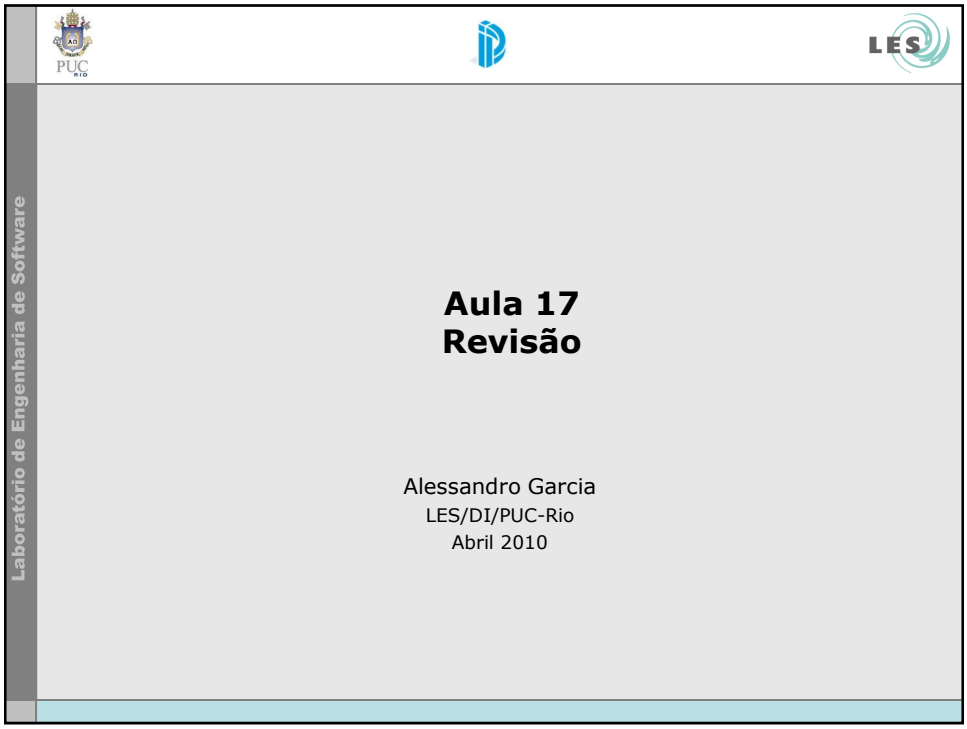
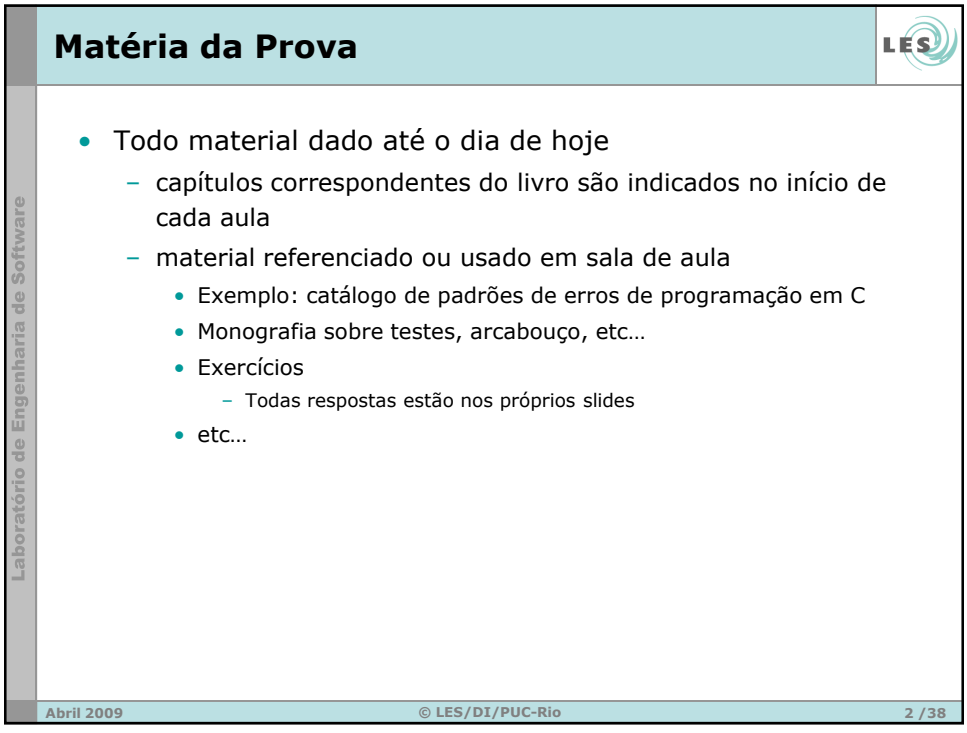

