OO – Engenharia Eletrônica

Orientação a Objetos
-
Programação em C++

Slides 5: Lista, Relações (via Ponteiros), Alocação Dinâmica.

Prof. Jean Marcelo SIMÃO

Exercício anterior

- Cada Objeto oriundo da Classe *Disciplina* poderá ter um número determinado de objetos Alunos relacionados (de uma Classe Aluno).

- Este número será determinado no construtor da classe, cujo valor padrão (*default*) será 45.

- As referências (*endereços*) dos objetos *Aluno* serão armazenados em uma lista duplamente encadeada em cada objeto Disciplina.

~~**- Os objetos *Aluno* serão "registrados" em ordem alfabética nos objetos *Disciplina* relacionados.**~~

- *Alunos* poderão ser incluídos e excluídos das listas das Disciplinas.

Diagrama de Classes – Análise - Associação

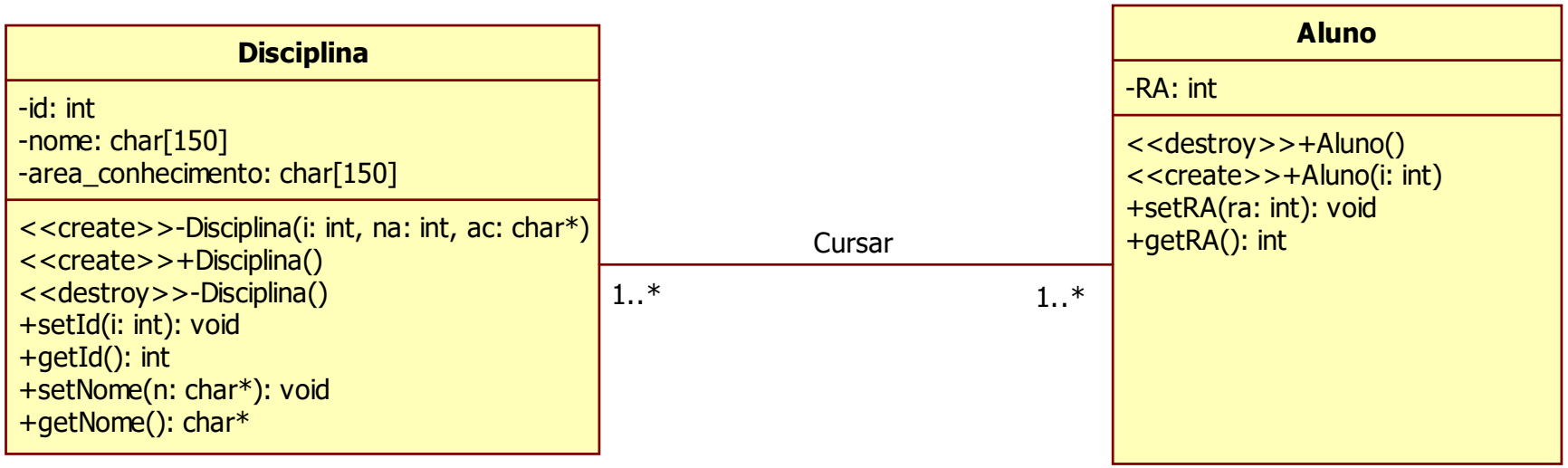


Diagrama de Classes – Análise – Agregação-Simples

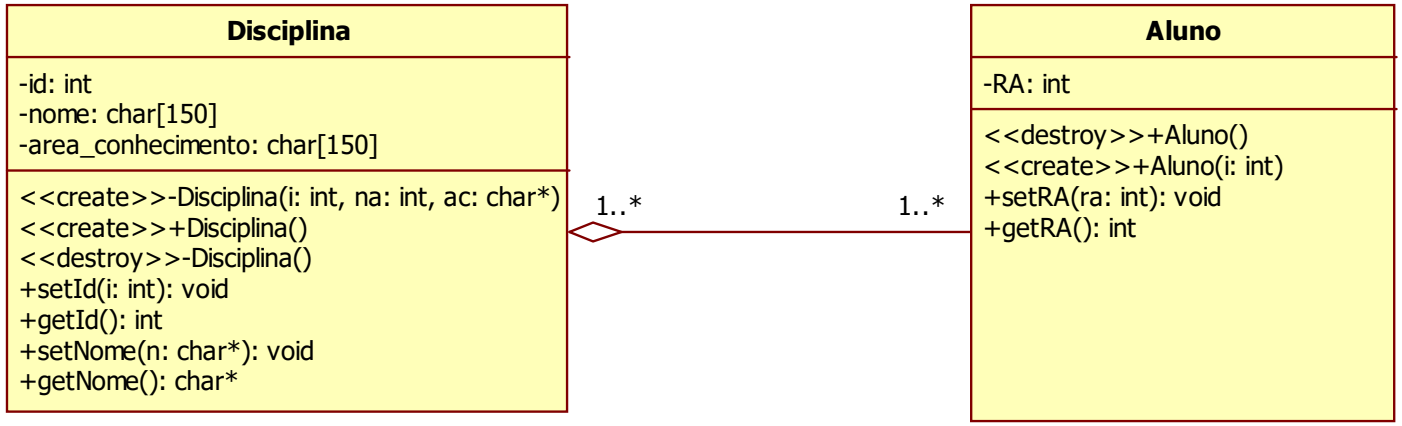
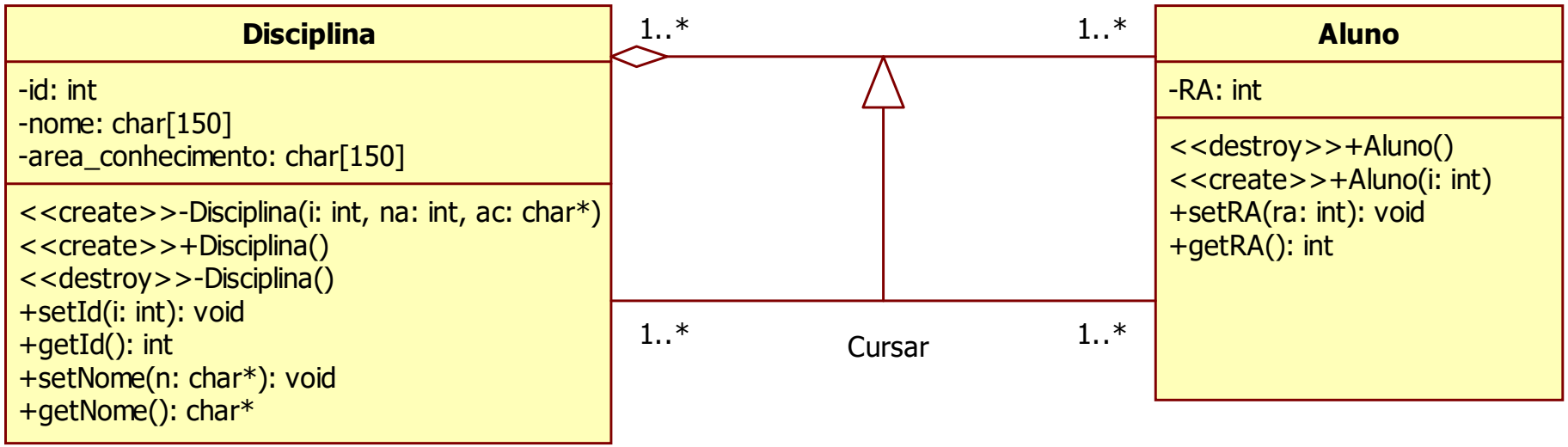
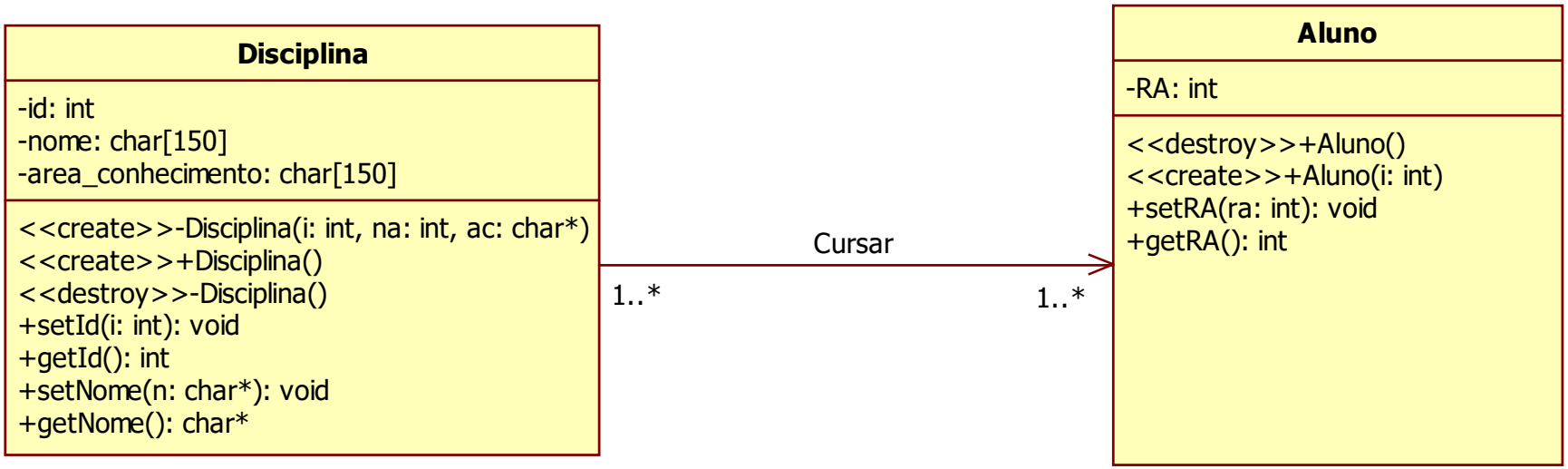


Diagrama de Classes – Análise



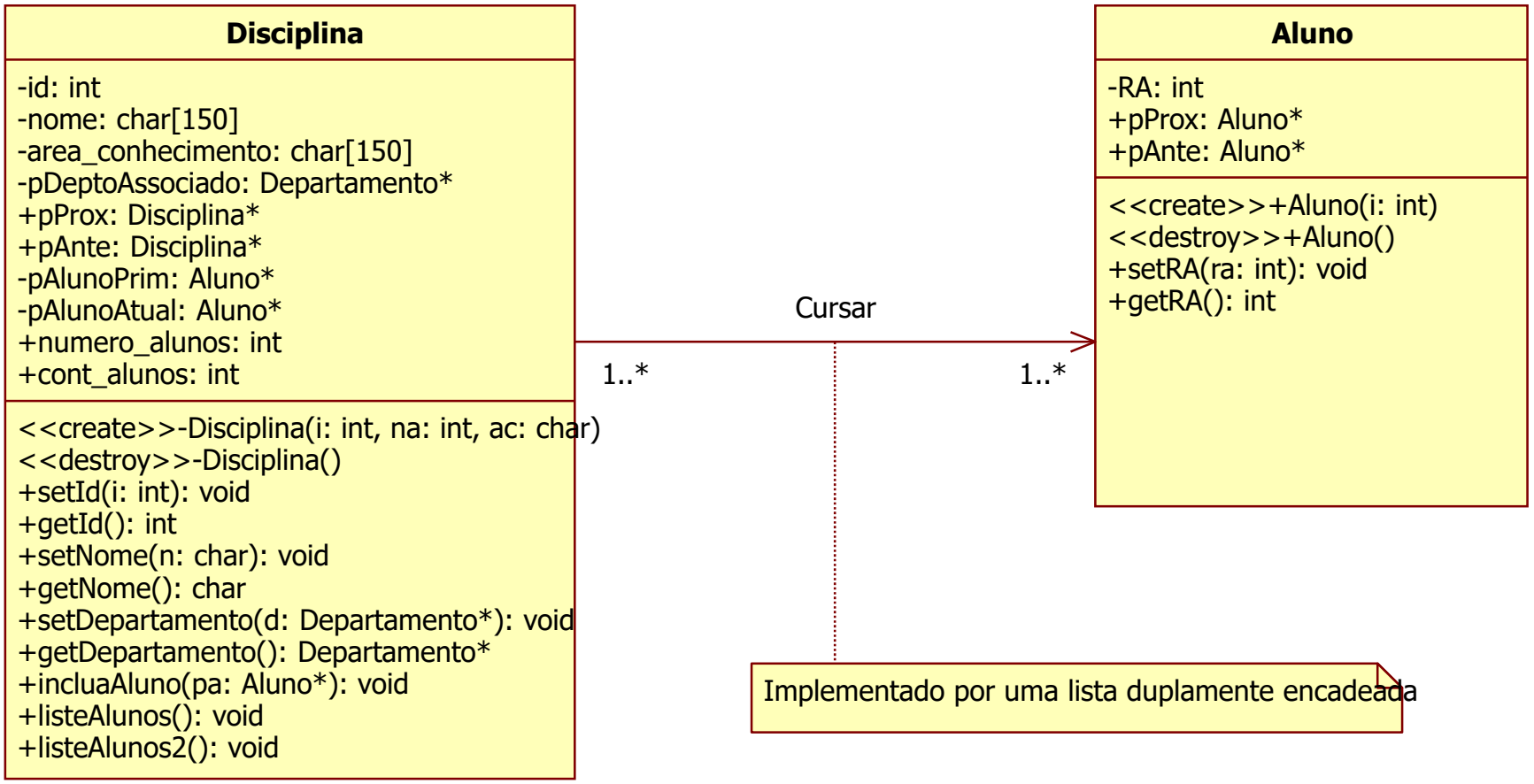
A agregação simples se REALIZA por meio de uma associação

Diagrama de Classes – Análise



A agregação simples se REALIZA por meio de uma associação

Diagrama de Classes – Projeto



‘Correção de Exercício’

```
#ifndef _DISCIPLINA_H_
#define _DISCIPLINA_H_
#include "Aluno.h“
#include “Departamento.h”
class Disciplina
{
private:
   int id;
   char nome [ 150 ];
   char area_conhecimento [ 150 ];
   int numero_alunos;
   int cont_alunos;
   Departamento* pDeptoAssociado;

   Aluno* pAlunoPrim;
   Aluno* pAlunoAtual;
 public:
   Disciplina ( int na = 45, char* ac = "" );
   ~Disciplina ( );

   Disciplina* pProx;
   Disciplina* pAnte;

   void setId ( int n ) ;
   int getId ( );

   void   setNome ( char* n );
   char* getNome ( );

   void setDepartamento ( Departamento* pdpto );
   Departamento* getDepartamento ( );

   void incluaAluno ( Aluno* pa );
   void listeAlunos ( );
   void listeAlunos2 ( );
};
#endif
```

Cada Objeto oriundo da Classe *Disciplina* poderá ter um número determinado de objetos Alunos relacionados.

As referências (*endereços*) dos objetos *Aluno* serão armazenados em uma lista duplamente encadeada em cada objeto Disciplina.

Este número será determinado no construtor da classe, cujo valor padrão (*default*) será 45.

```
#ifndef _ALUNO_H_
#define _ALUNO_H_
#include “Pessoa.h“

class Departamento;

class Aluno : public Pessoa
{
  private:
    int RA;

  public:

    Aluno *pProx;
    Aluno *pAnte;

    Aluno ( );
    ~Aluno ( );

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

};
#endif
```

```
#include "Disciplina.h"
#include "Departamento.h"
#include <stdio.h>
#include <string.h>

Disciplina::Disciplina ( int na, char* ac )
{
   pDeptoAssociado = NULL;
   pAlunoPrim      = NULL;
   pAlunoAtual     = NULL;
   pProx           = NULL;
   pAnte           = NULL;
   cont_alunos     = 0;
   numero_alunos   = na;
   strcpy ( area_conhecimento, ac ) ;
}

Disciplina::~Disciplina ( )
{
  pDeptoAssociado  = NULL;
  pAlunoPrim       = NULL;
  pAlunoAtual      = NULL;
  pProx            = NULL;
  pAnte            = NULL;
}

void   Disciplina::setId ( int n )          {  id = n;          }
int    Disciplina::getId ( )                {  return id;       }
void   Disciplina::setNome ( char* n )      {  strcpy ( nome, n );  }
char*  Disciplina::getNome ( )              {  return nome;  }

void Disciplina::setDepartamento ( Departamento* pd )
{
  // Cada vez que um Departamento é associado a uma Disciplina,
  // esta Disciplina passa a fazer parte da lista de disciplina
  // do Departamento, por meio do comando abaixo.
  pDeptoAssociado = pd;
  pd->setDisciplina ( this );
}

Departamento* Disciplina::getDepartamento ( )
{
  return  pDeptoAssociado;
}
```

```
void Disciplina::incluaAluno ( Aluno* a )
{
   if ( ( cont_alunos < numero_alunos ) && ( a != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim              = a;
        pAlunoAtual             = a;
     }
     else
     {
       pAlunoAtual->pProx       = a;
       a->pAnte                 = pAlunoAtual;
       pAlunoAtual              = a;
     }
     cont_alunos++;
   }
   else
   {
     printf ( " Aluno não incluído. Turma já lotada \n" );
   }
}

void Disciplina::listeAlunos ( )
{
  Aluno* paux;
  paux = pAlunoPrim;
  while ( paux != NULL )
  {
    printf ( " Aluno %s matriculado em  %s \n ", paux->getNome(), nome );
    paux = paux->pProx;
  }
}

void Disciplina::listeAlunos2 ( )
{
  Aluno* paux;
  paux = pAlunoAtual;
  while ( paux != NULL )
  {
    printf ( "Aluno %s matriculado em %s \n", paux->getNome( ), nome );
    paux = paux->pAnte;
  }
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim  = pa;
        pAlunoAtual  = pa;
     }
     else
     {
       pAlunoAtual->pProx = pa;
       pa->pAnte          = pAlunoAtual;
       pAlunoAtual        = pa;
     }
    cont_alunos++;
   }
   else
   {
     printf ( "Aluno não incluído. Turma já lotada \n" );
   }
}
```

Este slide apenas destaca/salienta os métodos *incluaAluno* e *listeAlunos* da classe *Disciplina*.

```
void Disciplina::listeAlunos ( )
{
  Aluno* paux;
  paux = pAlunoPrim;
  while ( paux != NULL )
  {
    printf ( "Aluno %s matriculado em  %s \n", paux->getNome ( ), nome);
    paux = paux->pProx;
  }
}

void Disciplina::listeAlunos2 ( )
{
  Aluno* paux;
  paux = pAlunoAtual;
  while ( paux != NULL )
  {
    printf ("Aluno %s matriculado em %s \n", paux->getNome(), nome);
    paux = paux->pAnte;
  }
}
```

```
#ifndef _PRINCIPAL_H_
#define _PRINCIPAL_H_
#include "Professor.h"
#include "Universidade.h"
#include "Aluno.h"
#include "Disciplina.h"

class Principal
{
 private:
   // Universidades
   Universidade UTFPR;
   Universidade Princeton;
   Universidade Cambridge;

   // Departamentos
   Departamento DAELN;
   Departamento MatematicaUTFPR;
   Departamento FisicaUTFPR;

   Departamento MatematicaPrinceton;
   Departamento FisicaPrinceton;

   Departamento MatematicaCambridge;
   Departamento FisicaCambridge;

   // Professores
   Professor Simao;
   Professor Einstein;
   Professor Newton;

   // Disciplinas
   Disciplina Computacao1_2006;
   Disciplina Introd_Alg_2007;
   Disciplina Computacao2_2007;
   Disciplina Metodos2_2007;
   // Alunos
   Aluno    AAA;
   Aluno    BBB;
   Aluno    CCC;
   Aluno    DDD;
   Aluno    EEE;

   int      diaAtual;
   int      mesAtual;
   int      anoAtual;

 public:

   Principal ( int dia, int mes, int ano );

   // Inicializações...
   void Inicializa ( );
   void InicializaUnivesidades ( );
   void InicializaDepartamentos ( );
   void InicializaProfessores ( );
   void InicializaAlunos ( );
   void InicializaDisciplinas ( );

   void Executar ( );

   void CalcIdadeProfs ( );
   void UnivOndeProfsTrabalham ( );
   void DepOndeProfsTrabalham ( );
   void ConhecPessoa ( );
   void ListeDiscDeptos ( );

};

#endif
```

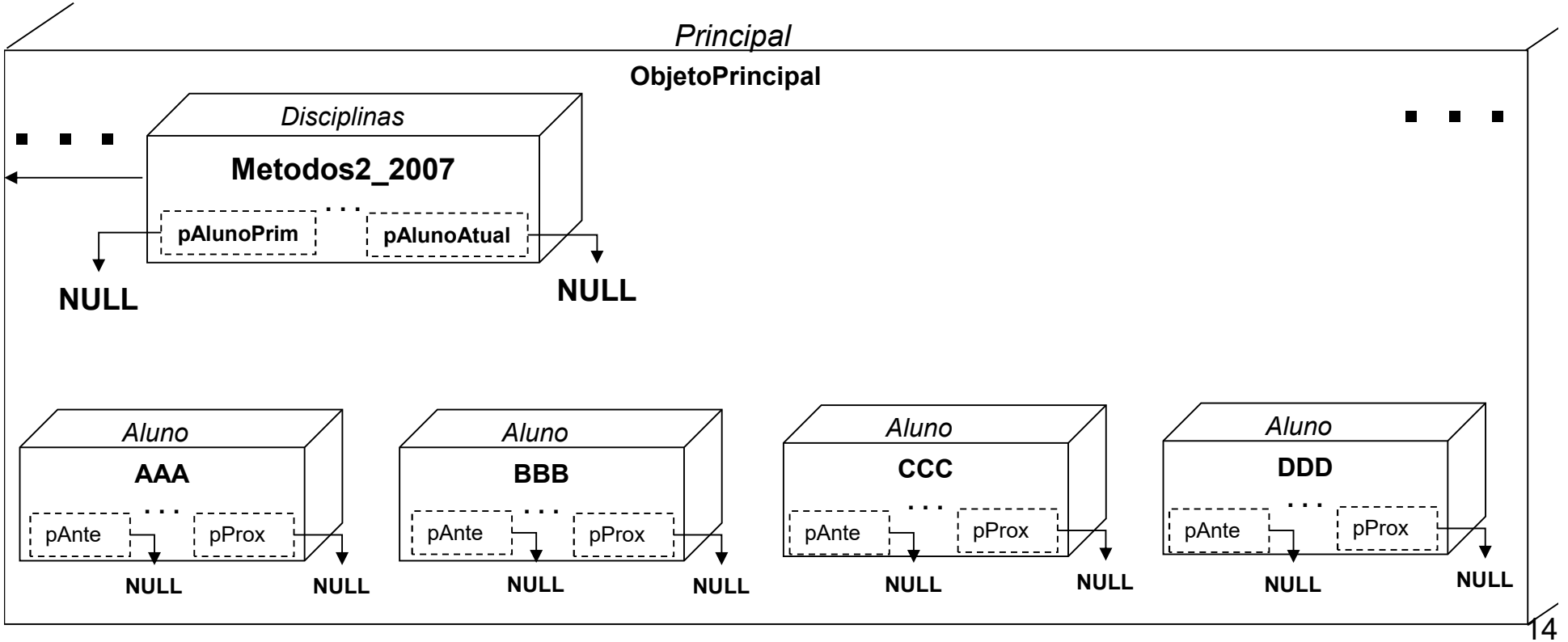
```
void Principal::InicializaAlunos ( )
{
 // . . .
}

void Principal::InicializaDisciplinas ( )
{
   Computacao1_2006.setNome ( "Computacao I 2006" );
   Introd_Alg_2007.setNome      ( "Introducao de Algoritmos de Programacao 2007" );
   Computacao2_2007.setNome ( "Computao II"   );
   Metodos2_2007.setNome       ( "Métodos II" );

   Computacao1_2006.setDepartamento ( &DAELN );
   Introd_Alg_2007.setDepartamento      ( &DAELN);
   Computacao2_2007.setDepartamento ( &DAELN);
   Metodos2_2007.setDepartamento       ( &DAELN);

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Principal::ListeAlunosDisc ( )
{
   Metodos2_2007.listeAlunos ( );
   printf ( "\n" );

   Metodos2_2007.listeAlunos2 ( );
   printf ( "\n" );
}

void Principal::Executar ( )
{
   CalcIdadeProfs ( );
   UnivOndeProfsTrabalham ( );
   DepOndeProfsTrabalham ( );
   ConhecPessoa ( );
   ListeDiscDeptos ( );
   ListeAlunosDisc ( );
}
```

Faça uma “simulação”...

