server = new DataServer2();  
5         int diasAtrasado = 5;  
6         for (DataItem item : server.getItemsExAtraso(diasAtrasado))  
7             processarItem(item);  
8     }  
9 }  
0 // Resto da classe  
1 }
```

Avalia o quanto um método de um Classe depende de outros Métodos de outras Classes para resolver a sua lógica.

Uma Classe é coesa se os seus Métodos são coesos.

Dois métodos de uma Classe X são coesos se utilizam os atributos da Classe X e não de outra.

$LCOM = (\text{número de Atributos usados no Método}) / (\text{número de atributos utilizados da Classe do Método pelo Método})$

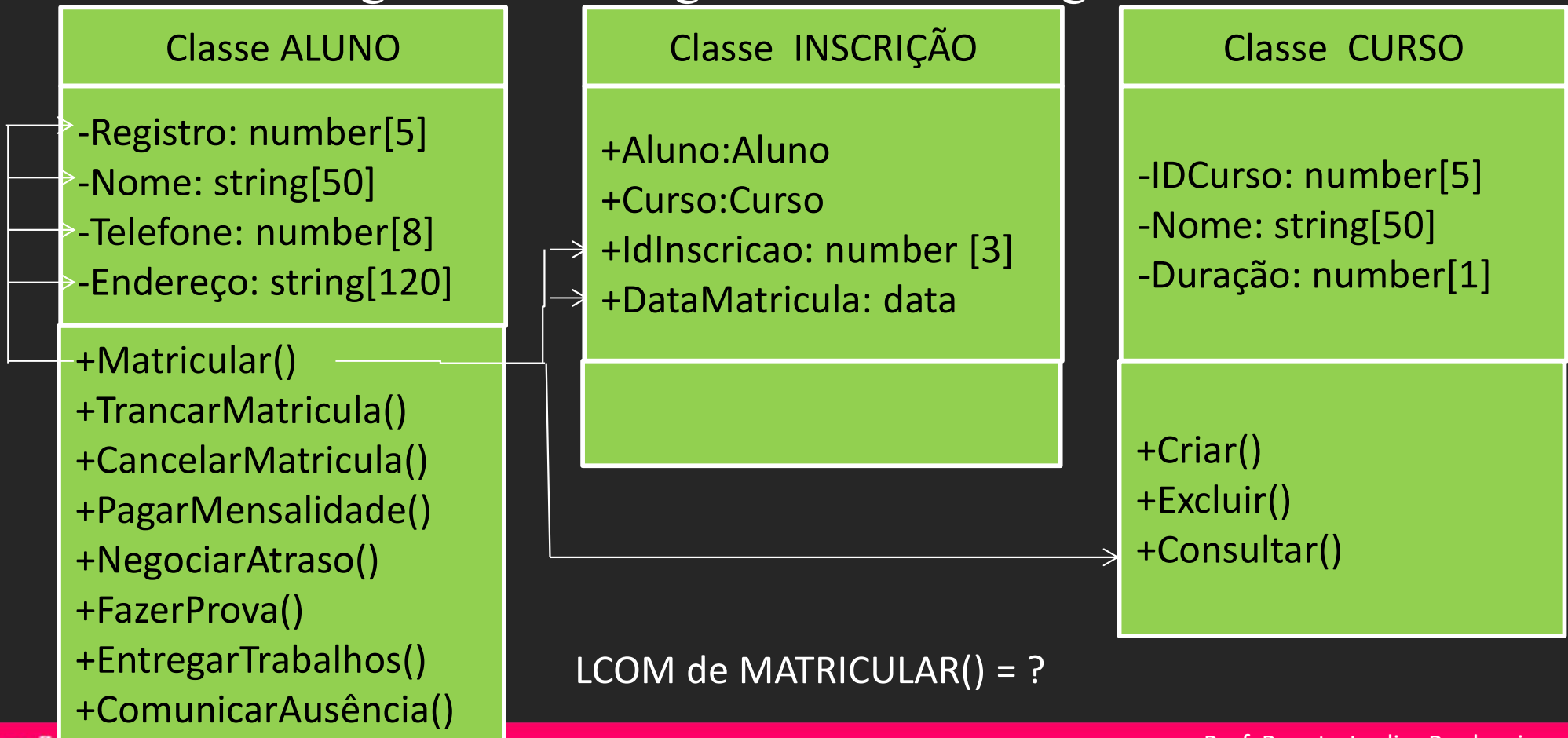
Calculado Método a Método

LCOM

- Coesão em métodos (LCOM)**

```
1 public class Pedido {
2
3     public void processar() {
4         DataServer2 server = new DataServer2();
5         int diasAtrasado = 5;
6         for (DataItem item : server.getItemsExAtraso(diasAtrasado))
7             processarItem(item);
8     }
9 }
10 // Resto da classe
11 }
```

- Considere que cada seta indica uma chamada de um Método de dentro do algoritmo / código do Método origem da seta.

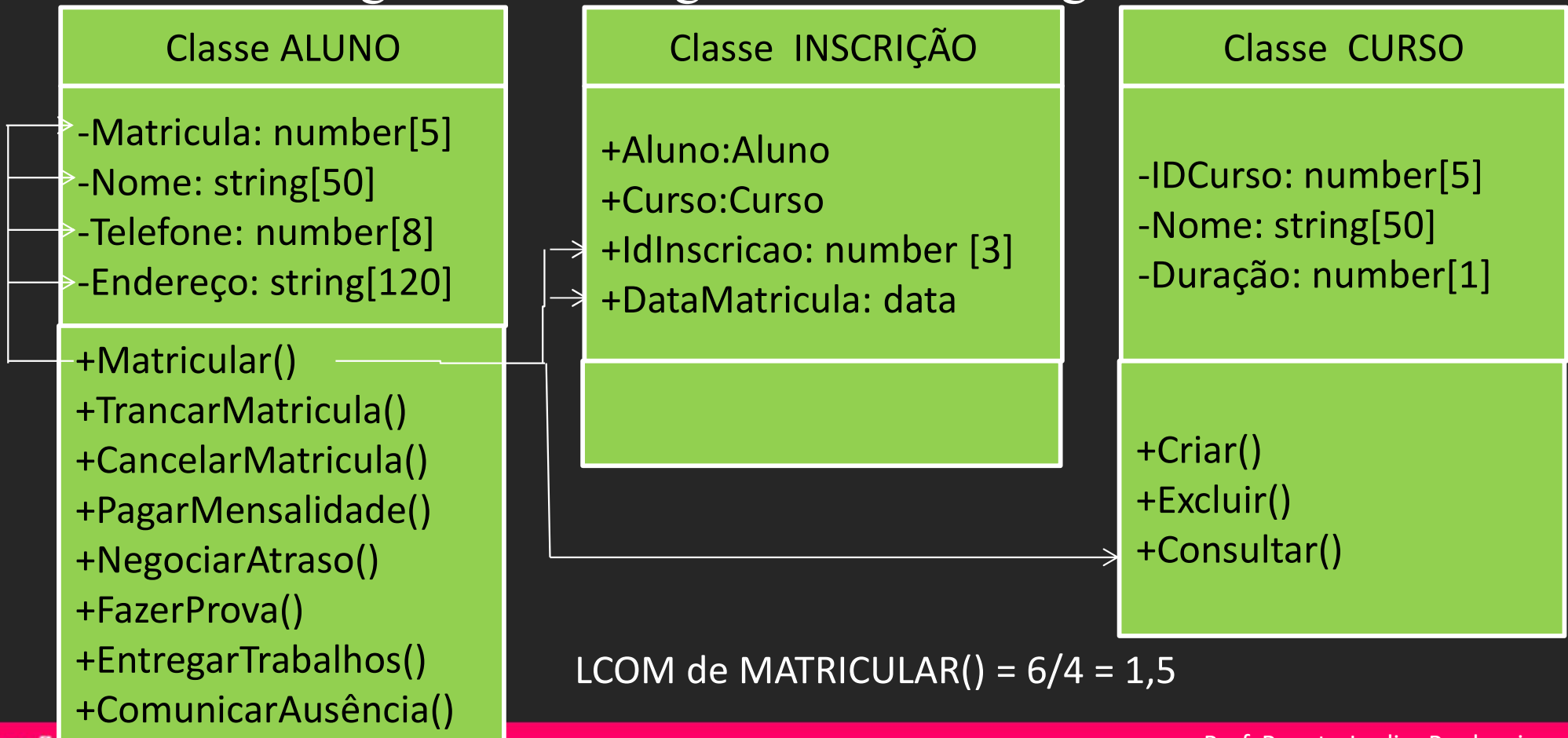


LCOM

- Coesão em métodos (LCOM)**

