UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

DCC004 - Algoritmos e Estruturas de Dados II

Conceito de especificação de software

Renato Martins

Email: renato.martins@dcc.ufmg.br

https://www.dcc.ufmg.br/~renato.martins/courses/DCC004

Material adaptado de PDS2 - Douglas Macharet e Flávio Figueiredo

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Introdução

- Análise/Modelagem/Design
  - Quais objetos devem ser criados?
  - Quais características eles devem possuir?
  - Dados, comportamentos, ...

Introdução

- Análise e Projeto
  - Quais as reais necessidades do cliente?
  - Requisitos ➜ Software
- Software complexo ➜ Planejamento
- Sistema mal projetado
  - Prejudica manutenção/extensão
  - Tempo ➜ Dinheiro

Duas Regras Simples

- Keep it Simple Stupid (KISS)!
  - Atingimos a perfeição não quando nada pode acrescentar-se a um projeto mas quando nada pode retirar-se.

Duas Regras Simples

- Keep it Simple Stupid (KISS)!
  - Atingimos a perfeição não quando nada pode acrescentar-se a um projeto mas quando nada pode retirar-se.
- Responsabilidade junto com os dados
  - Atribuir uma responsabilidade ao expert de informação - a classe que possui a informação necessária para preencher a responsabilidade

Dúvida

Entre as seguintes classes do mundo bancário, (Agencia, Conta, ContaCaixa, ContaSimples, Extrato, ExtratoHTML, Moeda, Movimento, Real, Transacao), quem deve ser responsável pela responsabilidade "Localizar a conta com certo número"?

User Stories

User Stories
Kent Beck

- Definição em linguagem natural de diferentes funcionalidades do programa
- Guiam o desenvolvimento do código
- Podemos desenvolver um user story por vez

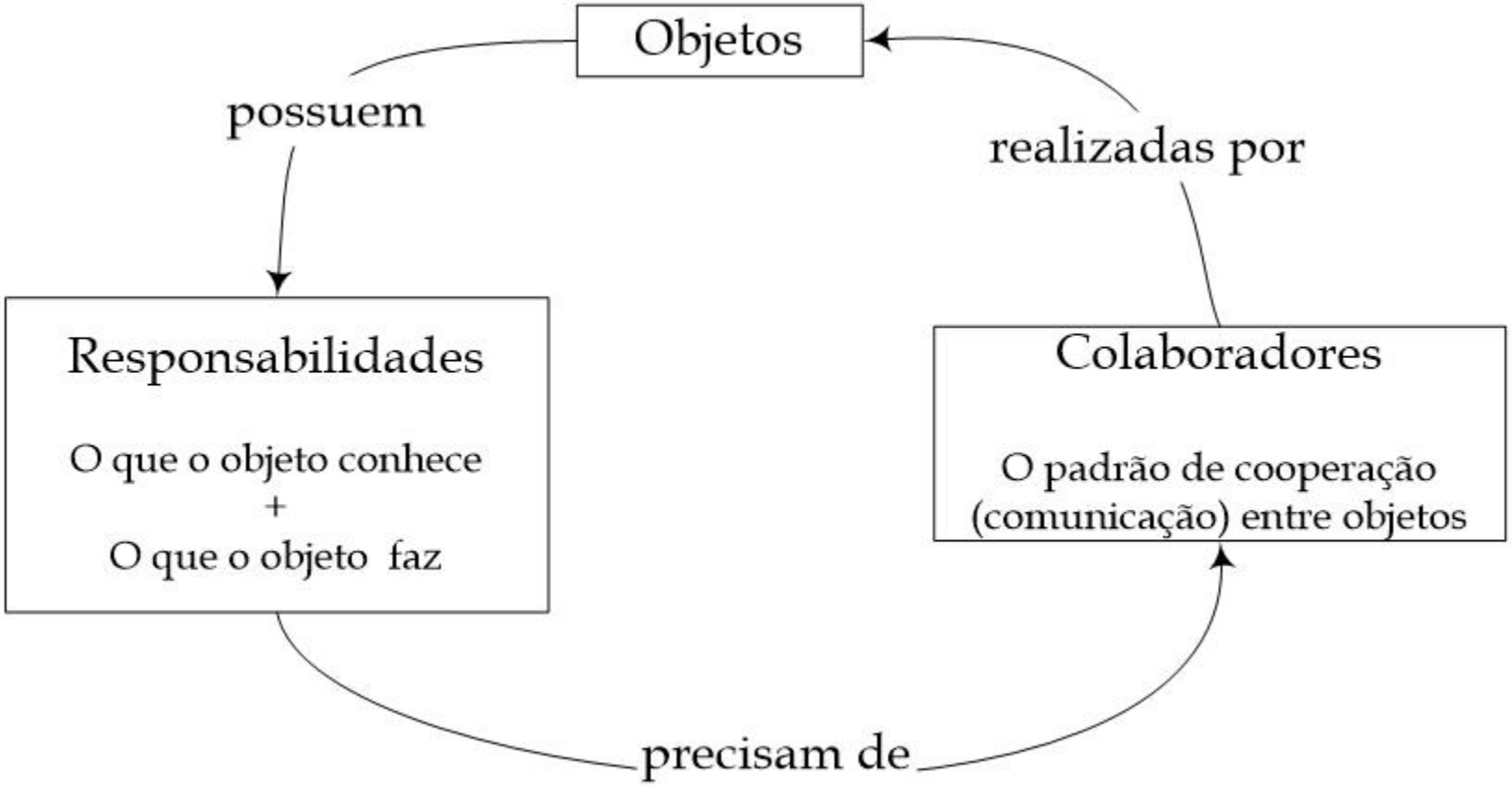
User Stories

Como [Quem?] eu quero [O quê?] para [Por quê?].