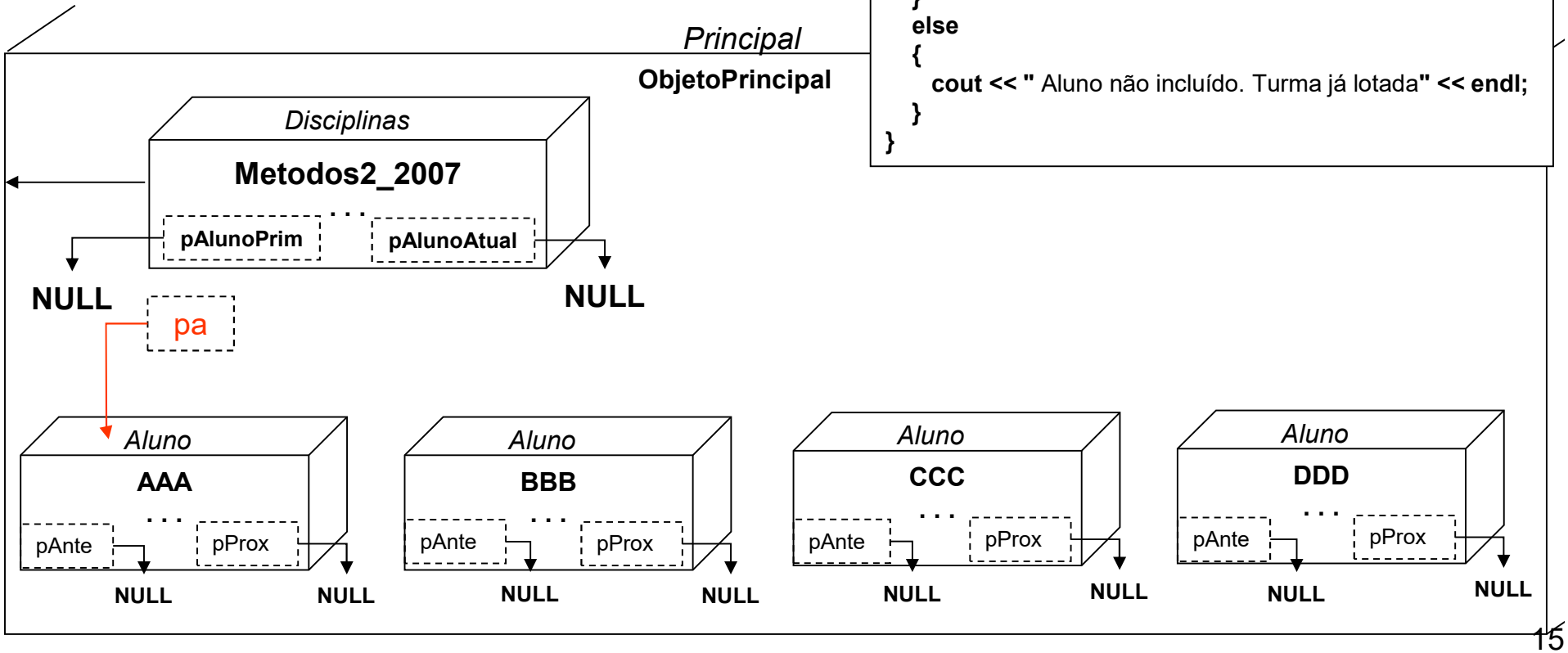
por meio do “diagrama de cubos” da constituição
da lista duplamente encadeada cubos”

```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```



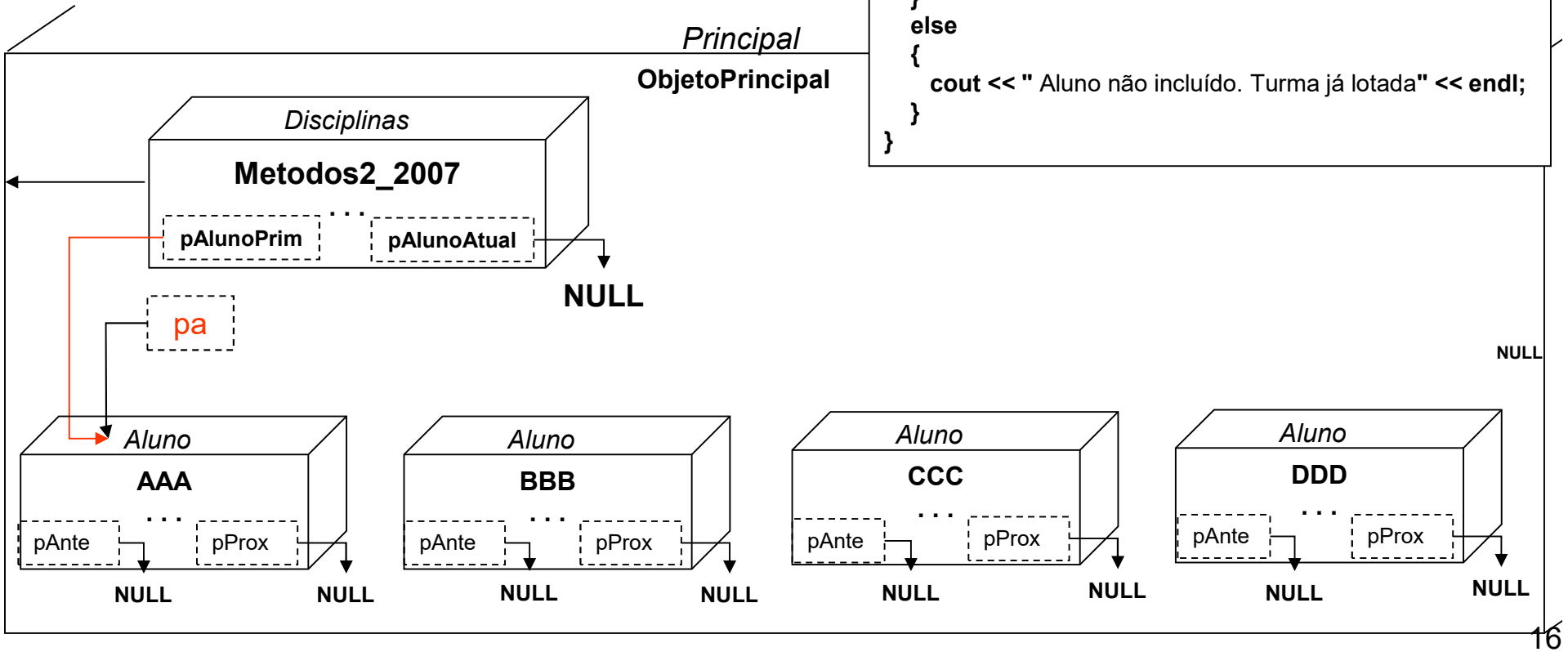
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim  = pa;
        pAlunoAtual  = pa;
     }
     else
     {
       pAlunoAtual->pProx    = pa;
       pa->pAnte             = pAlunoAtual;
       pAlunoAtual           = pa;
     }
    cont_alunos++;
   }
   else
   {
     cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```



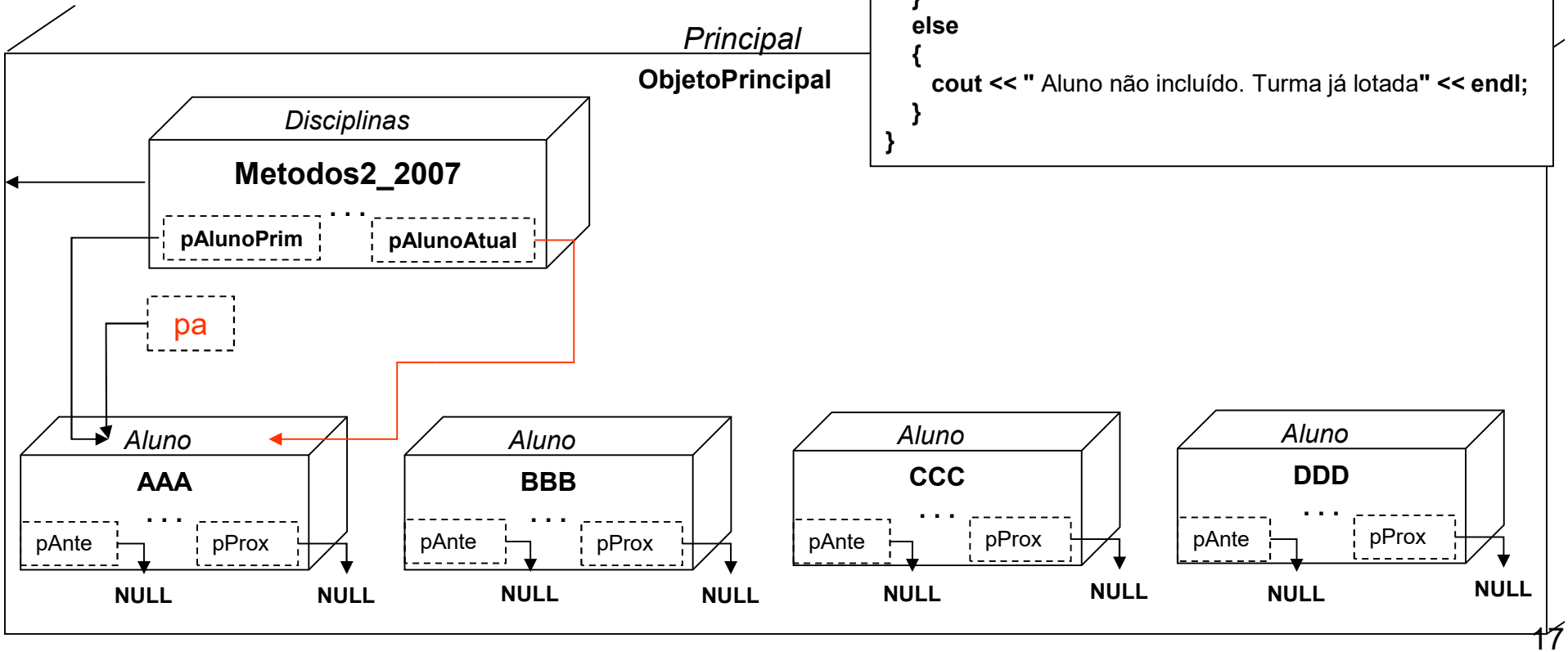
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte             = pAlunoAtual;
      pAlunoAtual           = pa;
    }
    cont_alunos++;
   }
   else
   {
    cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```

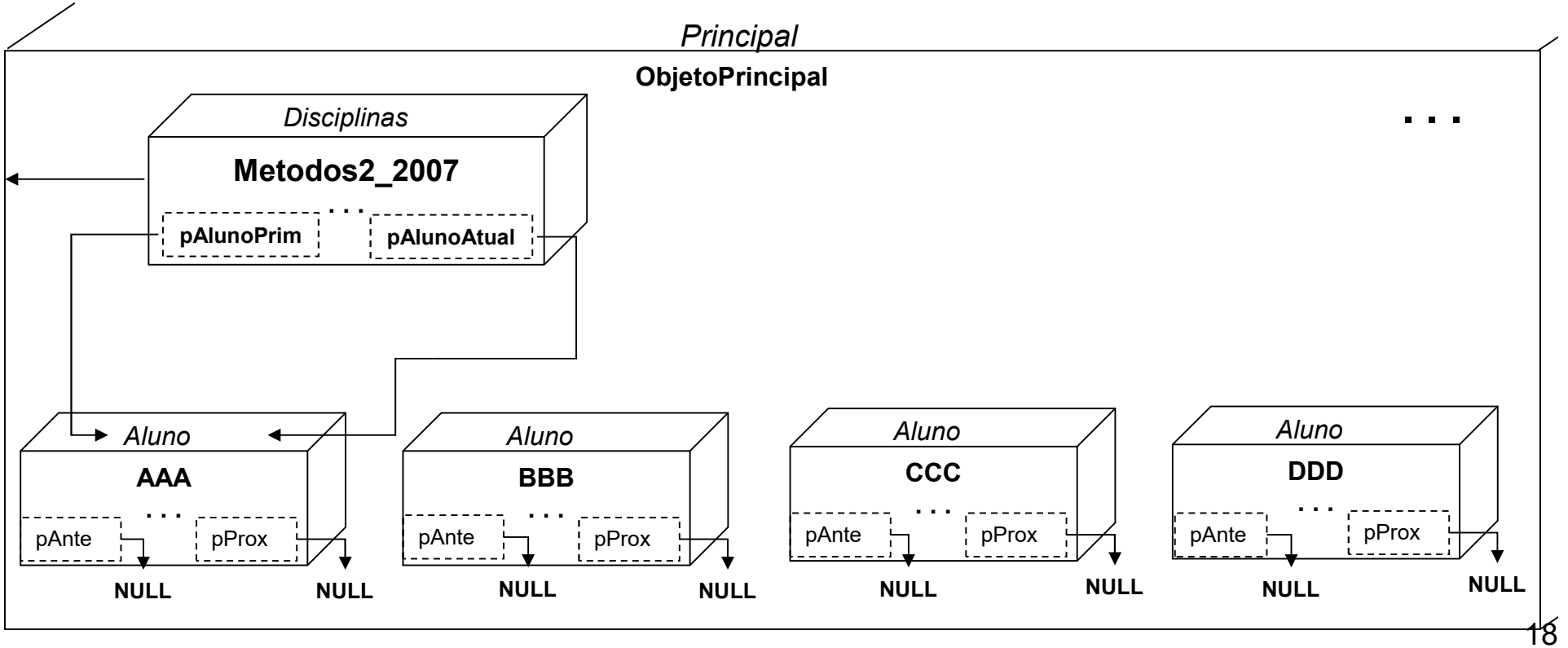


```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim  = pa;
        pAlunoAtual  = pa;
     }
     else
     {
       pAlunoAtual->pProx    = pa;
       pa->pAnte             = pAlunoAtual;
       pAlunoAtual           = pa;
     }
    cont_alunos++;
   }
   else
   {
     cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```