Laboratório de Engenharia de Software
Aula 17
Revisão
Alessandro Garcia
LES/DI/PUC-Rio
Abril 2010


Matéria da Prova
Laboratório de Engenharia de Software
• Todo material dado até o dia de hoje
– capítulos correspondentes do livro são indicados no início de cada aula
– material referenciado ou usado em sala de aula
• Exemplo: catálogo de padrões de erros de programação em C
• Monografia sobre testes, arcabouço, etc…
• Exercícios
– Todas respostas estão nos próprios slides
• etc…
Abril 2009
© LES/DI/PUC-Rio
2 / 38

Pontos a serem revisados

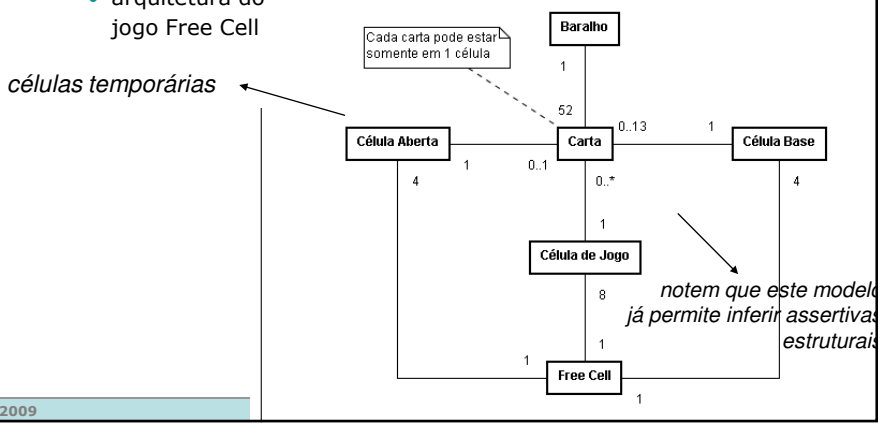
- Dúvidas e erros tipos cometidos em provas passadas (e/ou trabalhos)
  - Modelo conceitual vs. Modelo físico vs. Exemplo físico
    - Diferenças entre os diferentes modelos, diagramas e notações
  - Assertivas
    - Como especificar: linguagem formal ou natural?
  - Relacionamento entre faltas-erros-falhas e...
    - Inspeção, Teste, e Assertivas
  - Outras perguntas específicas
    - Linguagens de scripts, porcentagem da prova sobre o trabalho, respostas em C++/Java, Acoplamento vs. Coesão vs. Encapsulamento, etc...

Definição dos modelos

- Modelo conceitual vs. Modelo físico vs. Exemplo físico
  - Diferenças entre os diferentes modelos, diagramas e notações
  - Trabalho: boa parte dos erros não foi na notação, mas sim onde deve ser descrito o que... Exemplos:
    - inclusão de detalhes de implementação no modelo da arquitetura
    - confusão: modelo físico vs. exemplo físico
- Então: segue algumas dicas de boas práticas...

