OO – Engenharia Eletrônica

Orientação a Objetos
-
Programação em C++

Slides 3 - B:

Relacionamento de Herança

Prof. Jean Marcelo SIMÃO – DAELN/UTFPR

Herança

Introdução a Herança

Reflexão - Classe *Pessoa*

Está correto "semanticamente" que os objetos *Simão*, *Einstein* e *Newton* sejam da classe *Pessoa*?

Eles não seriam, na verdade, de uma classe *Professor*?

Entretanto, eles não seriam também *Pessoas*?

Herança

Pelo mecanismo de herança pode-se fazer com que um objeto pertença a duas classes!

Isto é possível definindo que sua classe herde as propriedades de uma classe base....

Exemplo de Herança

```
#ifndef _PESSOA_H_
#define _PESSOA_H_

class Pessoa
{
protected:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;
   char nomeP[ 30 ];
   int id;
public:
   Pessoa ( int diaNa, int mesNa, int anoNa, char* nome = "“ );
   Pessoa ( );
   ~Pessoa ( ) { }
   void Inicializa (int diaNa, int mesNa, int anoNa, char* nome = "" );
   void Calc_Idade (int diaAT, int mesAT, int anoAT );
   int  informaIdade ( );
   void  seId ( int i ) { id = i; }
   int   getId ( ) { return id; }
   void  seNome ( char*  n ) { strcpy(nomeP, n); }
   char* getNome ( ) { return nome; }
};

#endif
```

```
#ifndef _PROFESSOR_H_
#define _PROFESSOR_H_

#include "Pessoa.h"
#include "Universidade.h"

class Professor : public Pessoa
{
private:
  Universidade* pUnivFiliado;
  Departamento* pDptoFiliado;

public:
   Professor( int diaNa, int mesNa, int anoNa, char* nome = "" );
   Professor ( );
  ~Professor ( );

   void setUnivFiliado ( Universidade* pu);
   void setDepartamento(Departamento* pdpto);
   void OndeTrabalho ( );
   void QualDepartamentoTrabalho ( );
};

#endif
```

Exemplo de Herança

```
#ifndef _PESSOA_H_
#define _PESSOA_H_

class Pessoa
{
protected:
   int diaP;
   int mesP;
   int anoP;
   int idadeP;
   char nomeP[ 30 ];
   int id;
public:
   Pessoa ( int diaNa, int mesNa, int anoNa, char* nome = "" );
   Pessoa ( );
   ~Pessoa ( ) { }
   void Inicializa (int diaNa, int mesNa, int anoNa, char* nome = "“ );
   void Calc_Idade (int diaAT, int mesAT, int anoAT );
   int  informaIdade ( );
   void  seId ( int i ) { id = i; }
   int   getId ( ) { return id; }
   void  seNome ( char*  n ) { strcpy(nomeP, n); }
   char* getNome ( ) { return nome; }
};


#endif
```

Qual é a diferença entre um elemento privado e um protegido?

A diferença é que o elemento privado não é herdado, enquanto o protegido é herdado.