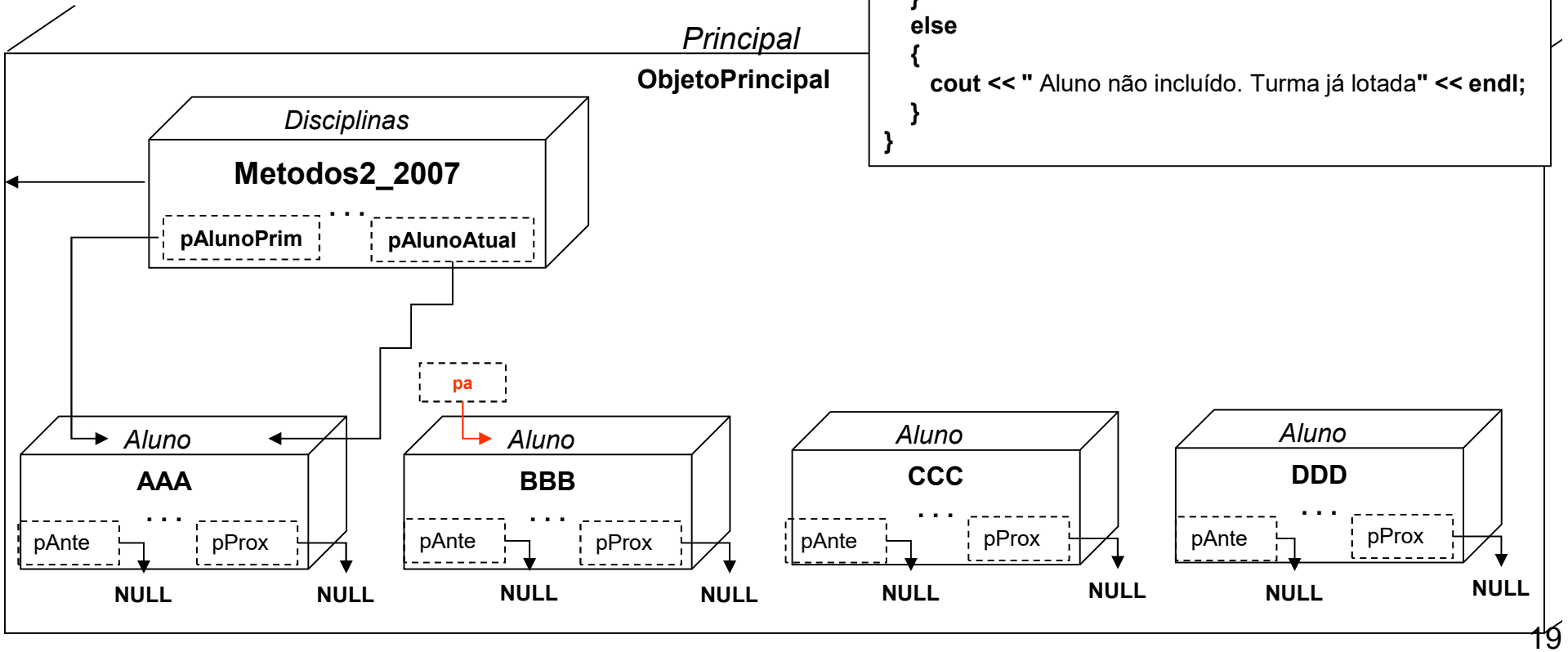


```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```



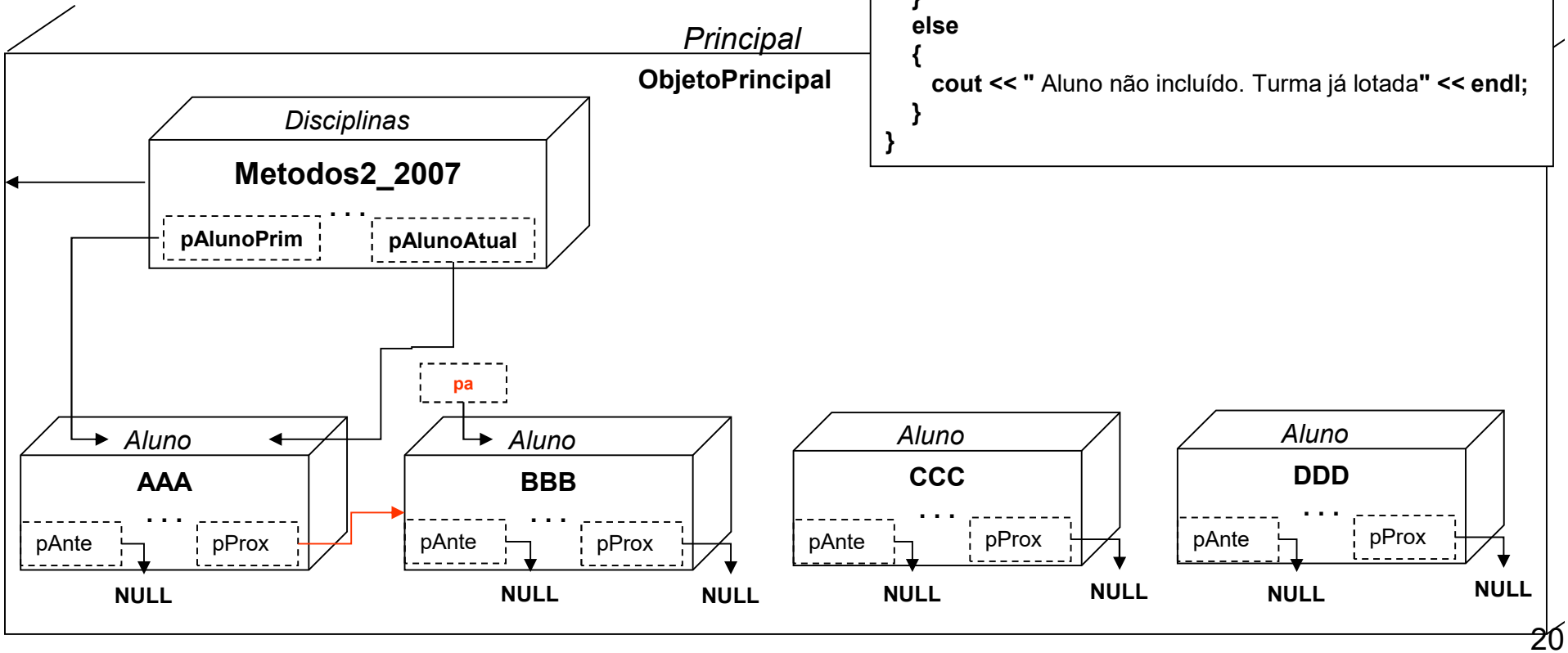
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte             = pAlunoAtual;
      pAlunoAtual           = pa;
    }
    cont_alunos++;
   }
   else
   {
     cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```



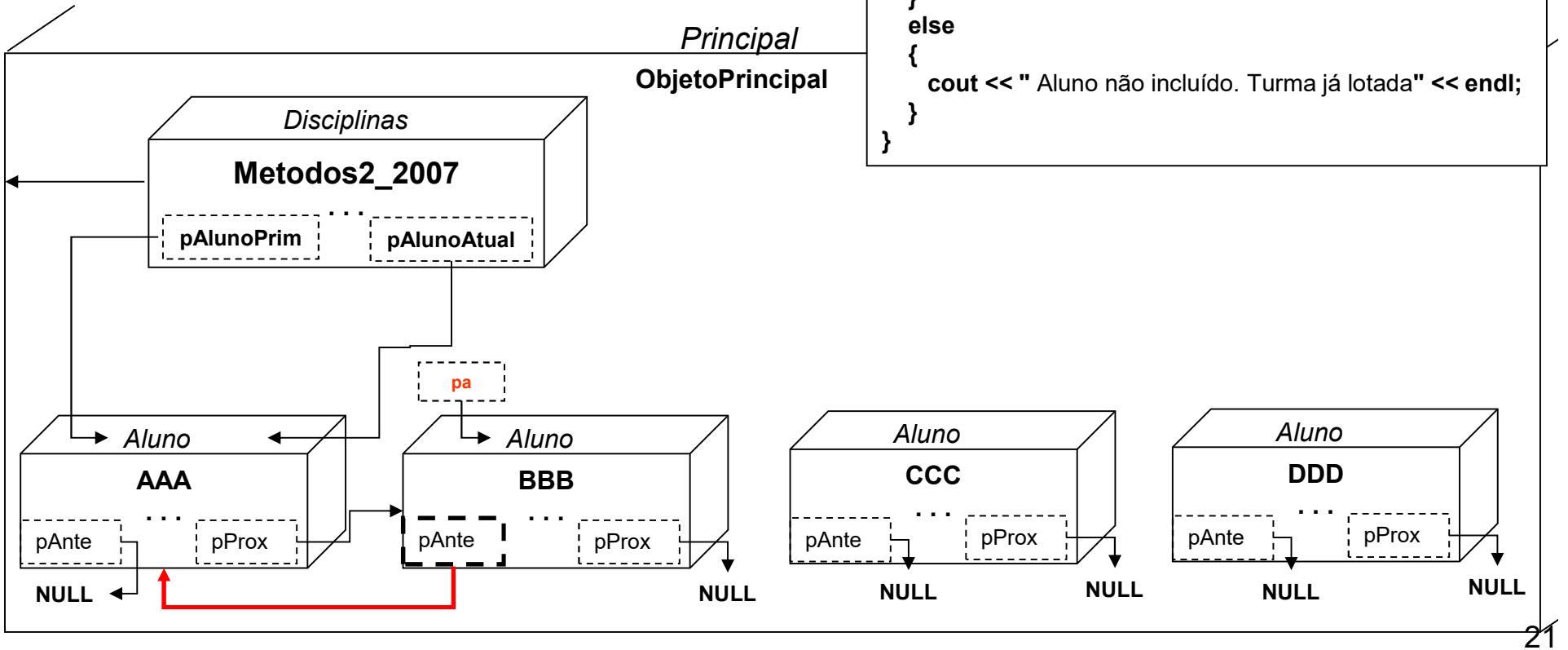
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim  = pa;
        pAlunoAtual  = pa;
     }
     else
     {
       pAlunoAtual->pProx   = pa;
       pa->pAnte            = pAlunoAtual;
       pAlunoAtual          = pa;
     }
     cont_alunos++;
   }
   else
   {
     cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```



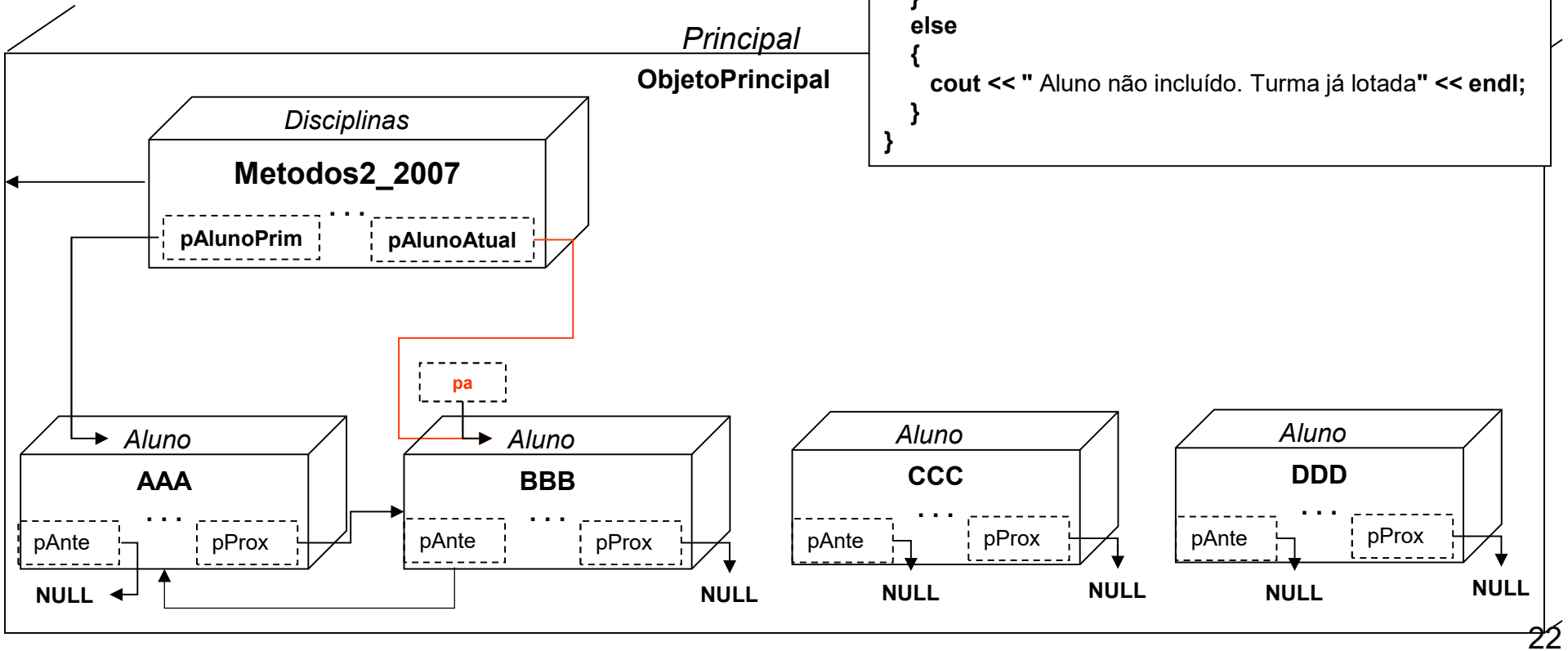
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim  = pa;
        pAlunoAtual  = pa;
     }
     else
     {
       pAlunoAtual->pProx    = pa;
       pa->pAnte             = pAlunoAtual;
       pAlunoAtual           = pa;
     }
     cont_alunos++;
   }
   else
   {
     cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```



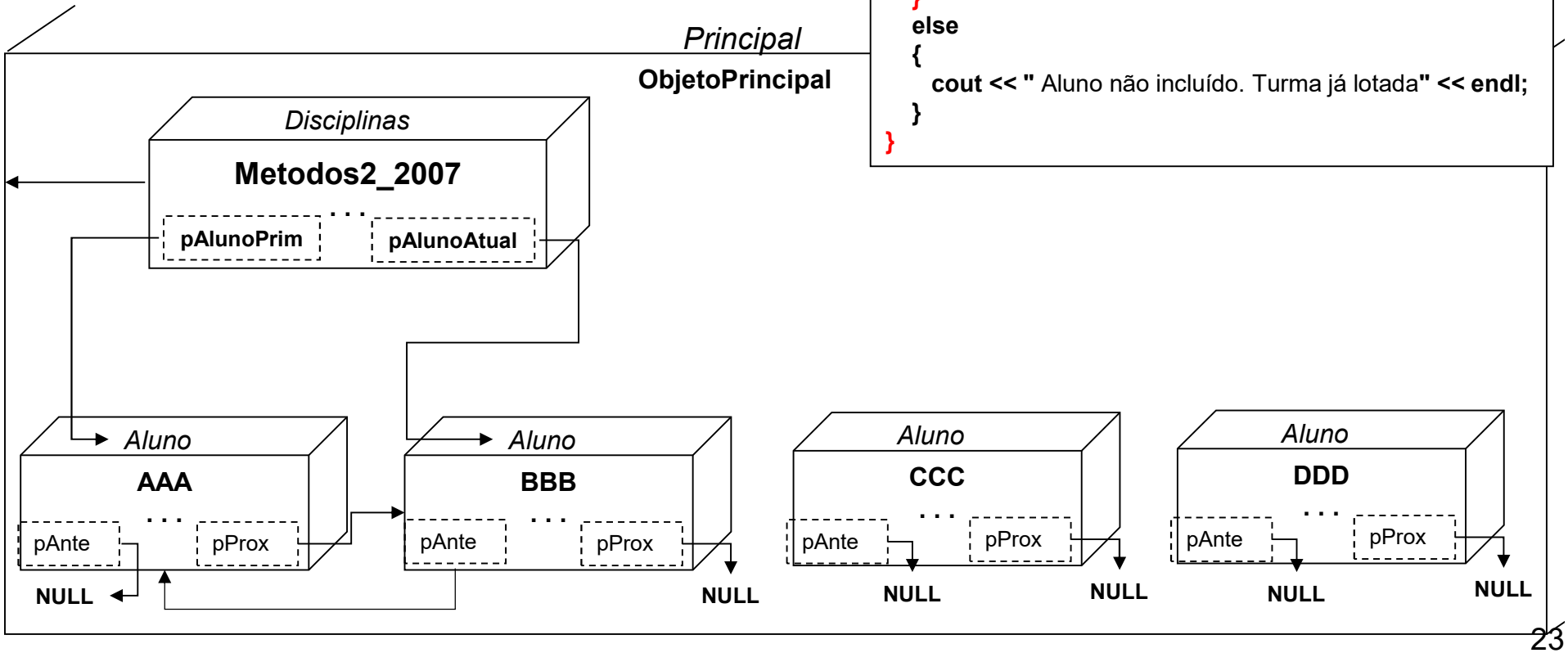
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim  = pa;
        pAlunoAtual  = pa;
     }
     else
     {
       pAlunoAtual->pProx    = pa;
       pa->pAnte             = pAlunoAtual;
       pAlunoAtual           = pa;
     }
     cont_alunos++;
   }
   else
   {
     cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```

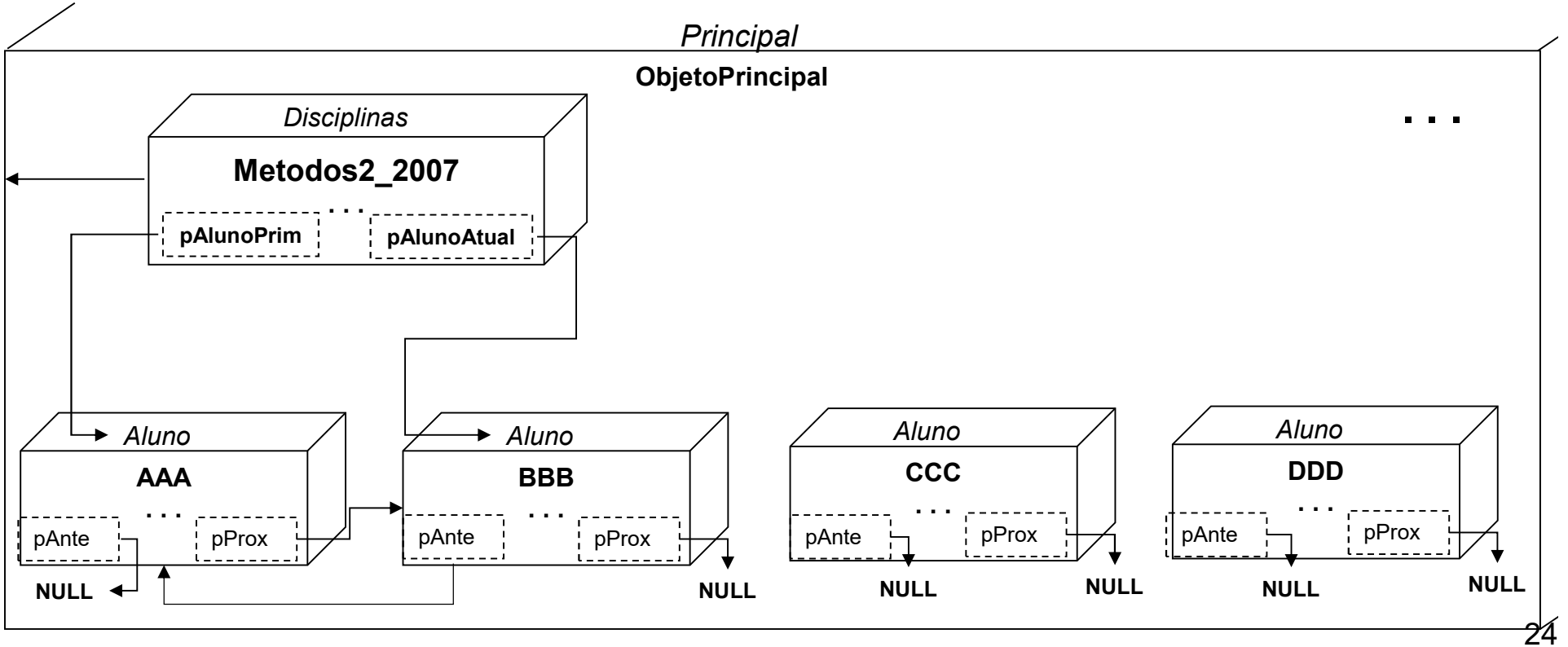


```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
     if ( pAlunoPrim == NULL )
     {
        pAlunoPrim  = pa;
        pAlunoAtual  = pa;
     }
     else
     {
       pAlunoAtual->pProx    = pa;
       pa->pAnte             = pAlunoAtual;
       pAlunoAtual           = pa;
     }
    cont_alunos++;
   }
   else
   {
     cout << " Aluno não incluído. Turma já lotada" << endl;
   }
}
```



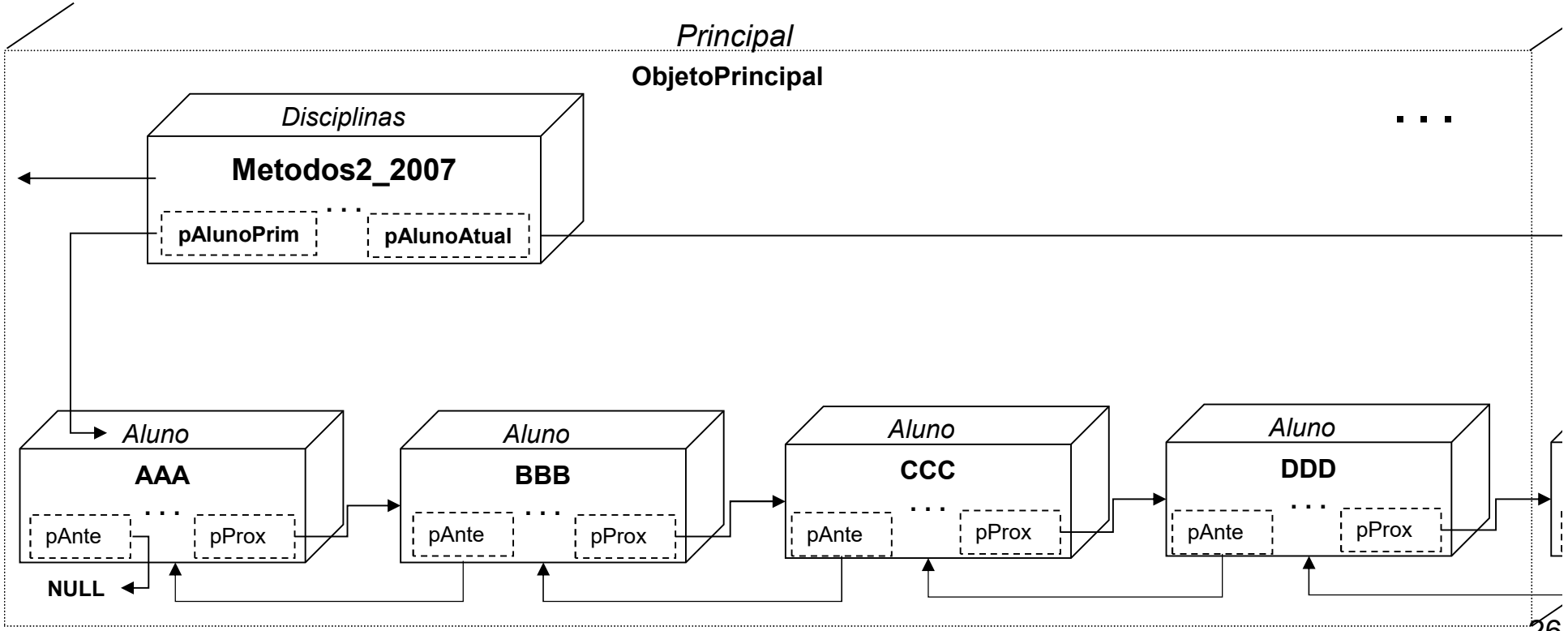
```
void Principal::InicializaDisciplinas ( )
{
    . . .
    Computacao1_2006.setDepartamento  ( &DAELN );
    Introd_Alg_2007.setDepartamento   ( &DAELN );
    Computacao2_2007.setDepartamento  ( &DAELN );
    Metodos2_2007.setDepartamento     ( &DAELN );

    Metodos2_2007.incluaAluno ( &AAA );
    Metodos2_2007.incluaAluno ( &BBB );
    Metodos2_2007.incluaAluno ( &CCC );
    Metodos2_2007.incluaAluno ( &DDD );
    Metodos2_2007.incluaAluno ( &EEE );
}
```



Continue a “simulação”

utilizando o “diagrama de cubos”.

```
void Principal::InicializaDisciplinas ( )
{
    . . .
    Computacao1_2006.setDepartamento  ( &DAELN );
    Introd_Alg_2007.setDepartamento   ( &DAELN );
    Computacao2_2007.setDepartamento  ( &DAELN );
    Metodos2_2007.setDepartamento     ( &DAELN );

    Metodos2_2007.incluaAluno ( &AAA );
    Metodos2_2007.incluaAluno ( &BBB );
    Metodos2_2007.incluaAluno ( &CCC );
    Metodos2_2007.incluaAluno ( &DDD );
    Metodos2_2007.incluaAluno ( &EEE );
}
```



```
void Principal::InicializaDisciplinas ( )
{

   Computacao1_2006.setNome ( "Computacao I 2006" );
   Introd_Alg_2007.setNome ( "Introducao de Algoritmos de Programacao 2007" );
   Computacao2_2007.setNome ( "Computao II" );
   Metodos2_2007.setNome ( "Métodos II" );

   Computacao1_2006.setDepartamento ( &DAELN );
   Introd_Alg_2007.setDepartamento      ( &DAELN );

   Computacao2_2007.setDepartamento ( & DAELN );
   Metodos2_2007.setDepartamento      ( & DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );

}
```

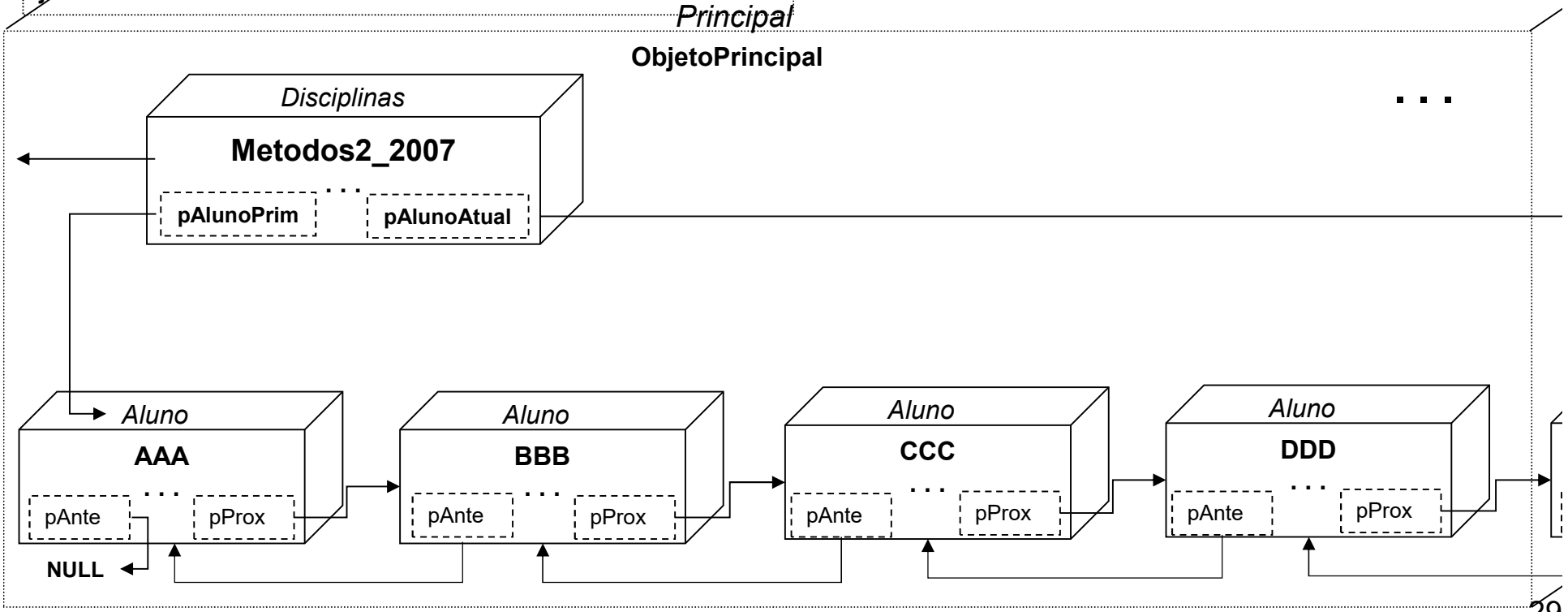
PROBLEMA!

Qual é o problema ?
Como resolver?

Faça uma “simulação”...

por meio do “diagrama de cubos”
para descobrir o problema

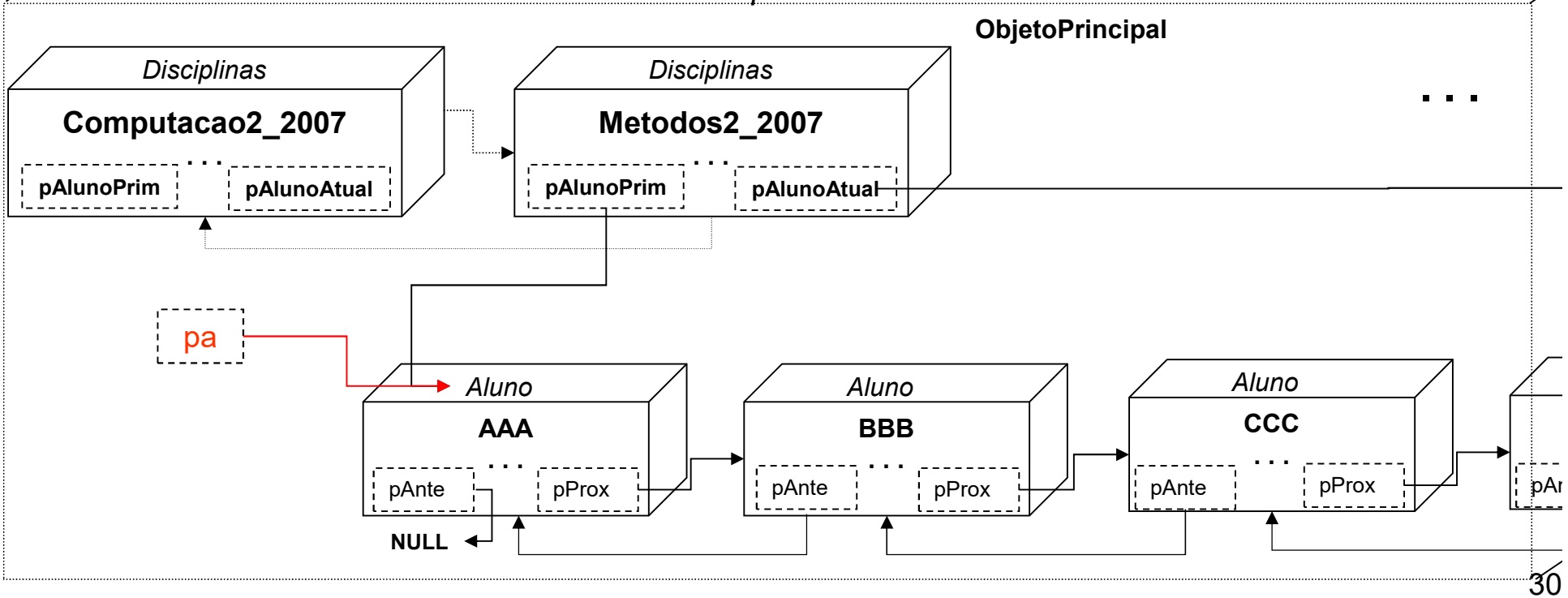
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```