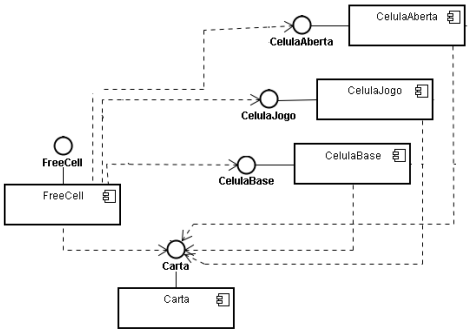
Se for pedido modelo conceitual...

- **... da arquitetura (baseado em um conjunto de requisitos)**
  - não esqueçam das interfaces!!!!
  - exemplos:
    - arquitetura do jogo Free Cell



Modelo conceitual da arquitetura

- Modelo conceitual da arquitetura
  - descreve um conceito sem se preocupar com a forma de sua implementação
  - exemplo (trabalho): descrição conceitual da arquitetura (não provê certos detalhes de implementação)
    - e.g. módulos que representam estruturas de dados
    - e.g. funções internas/auxiliares



Modelo conceitual

- **Modelo conceitual**
  - descreve um conceito sem se preocupar com a forma de sua implementação
  - exemplos:
    - modelo conceitual do problema Free Cell

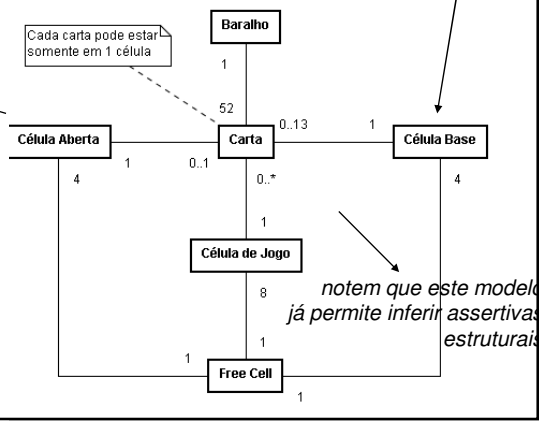
cada célula base é específica para um naipe e deve receber cartas em ordem crescente

células temporárias

um exemplo de assertiva que não pode ser expressa aqui no modelo?

uma instância de carta nunca pode ocupar espaço em mais em um tipo de célula

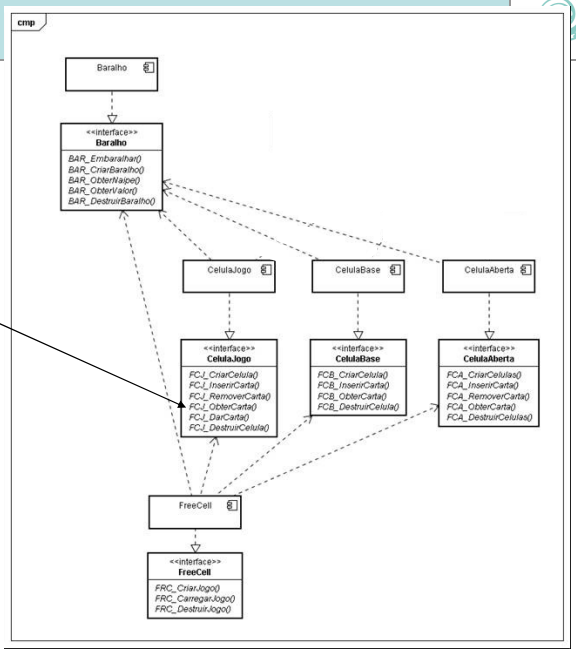
uma instância de carta deve ocupar pelo menos um espaço em uma das células



notem que este modelo já permite inferir assertivas estruturais

Especificação conceitual das interfaces

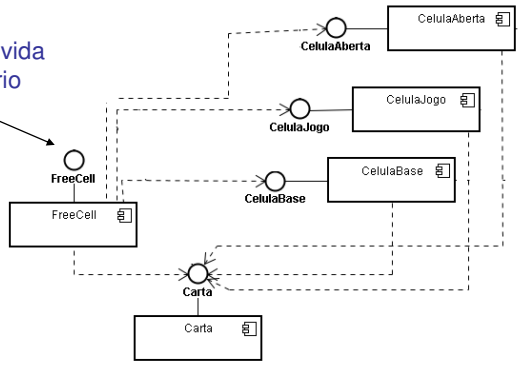
funções nos módulos sofrem nenhuma influência da estrutura de dados (lista)



