Associações e *Forward Declarations*

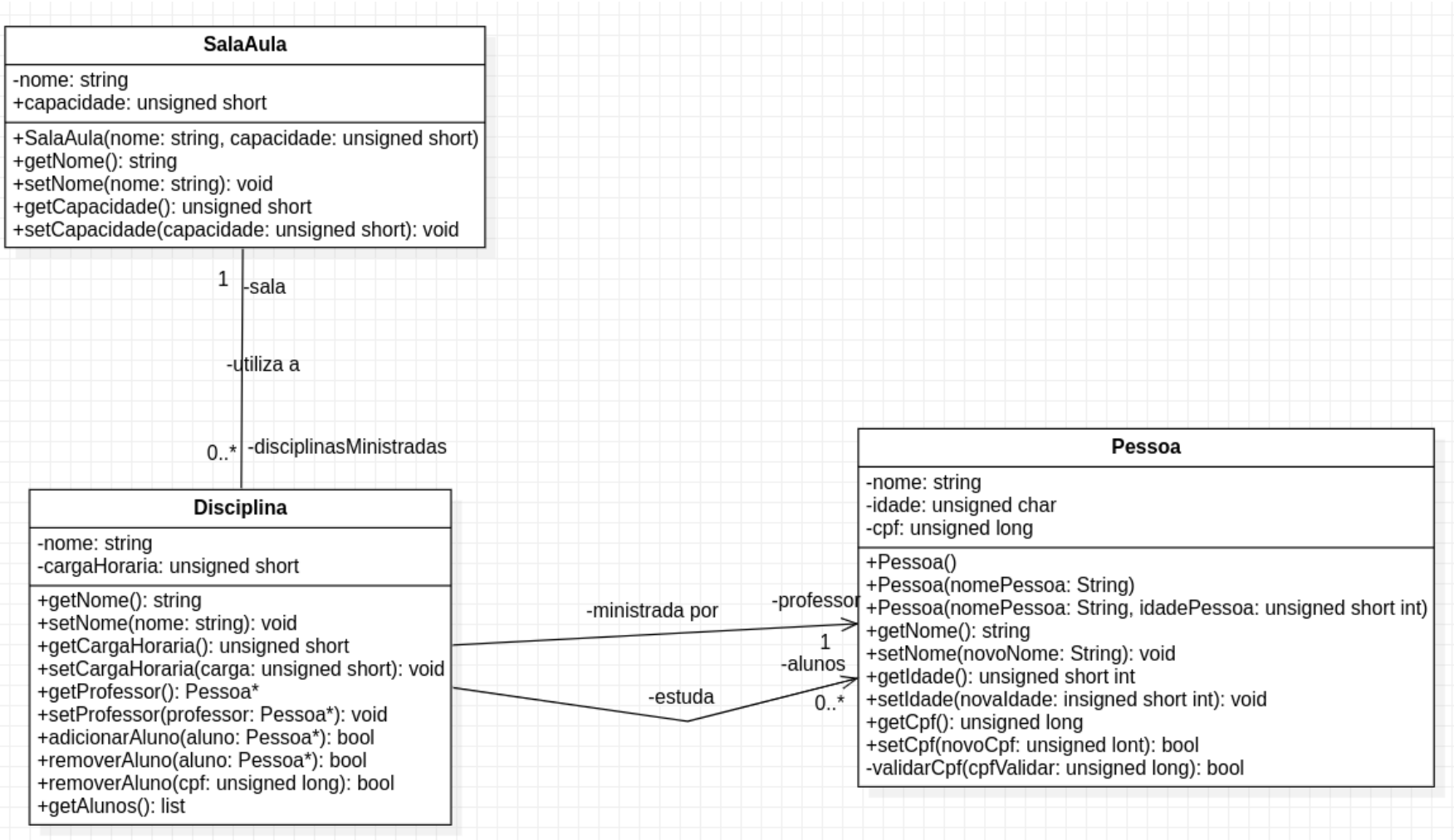
Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