Exemplo: Como um cliente da operadora de saúde eu quero procurar um médico pelo nome para obter o endereço do seu consultório.

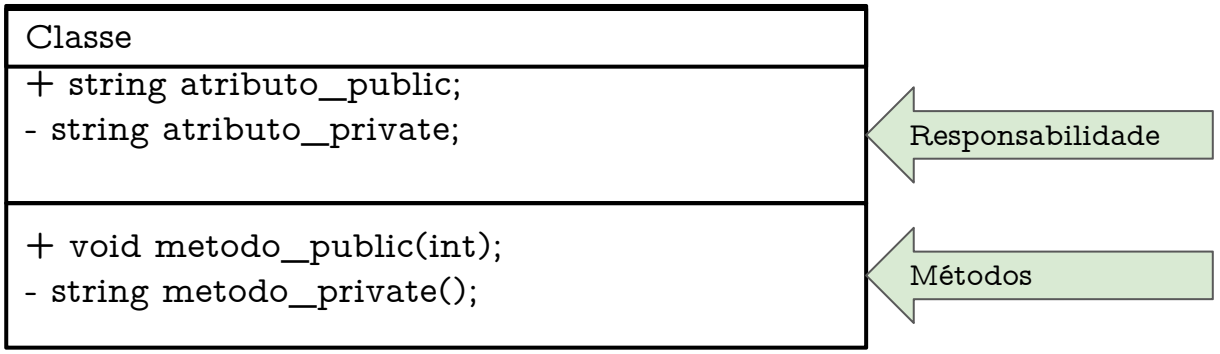
Modelos UML

Modelagem do Mundo Através de Objetos



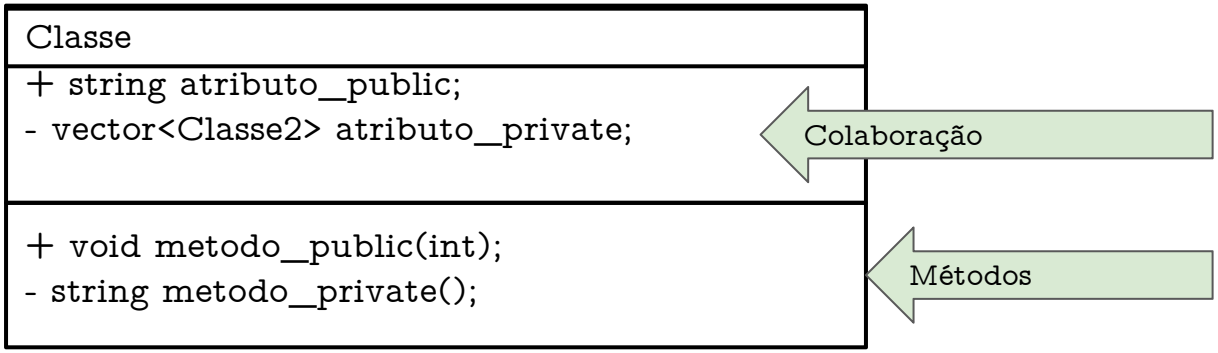
Diagramas de Classe

- Ajudam a definir as responsabilidades e as colaborações



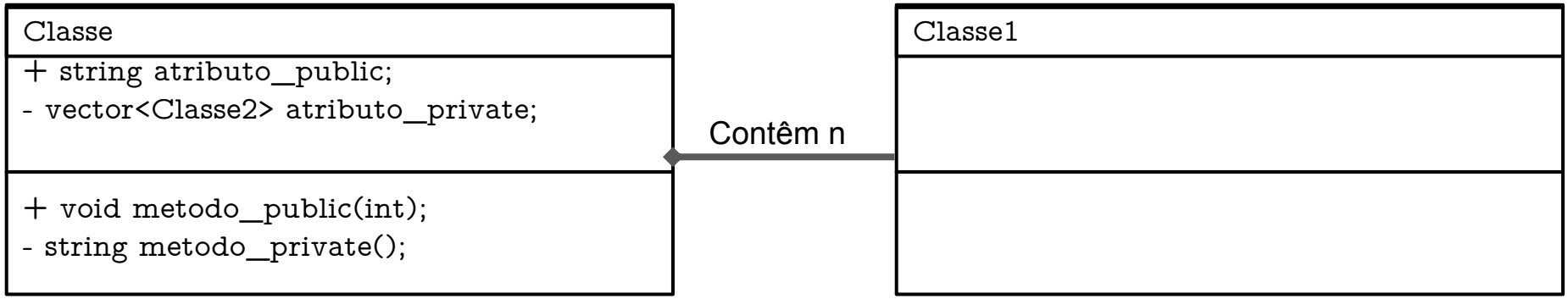
Diagramas de Classe

- Ajudam a definir as responsabilidades e as colaborações



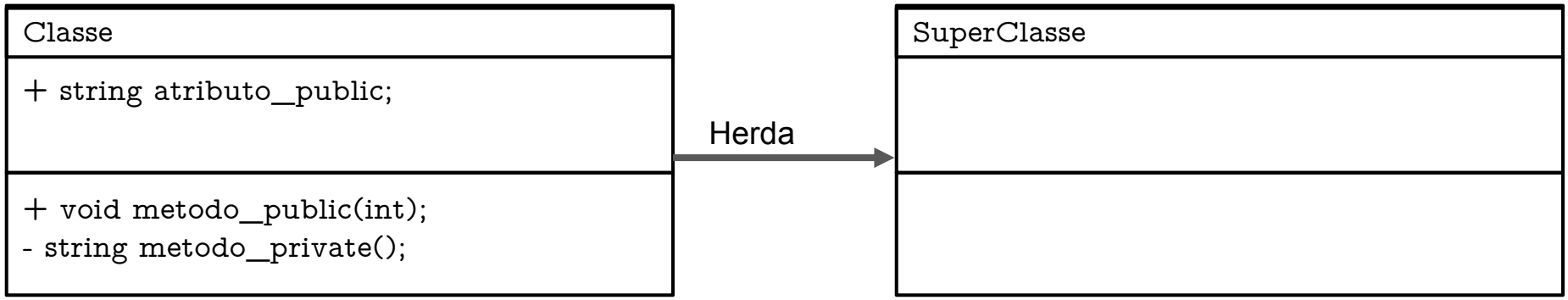
Diagramas de Classe

- Ajudam a definir as responsabilidades e as colaborações



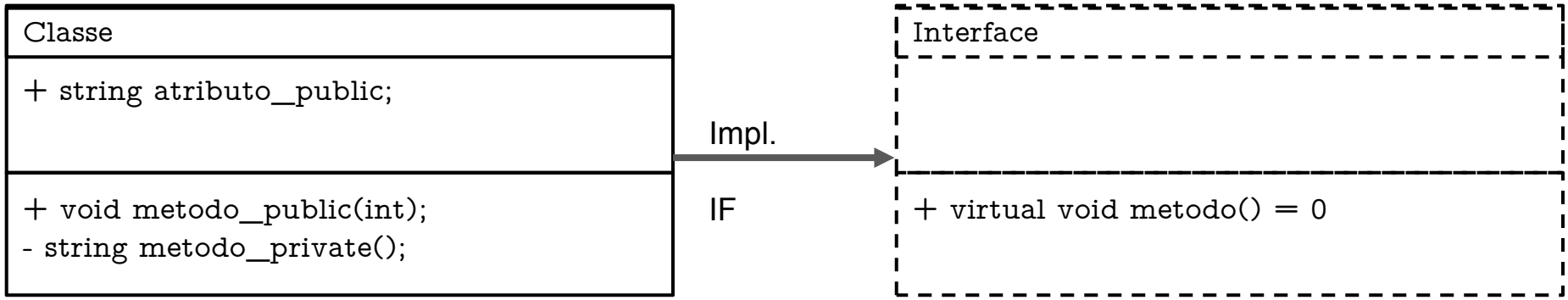
Diagramas de Classe

- Ajudam a definir as responsabilidades e as colaborações



Diagramas de Classe

- Ajudam a definir as responsabilidades e as colaborações



Baixo Acoplamento

Minimizar dependências

- Como minimizar as dependências e, ao mesmo tempo, maximizar o reuso?

Definição

- Como minimizar as dependências e, ao mesmo tempo, maximizar o reuso?
- Acoplamento mensura:
  - Quão conectada é uma classe