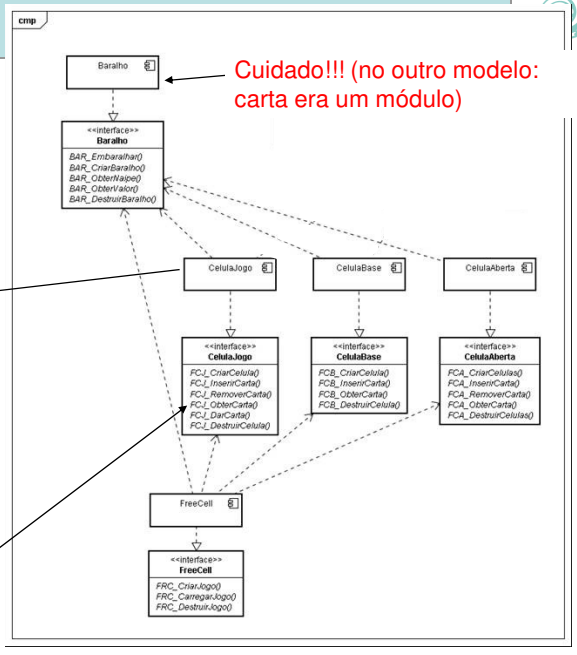
Exemplo de bom modelo arquitetural

- Propriedades de um bom modelo da arquitetura
  - simplicidade: principais módulos estão bem identificados e são derivados geralmente dos requisitos (descrição do problema)
  - notem que modelo arquitetural não requer conhecimento sobre implementação (não era precisa modelar lista neste modelo)
  - relacionamentos e dependências entre módulos são capturados
    - alguns dos relacionamentos somente podem ser capturados quando você raciona sobre as funções que devem estar disponíveis na interface

interface provida para o usuário



Rastreabilidade



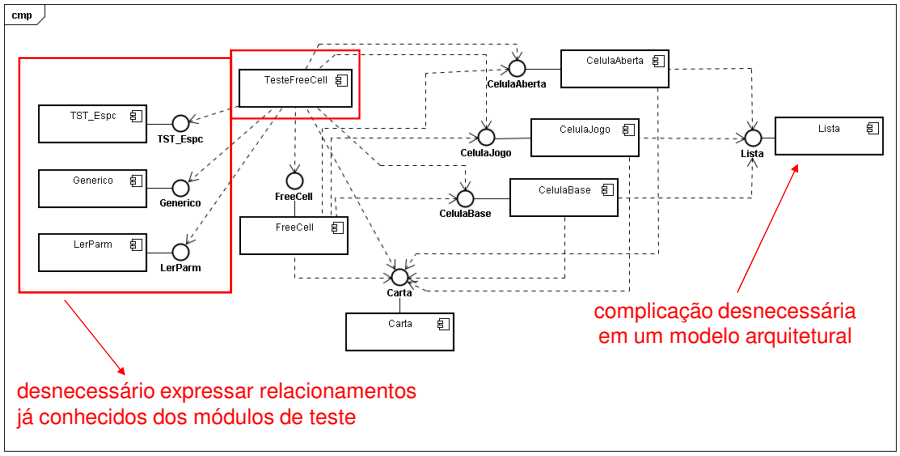
Cuidado!!! (no outro modelo: carta era um módulo)

Note como os mesmos termos são utilizados Consistentemente nas duas descrições

Note as boas escolhas de projeto modular: funções nos módulos sofrem nenhuma influência da estrutura de dados (lista)