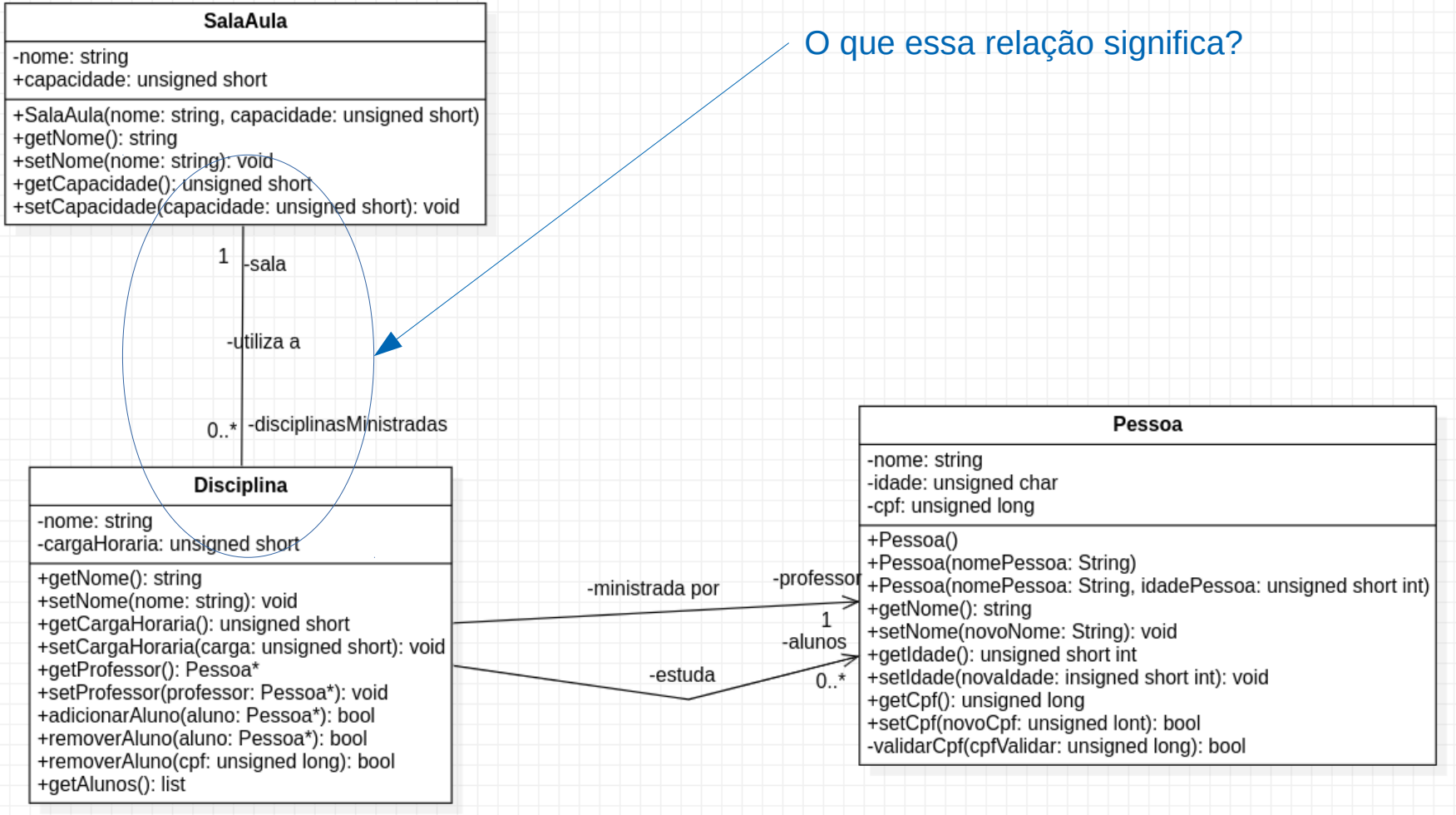
Antes de começar

- Faça o donwload do material disponibilizado no site
  - “*Programas Antes Aula*”
  - Entenda a classe SalaAula criada