  - Quanto ela possui conhecimento de outra

Forte Acoplamento

1. Mudança às classes correlatas
   a. Impactam na classe acoplada
2. Difícil entender o comportamento
   a. Comportamento depende de outras classes
3. Difícil re-utilizar
   a. Muita bagagem

Mensurando Acoplamento

- X tem um atributo que referencia uma instância de Y
- X tem um método que referencia uma instância de Y
  - Pode ser parâmetro, variável local, objeto retornado pelo método
- X é uma subclasse direta ou indireta de Y
- X implementa a interface Y

Acoplamento

- Tem que existir
- Porém queremos minimizar o mesmo

Problema

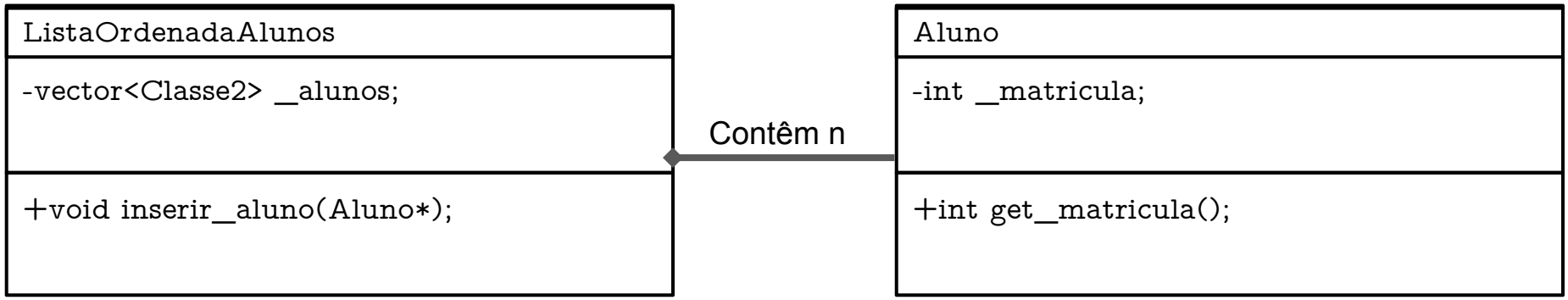
- Lista de Alunos Ordenados por Matrícula

Solução 0
Quais são os problemas?

```
#ifndef LISTA_ALUNO_PDS2_H
#define LISTA_ALUNO_PDS2_H

#include <vector>
#include "aluno.h"


class ListaOrdenadaAlunos {
private:
  std::vector<const Aluno *> _alunos;
public:
  void inserir_aluno(Aluno const &aluno);
};

#endif
```

Solução 0
Em UML

- Parece que temos um acoplamento baixo



Solução 0
Quais são os problemas?