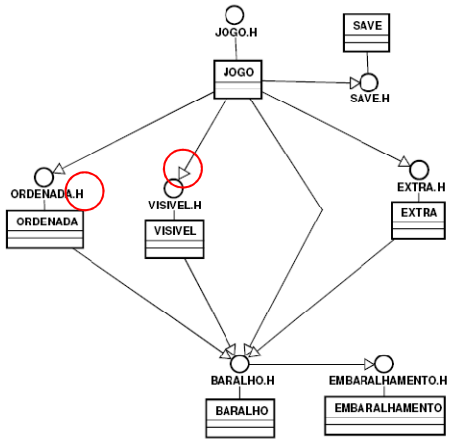
Possíveis melhorias ou simplificações

- Idealmente: **nomear relacionamentos**
- Na prova: se necessário, usar texto auxiliar (linguagem natural) para descrever um conceito que não aparece explicitamente na descrição do problema



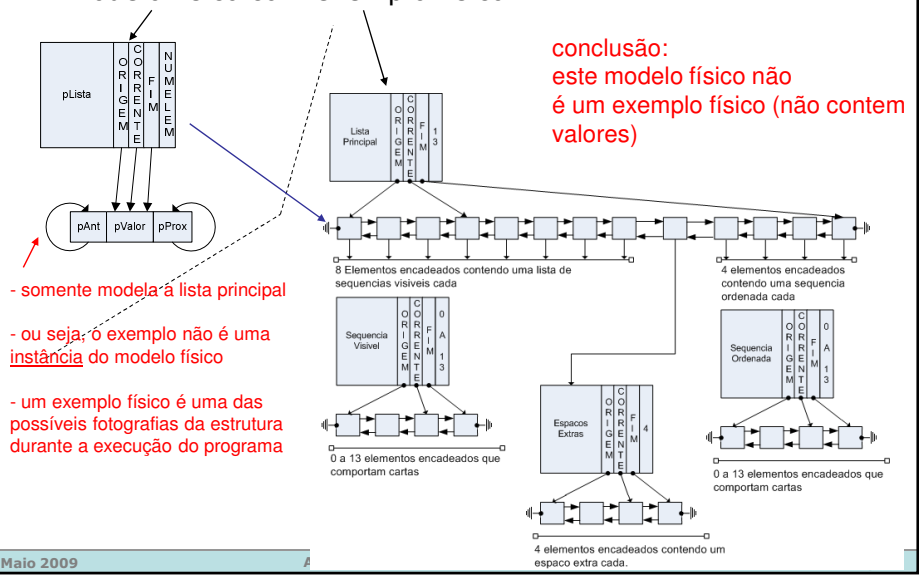
Cuidados especiais....

- *.h no modelo da arquitetura?
- cuidado em criar confusões com elementos da notação (neste caso, herança/especialização)



Cuidado em confundir...

- Modelo físico com exemplo físico



conclusão:
este modelo físico não
é um exemplo físico (não contem valores)