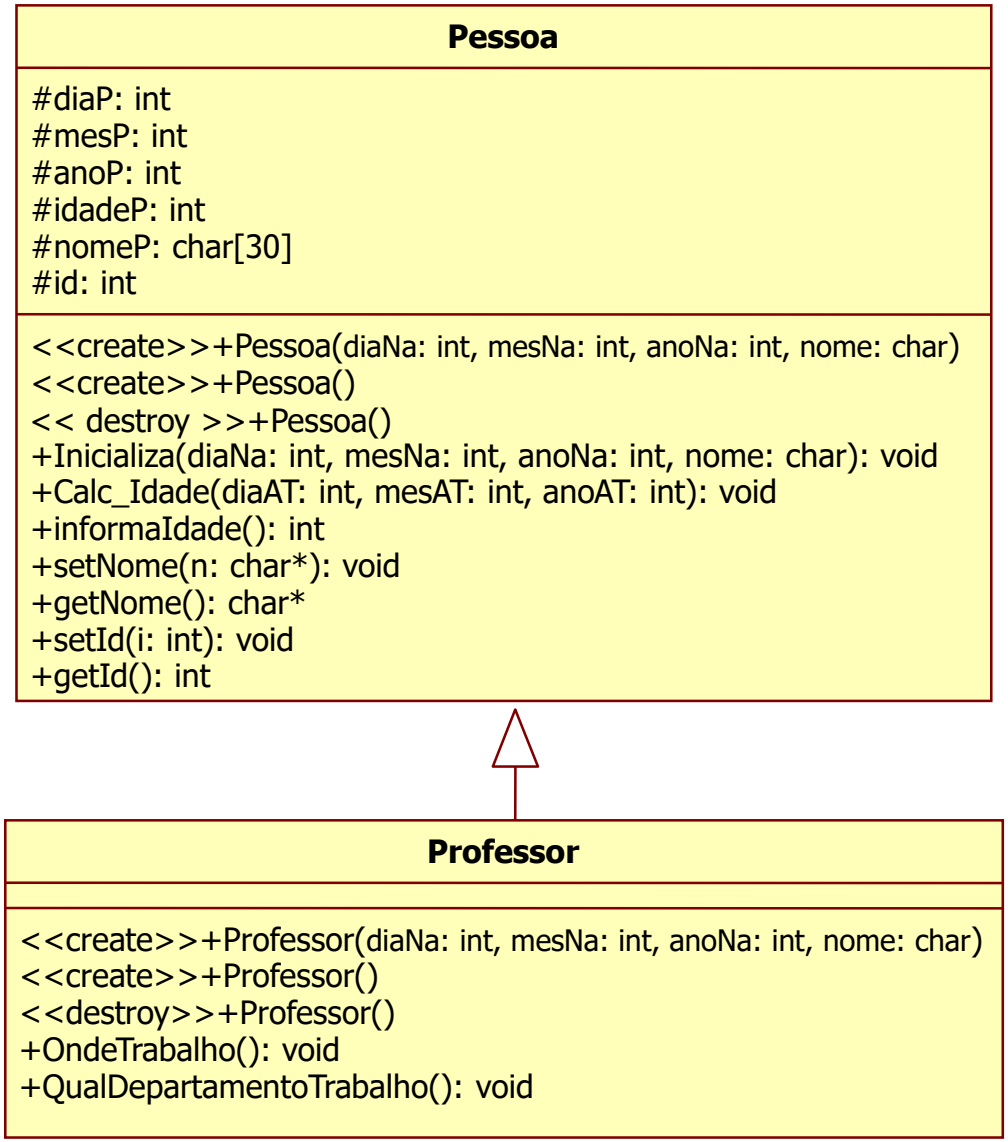
```
#ifndef _PROFESSOR_H_
#define _PROFESSOR_H_

#include "Pessoa.h"
#include "Universidade.h"

class Professor : public Pessoa
{
private:
  Universidade* pUnivFiliado;
  Departamento* pDptoFiliado;

public:
   Professor ( int diaNa, int mesNa, int anoNa, char* nome = "" );
   Professor ( );
  ~Professor ( );

   void setUnivFiliado ( Universidade* pu );
   void setDepartamento ( Departamento* pdpto);
   void OndeTrabalho ( );
   void QualDepartamentoTrabalho ( );
};

#endif
```

```
#include “stdafx.h”
#include "Pessoa.h"

Pessoa::Pessoa ( int diaNa, int mesNa, int anoNa, char* nome )
{
  Inicializa (diaNa, mesNa, anoNa, nome);
}

Pessoa::Pessoa ()
{
  Inicializa ( 0, 0, 0);
}

void Pessoa::Inicializa ( int diaNa, int mesNa, int anoNa, char* nome)
{
  …
}

void Pessoa::Calc_Idade ( int diaAT, int mesAT, int anoAT )
{
   …
}

int Pessoa::informaIdade()
{
  return idadeP;
}

. . .
```

```
#include “stdafx.h”
#include "Professor.h"

Professor::Professor (  int diaNa, int mesNa, int anoNa, char* nome ) :
Pessoa ( diaNa, mesNa, anoNa, nome)
{
  pUnivFiliado = NULL;     pDptoFiliado = NULL;
}

Professor::Professor ( ) :
Pessoa ( )
{