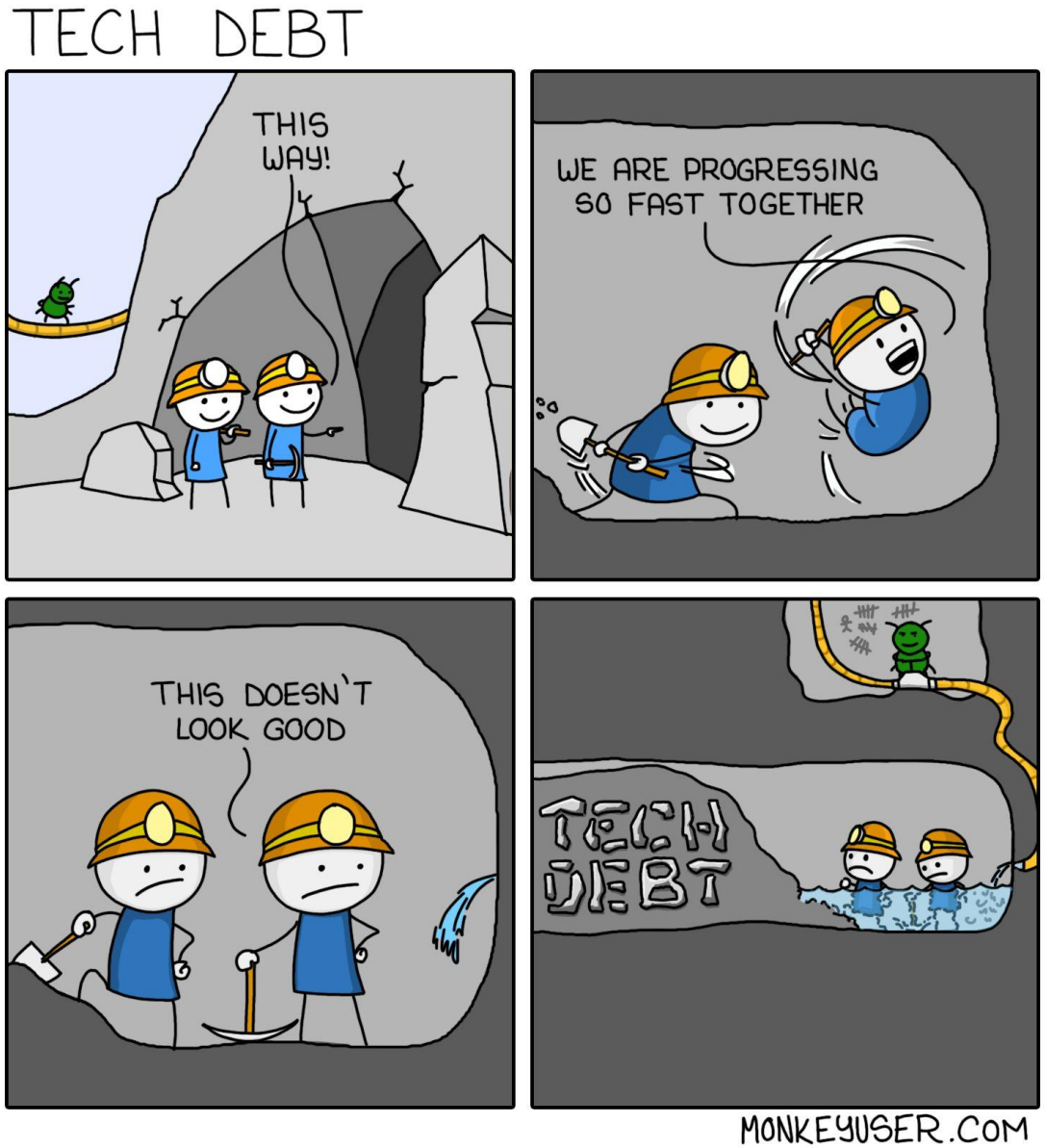
- A lista funciona para um tipo apenas
  - Alto acoplamento
- Nem sempre é sobre o agora
  - Pensar no futuro

```
#include "listaaluno.h"

void ListaOrdenadaAlunos::inserir_aluno(Aluno const &aluno) {
  auto it = this->_alunos.begin();
  auto ed = this->_alunos.end();
  while (it != ed && (*it)->get_matricula() < aluno.get_matricula()) {
            it++;
  }
  this->_alunos.insert(it, &aluno);
}
```