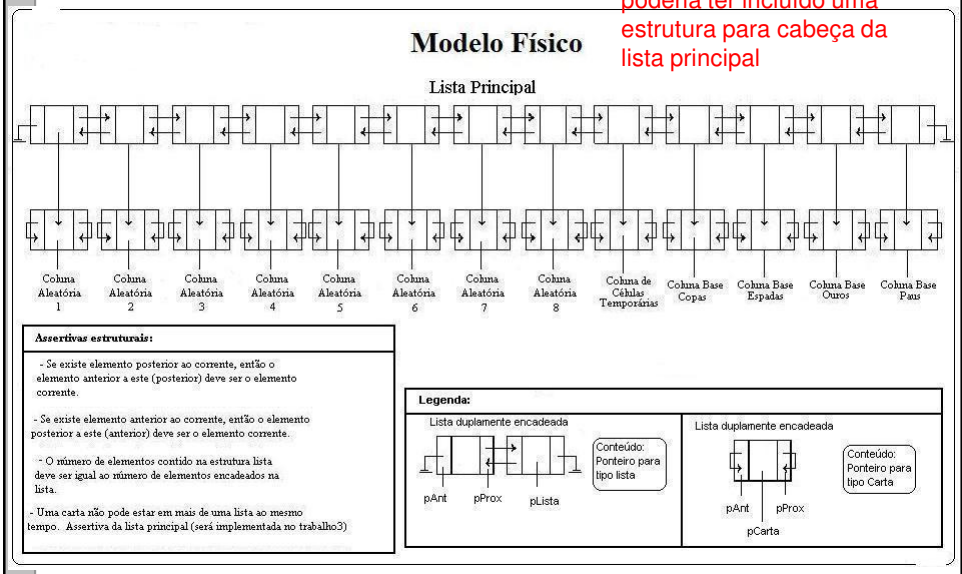
- somente modela a lista principal

- ou seja, o exemplo não é uma instância do modelo físico

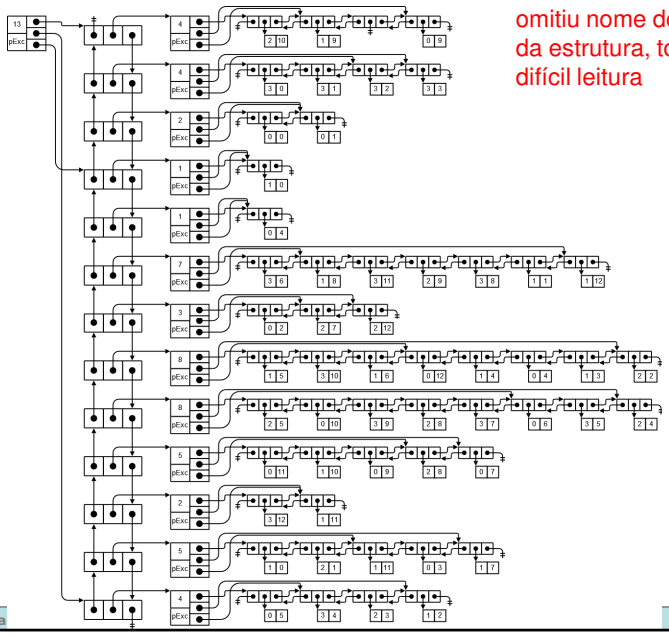
- um exemplo físico é uma das possíveis fotografias da estrutura durante a execução do programa

Bom modelo físico

poderia ter incluído uma estrutura para cabeça da lista principal



Bom exemplo físico



omitiu nome dos elementos da estrutura, tornando-a de difícil leitura

O que são assertivas?

- Assertivas são relações (expressões lógicas) envolvendo dados e estados manipulados
- São definidas em vários níveis de abstração
  - **programas**
    - devem estar satisfeitas para os dados persistentes (arquivos)
  - **módulos (ou classes)**
    - devem estar satisfeitas ao entrar e ao retornar de todas funções
    - assertivas invariantes:
      - assertivas estruturais: corretude das instâncias de estruturas de dados
      - também: certos limites inferiores e superiores associados com certos atributos de classes
  - **funções**
    - devem estar satisfeitas em determinados pontos do corpo da função
    - usualmente assertivas de entrada e assertivas de saída
      - pré e pós condições
      - na definição das pós-condições, assume-se que as pré-condições foram satisfeitas
    - dificilmente são expressas em modelos físicos

Notação para assertivas?

- Em uma lista duplamente encadeada
  1. *∀ pElem ∈ lista* **:** *pElem->pAnt != NULL =>*
     pElem->pAnt->pProx == pElem
     - note o uso da linguagem de programação com algumas extensões de notação

  Outra redação: para todos os elementos *elem* pertencentes a uma *lista duplamente encadeada*, se *elem* possui um antecessor, então o sucessor deste é o próprio *elem*

- É dado um *arquivo-A* contendo n >= 0 registros
  - cada registro contém um campo *chave*
  - *∀ registros* r_i e r_k **:** r_i , r_k ∈ *arquivo-A* **:**
    se r_i antecede r_k então $r_i.chave < r_k.chave$

  Outra redação: é dado um *arquivo-A* contendo n >= 0 registros
  - cada registro contém um campo *chave*
  - o arquivo é ordenado em ordem estritamente crescente segundo *chave*
    - estritamente: sem conter chave repetida

Assertivas – onde colocar?