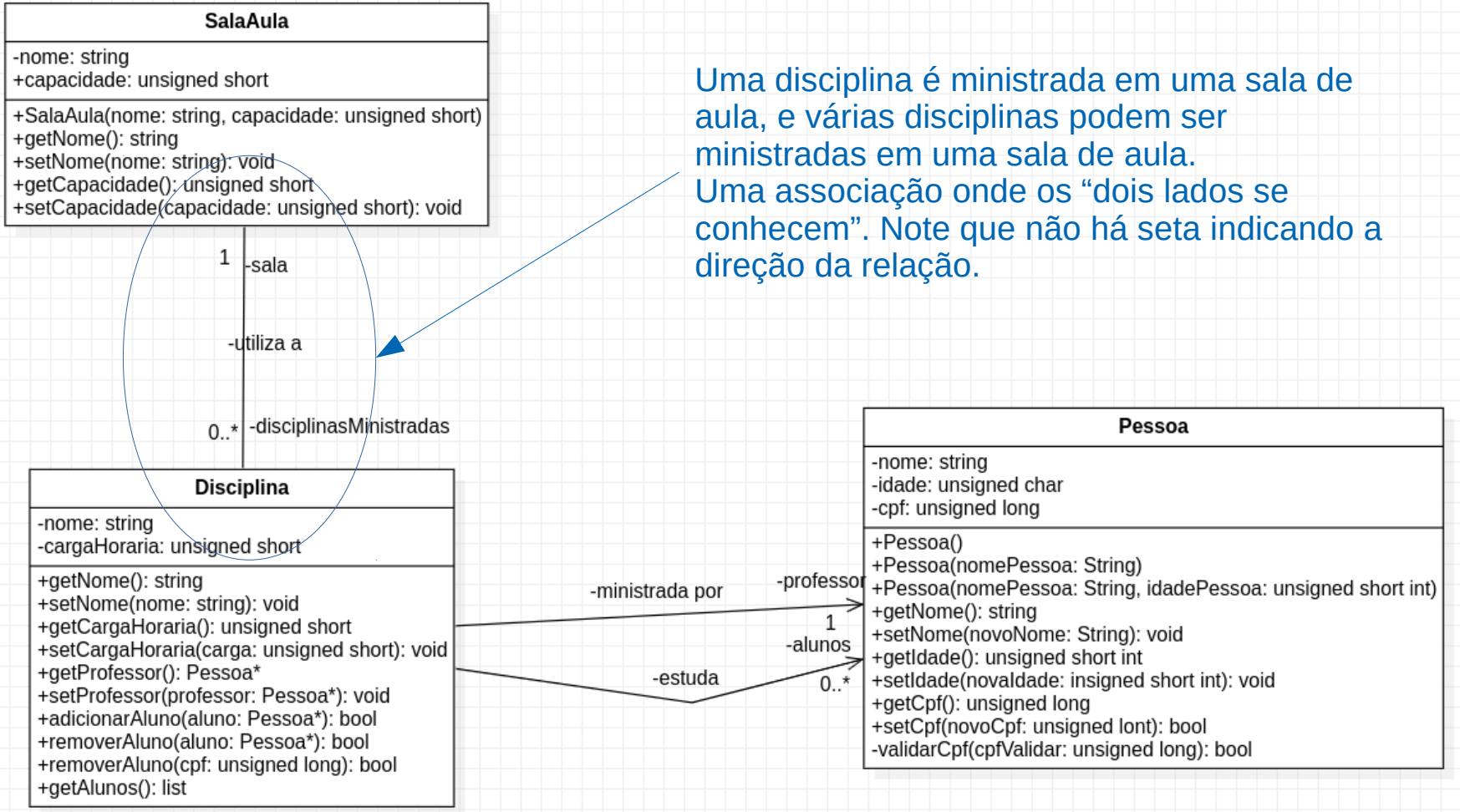
Modelagem


SalaAula
-nome: string
+capacidade: unsigned short
+SalaAula(nome: string, capacidade: unsigned short)
+getNome(): string
+setNome(nome: string): void
+getCapacidade(): unsigned short
+setCapacidade(capacidade: unsigned short): void
1
-sala
-utiliza a
0..*
-disciplinasMinistradas
Disciplina
-nome: string
-cargaHoraria: unsigned short
+getNome(): string
+setNome(nome: string): void
+getCargaHoraria(): unsigned short
+setCargaHoraria(carga: unsigned short): void
+getProfessor(): Pessoa*
+setProfessor(professor: Pessoa*): void
+adicionarAluno(aluno: Pessoa*): bool
+removerAluno(aluno: Pessoa*): bool
+removerAluno(cpf: unsigned long): bool
+getAlunos(): list
-ministrada por
-professor
1
-alunos
0..*
-estuda
Pessoa
-nome: string
-idade: unsigned char
-cpf: unsigned long
+Pessoa()
+Pessoa(nomePessoa: String)
+Pessoa(nomePessoa: String, idadePessoa: unsigned short int)
+getNome(): string
+setNome(novoNome: String): void
+getIdade(): unsigned short int
+setIdade(novaIdade: insigned short int): void
+getCpf(): unsigned long
+setCpf(novoCpf: unsigned lont): bool
-validarCpf(cpfValidar: unsigned long): bool