```
1 public class Pedido {
2
3     public void processar() {
4         DataServer2 server = new DataServer2();
5         int diasAtrasado = 5;
6         for (DataItem item : server.getItemsExAtraso(diasAtrasado))
7             processarItem(item);
8     }
9 }
10 // Resto da classe
11 }
```

- Considere que cada seta indica uma chamada de um Método de dentro do algoritmo / código do Método origem da seta.



LCOM

- Coesão em métodos (LCOM)**

```
1 public class Pedido {
2
3     public void processar() {
4         DataServer2 server = new DataServer2();
5         int diasAtrasado = 5;
6         for (DataItem item : server.getItemsExAtraso(diasAtrasado))
7             processarItem(item);
8     }
9 }
10 // Resto da classe
11 }
```

- Considere que cada seta indica uma chamada de um Método de dentro do algoritmo / código do Método origem da seta.

Classe ALUNO

-Matricula: number[5]
 -Nome: string[50]
 -Telefone: number[8]
 -Endereço: string[120]

+Matricular()
 +TrancarMatricula()
 +CancelarMatricula()
 +PagarMensalidade()
 +NegociarAtraso()
 +FazerProva()
 +EntregarTrabalhos()
 +ComunicarAusência()
 +Consultar()

Classe INSCRIÇÃO

+Aluno:Aluno
 +Curso:Curso
 +IdInscricao: number [3]
 +DataMatricula: data

Classe CURSO

-IDCurso: number[5]
 -Nome: string[50]
 -Duração: number[1]

+Criar()
 +Excluir()
 +Consultar()

LCOM de CONSULTAR() = ?

LCOM

• Coesão em métodos (LCOM)

```
1 public class Pedido {
2
3     public void processar() {
4         DataServer2 server = new DataServer2();
5         int diasAtrasado = 5;
6         for (DataItem item : server.getItemsExAtraso(diasAtrasado))
7             processarItem(item);
8     }
9 }
10 // Resto da classe
11 }
```

- Considere que cada seta indica uma chamada de um Método de dentro do algoritmo / código do Método origem da seta.

Classe ALUNO

-Matricula: number[5]
 -Nome: string[50]
 -Telefone: number[8]
 -Endereço: string[120]

+Matricular()
 +TrancarMatricula()
 +CancelarMatricula()
 +PagarMensalidade()
 +NegociarAtraso()
 +FazerProva()
 +EntregarTrabalhos()
 +ComunicarAusência()
 +Consultar()

Classe INSCRIÇÃO

+Aluno:Aluno
 +Curso:Curso
 +IdInscricao: number [3]
 +DataMatricula: data

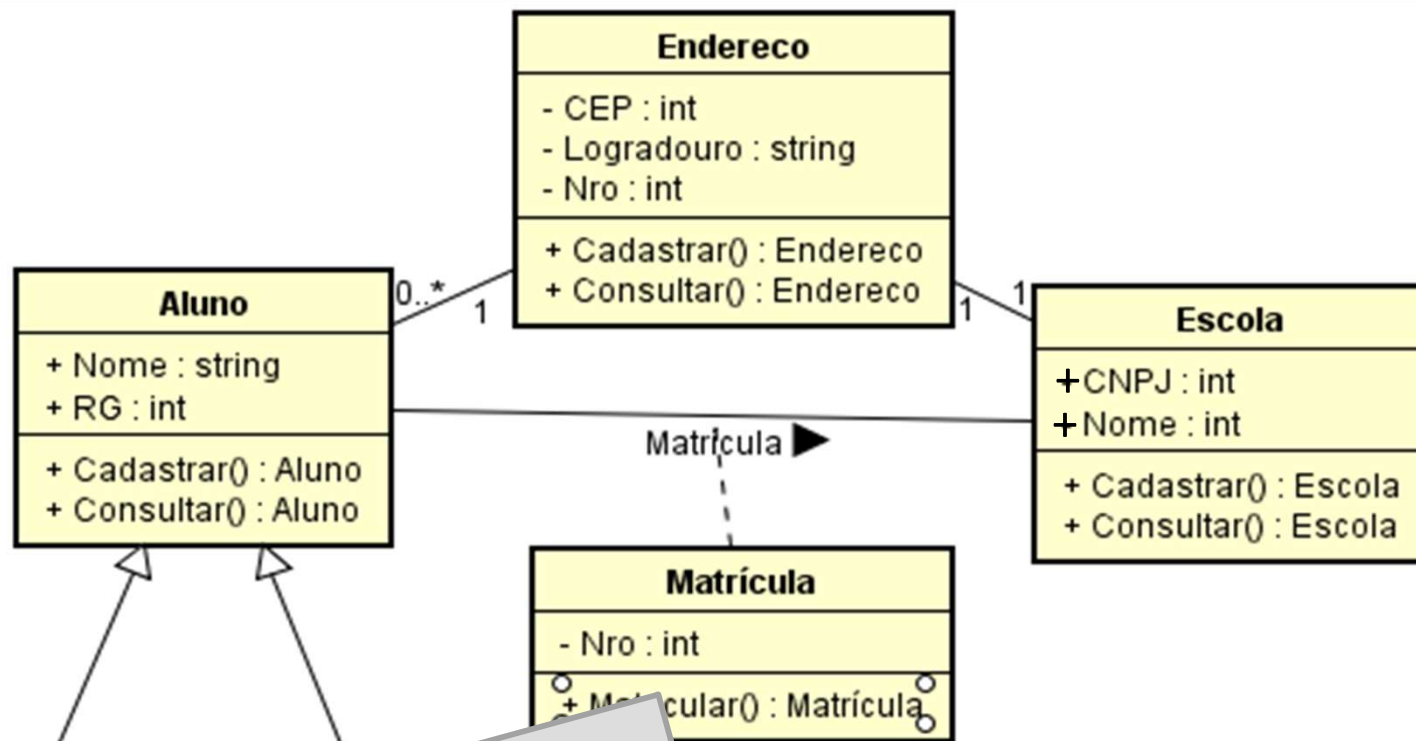
Classe CURSO

-IDCurso: number[5]
 -Nome: string[50]
 -Duração: number[1]

+Criar()
 +Excluir()
 +Consultar()

LCOM de CONSULTAR() = 4/4 = 1

Calcule o LCOM



Algoritmo de Matricular

```

Inicio
  Nova Matricula
  Matricula.Nro := Aluno.RG + Escola.CNPJ + Random(10)
Fim
    
```

Privado

omico : string

+ InformarEstrutura() : void

+ InformarSetor() : void

LCOM

- Coesão em métodos (LCOM)**