- É comum que parte das assertivas possam ser alternativamente incluídas em vários pontos:
  - Na definição do módulo, OU
  - Na definição da estrutura de dados, OU
  - Nas funções contidas no módulo, etc...

- **Solução**: para resolver esta ambigüidade de posição, procure incorporar a especificação no elemento *menos abstrato*

Relações entre: Falta, Erro, Falha, ...

- ... Assertivas, Teste, etc...
- Programas podem conter defeitos (ou faltas) que, quando exercitados, provocam erros de funcionamento. Quando observados estes erros passam a ser falhas.
  - defeito: código errado (falta: a mesma coisa que defeito)
    - trecho que código onde é inserido um elemento à mais em um vetor de 10 elementos (estático)
  - erro: estado diferente do esperado/desejado, ainda não observado
    - erro: no momento em que vetor fica com 11 elementos (dinâmico)
    - ou seja, observada a violação da assertiva estrutural
  - falha: estado diferente do esperado ou desejado, observado
    - quando o usuário percebe o erro interno do sistema (dinâmico)

Linguagem de script para testar um módulo

- É o conjunto de palavras-chave necessárias para definir os casos de teste em um arquivo/script de teste

```
== Excluir nó corrente de árvore contendo somente raiz
=excluirNo      0  0
=ehVazia        0  T
=irPai          0  5
=irFilho        0  5
=irEsquerda     0  5
=irDireita      0  5
=obterValor     0  #  5
```

... Palavras-chave para possíveis resultados tbém são parte da linguagem de script

Item de interface sendo testado

resultado esperado

```
  . .
= Criar árvore
criar       OK
irdir       ArvoreVazia

= Inserir à esquerda
insdir      CharA   OK

= Obter o valor inserido
obter       CharA   OK

= Ir para no pai, não tem
irpai       EhRaiz

= Inserir à esquerda
insesq      CharB   OK
obter       CharB   OK

= Ir para no pai, tem
irpai       OK
obter       CharA   OK
  . .
```


Logo...

- Altamente recomendado que vocês **não ignorem** tópicos do curso menos explorados (pelo menos, explicitamente) nos trabalhos
  - inspeção de código
  - imposição e conversão de tipos
  - acoplamento, coesão, encapsulamento, etc...
- Revisem exercícios da sala de aula

Perguntas/dúvidas Específicas

- Questões de prova poderão ser respondidas em C++/Java?
  - Somente respostas em C
    - afinal, foi a linguagem utilizada nos trabalhos
  - O objetivo deste curso não é um curso específico de uma linguagem de programação; portanto:
    - em certas aulas, usamos exemplos em diferentes linguagens (p.e. C++/Java) para ilustrar que conceitos/princípios se aplicar a programas em qualquer linguagem
    - raramente se pedirá para você escrever código na prova
      - logo, esta questão se torna praticamente irrelevante para provas
  - Casos especiais:
    - obviamente, certos padrões de organizações de módulos ou padrões de faltas discutidos no curso são específicos para C
      - por exemplo, separação entre interface e implementação: *.h vs. *.c
      - imposição de tipos em C