Modelagem

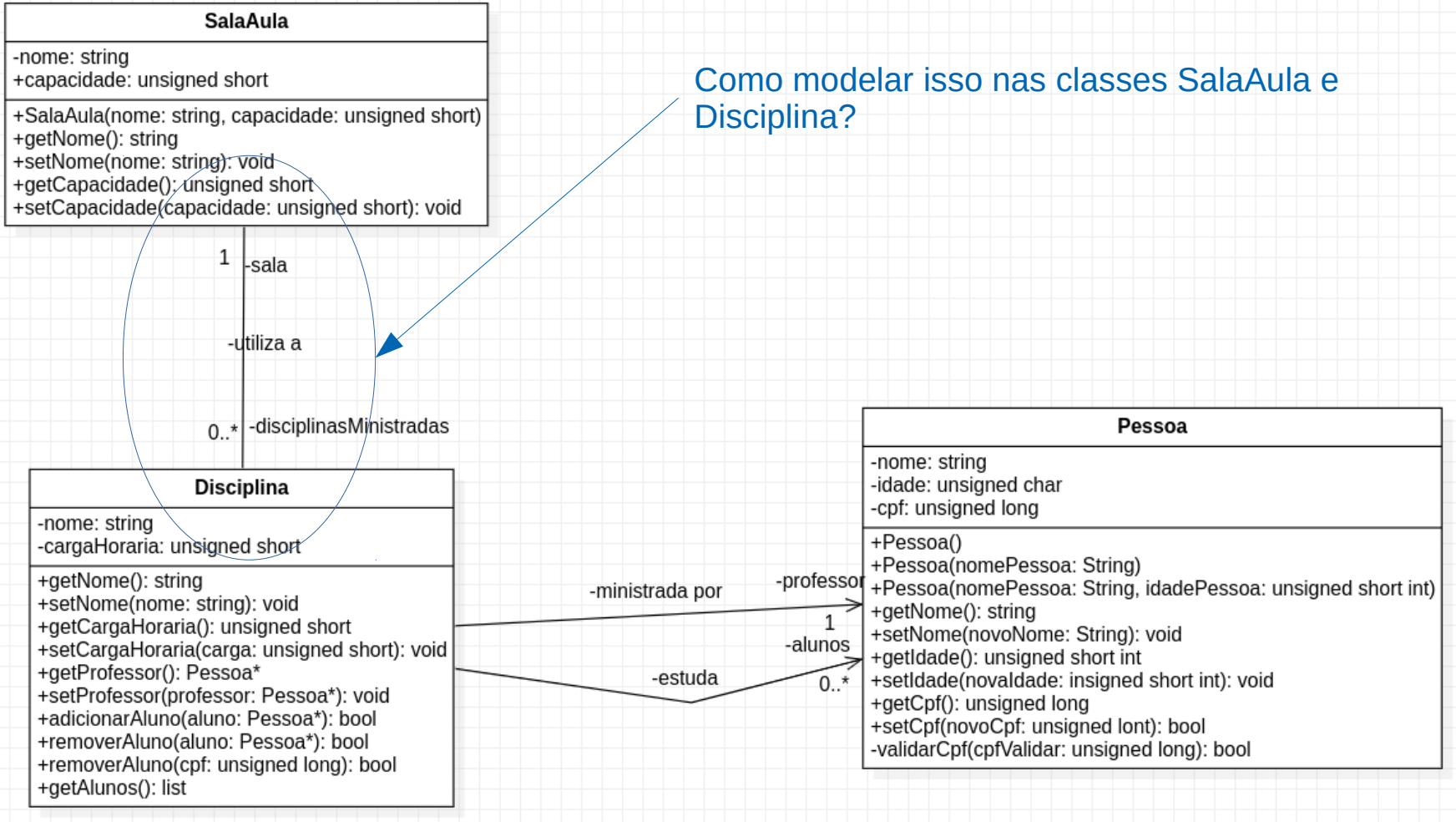

SalaAula
-nome: string
+capacidade: unsigned short
+SalaAula(nome: string, capacidade: unsigned short)
+getNome(): string
+setNome(nome: string): void
+getCapacidade(): unsigned short
+setCapacidade(capacidade: unsigned short): void
1
-sala
-utiliza a
0..*
-disciplinasMinistradas
O que essa relação significa?
Disciplina
-nome: string
-cargaHoraria: unsigned short
+getNome(): string
+setNome(nome: string): void
+getCargaHoraria(): unsigned short
+setCargaHoraria(carga: unsigned short): void
+getProfessor(): Pessoa*
+setProfessor(professor: Pessoa*): void
+adicionarAluno(aluno: Pessoa*): bool
+removerAluno(aluno: Pessoa*): bool
+removerAluno(cpf: unsigned long): bool
+getAlunos(): list
-ministrada por
-professor
1
-estuda
-alunos
0..*
Pessoa
-nome: string
-idade: unsigned char
-cpf: unsigned long
+Pessoa()
+Pessoa(nomePessoa: String)
+Pessoa(nomePessoa: String, idadePessoa: unsigned short int)
+getNome(): string
+setNome(novoNome: String): void
+getIdade(): unsigned short int
+setIdade(novaIdade: insigned short int): void
+getCpf(): unsigned long
+setCpf(novoCpf: unsigned lont): bool
-validarCpf(cpfValidar: unsigned long): bool