```
1 public class Pedido {
2
3     public void processar() {
4         DataServer2 server = new DataServer2();
5         int diasAtrasado = 5;
6         for (DataItem item : server.getItemsExAtraso(diasAtrasado))
7             processarItem(item);
8     }
9 }
10 // Resto da classe
11 }
```

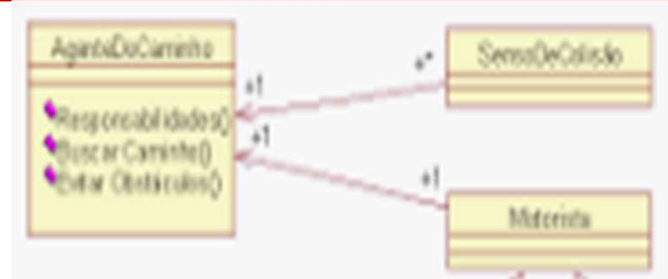
Qual decisão você pode tomar com base nessa métrica?

Análise: se algum método da Classe utilizar atributos que não são dela, ele é avaliado como “não coeso”. O nível de coesão é dado pelo número de atributos utilizados pelo método sobre o total de atributos da Classe. Se a divisão der 1, a coesão é total. Quanto maior o valor obtido da divisão, pior a coesão.

Se um método não está totalmente coeso, ele rompe com o princípio de Encapsulamento dos Objetos.

CBO

- **Acoplamento entre Classes (CBO)**



Avalia o quanto uma Classe depende de associações com outras Classes para realizar suas atividades.

Uma Classe está acoplada a outra quando ela referencia objetos de outra Classe (associa-se com outra Classe).

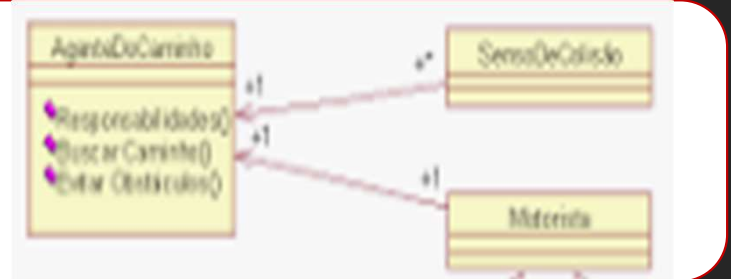
Essa avaliação é feita, observando as relações entre Classes.

CBO = número de Associações que partem da Classe analisada, apontando para outras Classes, o que implica na criação de ligação com Objetos da Classe associada.

Calculado Classe a Classe

CBO

- Acoplamento entre Classes (CBO)**



Classe ALUNO

-Matricula: integer
 -Nome: char
 -Telefone: integer
-Endereço: MoradiaFixa

+Matricular()
 +TrancarMatricula()
 +CancelarMatricula()
 +PagarMensalidade()
 +NegociarAtraso()
 +FazerProva()
 +EntregarTrabalhos()
 +ComunicarAusência()
 +Consultar()
 +AtualizarEndereco()

RESIDE

0..*

1

Classe MoradiaFixa

-CEP: integer
 -NomeRua: char
 -Numero: integer

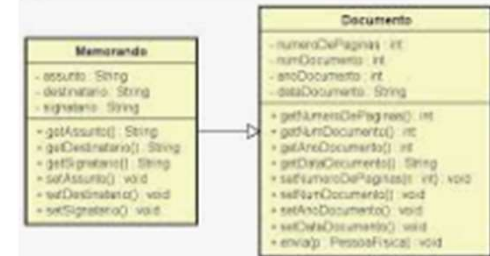
+Criar()
 +Excluir()