Questões sobre Modelos

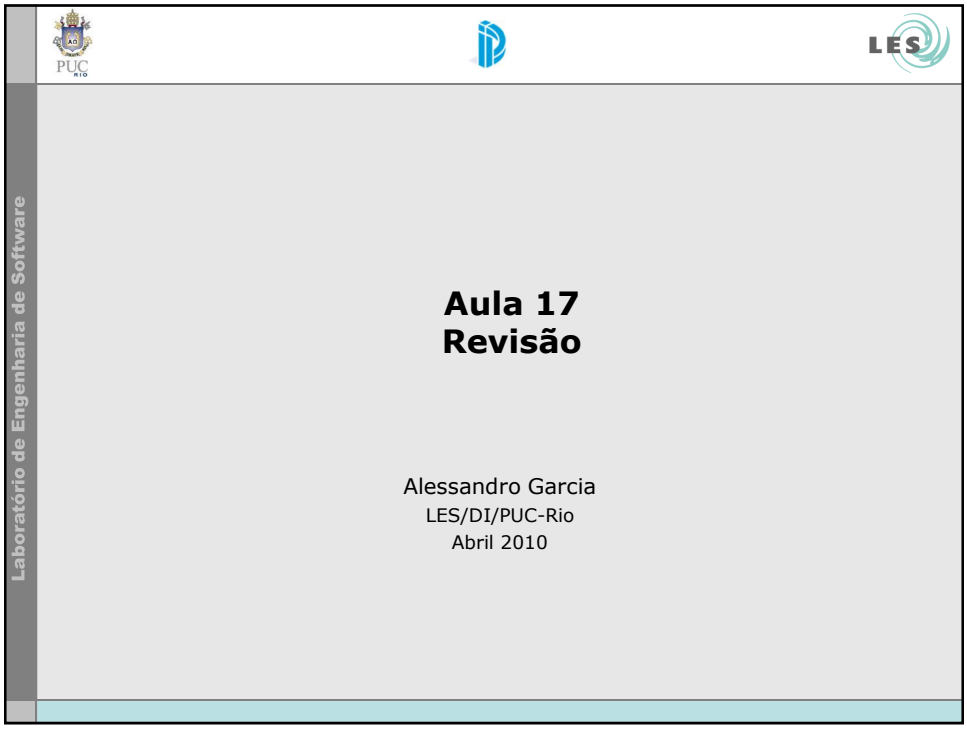
- **Evite "inventar"** elementos da **notação** de modelagem
  - Por exemplo, cada triângulo é uma interface...
    - assim que lê sua especificação não sabe qual foi a intenção do modelador
  - Na pior das hipóteses, crie uma legenda para comunicar a semântica de cada elemento da "sua notação"
- Caso você for usar diferentes estilos de setas para expressar diferentes de relacionamentos entre módulos, **use legendas**...
- **Cuidado com uso errado da notação**, por exemplo:
  - referência ao invés de usar agregação
- **Não esqueça de dar nomes** aos elementos do modelo: nomes dos módulos, das interfaces, relacionamentos, etc...

Dicas gerais

- Se houverem questões do tipo:
  - "Dado o código/modelo abaixo, redija um texto sobre os princípios de de modularidade seguidos ou violados."
- Busque **completude** e coerência na sua resposta, mas escreva **somente o necessário**
  - **Evite divagar** sobre o assunto
  - Não fantasie e escreva sobre o que você não tem certeza do que está falando
  - Tamanho da resposta não é proporcional à nota
- Seja preciso e maximize o uso de **terminologia básica da Programação Modular**
  - por exemplo, use: acoplamento, interface, encapsulamento, assertivas, padrão de programação, faltas, erros, falhas, etc...
  - evite termos demasiadamente genéricos ou não definidos/relevantes neste curso, exemplo: coisa, elemento, classe, etc...

Dicas gerais

- Justifique suas respostas!
  - Evite respostas vagas e sem evidências do tipo
    - "Sim, a função X viola padrões de programação."
    - Prefira respostas com argumentação e precisão apropriada:
      - "A 5ª linha de código da função X viola o padrão de identificadores X porque o prefixo do nome da variável Z...."
      - "A expressão A na linha 3 exibe um padrão de falta clássico pois...."
      - "O uso de malloc e sizeof na linha 7..."


Laboratório de Engenharia de Software
PUC
LES
Aula 17
Revisão
Alessandro Garcia
LES/DI/PUC-Rio
Abril 2010