Modelagem

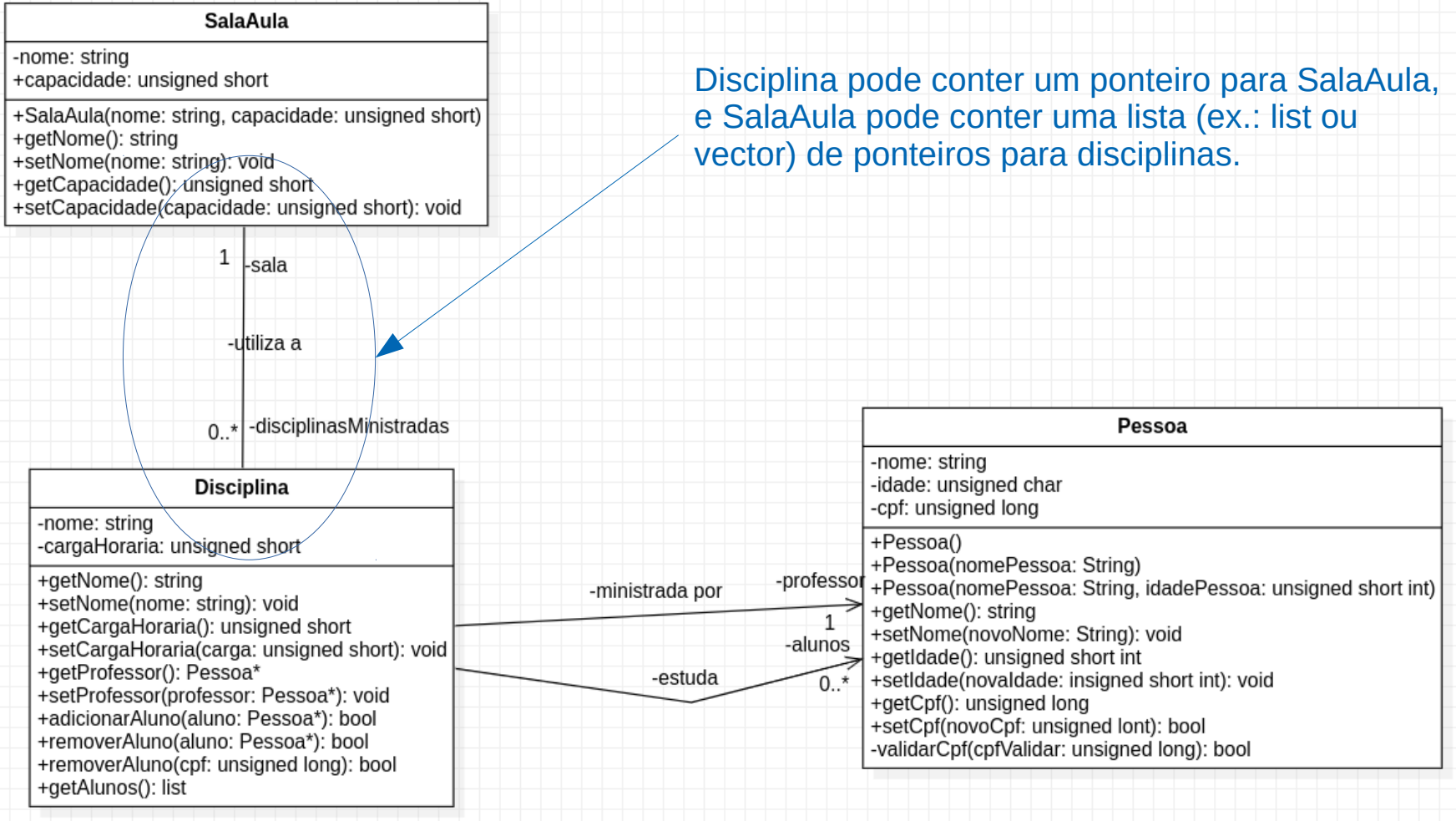


Uma disciplina é ministrada em uma sala de aula, e várias disciplinas podem ser ministradas em uma sala de aula.
Uma associação onde os “dois lados se conhecem”. Note que não há seta indicando a direção da relação.

Modelagem


SalaAula
-nome: string
+capacidade: unsigned short
+SalaAula(nome: string, capacidade: unsigned short)
+getNome(): string
+setNome(nome: string): void
+getCapacidade(): unsigned short
+setCapacidade(capacidade: unsigned short): void
1
-sala
-utiliza a
0..*
-disciplinasMinistradas
Como modelar isso nas classes SalaAula e Disciplina?
Disciplina
-nome: string
-cargaHoraria: unsigned short
+getNome(): string
+setNome(nome: string): void
+getCargaHoraria(): unsigned short
+setCargaHoraria(carga: unsigned short): void
+getProfessor(): Pessoa*
+setProfessor(professor: Pessoa*): void
+adicionarAluno(aluno: Pessoa*): bool
+removerAluno(aluno: Pessoa*): bool
+removerAluno(cpf: unsigned long): bool
+getAlunos(): list
-ministrada por
-professor
1
-alunos
0..*
-estuda
Pessoa
-nome: string
-idade: unsigned char
-cpf: unsigned long
+Pessoa()
+Pessoa(nomePessoa: String)
+Pessoa(nomePessoa: String, idadePessoa: unsigned short int)
+getNome(): string
+setNome(novoNome: String): void
+getIdade(): unsigned short int
+setIdade(novaIdade: insigned short int): void
+getCpf(): unsigned long