 +Consultar()

CBO de ALUNO = ?

CBO de MORADIA FIXA= ?

CBO

- Acoplamento entre Classes (CBO)**



Classe ALUNO

-Matricula: integer
 -Nome: char
 -Telefone: integer
-Endereço: MoradiaFixa

+Matricular()
 +TrancarMatricula()
 +CancelarMatricula()
 +PagarMensalidade()
 +NegociarAtraso()
 +FazerProva()
 +EntregarTrabalhos()
 +ComunicarAusência()
 +Consultar()
 +AtualizarEndereco()

RESIDE

0..*

1

Classe MoradiaFixa

-CEP: integer
 -NomeRua: char
 -Numero: integer

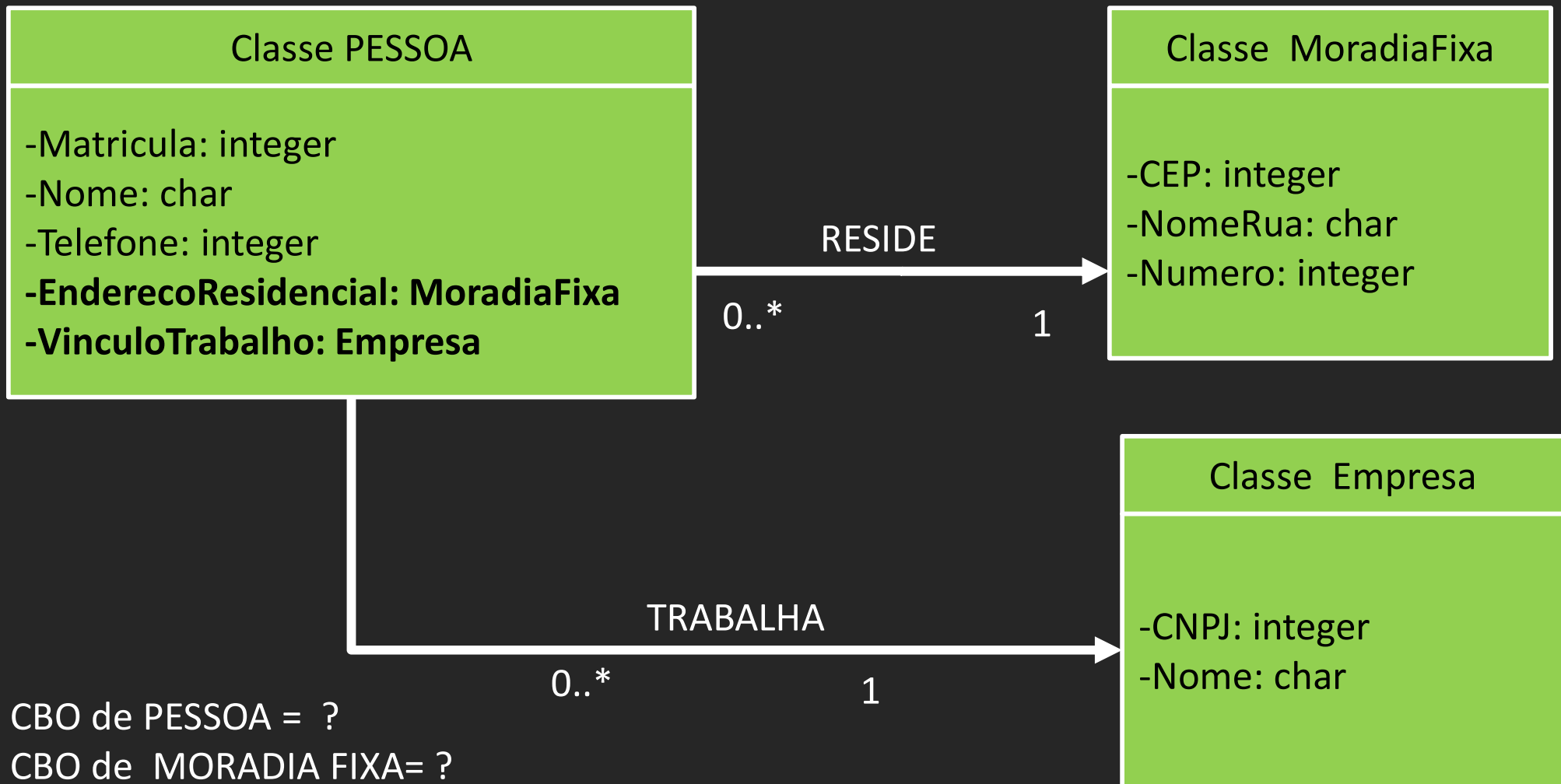
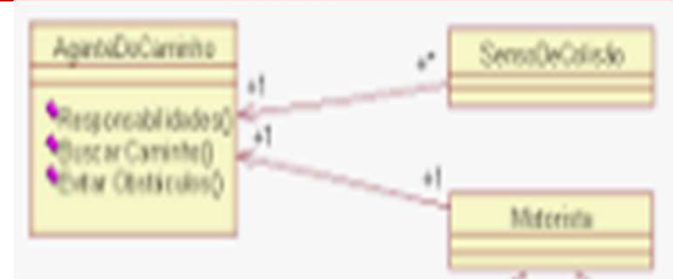
+Criar()
 +Excluir()
 +Consultar()

CBO de ALUNO = 1, pois aponta para uma relação com outra Classe

CBO de MORADIAFIXA= 0, pois não depende obrigatoriamente de outra Classe

CBO

- **Acoplamento entre Classes (CBO)**



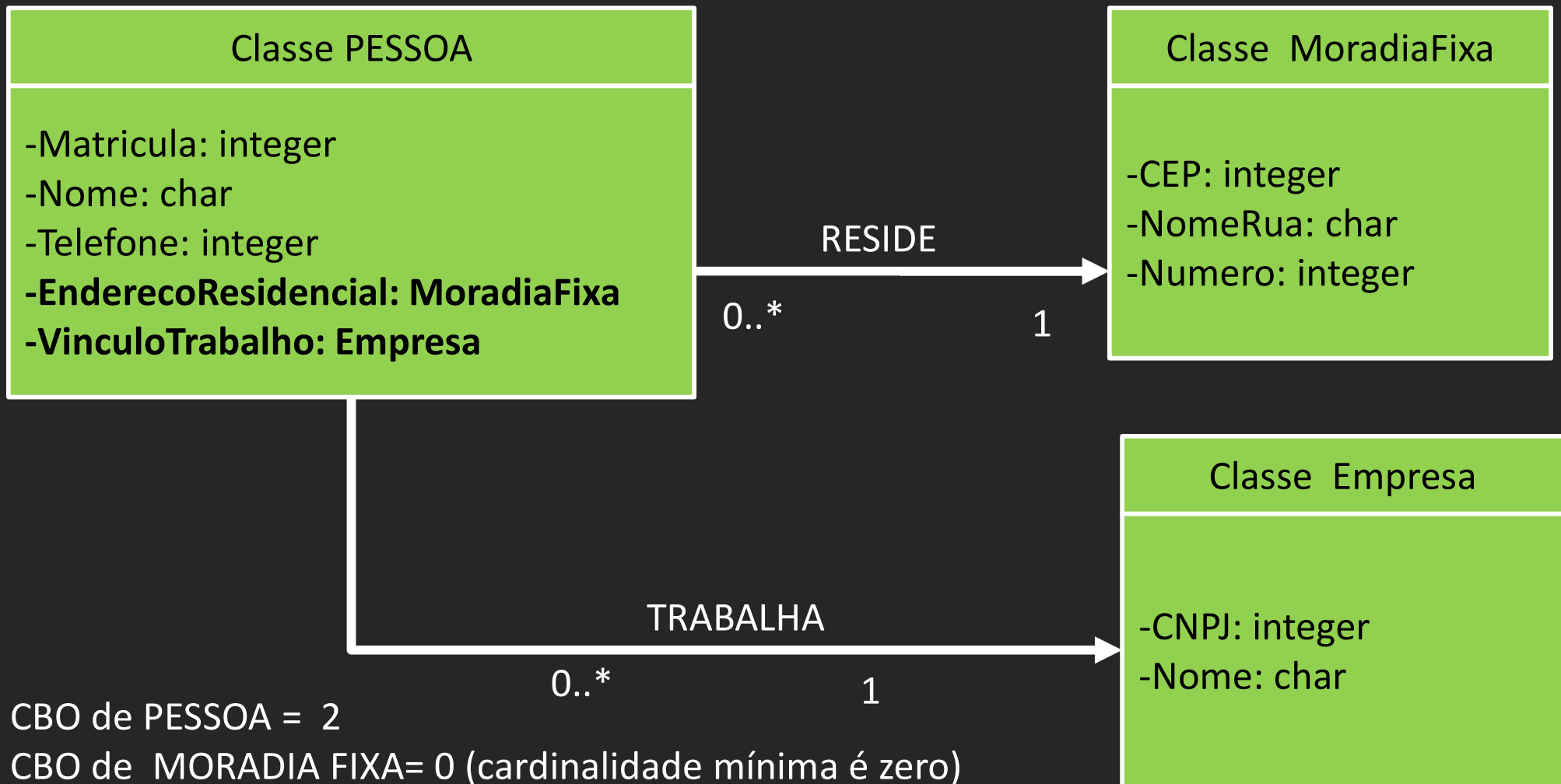
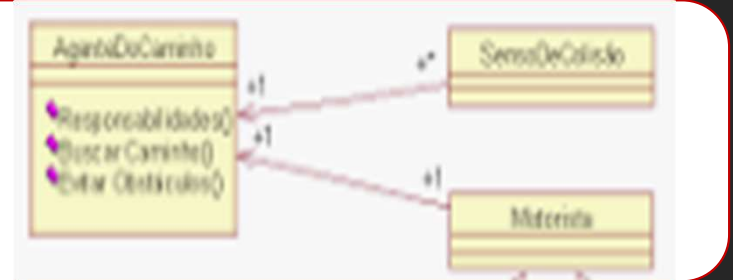
CBO de PESSOA = ?

CBO de MORADIA FIXA= ?

CBO de EMPRESA = ?

CBO

- Acoplamento entre Classes (CBO)**



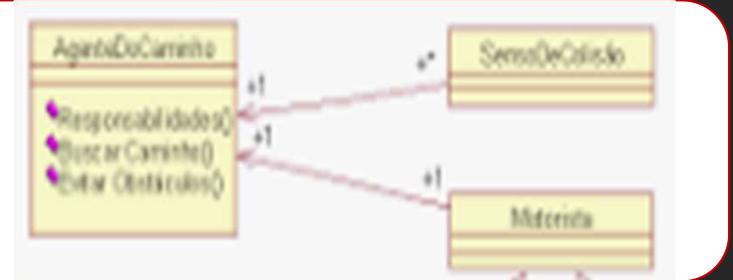
CBO de PESSOA = 2

CBO de MORADIA FIXA= 0 (cardinalidade mínima é zero)

CBO de EMPRESA = 0 (dependência não é obrigatória)

CBO

- Acoplamento entre Classes (CBO)**



Classe ALUNO

-Matricula: number[5]
 -Nome: string[50]
 -Telefone: number[8]
 -Endereço: string[120]

+Matricular()
 +TrancarMatricula()
 +CancelarMatricula()
 +PagarMensalidade()
 +NegociarAtraso()
 +FazerProva()
 +EntregarTrabalhos()
 +ComunicarAusência()
 +Consultar()

Classe INSCRIÇÃO

+Aluno: Aluno
 +Curso: Curso
 +IdInscricao: number [3]
 +DataMatricula: data

Classe CURSO

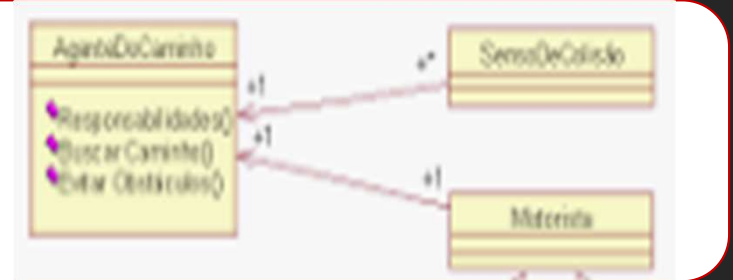
-IDCurso: number[5]
 -Nome: string[50]
 -Duração: number[1]

+Criar()
 +Excluir()
 +Consultar()

CBO de ALUNO = ?
 CBO de INSCRIÇÃO = ?
 CBO de CURSO = ?

CBO

- Acoplamento entre Classes (CBO)**



Classe ALUNO

-Matricula: number[5]