```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte                 = pAlunoAtual;
      pAlunoAtual               = pa;
    }
    cont_alunos++;
   }
   else  { cout << " Aluno não incluído. Turma já lotada" << endl; }
}
```

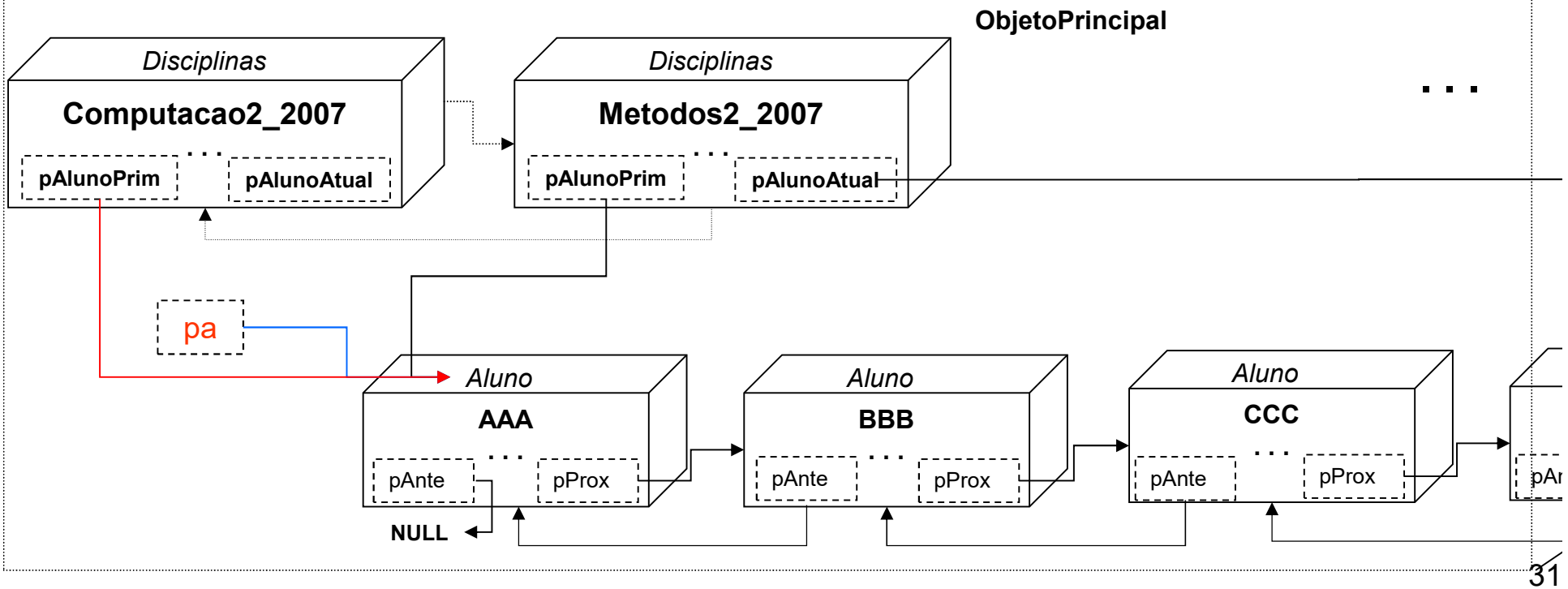
Principa



```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte                 = pAlunoAtual;
      pAlunoAtual               = pa;
    }
    cont_alunos++;
   }
   else  { cout << " Aluno não incluído. Turma já lotada" << endl; }
}
```

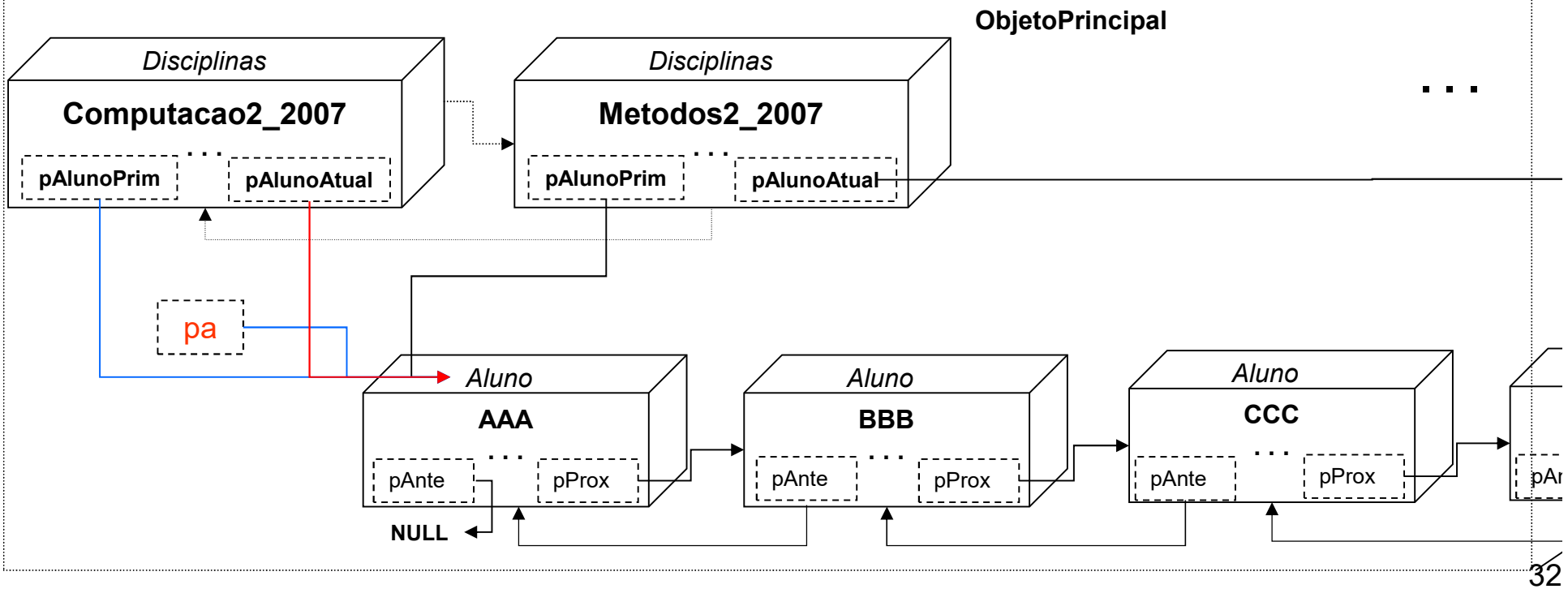
Principal



```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte                 = pAlunoAtual;
      pAlunoAtual               = pa;
    }
    cont_alunos++;
   }
   else  { cout << " Aluno não incluído. Turma já lotada" << endl; }
}
```

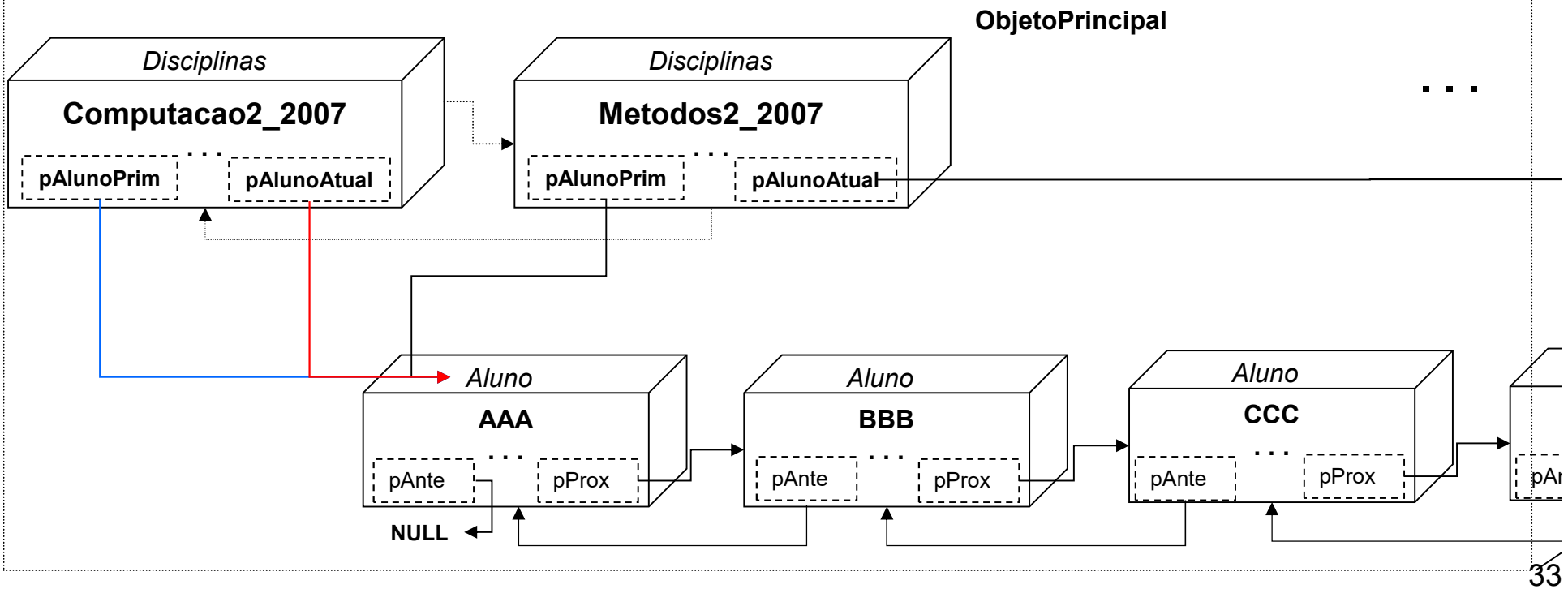
Principa



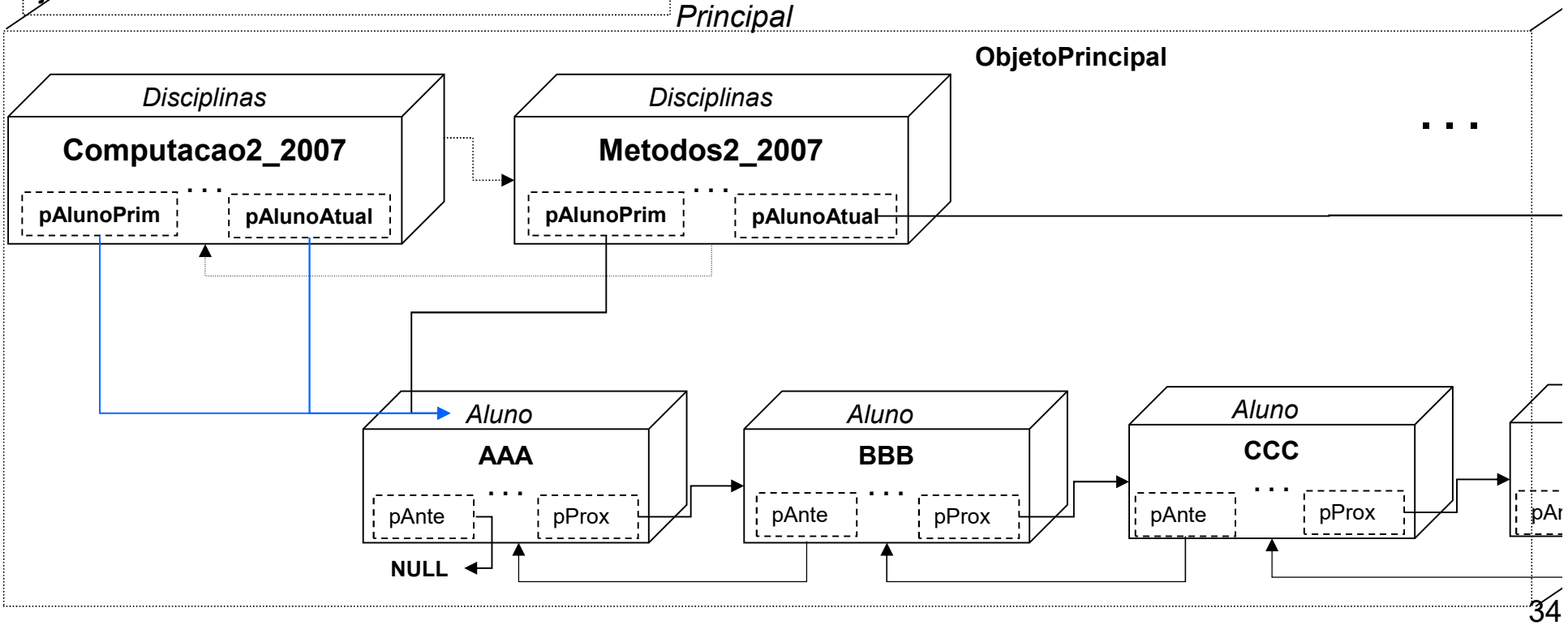
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte                 = pAlunoAtual;
      pAlunoAtual               = pa;
    }
    cont_alunos++;
   }
   else  { cout << " Aluno não incluído. Turma já lotada" << endl; }
}
```

Principa



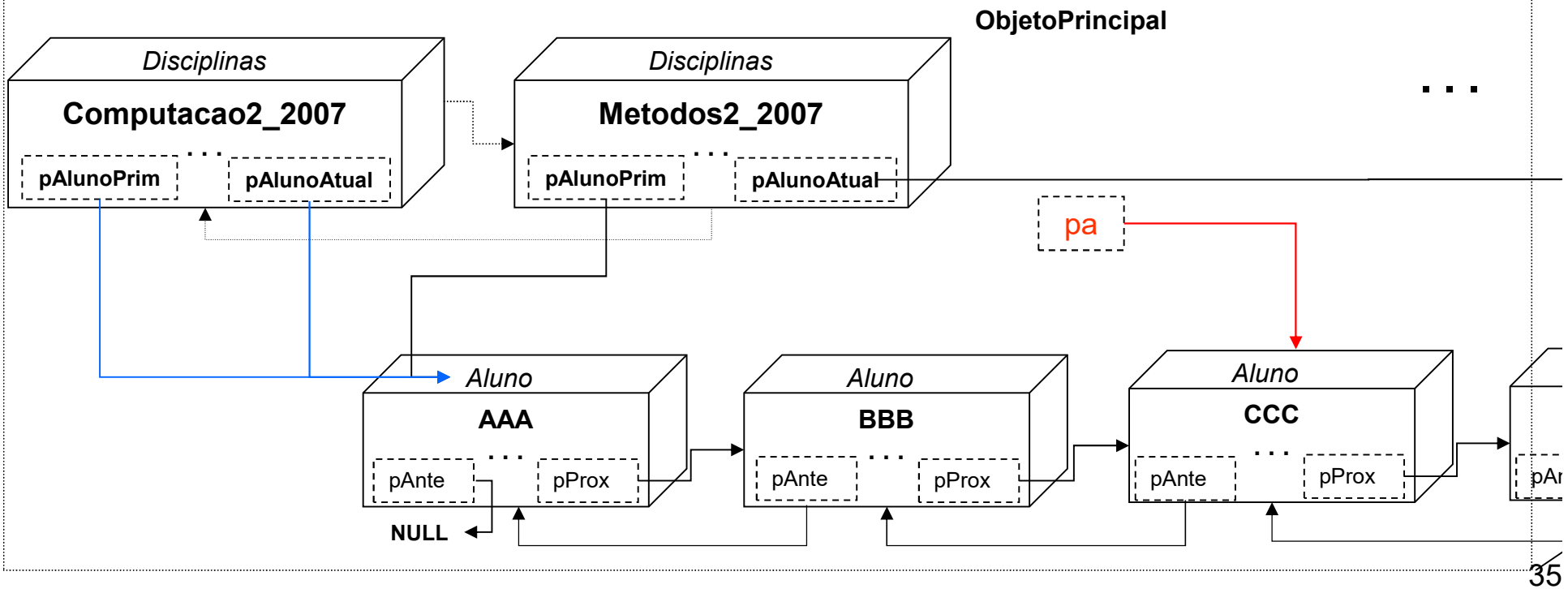
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```



```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte                 = pAlunoAtual;
      pAlunoAtual               = pa;
    }
    cont_alunos++;
   }
   else  { cout << " Aluno não incluído. Turma já lotada" << endl; }
}
```

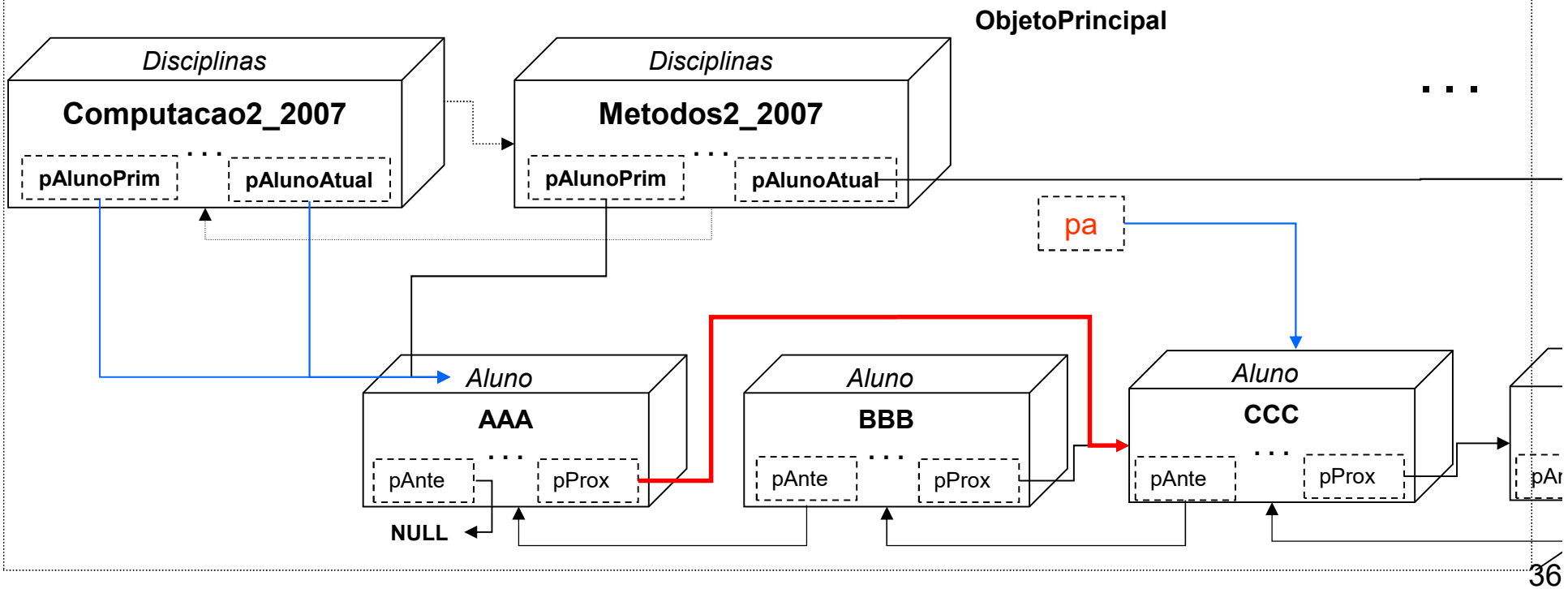
Principa...



```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte                 = pAlunoAtual;
      pAlunoAtual               = pa;
    }
    cont_alunos++;
   }
   else  { cout << " Aluno não incluído. Turma já lotada" << endl; }
}
```