+setCpf(novoCpf: unsigned lont): bool
-validarCpf(cpfValidar: unsigned long): bool

Modelagem


SalaAula
-nome: string
+capacidade: unsigned short
+SalaAula(nome: string, capacidade: unsigned short)
+getNome(): string
+setNome(nome: string): void
+getCapacidade(): unsigned short
+setCapacidade(capacidade: unsigned short): void
1
-sala
-utiliza a
0..*
-disciplinasMinistradas
Disciplina pode conter um ponteiro para SalaAula, e SalaAula pode conter uma lista (ex.: list ou vector) de ponteiros para disciplinas.
Disciplina
-nome: string
-cargaHoraria: unsigned short
+getNome(): string
+setNome(nome: string): void
+getCargaHoraria(): unsigned short
+setCargaHoraria(carga: unsigned short): void
+getProfessor(): Pessoa*
+setProfessor(professor: Pessoa*): void
+adicionarAluno(aluno: Pessoa*): bool
+removerAluno(aluno: Pessoa*): bool
+removerAluno(cpf: unsigned long): bool
+getAlunos(): list
-ministrada por
-professor
1
-estuda
-alunos
0..*
Pessoa
-nome: string
-idade: unsigned char
-cpf: unsigned long
+Pessoa()
+Pessoa(nomePessoa: String)
+Pessoa(nomePessoa: String, idadePessoa: unsigned short int)
+getNome(): string
+setNome(novoNome: String): void
+getIdade(): unsigned short int
+setIdade(novaIdade: insigned short int): void
+getCpf(): unsigned long
+setCpf(novoCpf: unsigned lont): bool
-validarCpf(cpfValidar: unsigned long): bool