   pUnivFiliado = NULL;    pDptoFiliado = NULL;
}

Professor::~Professor ( )
{
   pUnivFiliado = NULL; ‘   pDptoFiliado = NULL;
}

void Professor::setUnivFiliado ( Universidade* pu )
{
    pUnivFiliado = pu;
}

void Professor::setDepartamento ( Departamento* pdpto)
{
   pDptoFiliado = pdpto;
}

void Professor::OndeTrabalho ( )
{
     cout << nomeP << “trabalha para a“
          << pUnivFiliado->getNome() << endl;
}

void Professor::QualDepartamentoTrabalho()
{
   cout << nomeP << “trabalha para a“
        << pUnivFiliado->getNome()
        << pDptoFiliado->getNome()
        << endl;
}
```

```
#ifndef _PRINCIPAL_H_
#define _PRINCIPAL_H_

#include "Professor.h"

class Principal
{ private:
   . . .

   Universidade UTFPR;
   Universidade Princeton;
   Universidade Cambridge;

   Departamento EletronicaUTFPR;
   Departamento MatematicaUTFPR;
   Departamento FisicaUTFPR;
   Departamento MatematicaPrinceton;
   Departamento FisicaPrinceton;
   Departamento MatematicaCambridge;
   Departamento FisicaCambridge;

   Professor Simao;
   Professor Einstein;
   Professor Newton;

   int          diaAtual, mesAtual, anoAtual;

 public:
   Principal ( );
   void Executar ();
};
#endif
```