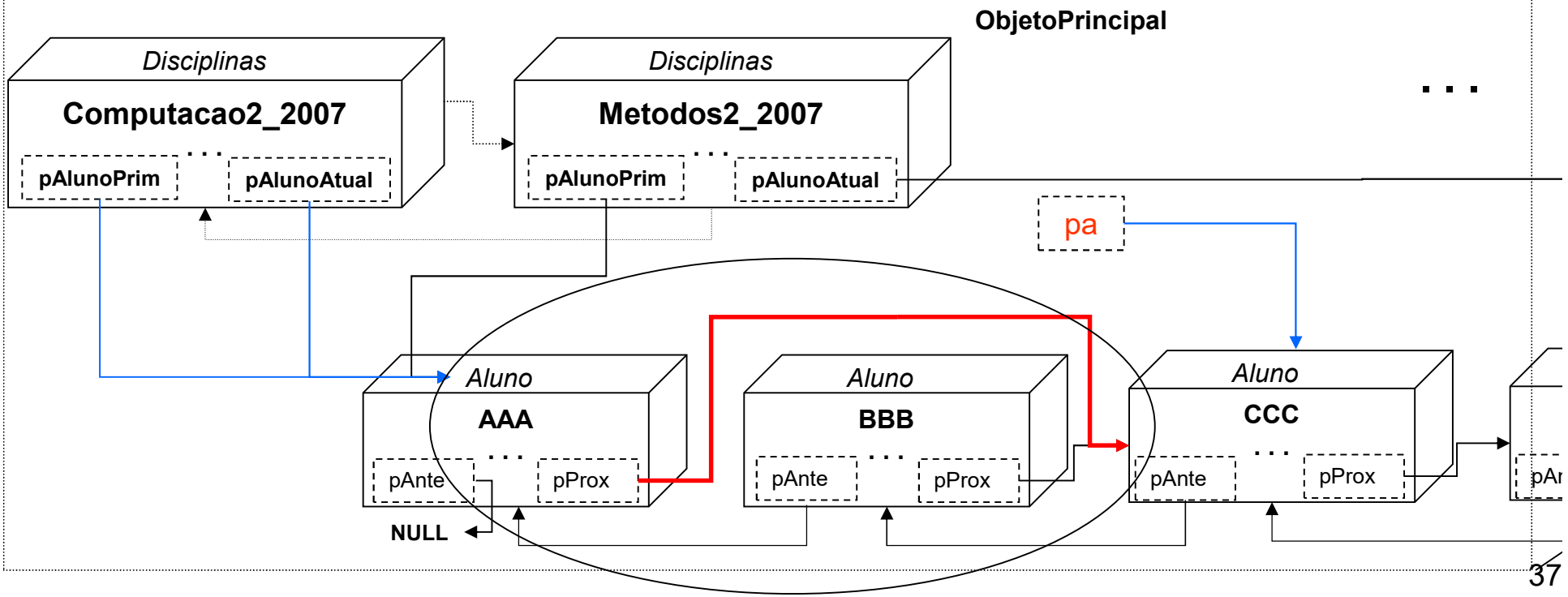
Principa



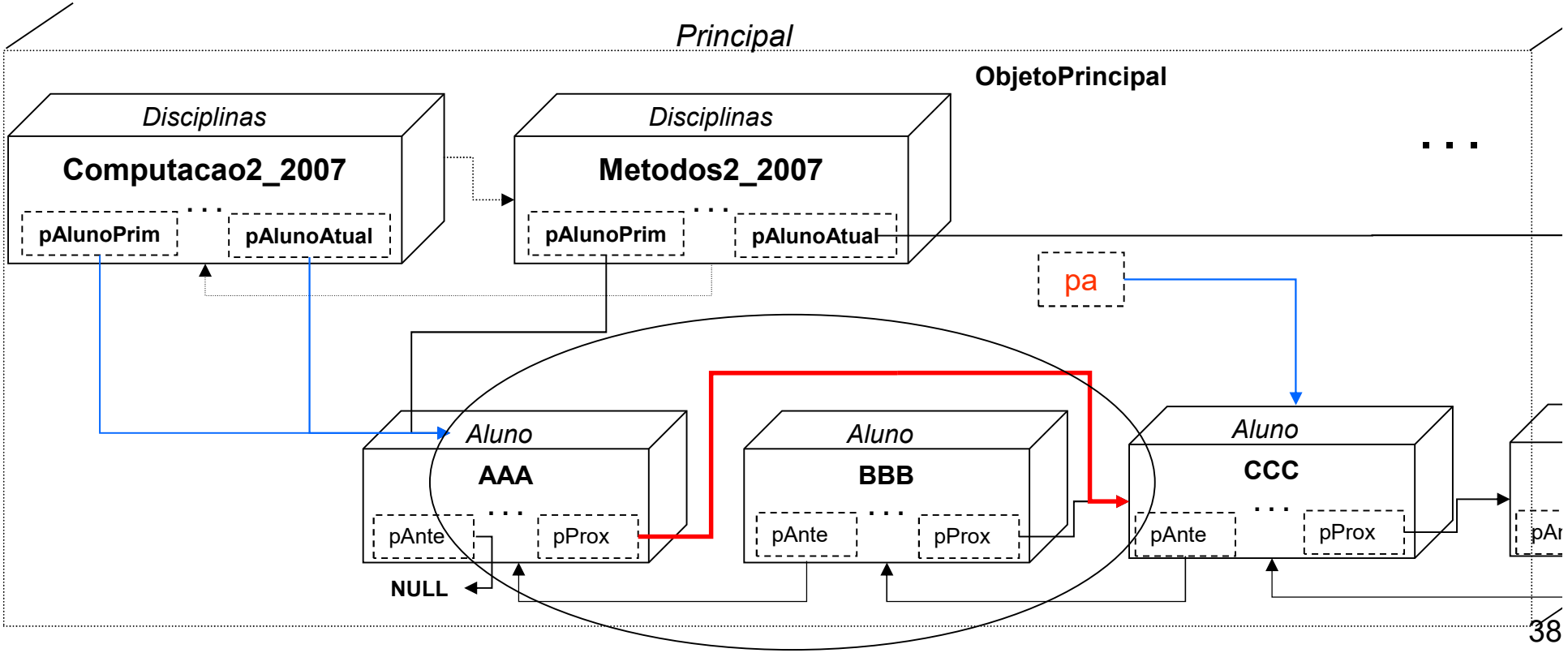
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```

```
void Disciplina:: incluaAluno ( Aluno* pa )
{
   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
    if ( pAlunoPrim == NULL )
    {
       pAlunoPrim  = pa;
       pAlunoAtual  = pa;
    }
    else
    {
      pAlunoAtual->pProx    = pa;
      pa->pAnte                 = pAlunoAtual;
      pAlunoAtual               = pa;
    }
    cont_alunos++;
   }
   else  { cout << " Aluno não incluído. Turma já lotada" << endl; }
}
```

Principa...



Perdeu-se a conexão de AAA para BBB necessária a lista (duplamente) encadeada de Metodos2_2007.



Uma primeira solução

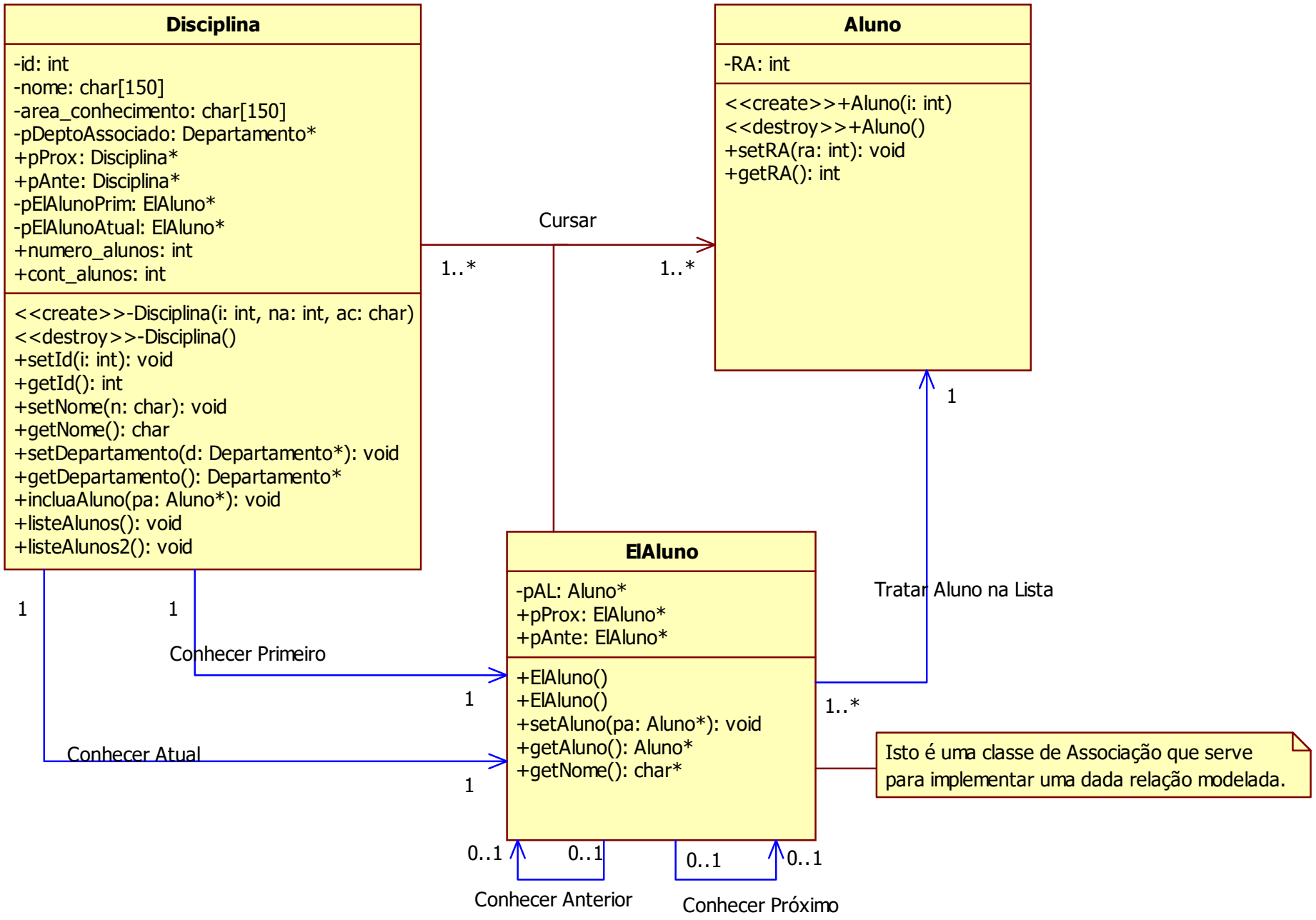
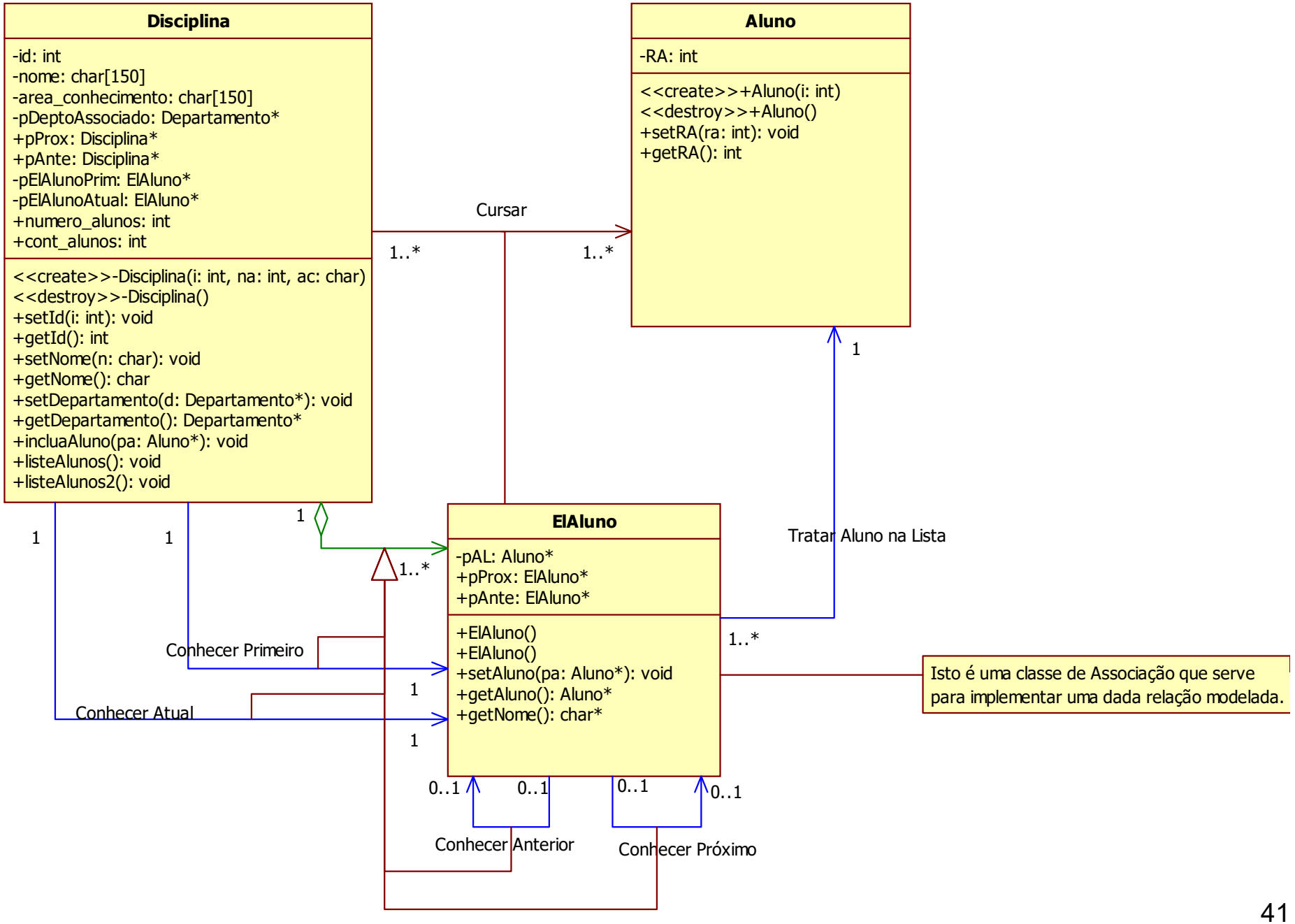
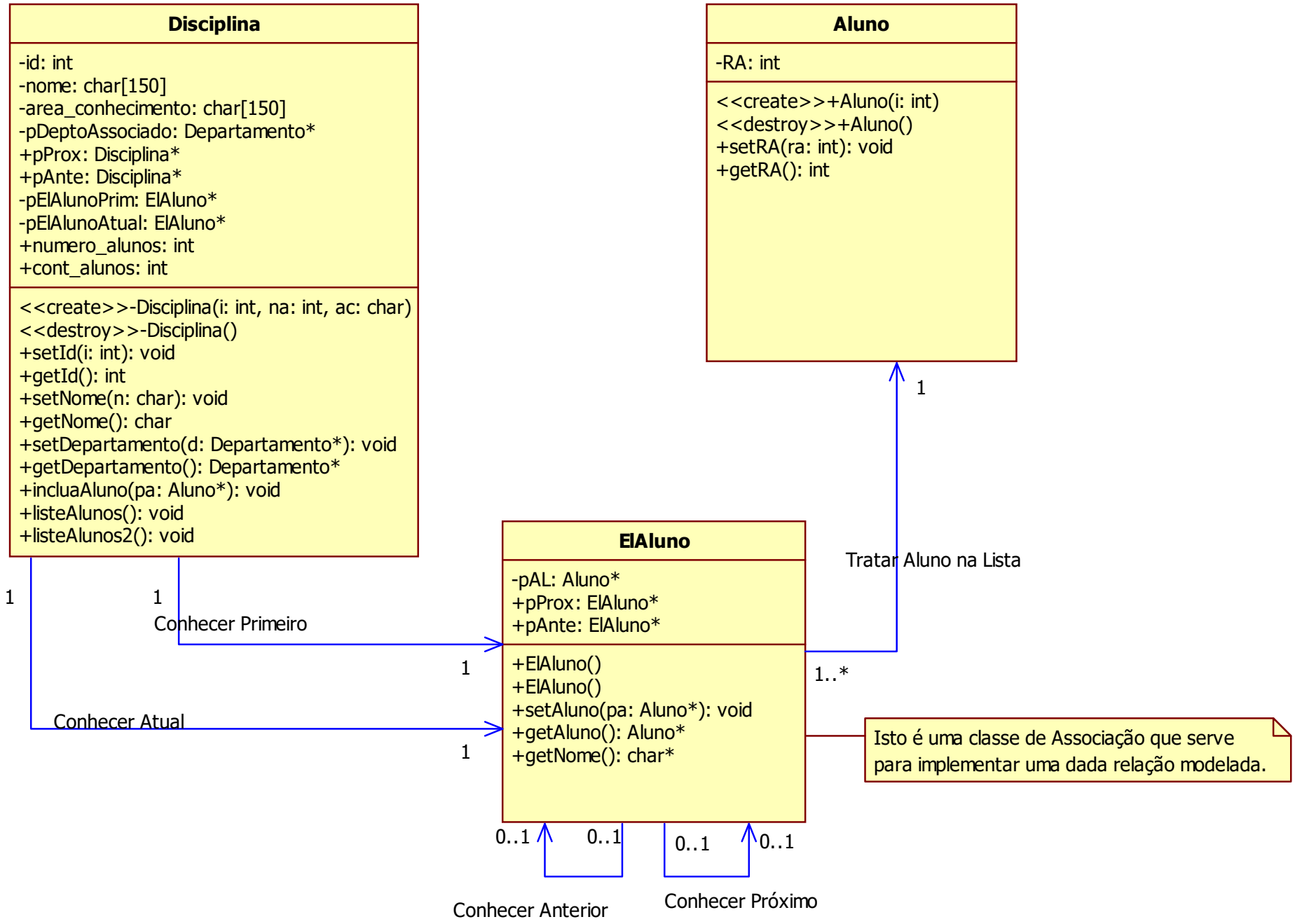

Disciplina
-id: int
-nome: char[150]
-area_conhecimento: char[150]
-pDeptoAssociado: Departamento*
+pProx: Disciplina*
+pAnte: Disciplina*
-pElAlunoPrim: ElAluno*
-pElAlunoAtual: ElAluno*
+numero_alunos: int
+cont_alunos: int
<<create>>-Disciplina(i: int, na: int, ac: char)
<<destroy>>-Disciplina()
+setId(i: int): void
+getId(): int
+setNome(n: char): void
+getNome(): char
+setDepartamento(d: Departamento*): void
+getDepartamento(): Departamento*
+incluaAluno(pa: Aluno*): void
+listeAlunos(): void
+listeAlunos2(): void
Aluno
-RA: int
<<create>>+Aluno(i: int)
<<destroy>>+Aluno()
+setRA(ra: int): void
+getRA(): int
Cursar
1..*
1..*
1
ElAluno
-pAL: Aluno*
+pProx: ElAluno*
+pAnte: ElAluno*
+ElAluno()
+ElAluno()
+setAluno(pa: Aluno*): void
+getAluno(): Aluno*
+getNome(): char*
Tratar Aluno na Lista
1..*
1
1
Conhecer Primeiro
1
Conhecer Atual
1
Isto é uma classe de Associação que serve para implementar uma dada relação modelada.
0..1
0..1
Conhecer Anterior
0..1
0..1
Conhecer Próximo

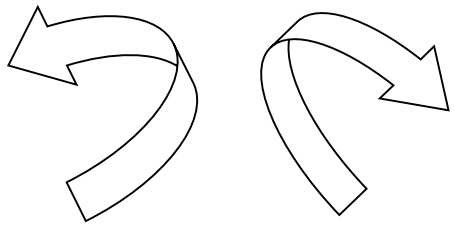
Diagrama de Classes – Projeto – Classe de Associação




Disciplina
-id: int
-nome: char[150]
-area_conhecimento: char[150]
-pDeptoAssociado: Departamento*
+pProx: Disciplina*
+pAnte: Disciplina*
-pElAlunoPrim: ElAluno*
-pElAlunoAtual: ElAluno*
+numero_alunos: int
+cont_alunos: int
<<create>>-Disciplina(i: int, na: int, ac: char)
<<destroy>>-Disciplina()
+setId(i: int): void
+getId(): int
+setNome(n: char): void
+getNome(): char
+setDepartamento(d: Departamento*): void
+getDepartamento(): Departamento*
+incluaAluno(pa: Aluno*): void
+listeAlunos(): void
+listeAlunos2(): void
Aluno
-RA: int
<<create>>+Aluno(i: int)
<<destroy>>+Aluno()
+setRA(ra: int): void
+getRA(): int
1
Tratar Aluno na Lista
ElAluno
-pAL: Aluno*
+pProx: ElAluno*
+pAnte: ElAluno*
+ElAluno()
+ElAluno()
+setAluno(pa: Aluno*): void
+getAluno(): Aluno*
+getNome(): char*
1..*
1
1
Conhecer Primeiro
1
Conhecer Atual
1
Isto é uma classe de Associação que serve para implementar uma dada relação modelada.
0..1
0..1
Conhecer Anterior
0..1
0..1
Conhecer Próximo

```
#ifndef _LALUNO_H_
#define _LALUNO_H_
#include "Aluno.h"
class ElAluno
{
 private:
    Aluno* pAl;
 public:
    ElAluno ( );
    ~ElAluno ( );
    ElAluno *pProx;
    ElAluno *pAnte;
    void setAluno( Aluno* pa );
    Aluno* getAluno ( );
    char* getNome ( );
};
#endif
```

```
#include "LAluno.h"
#include <stdio.h>
// ...

void ElAluno::setAluno ( Aluno *pa )
{
   pAl = pa;
}

Aluno* ElAluno::getAluno ( )
{
   return pAl;
}

char* ElAluno::getNome ( )
{
 return pAl->getNome ( );
}
```



A lista não será tratada na Classe *Aluno*, mas sim em uma outra classe relacionada.