Disciplina

Disciplina.hpp

```
//...

#include "Pessoa.hpp"
#include "SalaAula.hpp"

class Disciplina{
	public:
		//...

		void setSalaAula(SalaAula* salaAula);
		SalaAula* getSalaAula();
	private:
		std::string nome;
		unsigned short int cargaHoraria;


		Pessoa* professor;
		std::list<Pessoa*> alunos;
		SalaAula* salaAula;
};
```

Disciplina.cpp

```
//...

void Disciplina::setSalaAula(SalaAula* salaAula){
	this->salaAula = salaAula;
}


SalaAula* Disciplina::getSalaAula(){
	return salaAula;
}
```

UDESC

SalaAula

SalaAula.hpp

```
//...

#include "Disciplina.hpp"

class SalaAula{
        public:

                //...
                void adicionarDisciplina(Disciplina* disciplina);
                void removerDisciplina(Disciplina* disciplina);
                std::list<Disciplina*>& getDisciplinas();
        private:
                std::string nome;
                unsigned int capacidade;
                std::list<Disciplina*> disciplinasMinistradas;
};
#endif
```

SalaAula.cpp

```
//...

void SalaAula::adicionarDisciplina(Disciplina* disciplina){
        disciplinasMinistradas.push_back(disciplina);
}

void SalaAula::removerDisciplina(Disciplina* disciplina){
        disciplinasMinistradas.remove(disciplina);
}

std::list<Disciplina*>& SalaAula::getDisciplinas(){
        return disciplinasMinistradas;
}
```

UDESC

Teste você mesmo

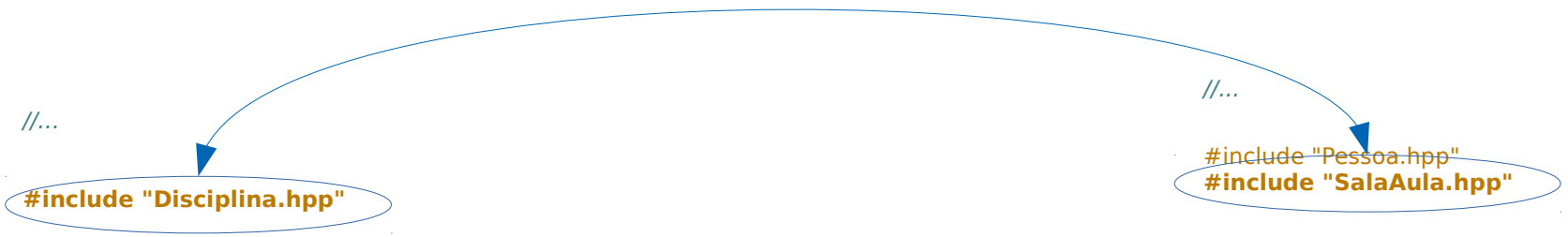
- make clean
- make
- O que acontece?

Teste você mesmo

- make clean
- make
- O que acontece?
  - Desastre!
  - Vários erros estranhos surgem no log de compilação
  - O que cargas d’água pode estar errado?

O problema!

Para compilar Disciplina, o compilador precisa compilar SalaAula, e para compilar SalaAula, o compilador precisa compilar Disciplina!!!



```
//...

#include "Disciplina.hpp"

class SalaAula{
        public:

                //...
                void adicionarDisciplina(Disciplina* disciplina);
                void removerDisciplina(Disciplina* disciplina);
                std::list<Disciplina*>& getDisciplinas();
        private:
                std::string nome;
                unsigned int capacidade;
                std::list<Disciplina*> disciplinasMinistradas;
};
#endif
```

```
//...

#include "Pessoa.hpp"
#include "SalaAula.hpp"

class Disciplina{
        public:
                //...

                void setSalaAula(SalaAula* salaAula);
                SalaAula* getSalaAula();
        private:
                std::string nome;
                unsigned short int cargaHoraria;


                Pessoa* professor;
                std::list<Pessoa*> alunos;
                SalaAula* salaAula;
};
```