 -Nome: string[50]
 -Telefone: number[8]
 -Endereço: string[120]

+Matricular()
 +TrancarMatricula()
 +CancelarMatricula()
 +PagarMensalidade()
 +NegociarAtraso()
 +FazerProva()
 +EntregarTrabalhos()
 +ComunicarAusência()
 +Consultar()

Classe INSCRIÇÃO

+Aluno: Aluno
 +Curso: Curso
 +IdInscricao: number [3]
 +DataMatricula: data

0..1

Classe CURSO

-IDCurso: number[5]
 -Nome: string[50]
 -Duração: number[1]

+Criar()
 +Excluir()
 +Consultar()

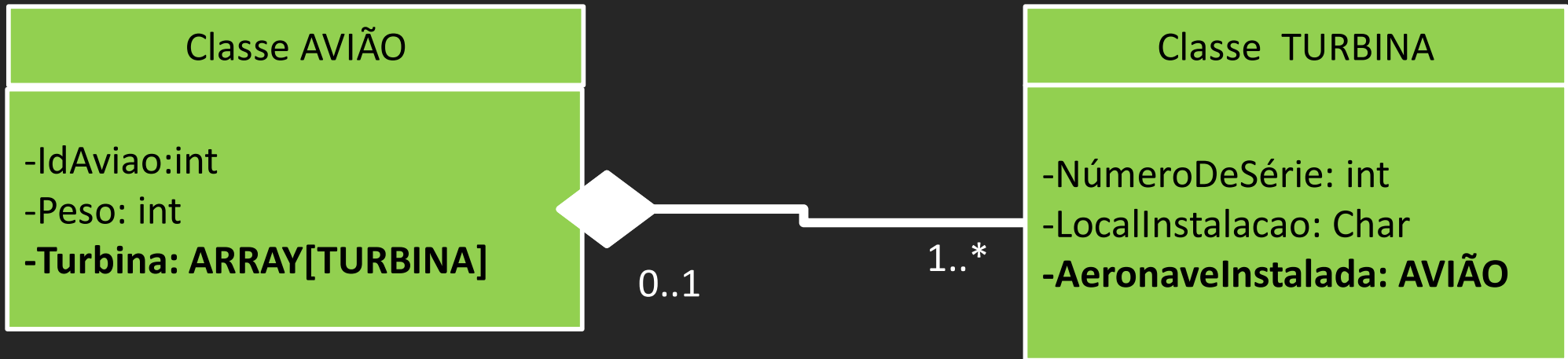
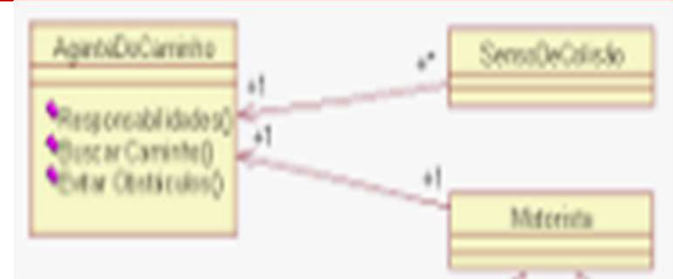
CBO de ALUNO = 0 associação com Curso

CBO de INSCRIÇÃO = 2 associações, com Curso e Aluno

CBO de CURSO = 0 associação com Alunos do Curso

CBO

- **Acoplamento entre Classes (CBO)**

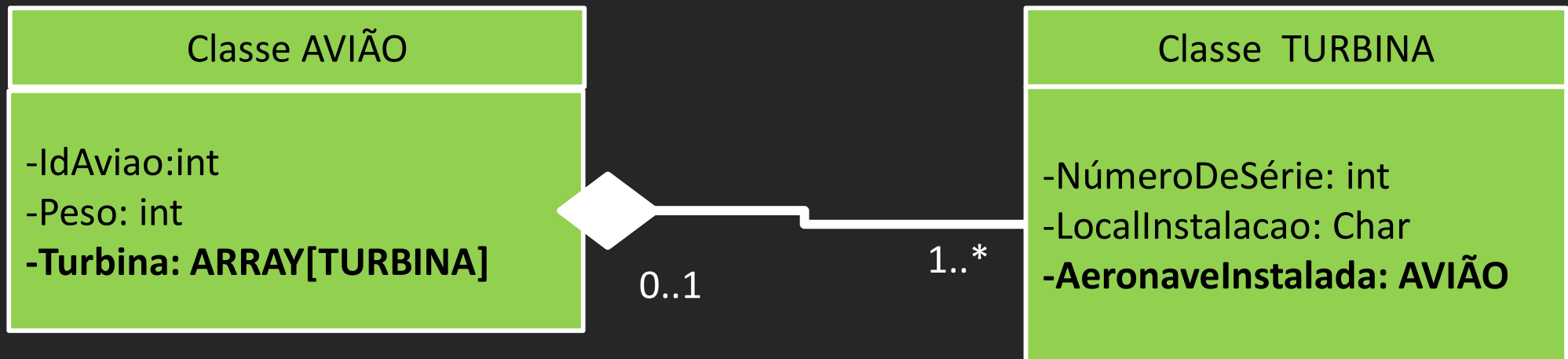
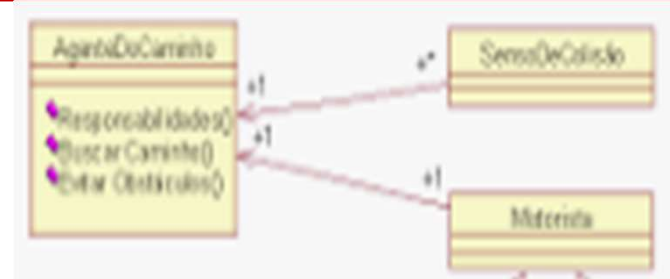


CBO de AVIÃO = ?

CBO de TURBINA = ?

CBO

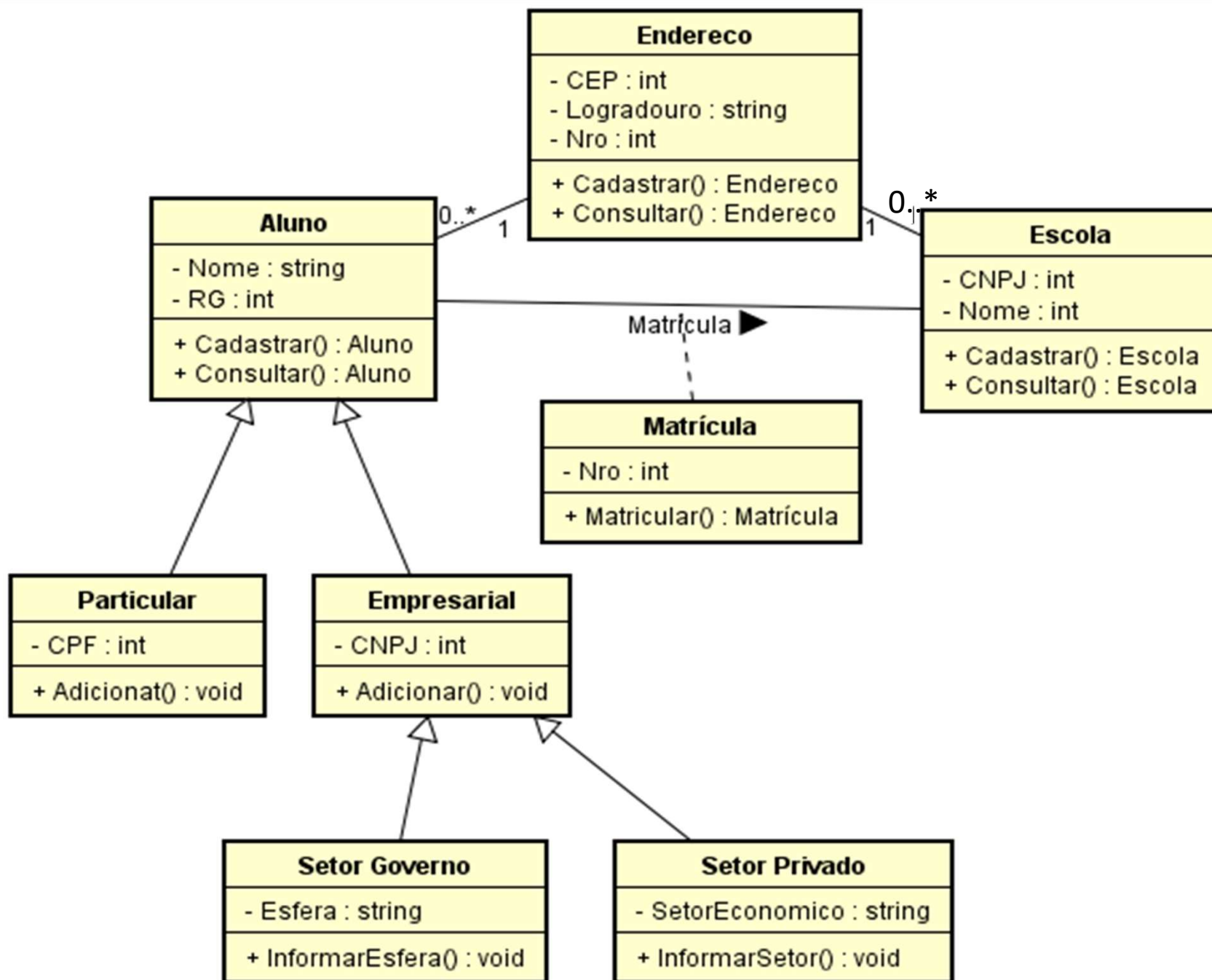
- **Acoplamento entre Classes (CBO)**



CBO de TURBINA = 0, pois aponta para uma relação não obrigatória com outra Classe

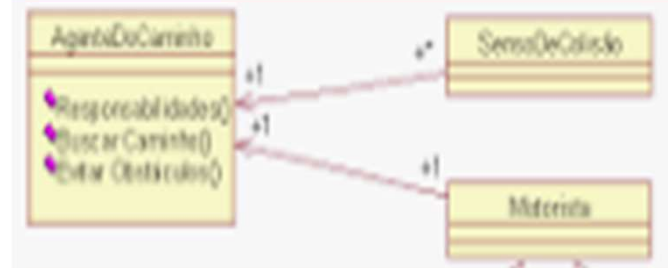
CBO de AVIÃO= 1 , pois depende de outra Classe, tendo relação OBRIGATÓRIA (cardinalidade mínima 1 na TURBINA)

Calcule o CBO



CBO

- **Acoplamento entre Classes (CBO)**



Qual decisão você pode tomar com base nessa métrica?

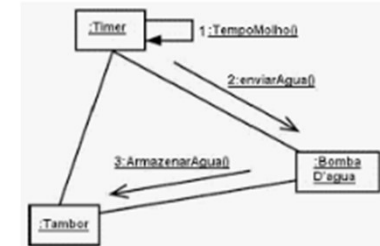
Análise: se o número de acoplamento for alto entre duas Classes, elas podem eventualmente ser Generalizadas ou seja, Unificadas.

OU

Se uma Classe aciona muitas outras Classes, isso pode indicar a necessidade de uma revisão do seu tipo e conteúdo (de Entidade para Controle, por exemplo) e portanto, pode implicar em reengenharia do projeto.

RFC

- Resposta para uma Classe (RFC)



Avalia quantas vezes um Método de uma Classe chama Métodos de outra Classe.

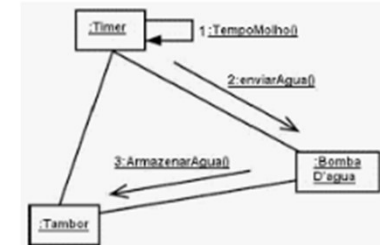