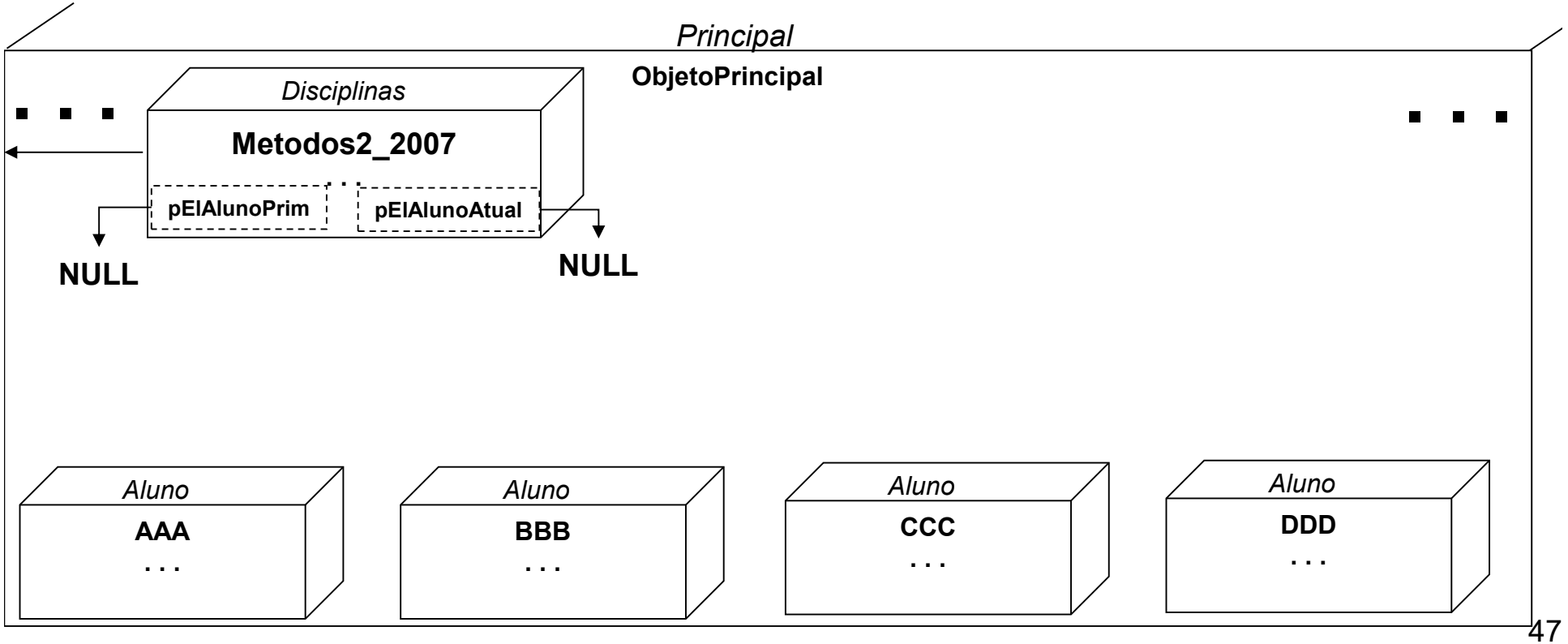
```
#ifndef _ALUNO_H_
#define _ALUNO_H_
#include “Pessoa.h"

class Departamento;

class Aluno : public Pessoa
{
  private:
    int RA;

  public:

    Aluno ( );
    ~Aluno ( );

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


};
#endif
```

```
#ifndef _DISCIPLINA_H_
#define _DISCIPLINA_H_
#include “ElAluno.h“
#include “Departamento.h”

class Disciplina
{
private:
    int id;
    char nome [ 150 ];
    char area_conhecimento [ 150 ];
    int numero_alunos;
    int cont_alunos;
    Departamento* pDeptoAssociado;

    ElAluno *pElAlunoPrim;
    ElAluno *pElAlunoAtual;

  public:
    Disciplina ( int na = 45, char* ac = "" );
    ~Disciplina ( );

     ...

    void incluaAluno ( Aluno* pa );
    void listeAlunos ( );
    void listeAlunos2 ( );
};
#endif
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{
   // Aqui é criado um ponteiro para LAluno
   ElAluno* paux = NULL;
   // Aqui é criado um objeto LAluno, sendo seu endereço armazenado em aux
   paux = new ElAluno ( );

   // Aqui recebe uma cópia do objeto interm.
   paux->setAluno ( pa );


  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {

     if ( pElAlunoPrim == NULL )
     {
        pElAlunoPrim   = paux;
        pElAlunoAtual  = paux;
     }
     else
     {
       pElAlunoAtual->pProx = paux;
       paux->pAnte          = pElAlunoAtual;
       pElAlunoAtual        =  paux;
     }
     cont_alunos++;
  }
  else
  {
      printf ("Aluno não incluído. Turma já lotada em %i alunos \n", numero_alunos );
  }

}
```

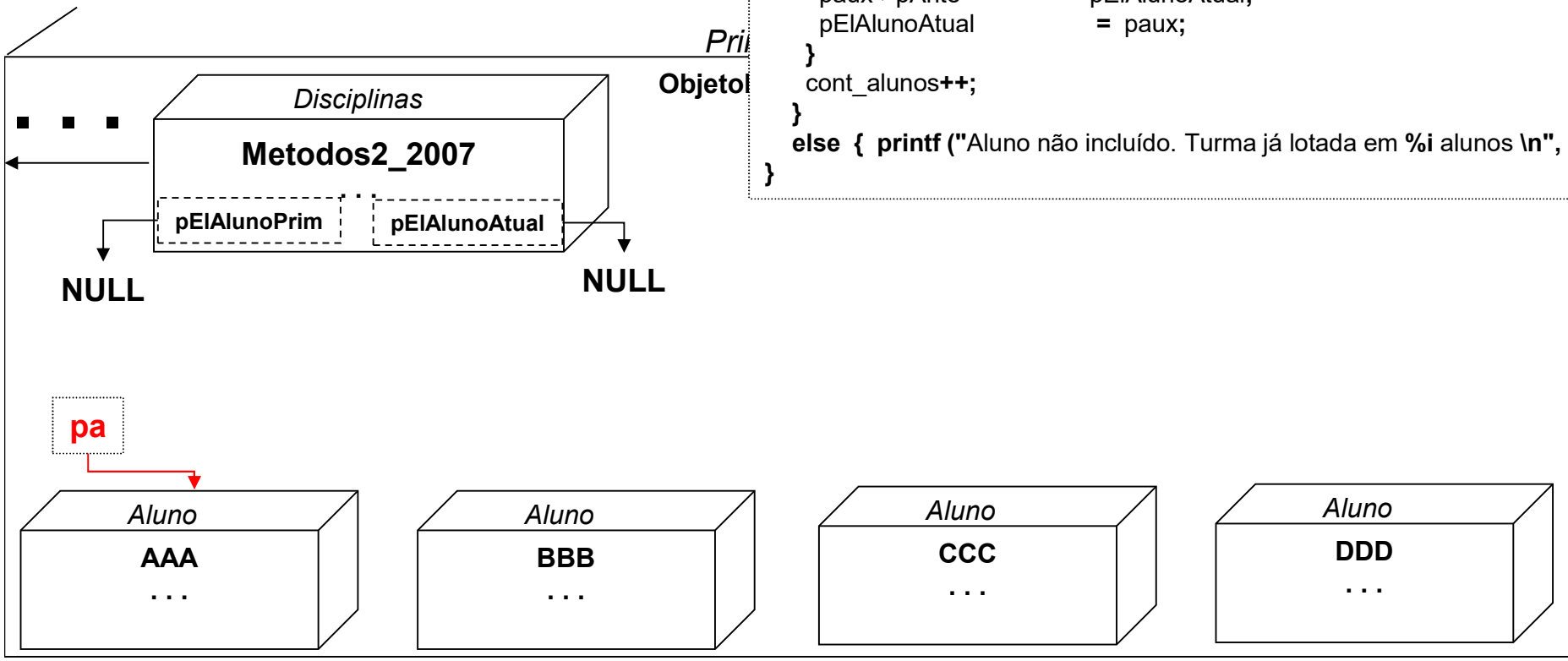
```
void Disciplina::listeAlunos()
{
   ElAluno* paux;
   paux = pElAlunoPrim;

   while ( paux != NULL )
   {
       printf(" Aluno %s matriculado na Disciplina %s \n", paux->getNome(), nome);
       paux = paux->pProx;
   }
}


void Disciplina::listeAlunos2 ( )
{
   ElAluno* pAux;
   pAux = pElAlunoAtual;

   while ( paux != NULL )
   {
     printf (" Aluno %s matriculado na Disciplina %s \n", paux->getNome(), nome);
     paux = paux->pAnte;
   }

}
```

```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```



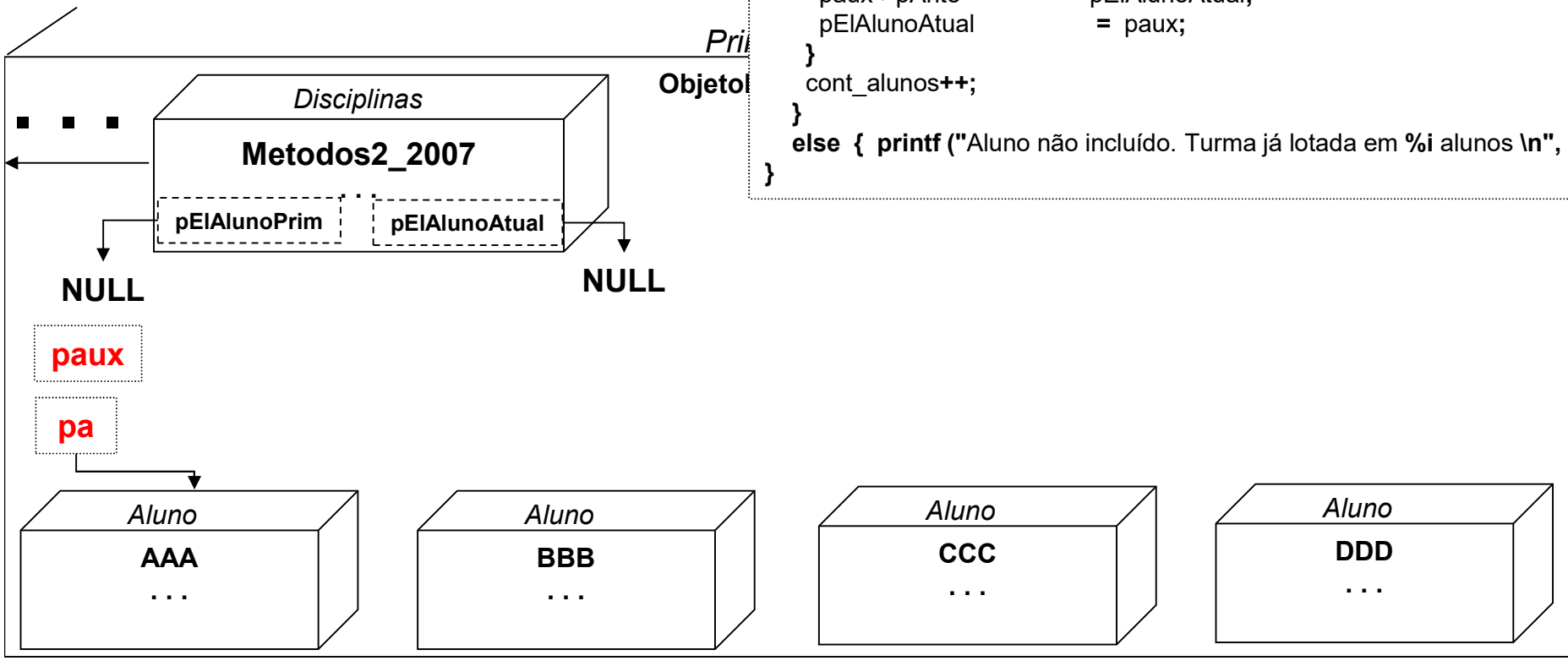
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento       ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento       ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007. incluaAluno ( &BBB );
   Metodos2_2007. incluaAluno ( &CCC );
   Metodos2_2007. incluaAluno ( &DDD );
   Metodos2_2007. incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina:: incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



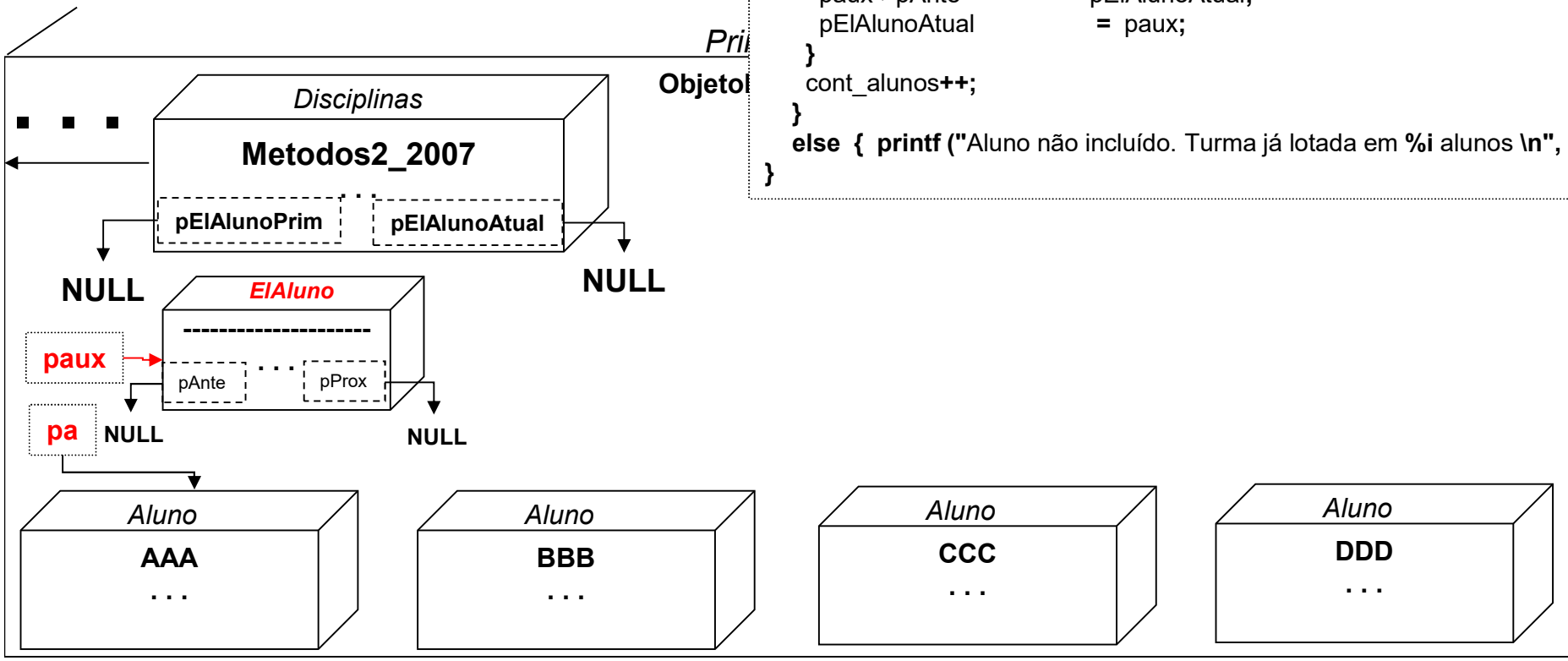
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



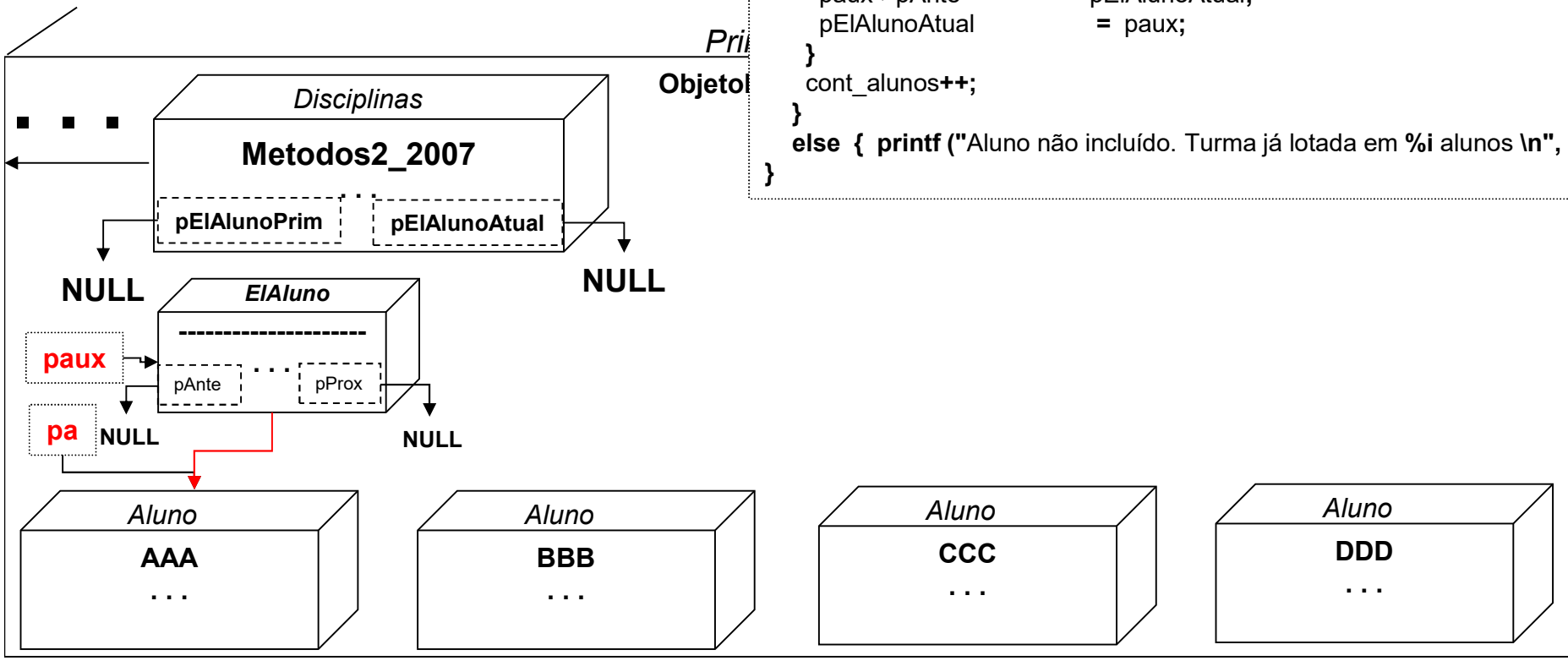
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



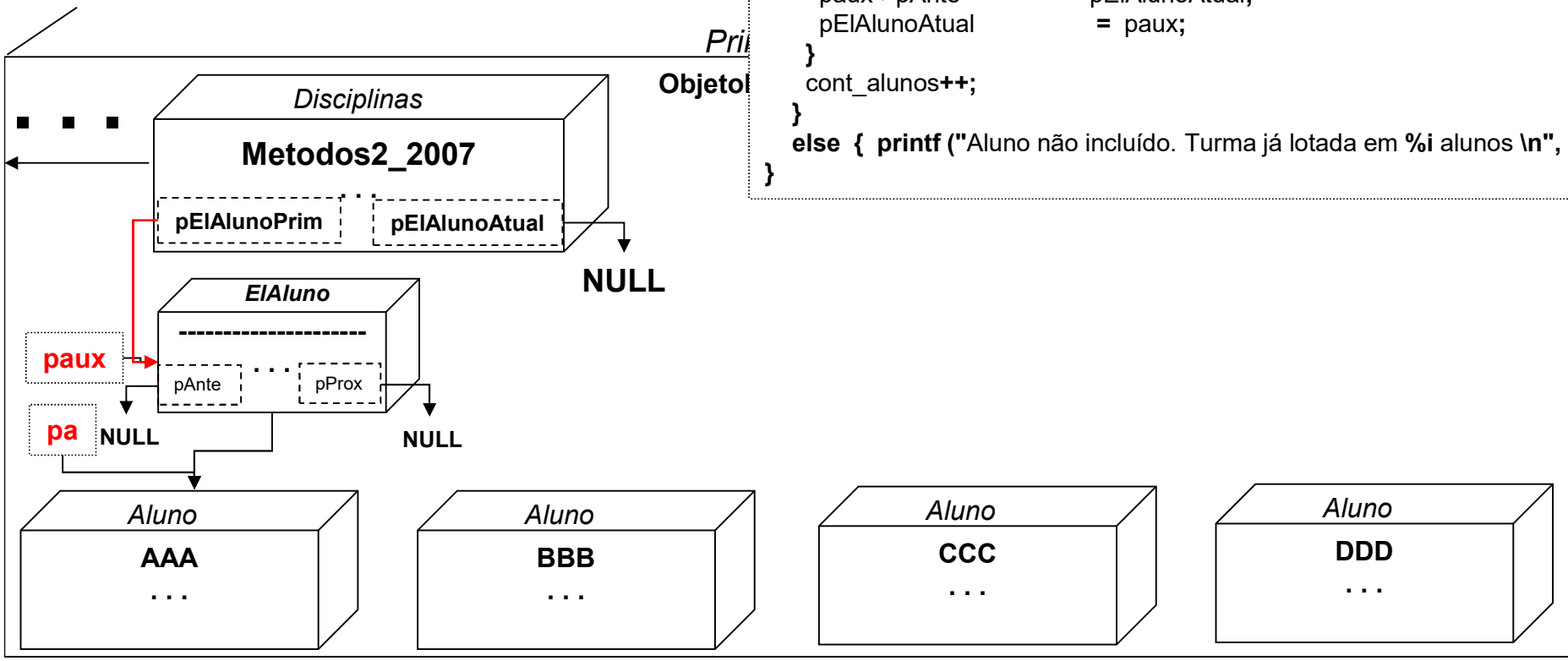
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



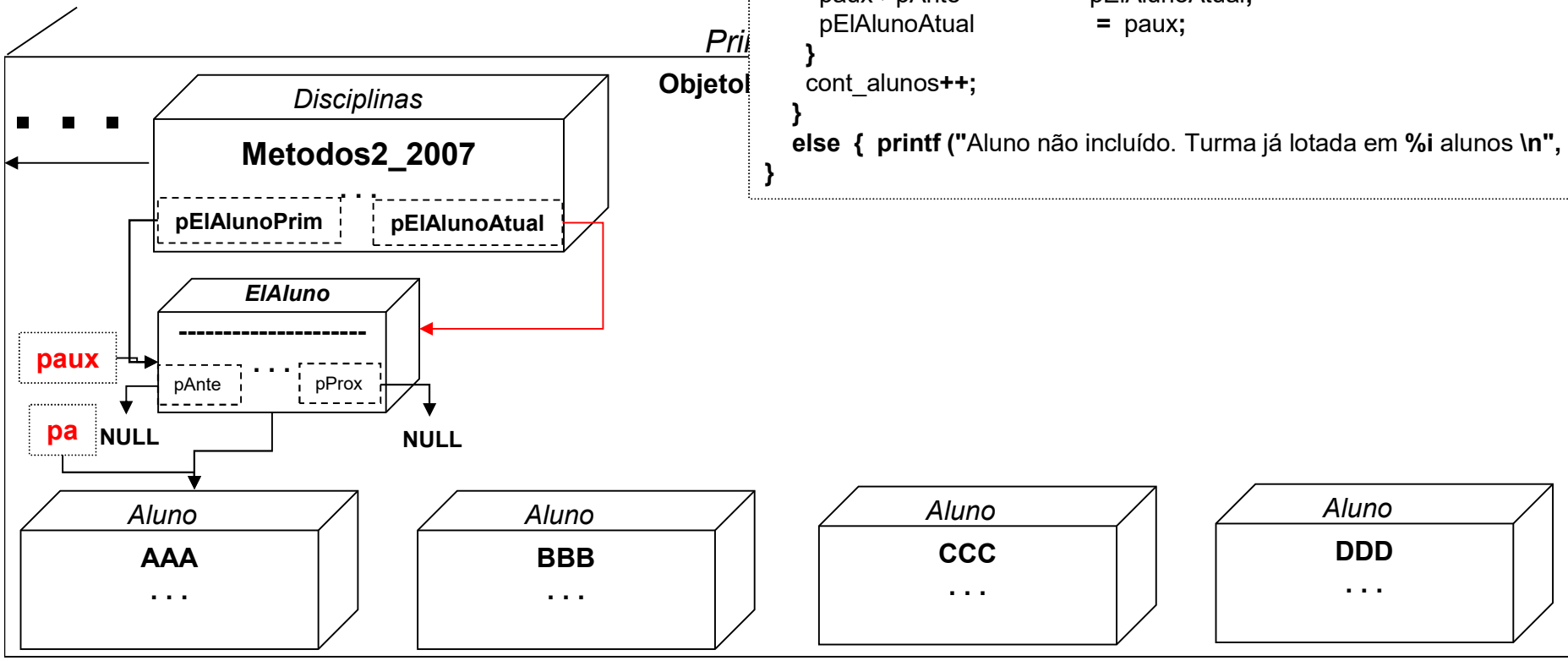
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



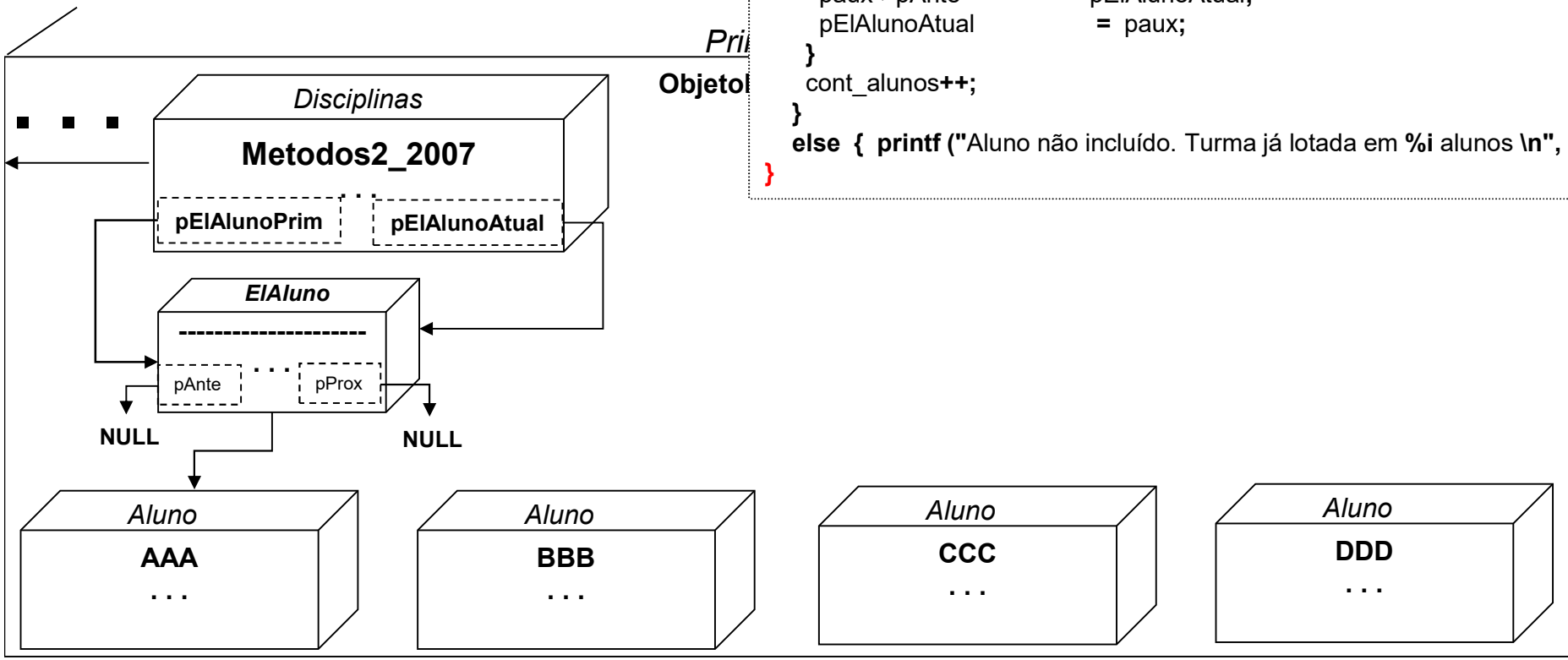
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```

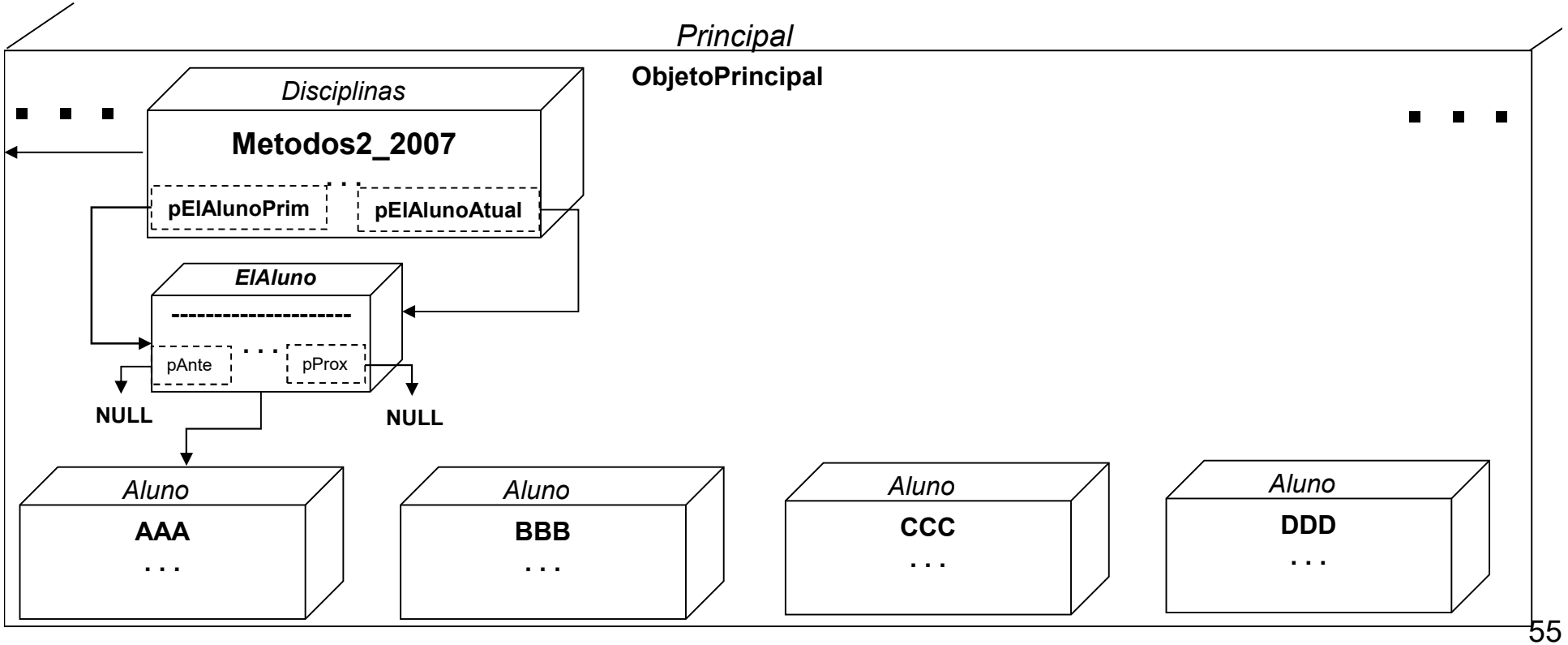