UDESC

Forward Declaration

- O Forward Declaration de uma classe tem o formato:
  - *class NomeClasse;*
  - Indica para o compilador que a classe existe, mas que ainda não foi compilada
    - Fazemos uma “promessa” ao compilador, dizendo que vamos mostrar onde compilar essa classe assim que pudermos

Forward Declaration

- Em Disciplina.hpp Substitua

  #include "Disciplina.hpp"

- Por

  class SalaAula;//Forward Declaration

Disciplina.hpp

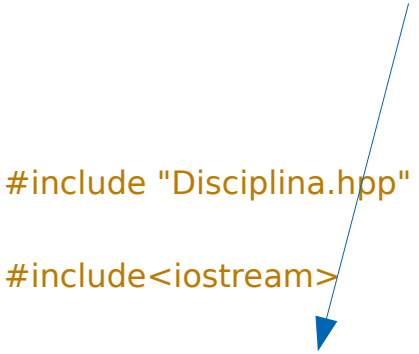
```
#ifndef DISCIPLINA_H
#define DISCIPLINA_H

#include <string>
#include <list>

#include "Pessoa.hpp"
//#include "SalaAula.hpp"
class SalaAula;//forward declaration
//o include de SalaAula deve ir agora para o cpp

class Disciplina{
        public:
                Disciplina(std::string nome);


                std::string getNome();
                void setNome(std::string nome);

                //...
};
#endif
```

UDESC

Forward Declaration

- Apenas informamos ao compilador que existe uma classe SalaAula
- Mas não a incluímos em Disciplina
- Com o *forward declaration*, o include da classe SalaAula **deve** ficar no **.cpp** de Disciplina

Somos obrigados incluir no .cpp devido ao forward declaration



```
#include "Disciplina.hpp"

#include<iostream>

#include "SalaAula.hpp"

Disciplina::Disciplina(std::string nome)
        :nome{nome} {
}

void Disciplina::adicionarAluno(Pessoa* aluno){
        this->alunos.push_back(aluno);
}

void Disciplina::removerAluno(Pessoa* aluno){
        this->alunos.remove(aluno);
}

//...
```

Forward Declaration

- Use com **parcimônia**
  - *Forward declarations* dificultam o trabalho do compilador
    - E dificultam o seu também, já que em caso de erro o compilador pode te dar um aviso maluco!
    - A compilação se torna mais complexa e lenta
  - Em casos extremos, foward declarations incorretos podem gerar linkedições incorretas!
    - O linkeditor pode utilizar uma função membro de uma classe que não era a que você tinha em mente!
- Resumo
  - **Use somente quando estritamente necessário!**

Exercícios

- Caso o programador modifique a sala de aula da *Disciplina* via *setSalaAula*, e esqueça de adicionar a disciplina na *SalaAula* via *adicionarDisciplina*, os objetos ficam inconsistentes.
  - Resolva esse problema, para que os objetos da classe Disciplina comuniquem as suas alterações para objetos SalaAula.
    - A *Disciplina* é incluída automaticamente na *SalaAula*
  - Note que os objetos ainda poderão ficar inconsistentes se o programador utilizar o *adicionarDisciplina()* de disciplina
    - Estudaremos maneiras de mitigar o problema na próxima aula
    - **Opcional**
      - Tente fazer com que ao adicionar a *Disciplina* em *SalaAula* via *adicionarDisciplina*, a SalaAula de Disciplina **também** seja atualizada automaticamente

Referências

- DEITEL, P.; DEITEL, H. **C++ how to Program**. [S.l.]: Pearson, 2017. ISBN 9780134448237
- STROUSTRUP, B. **The C++ Programming Language**. Pearson Education, 2013. ISBN 9780133522853.
- https://www.learncpp.com/cpp-tutorial/10-4-association/
- Pressman, R.;Maxim, B. **Engenharia de Software: uma abordagem Profissional.** McGraw Hill Brasil, 2016. 8 ed. ISBN 9788580555349.