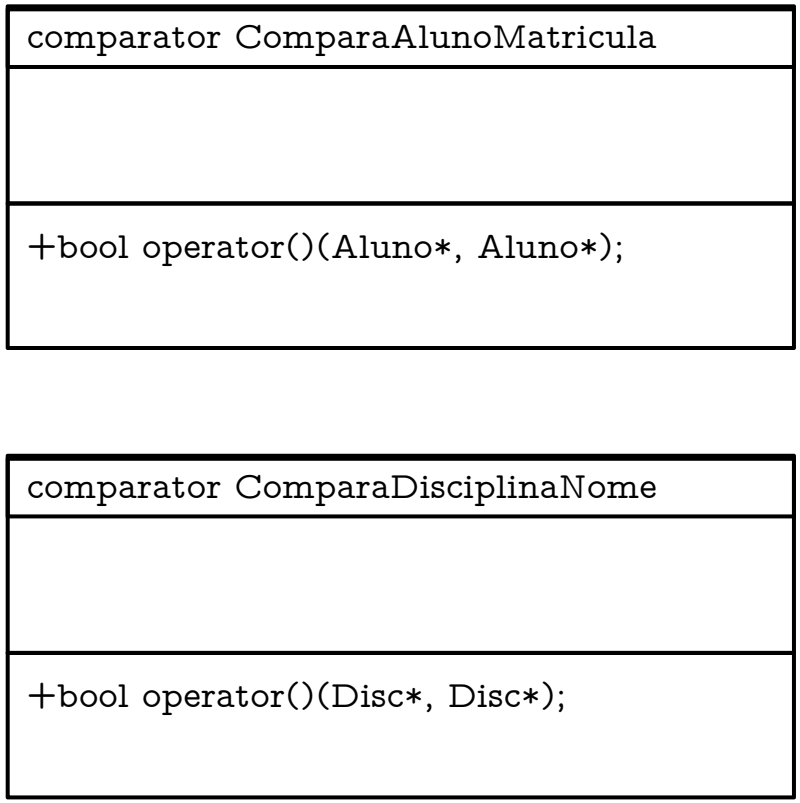
Pensando no futuro
Em UML

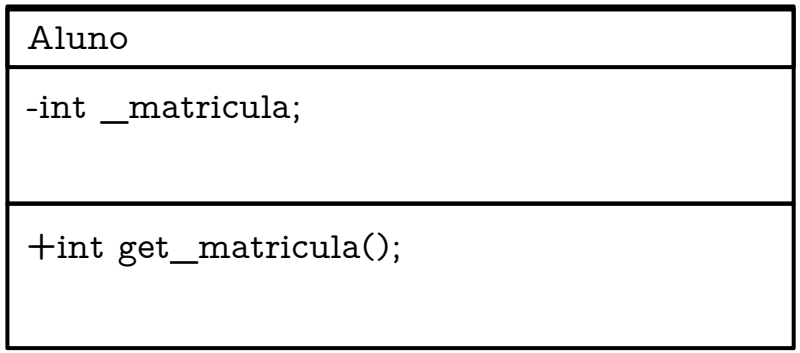
- Uma lista ordenada de disciplinas?!
- Ordenar Alunos pelo nome?
- Ordenar Disciplinas pelo nome?
- Temos um tipo muito específico
  - Acoplamento nem sempre é contar arestas em um código UML

Technical Debt



Solução Boa


comparator ComparaAlunoMatricula
+bool operator()(Aluno*, Aluno*);
comparator ComparaDisciplinaNome
+bool operator()(Disc*, Disc*);


Aluno
-int _matricula;
+int get_matricula();

Usando o Comparator

- Podemos reutilizar
- Basta combinar com um set
  - Elementos ordenados
- Podemos usar em outros contextos

```
struct aluno_comparator_f {
  bool operator()(const Aluno &aluno1, const Aluno &aluno2) const {
    return aluno1.get_matricula() < aluno2.get_matricula();
  }
};
```

Alta Coesão

Coesão

- A coesão mede quão relacionados ou focados estão as responsabilidades da classe
- Uma maior coesão:
  - Classes com propósitos bem definidos
- Com baixa coesão:
  - God Classes

Software de Álgebra Linear

- Precisamos representar:
  - Vetores
  - Vetores esparsos
  - Matrizes
  - Matrizes esparsas
- Cada um com operações
- Salvar dados no disco

Classe Muito Grande

DataClass
-double **_dados; -int _ndim; -bool _esparsa;
+double media(); +double mediana(); +double desvio_padrao(); +double produto_matricial(DataClass); +double produto_interno(DataClass); +double produto_matricial_sparse(DataClass); +double produto_interno_sparse(DataClass); +DataClass ordenar(DataClass); +void salvar_disco(std::string arquivo); +void carregar_dados_do_disco(std::string arquivo); . . .

Nova Solução
Classes com responsabilidades bem definidas

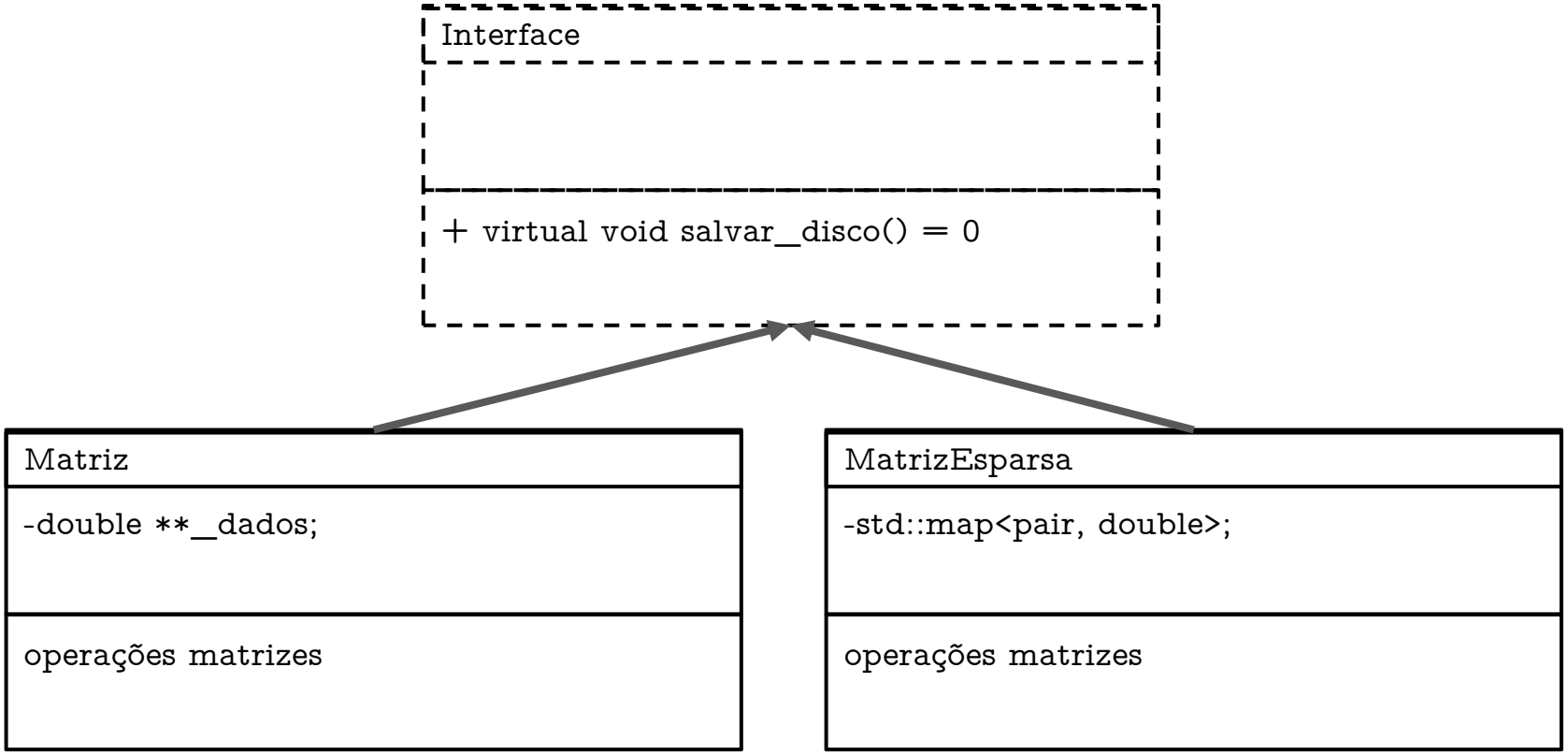
Vetor
-double *_dados;
operações vetores