Herança em UML



Obs.: Este é um diagrama de classes que pode ser considerado de análise

Diagrama de Classes - Análise

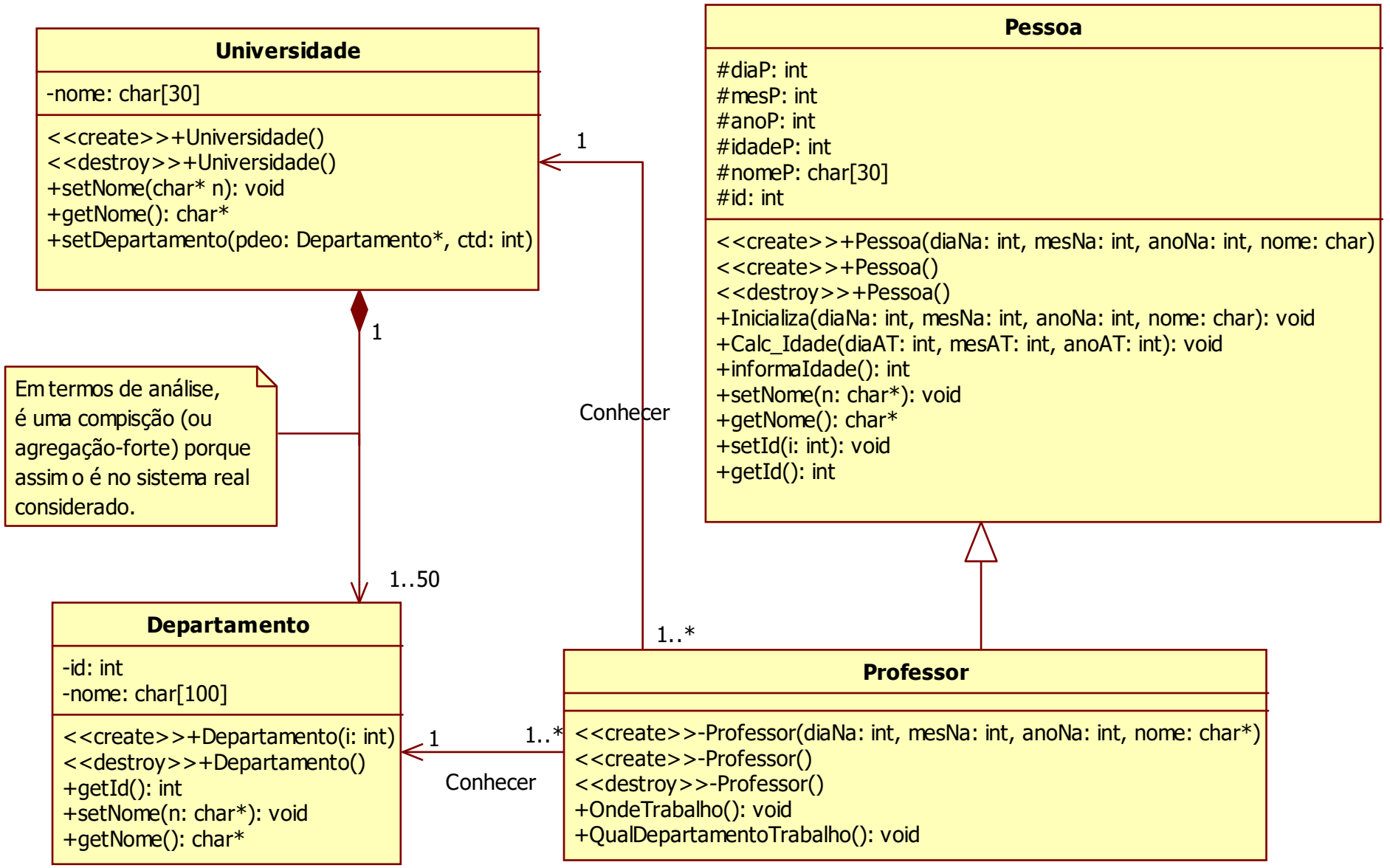
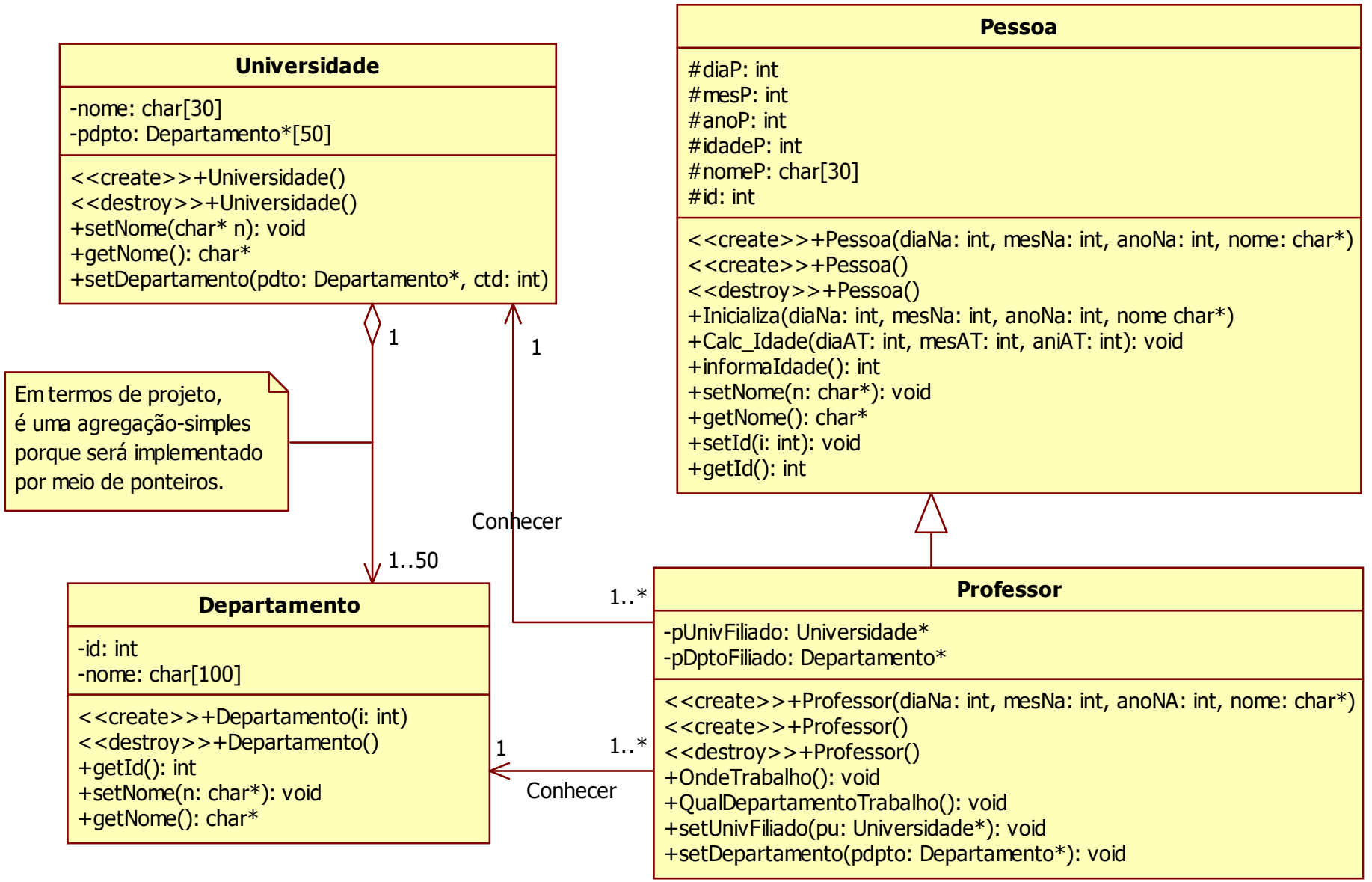