```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```

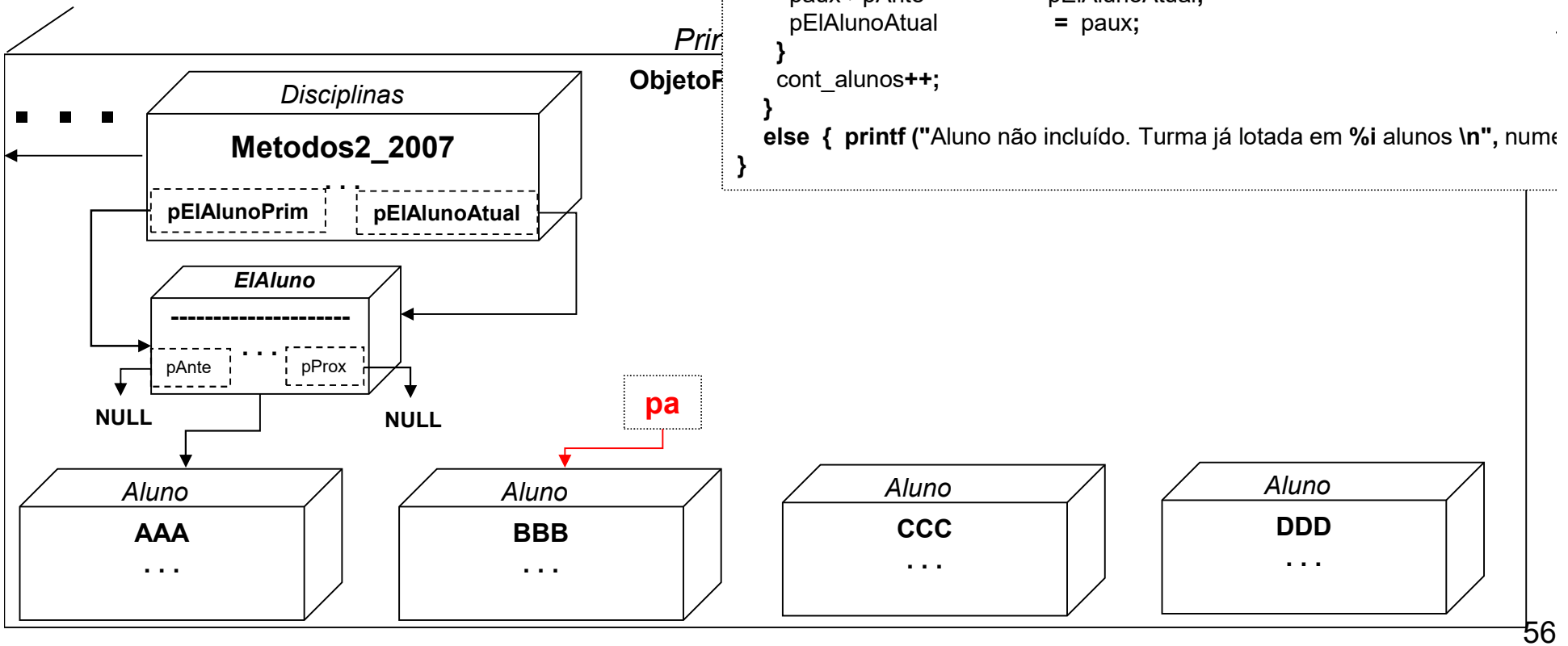


```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```



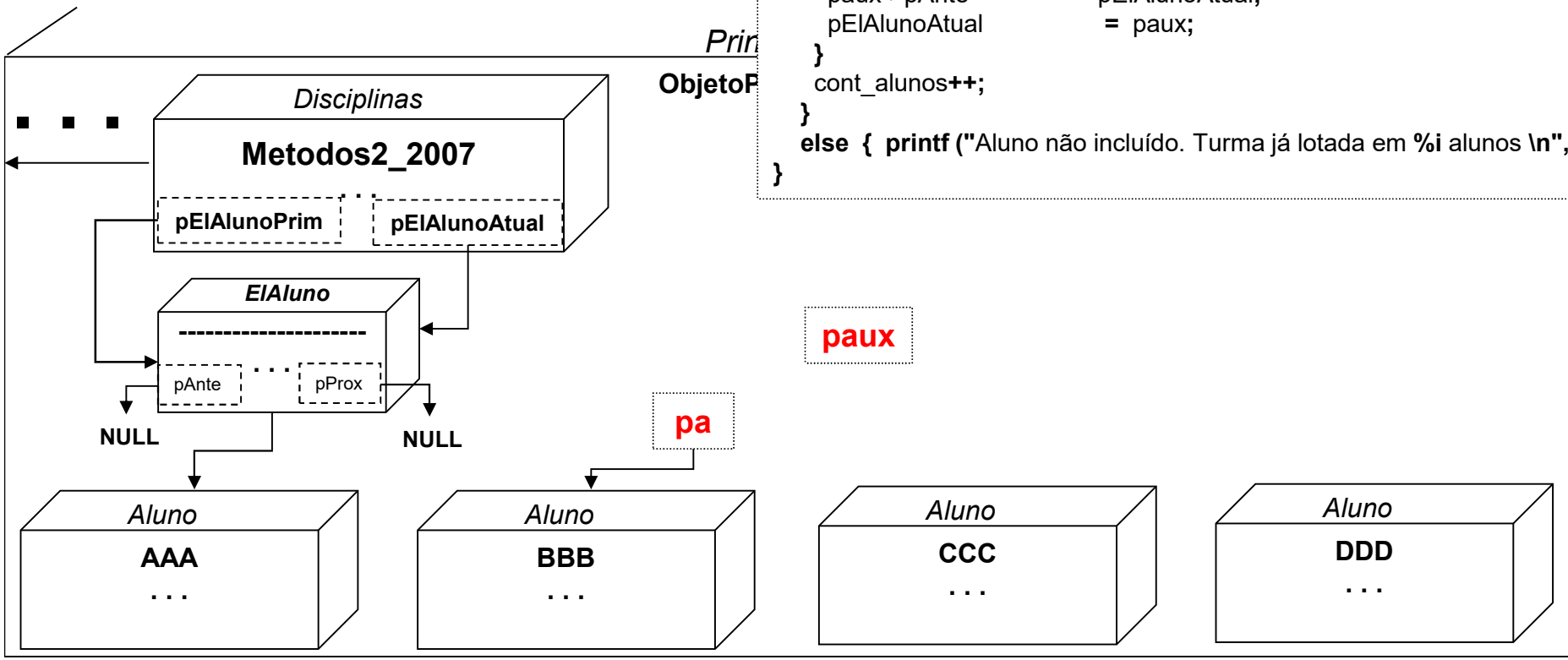
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento       ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento       ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



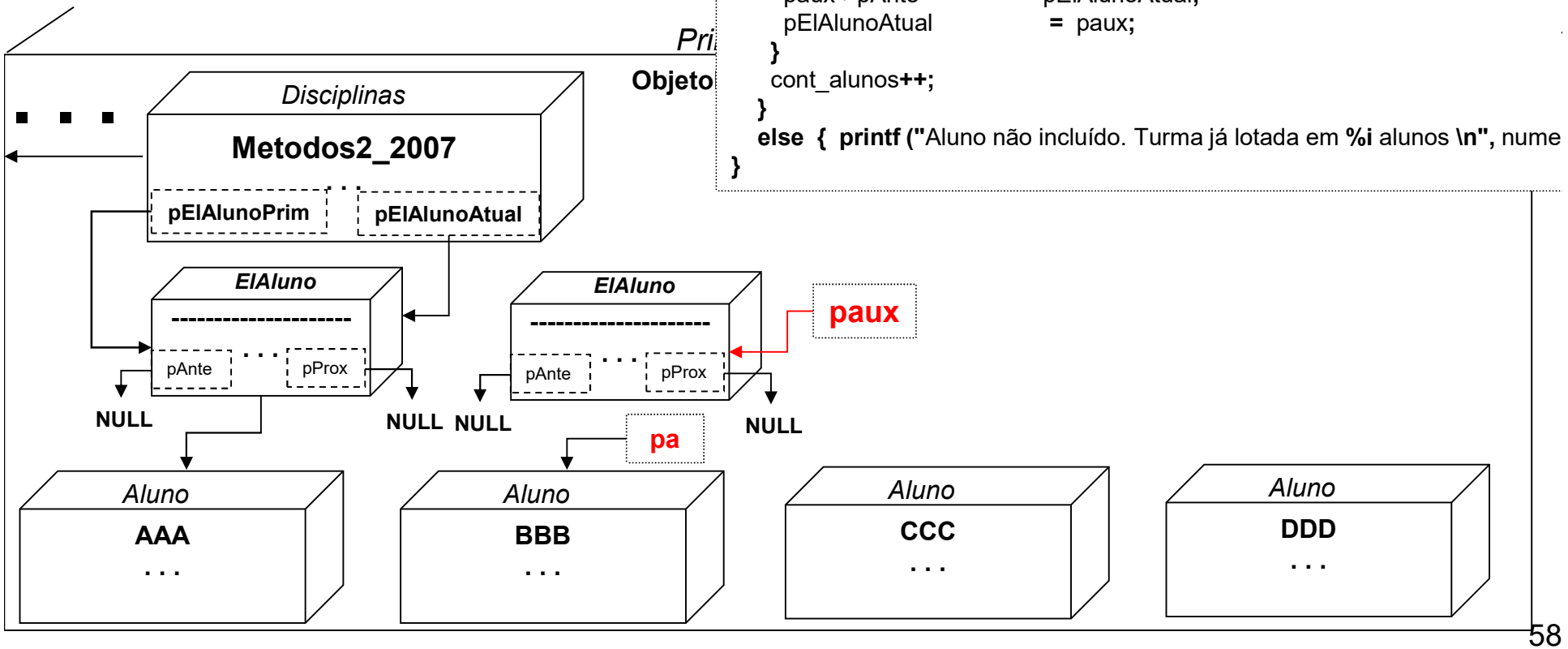
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
      pElAlunoAtual->pProx = paux;
      paux->pAnte          = pElAlunoAtual;
      pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



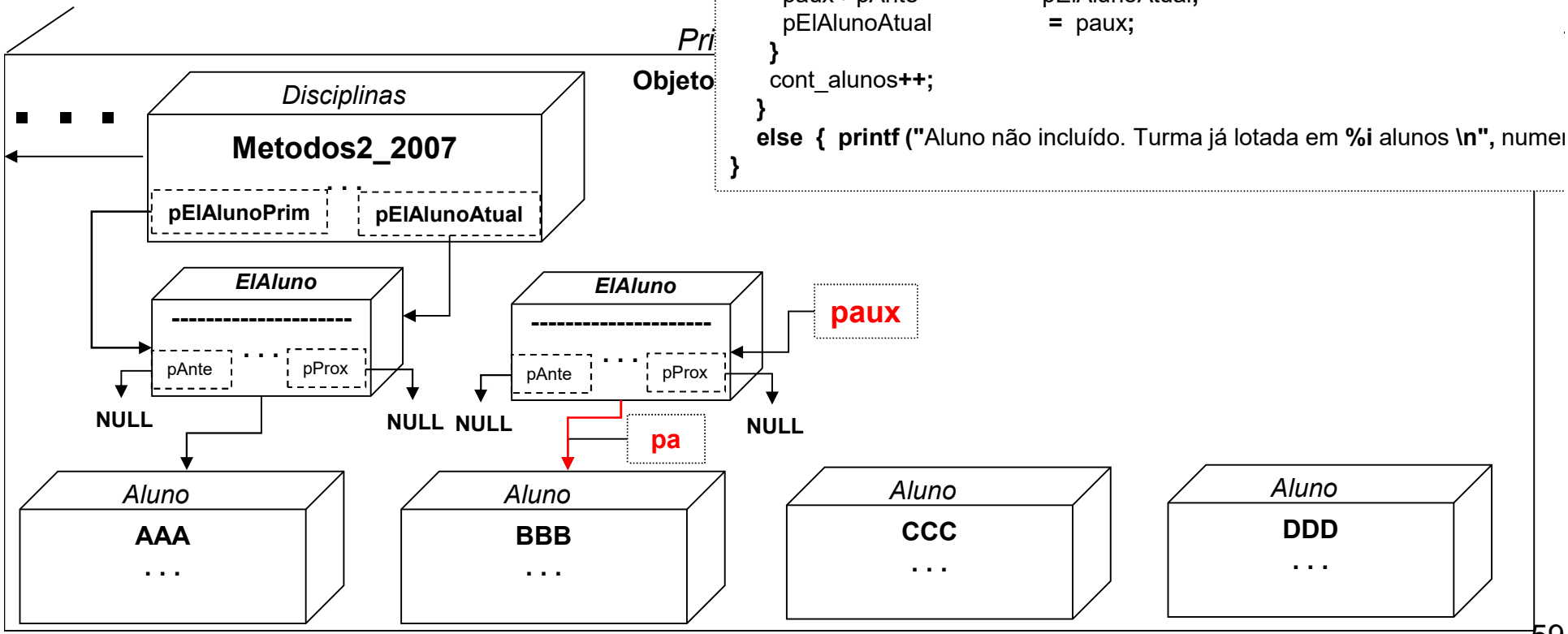
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento       ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento       ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
      pElAlunoAtual->pProx = paux;
      paux->pAnte          = pElAlunoAtual;
      pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", nume
}
```