Indica o nível de complexidade lógica que uma Classe tem para resolver um processo de negócio (rotina da aplicação por completo).

RFC = número de vezes em que ocorrem acionamentos (mensagens) que os Métodos da Classe fazem para Métodos de outras Classes.

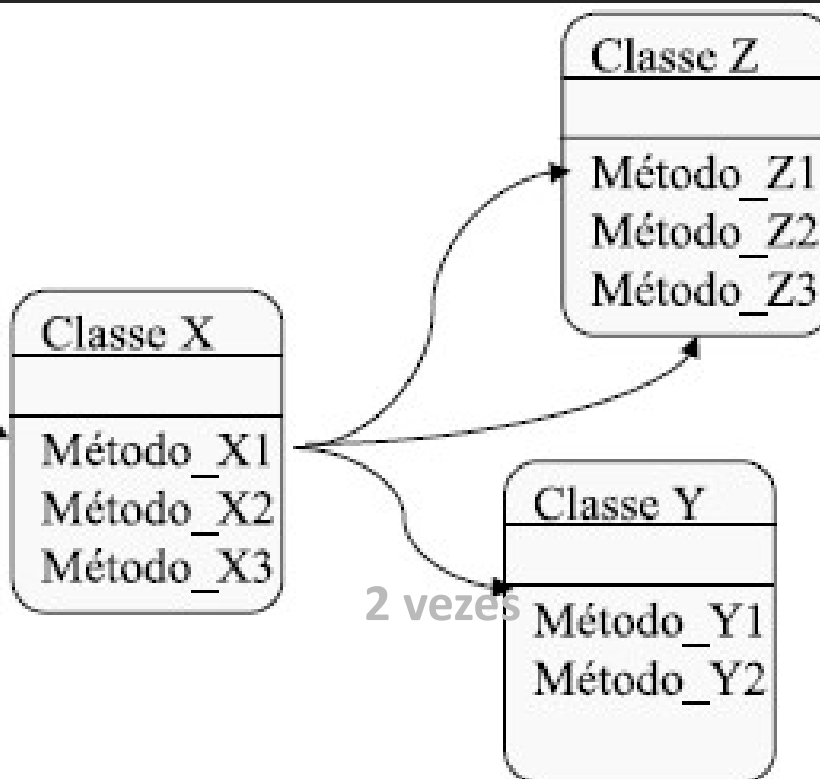
Calculado Classe a Classe, Método a Método.

RFC

- Resposta para uma Classe (RFC)



Chegada de mensagem para objX



Algoritmo do Método_X1

Inteiro A;
Inteiro B;
Boolean C;
Inteiro D;

*Obtém input de A
Chama **Método_Z1 (A)**;

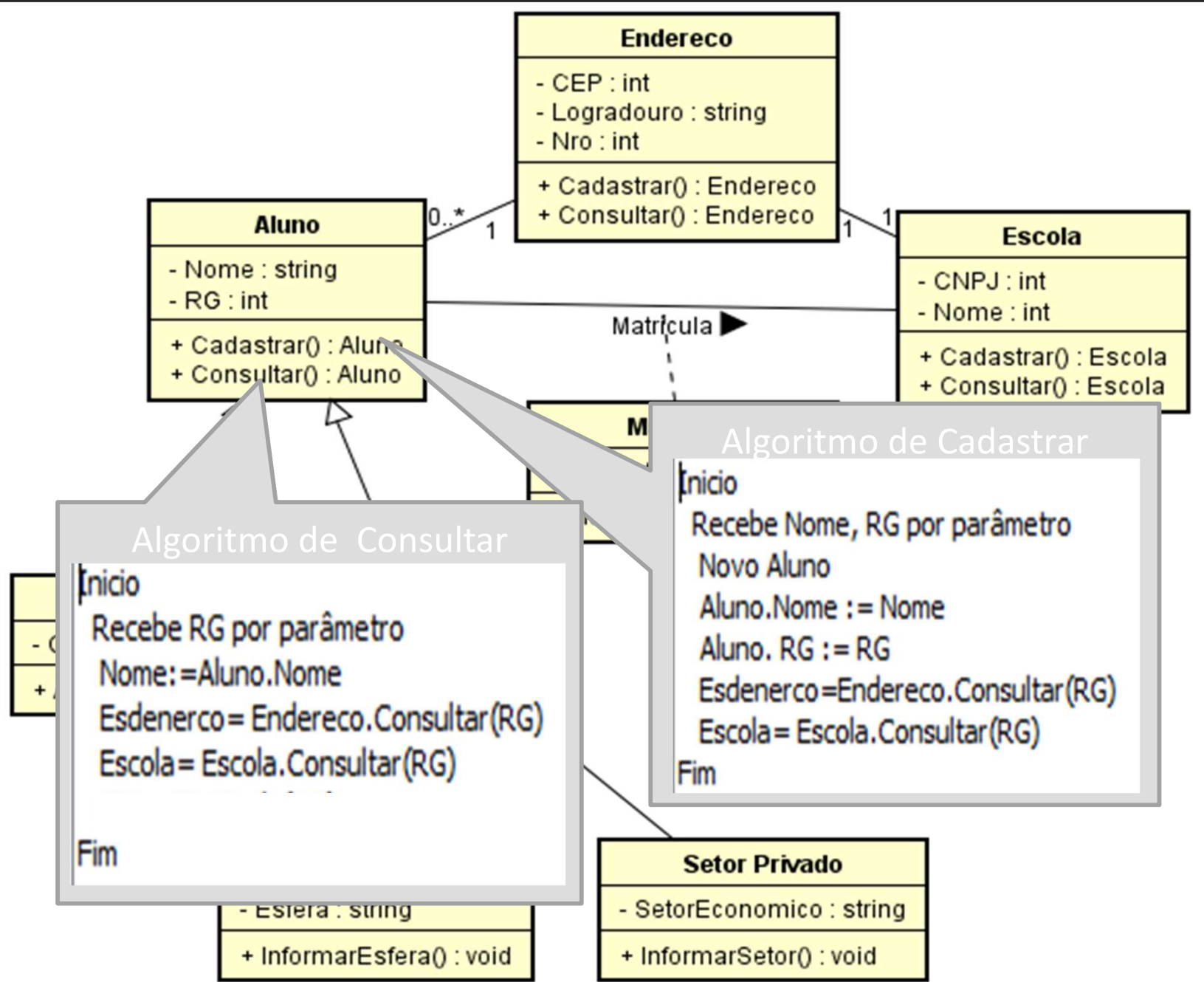
*Obtém input de B
Chama **Método_Z3 (B)**;

*Avalia o maior número entre A e B
C = Chama **Método_Y1 (A, B)**;

*Avalia o maior número entre A e D
D = 10;
C = Chama **Método_Y1 (A, D)**;
Fim_Algoritmo

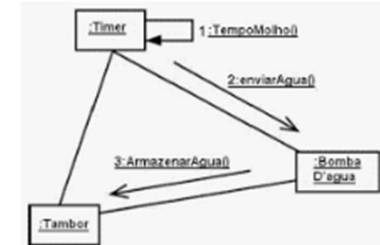
RFC da Classe X = 4

Calcule o RFC



RFC

- Resposta para uma Classe (RFC)



Qual decisão você pode tomar com base nessa métrica?

Análise: o nível de RFC é baseado na quantidade de métodos que uma Classe aciona de outras e de si própria.

Se uma Classe aciona demasiadamente uma única outra Classe, isso aponta que seus Métodos não lhe garantem auto suficiência, apontando uma provável necessidade de Agregação de Classes (Generalização) ou incorporação da lógica de um método de outra classe no método chamador, (revisão na lógica dos métodos, mudando escopos).

MÉTRICAS DE AVALIAÇÃO ESTRUTURAL

ATIVIDADE PRÁTICA



1º) Usando o arquivo Solucao-CASO-COMPLETO-FACULDADE.ASTAH,...
calcule o DIT, NOC e CBO das Classes.

2º) Usando o arquivo Pedidos-com-classes.pptx...
calcule o WMC, NOC, DIT, CBO das Classes.

3º) Usando o arquivo ClasseCarro.pptx
Calcule o LCOM dos Métodos e RFC da Classe

MÉTRICAS DE AVALIAÇÃO ESTRUTURAL

ATIVIDADE PRÁTICA



3º) Qual o LCOM e o RFC do método a seguir?