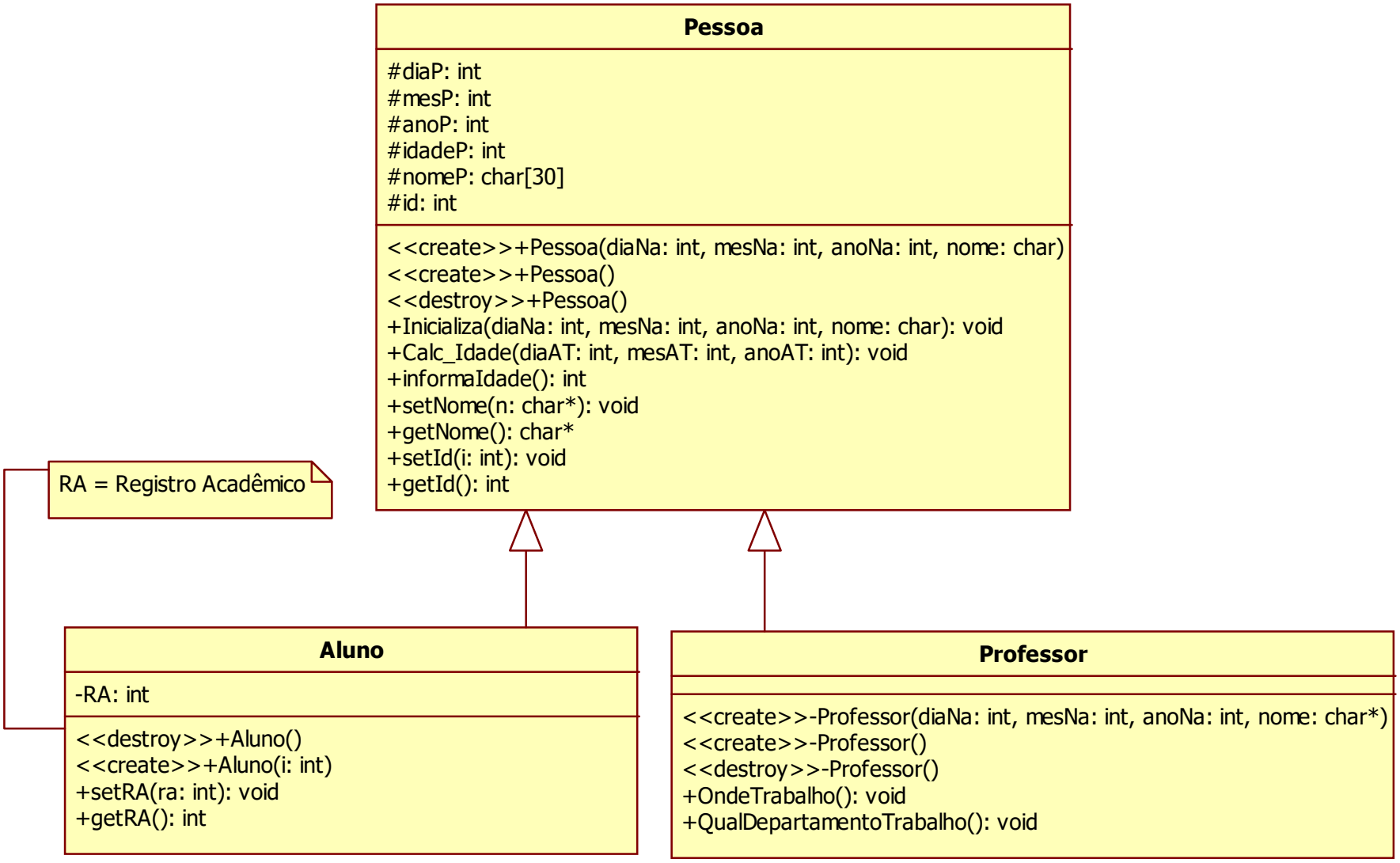
Diagrama de Classes - Projeto



Exercício

- **Criar uma classe *Aluno* que seja derivada da classe *Pessoa*.**

- **A classe Aluno terá uma atributo inteiro chamado RA, que é acrônimo de Registro Acadêmico.**



Obs.: Este é um diagrama de classes que pode ser considerado de análise

```
#ifndef _ALUNO_H_
#define _ALUNO_H_

#include "Pessoa.h"

class Aluno : public Pessoa
{
  private:

    int RA;

  public:


    Aluno( int diaNa, int mesNa, int anoNa, char* nome = "“ );
    // Faltaria atualizar o diagrama UML sobre esta contrutora.

    Aluno (int i = -1 );

    ~Aluno ( );

    void setRA ( int ra );
    int    getRA ( );

};
#endif
```

Tipos de Herança

```
class Derivada : public Base
{
  private:
  protected:
    // . . .
  public:
    // . . .
};
```

Uma herança pública (*public*) faz com que os elementos *protected* e *public* da classe base continuem como são na classe derivada.

```
class Derivada : protected Base
{
  private:
  protected:
    // . . .
  public:
    // . . .
};
```

Uma herança protegida *(protected)* faz com que os elementos *protected* e *public* da classe base sejam todos *protected* na classe derivada.

```
class Derivada : private Base
{
  private:
  protected:
    // . . .
  public:
    // . . .
};
```

Uma herança privada *(private)* faz com que os elementos *protected* e *public* da classe base sejam todos *private* na classe derivada.