VetorEsparso
-std::map<int, double>;
operações vetor esparso

Matriz
-double **_dados;
operações matrizes

MatrizEsparsa
-std::map<pair, double>;
operações matrizes

Interface Comum
Define operações como salvar no disco

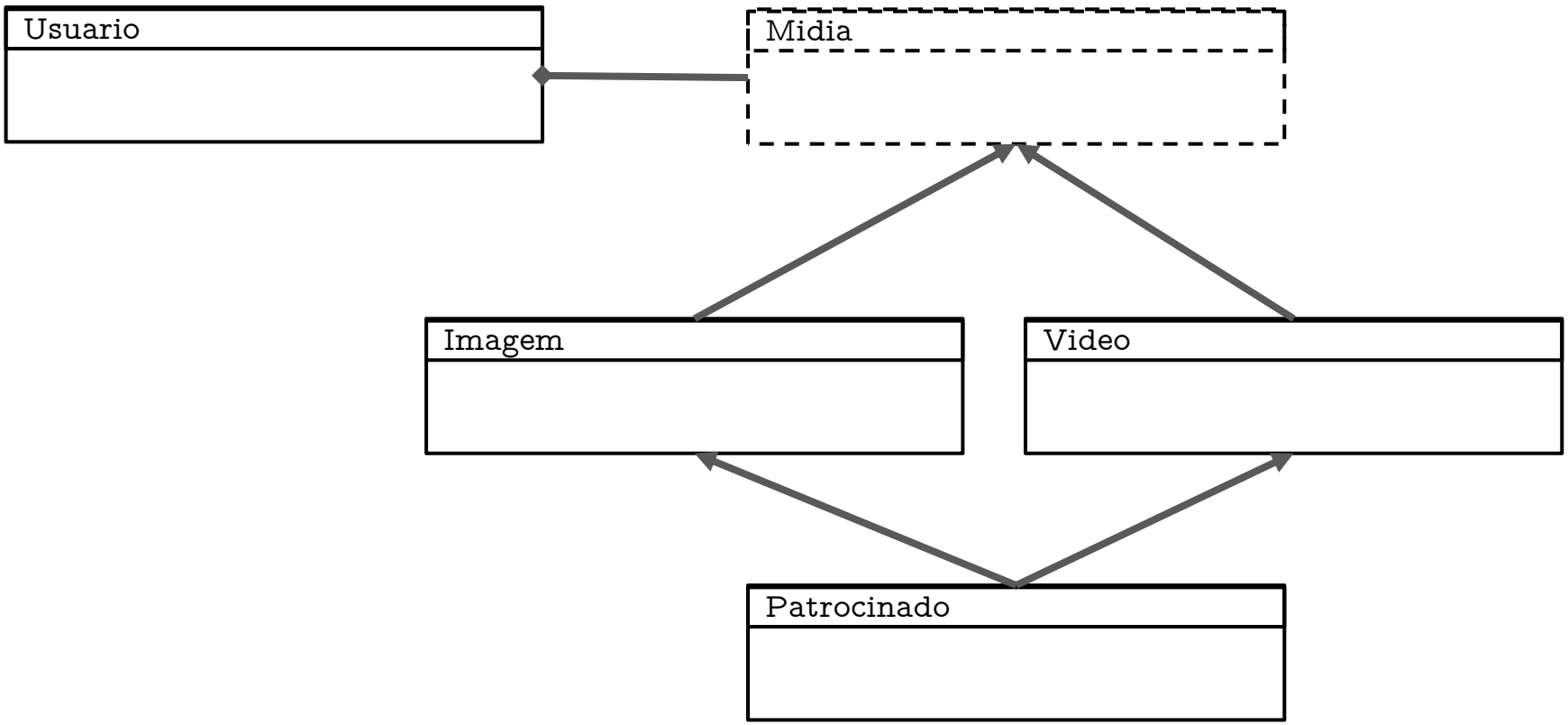


Instagram

Problema

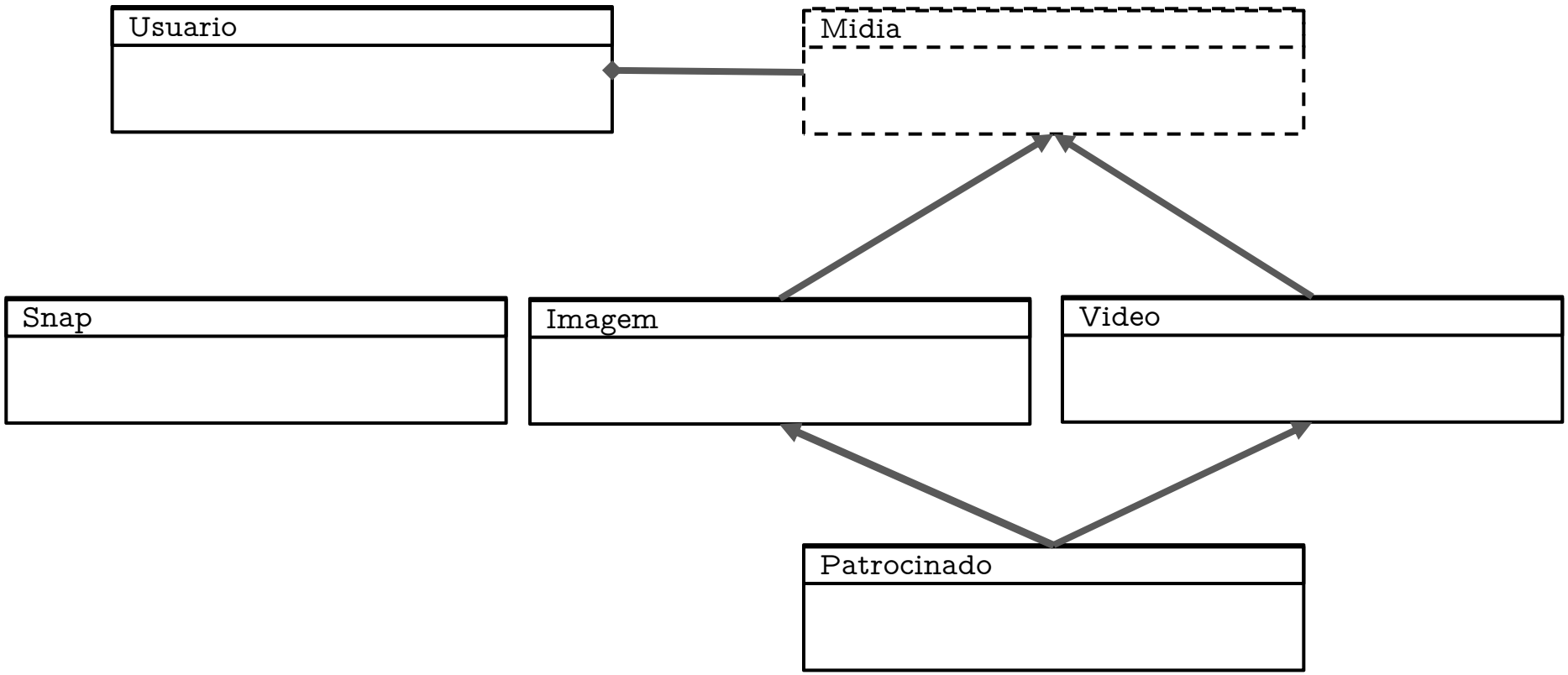