```
package cursojava00aula03;

/**
 *
 * @author maddo
 */
public class CursoJava00Aula03 {

    public static void main(String[] args) {

        Pessoa maddo = new Pessoa();

        maddo.setNome("Marco Aurélio");
        maddo.setCidadeNatal("Brasília");
        maddo.setNacionalidade("Brasileira");

        System.out.println("Nome: "+maddo.getNome());
        System.out.println("Cidade Natal: "+maddo.getCidadeNatal());
        System.out.println("Nacionalidade: "+maddo.getNacionalidade());

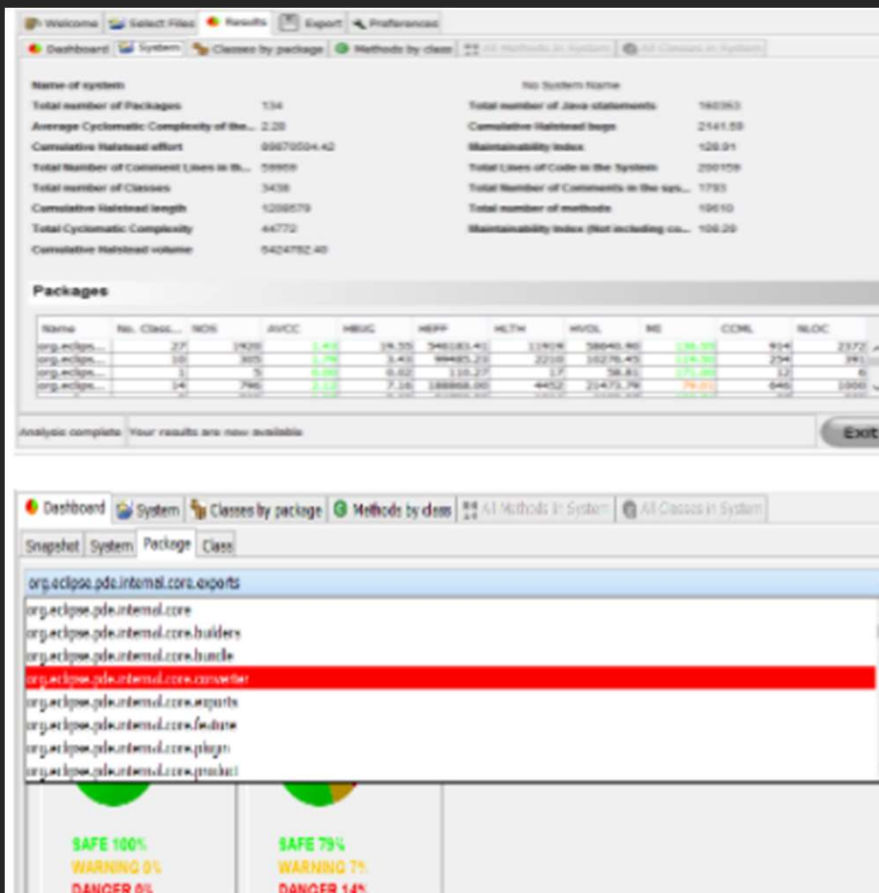
    }

}
```

Existem softwares de inspeção diagramas que calculam automaticamente essas métricas estudadas!

Exemplo:

VIRTUAL MACHINERY



Download
JHawk Demo

WMC, CBO, RFC, LCOM, DIT, NOC - 'The Chidamber and Kemerer Metrics'

Existem softwares de inspeção de código que podem gerar ainda mais métricas avaliativas (calcular número de linhas de código, linhas de decisão, de comando, desvios lógicos, etc.)

Um desses softwares é o  SourceMonitor

Com ele, é possível inspecionar códigos escritos em C, JAVA e outras linguagens.

Basta você apontar o diretório onde estão seus fontes e ele vai calcular uma série de métricas avaliativas sobre a complexidade da sua aplicação que influenciará nas manutenções futuras do software.

É possível visualizar todos os fontes de uma pasta e comparar suas complexidades!



SourceMonitor

Abrindo fontes para inspeção

Selecione a linguagem de programação usada nos fontes a inspecionar



Select Language

Create New Project

Step 1 of 7: Specify source code language. In order to create a new SourceMonitor project, you need to specify a language and source file extensions. The extensions will be used to select the initial file list when you create a new checkpoint. (You can edit this list as needed.)

Project Source Code Language

☐ C++
 ☐ C#
 ☒ Java
 ☐ VB.NET
 ☐ Delphi
 ☐ C
 ☐ HTML
 ☐ Visual Basic

Source File Extensions

Include:

Exclude:



SourceMonitor

Abrindo fontes para inspeção

Select Language

Step 2 of 7: Specify the project name and directory of the project file. The metrics data is saved in this project file (name = project name, extension = ".smproj"). Enter only valid file pathname characters (don't use <, >, :, ", /, \, ?, *, or |). Enter an absolute directory path (not relative).

Create New Project

Project File Name:
TesteComparado

Project File Directory:
-3SI-2020\Aula16-BoasPraticasGovernancaQualidade\ ...

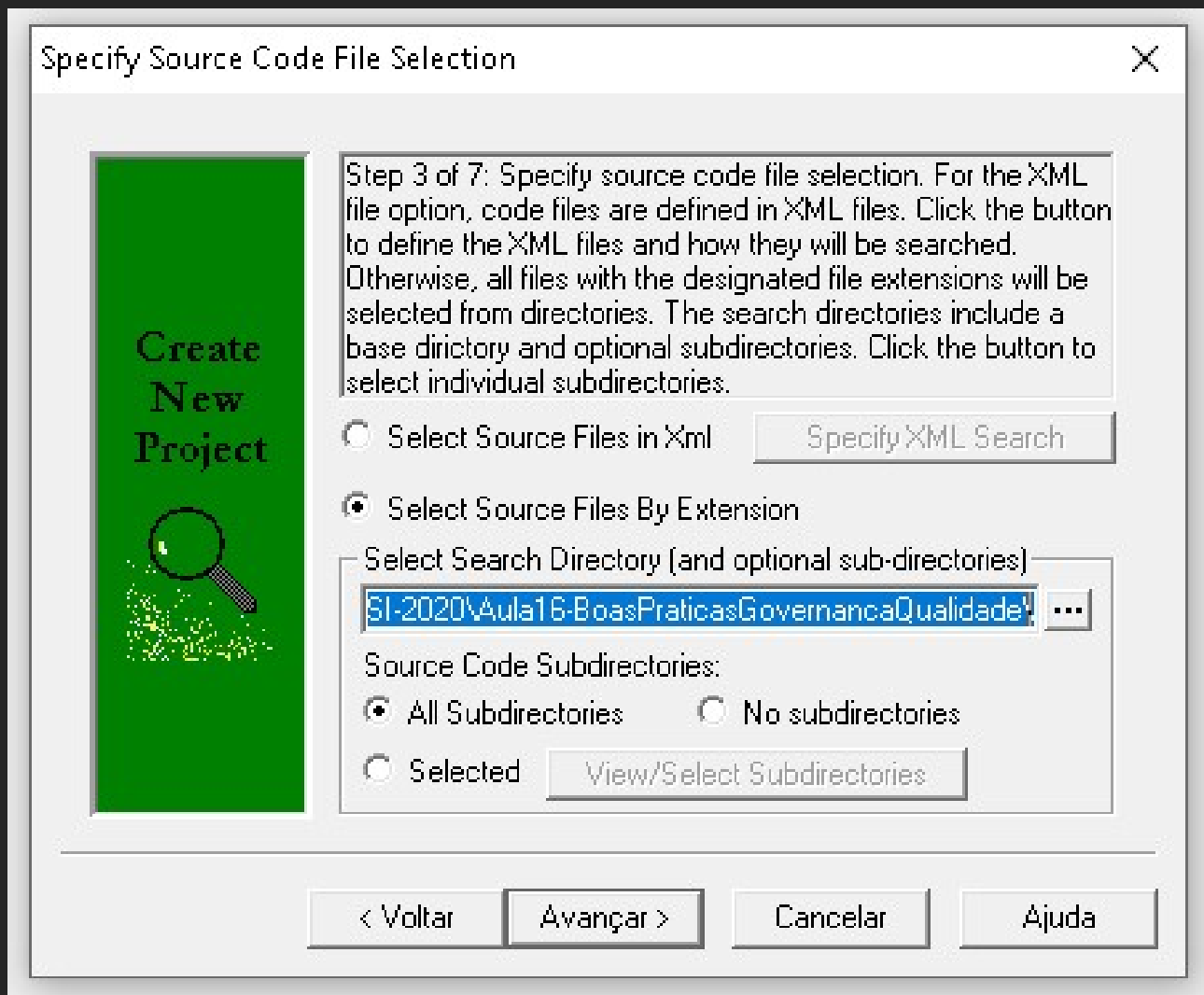
< Voltar Avançar > Cancelar Ajuda

Selecione a pasta e dê um nome para o seu relatório de métricas



SourceMonitor

Abrindo fontes para inspeção



Selecione a pasta onde estão os fontes e dê um nome para o seu relatório de métricas



SourceMonitor

Abrindo fontes para inspeção

Specify Initial Checkpoint

Create New Project

Step 6 of 7: Specify an optional initial project checkpoint. The metrics for all of the source files in a project are saved in checkpoints so that you can track the progress of your work. If you wish, you can specify the name of an initial checkpoint so that, as soon as your project is created, you will see a display of the files in your first checkpoint.

☒ Create First Project Checkpoint

Checkpoint Name:

Note: The option to allow parsing of UTF-8 files applies to all projects. It can be changed on the "General" tab in the "Options" dialog. The current value is shown below.

☐ Allow parsing of UTF-8 files

< Voltar

Avançar >

Cancelar

Ajuda

Defina um checkpoint para que o software guarde um histórico das métricas geradas ao longo do projeto



SourceMonitor

Navegação

SourceMonitor - [Java Checkpoints In Project 'CC' [Base Directory: 'C:\Users\renat\Documents\Docencia\FIAP-2020\QualidadeSW-3SI-2020\Aula16-BoasPraticasGover]

File Edit View Checkpoint Window Help

Checkpoint Name	Created ...	Files	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmtns/Method	Max Complexity	Max Depth	Avg Depth	Avg Com
Baseline	15 Jun 2020	2	112	76	36,8	38	8,9	2	1,00	34,00	15	5	3,18	