- Usuários representados como conjuntos de mídias
- Diferentes tipos de mídias
  - Imagens
  - Vídeos
- Podemos patrocinar uma mídia

Solução com Herança Múltipla
Qual o problema da solução?



Solução com Herança Múltipla

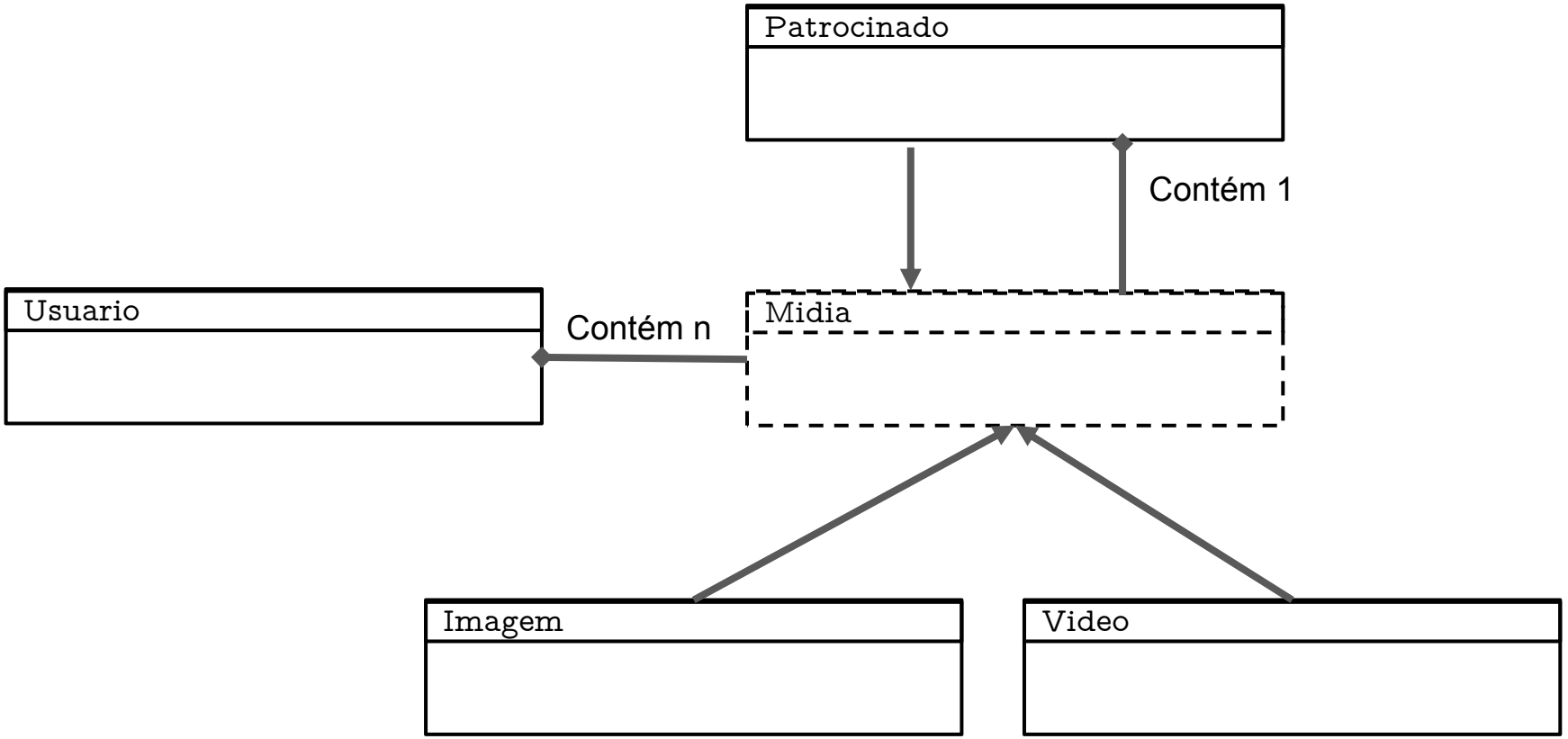
Caso exista uma nova mídia, Snap, como patrocinar?