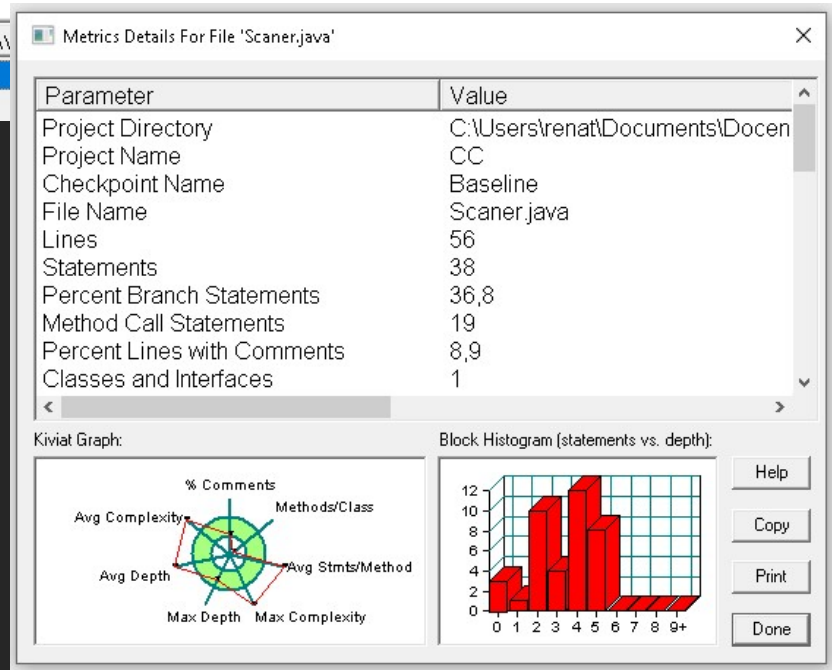
Double click

SourceMonitor - [Files in Java Project 'CC', Checkpoint 'Baseline' [Base Directory: 'C:\Users\renat\Documents\Docencia\FIAP-2020\QualidadeSW-3SI-2020\Aula16-]

File Edit View Window Help

File Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg
Scanner.java	56	38	36,8	19	8,9	1	1,00	
TesteComple...	56	38	36,8	19	8,9	1	1,00	

Double click





SourceMonitor

Exemplo do relatório gerado

SourceMonitor

File Edit View Window Help

Java Checkpoints In Project 'TesteComparado' [Base Directory: 'C:\Users\renat\Documents\Docencia\FIAP-2020\QualidadeSW-3SI-2020\Aula16-BoasPraticasGovernancaQualidade\']

Checkpoint Name	Created On	Files	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmt/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
Checkpoint1	15 Jun 2020	3	133	80	27,5	33	7,5	3	2,67	7,88	15	5	2,69	3,75
ComparativoGeral1	15 Jun 2020	0	0	0	0,0	0	0,0	0	0,00	0,00	0	0	0,00	0,00

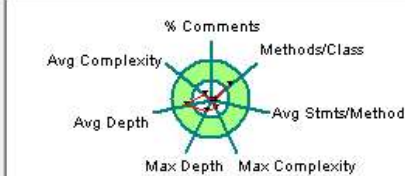
Files in Java Project 'TesteComparado', Checkpoint 'Checkpoint1' [Base Directory: 'C:\Users\renat\Documents\Docencia\FIAP-2020\QualidadeSW-3SI-2020\Aula16-BoasPraticasGovernancaQ...']

File Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmt/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
Produto.java	34	15	0,0	0	0,0	1	6,00	1,00	1	2	1,33	1,00
Scanner.java	56	38	36,8	19	8,9	1	1,00	34,00	15	5	3,18	15,00
TesteComplexity.JAVA	43	27	29,6	14	11,6	1	1,00	23,00	9	5	2,74	9,00

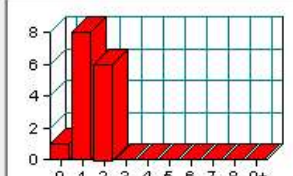
Metrics Details For File 'Produto.java'

Parameter	Value
Project Directory	C:\Users\renat\Documents\Docencia\FIAP-2020\QualidadeSW-3SI-2020\Aula16-BoasPraticasGovernancaQualidade\
Project Name	TesteComparado
Checkpoint Name	Checkpoint1
File Name	Produto.java
Lines	34
Statements	15
Percent Branch Statements	0,0
Method Call Statements	0
Percent Lines with Comments	0,0
Classes and Interfaces	1
Methods per Class	6,00
Average Statements per Method	1,00
Line Number of Most Complex Method	7
Name of Most Complex Method	Produto.setAltura()

Kiviatt Graph:



Block Histogram (statements vs. depth):



Help
Copy
Print
Done



SourceMonitor

Interpretação das principais métricas

Número de linhas do fonte

Número de linhas de comando

% linhas de desvio lógico

% linhas de comentário

Profundidade da árvore de decisão em função dos aninhamentos

Fator de complexidade geral, calculado pelo software para comparar os diversos fontes

Parameter	Value
Lines	56
Statements	38
Percent Branch Statements	36.8
Method Call Statements	19
Percent Lines with Comments	8.9
Classes and Interfaces	1
Methods per Class	1,00
Average Statements per Method	34,00
Line Number of Most Complex Method	12
Name of Most Complex Method	Scanner.main()
Maximum Complexity	15
Line Number of Deepest Block	25
Maximum Block Depth	5
Average Block Depth	3.18
Average Complexity	15.00

Most Complex Methods in 1 Class(es):	Complexity, Statements, Max Dep
Scanner.main()	15, 34, 5, 19

Block Depth	Statements
0	3
1	1
2	10
<	>

FERRAMENTAS DE CONTROLE DE VERSÃO



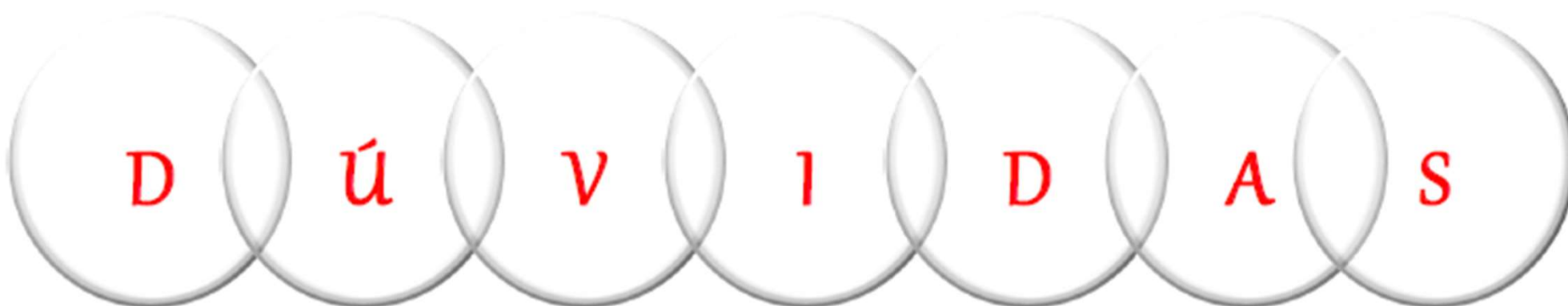
PROJETO

CONVITE...

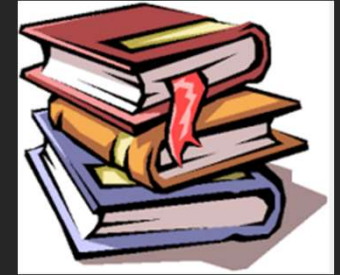
Calcule WMC, DIT, NOC, CBO com base no Diagrama de Classes do seu projeto Challenge.

Calcule também o LCOM e RFC com base no estudo de 1 Classe com Métodos que acionam outros Métodos da própria Classe ou de outra (use a Classe mais simples possível).

VALIDE COM O PROFESSOR!



Referência bibliográficas



BIBLIOGRAFIA :

- MOLINARI, Leonardo. Testes de Software – Produzindo Sistemas Melhores e Mais Confiáveis, 4a. Edição. Editora Erica, 2013.
- MOLINARI, Leonardo. Inovação e Automação de Testes de Software, 1ª edição. Érica, 2010.
- CMMi V3. SEI - Software Engineering Institute., USA, 2007. Disponpivel na biblioteca online da Carnegie Melon University.
- Reis, Luís Filipe Souza. ISO 9000/Auditorias de sistemas da qualidade.Editora: Érica, 1995.

TESTE DE SOFTWARE

Continua na próxima aula...

PROFESSOR:
RENATO JARDIM PARDUCCI

PROFRENATO.PARDUCCI@FIAP.COM.BR