Herança
Problemas

- Já mencionamos os problemas de herança
- No contexto da aula de hoje:
  - Causa um forte acoplamento
    - Do mesmo tipo
  - Baixa a coesão
    - Ao herdar várias classes
    - Ou uma árvore profunda de herança

Uma boa solução
Composição + Interfaces (Decorator)



User Stories
Instagram

- Postar Mídia
  - Who: User
  - What: Mídia
  - Why? Adicionar na coleção
- Patrocinar Mídia
  - Who: User
  - What: Mídia
  - Why? Alcançar + views

Uno

Jogo de Uno / 8 maluco / Mau Mau

- Cada Jogador tem recebe 7 cartas
- O resto do Baralho é oculto

Jogo de Uno / 8 maluco / Mau Mau

- Cada Jogador tem recebe 7 cartas
- O resto do Baralho é oculto
- Quais as classes até agora?

Jogo de Uno / 8 maluco / Mau Mau

- Cada **Jogador** tem recebe 7 **Cartas**
- O resto do **Baralho** é oculto

Jogo de Uno / 8 maluco / Mau Mau

- Cada **Jogador** tem recebe 7 **Cartas**
- O resto do **Baralho** é oculto
- Um baralho é composto de?

Jogo de Uno / 8 maluco / Mau Mau

- Cada **Jogador** tem recebe 7 **Cartas**
- O resto do **Baralho** é oculto
- Um baralho é composto de?
  - Coleção de **Cartas**

Jogo de Uno / 8 maluco / Mau Mau

- Cada **Jogador** tem recebe 7 **Cartas**
- O resto do **Baralho** é oculto
- Um baralho é composto de?
  - Coleção de **Cartas**
- Uno é um Jogo interessante.
  - Inicia no sentido horário, pode mudar

Jogo de Uno / 8 maluco / Mau Mau

- Cada **Jogador** tem recebe 7 **Cartas**
- O resto do **Baralho** é oculto
- Um baralho é composto de?
  - Coleção de **Cartas**
- Uno é um Jogo interessante.
  - Inicia no sentido horário, pode mudar
  - Isto é? Mantém um _____

Jogo de Uno / 8 maluco / Mau Mau

- Cada **Jogador** tem recebe 7 **Cartas**
- O resto do **Baralho** é oculto
- Um baralho é composto de?
  - Coleção de **Cartas**
- Uno é um Jogo interessante.
  - Inicia no sentido horário, pode mudar
  - Isto é? Mantém um **estado**

Jogo de Uno / 8 maluco / Mau Mau

- Cada **Jogador** tem recebe 7 **Cartas**
- O resto do **Baralho** é oculto
- Um baralho é composto de?
  - Coleção de **Cartas**
- Uno é um **Jogo** interessante.
  - Inicia no sentido horário, pode mudar
  - Isto é? Mantém um **estado**
  - Nova classe, atributo **sentido**

Jogo de Uno / 8 maluco / Mau Mau

- Ao modelar o mundo real:
  - **Definir objetos**
  - **Definir responsabilidades**
  - **Definir iterações**

Cartas

- Cada carta tem uma cor e um número
- Existem cartas especiais

Cartas

- Cada carta tem uma cor e um número
- Existem cartas especiais
  - Bom local para fazer uso de?
  - **Polimorfismo**
- Cartas especiais podem:
  - Alterar o sentido do jogo
  - Pular jogadores
  - Ser jogada em qualquer momento
  - Aumentar número de cartas do adversário

Jogadores

- Tem uma pontuação
- 7 cartas iniciais.
- **Porém**
  - Pode aumentar, com uma carta especial de um adversário
- **Vector/Set**

User Stories

- Iniciar Jogo
- Realizar Jogada
- Fechar programa
  - Desistir
- Salvar jogo
  - Continuar no futuro