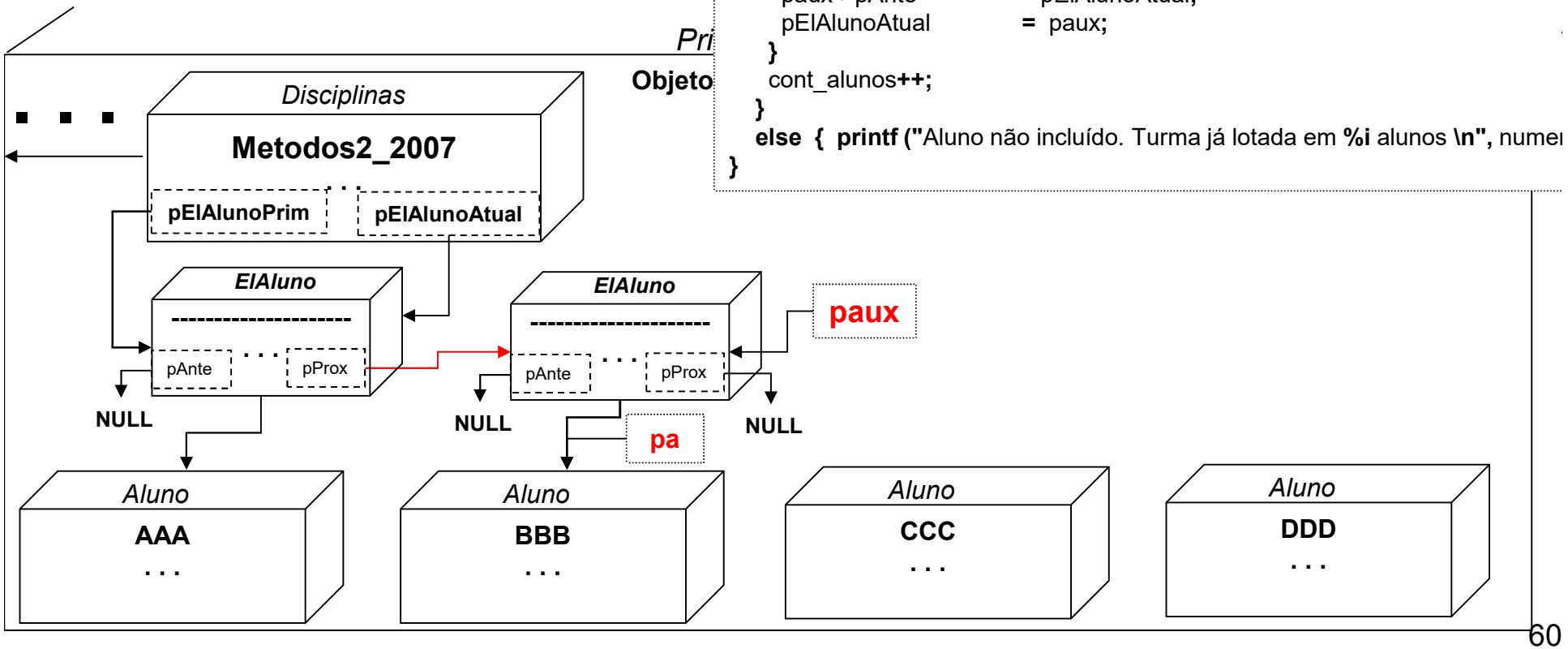
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento       ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento       ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina:: incluaAluno( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", numer
}
```



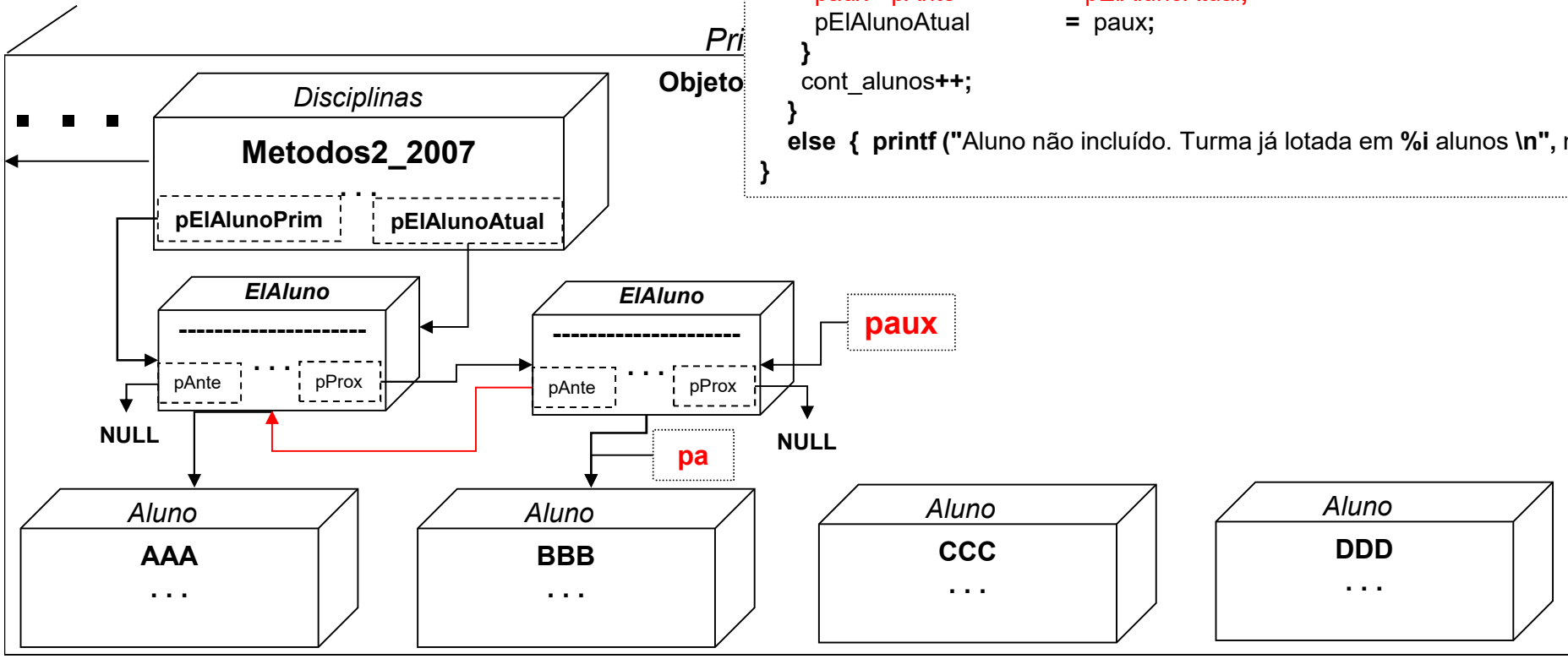
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", numer
}
```



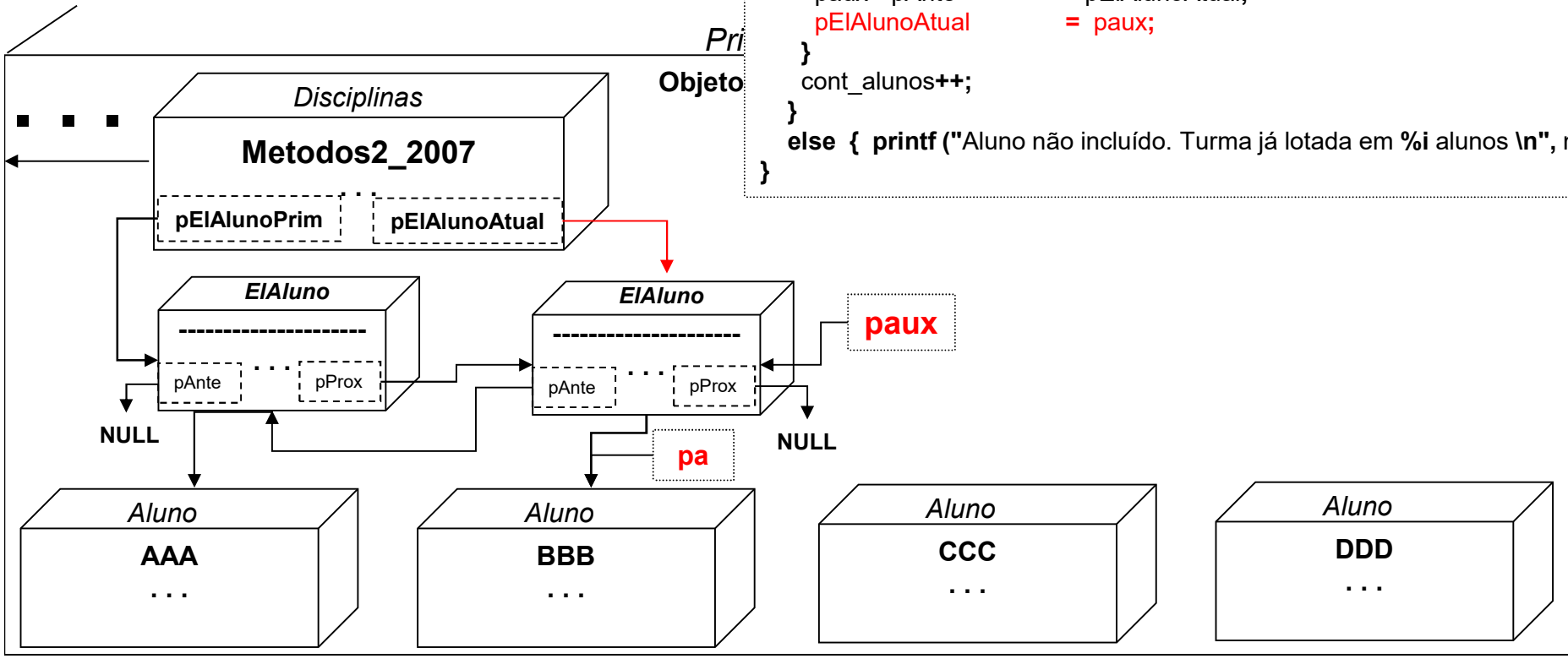
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
      pElAlunoAtual->pProx = paux;
      paux->pAnte          = pElAlunoAtual;
      pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", numer
}
```



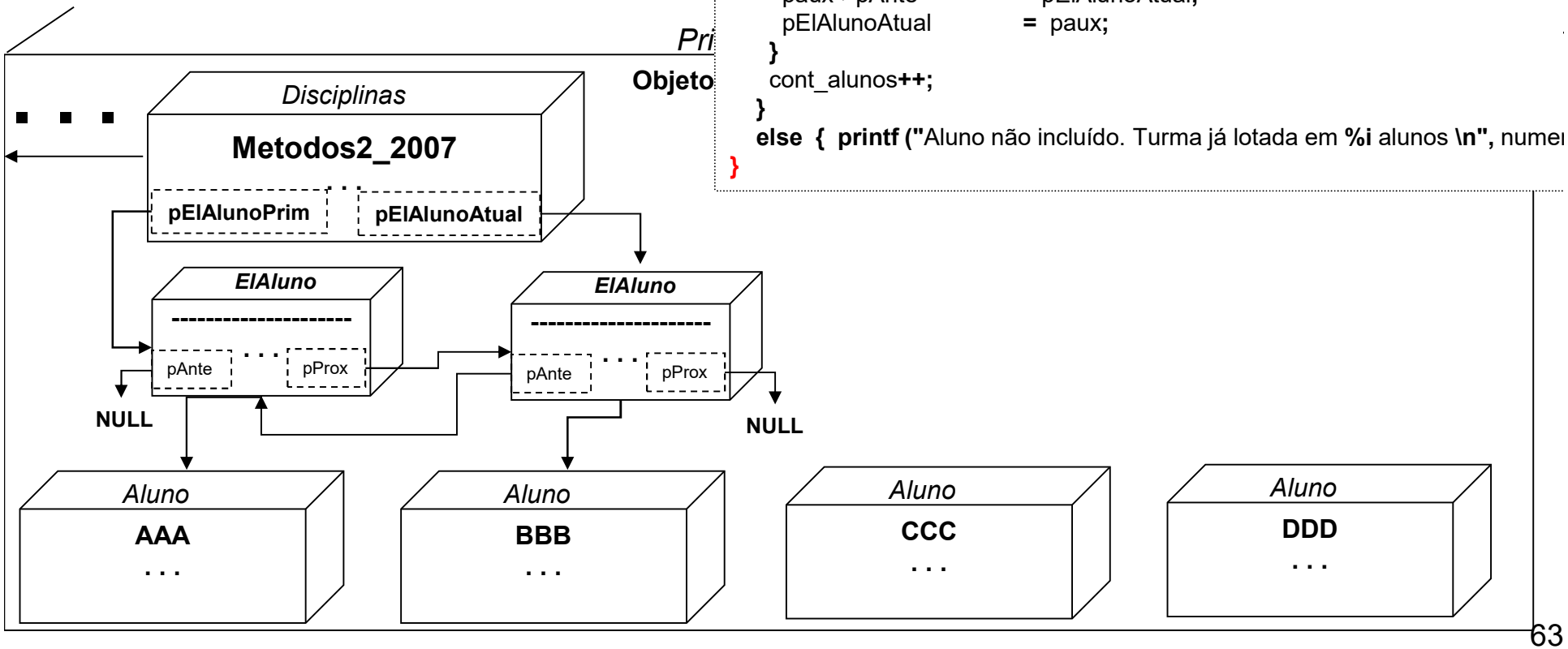
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
      pElAlunoAtual->pProx = paux;
      paux->pAnte          = pElAlunoAtual;
      pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", numer
}
```



```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```

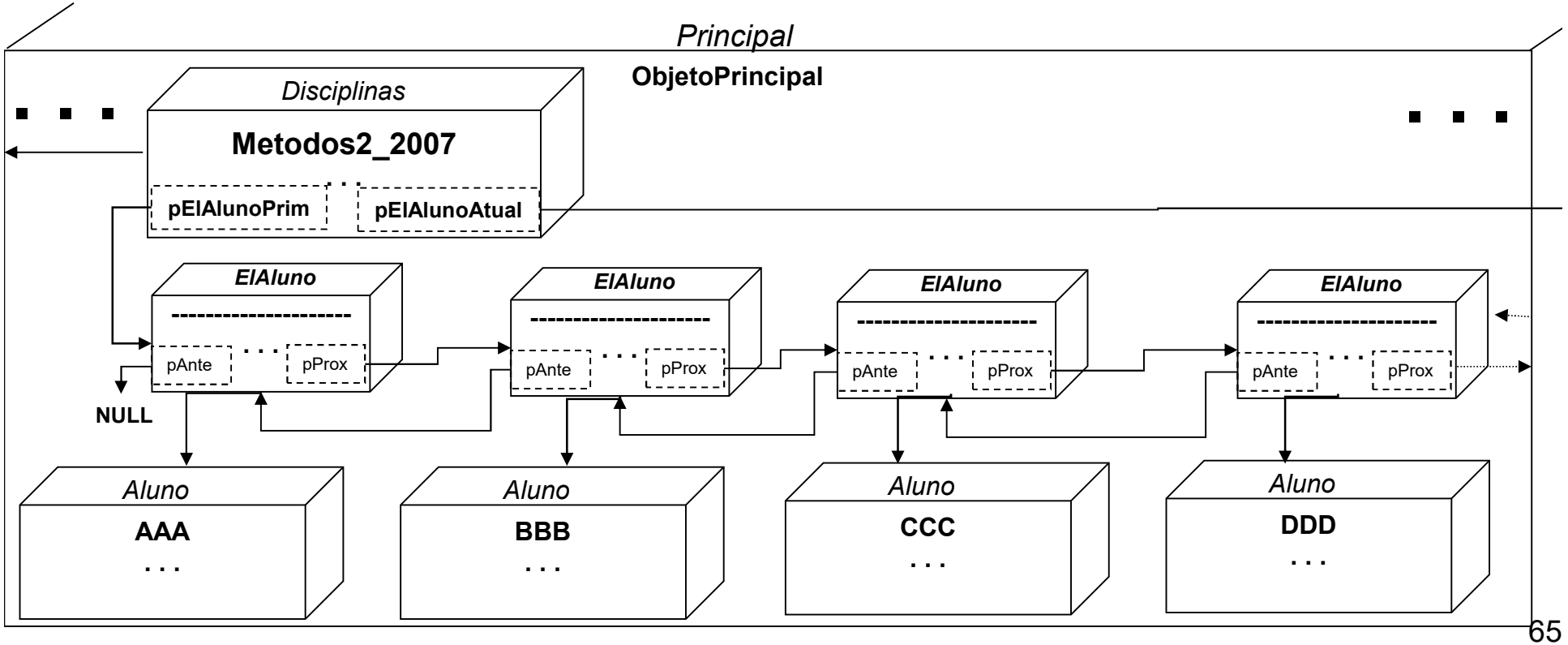
```
void Disciplina::incluaAluno ( Aluno* pa )
{ // Aqui abaixo é criado um ponteiro para LAluno
  ElAluno* paux;
  // Aqui abaixo é criado um objeto LAluno, sendo seu endereço armazenado e
  paux = new ElAluno ( );
  // Aqui abaixo recebe uma cópia do objeto interm.
  paux->setAluno ( pa );
  if ( ( cont_alunos < numero_alunos ) && ( pa != NULL) )
  {
    if ( pElAlunoPrim == NULL )
    {
      pElAlunoPrim   = paux;
      pElAlunoAtual  = paux;
    }
    else
    {
     pElAlunoAtual->pProx = paux;
     paux->pAnte          = pElAlunoAtual;
     pElAlunoAtual        =  paux;
    }
    cont_alunos++;
  }
  else  {  printf ("Aluno não incluído. Turma já lotada em %i alunos \n", numer
}
```



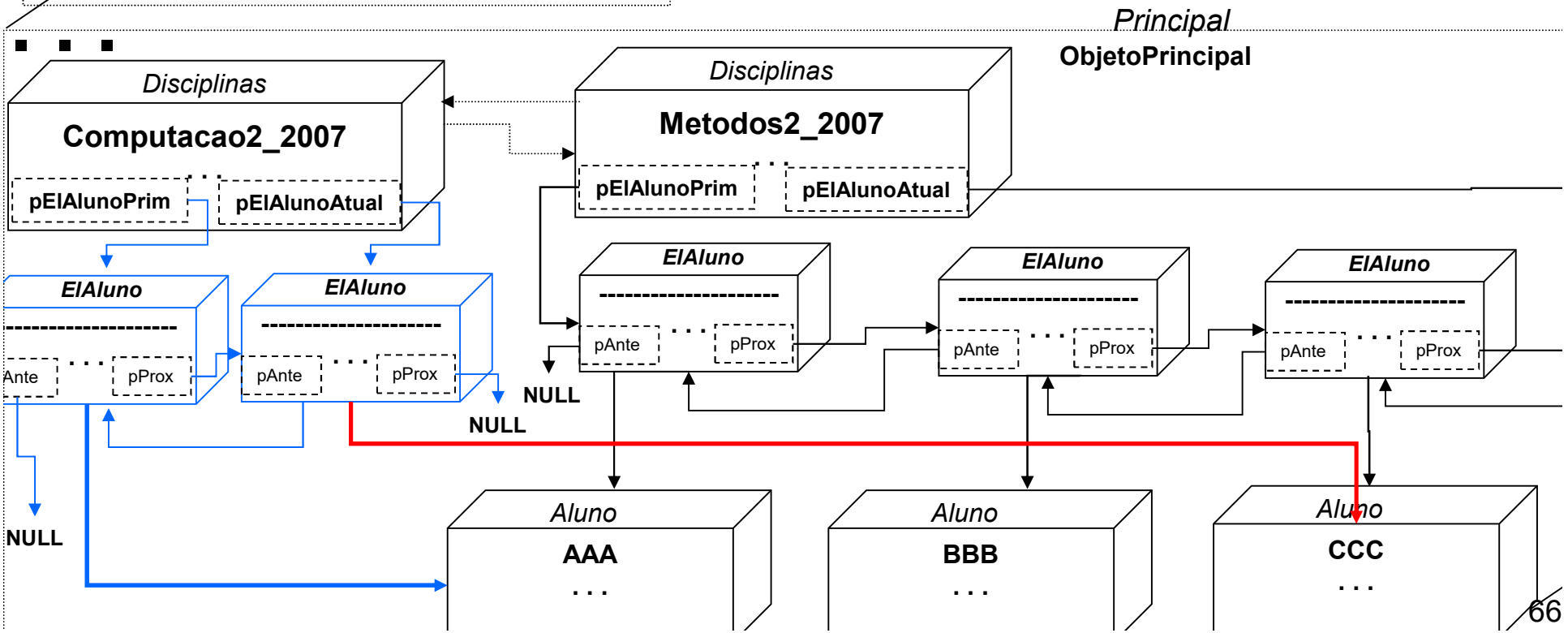
Continue a “simulação”

utilizando o “diagrama de cubos”.

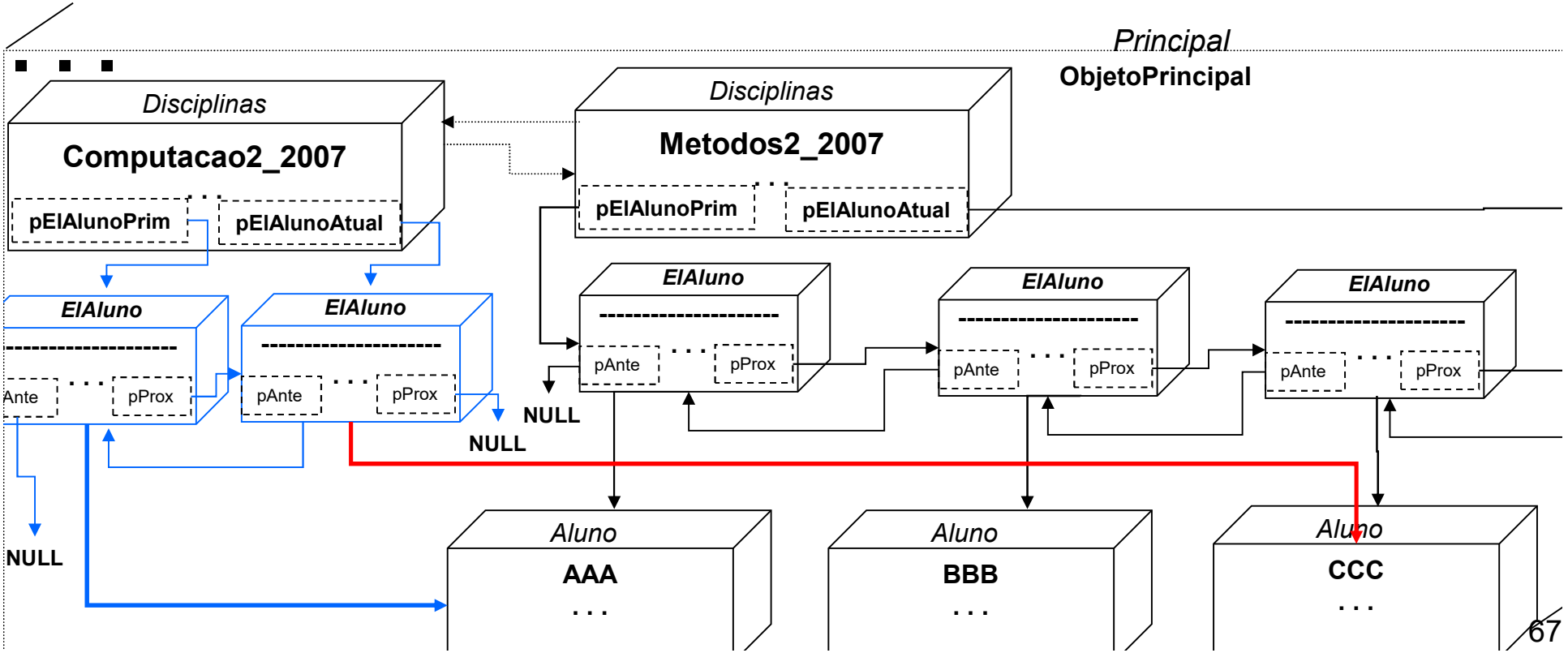
```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN );
   Introd_Alg_2007.setDepartamento   ( &DAELN );
   Computacao2_2007.setDepartamento  ( &DAELN );
   Metodos2_2007.setDepartamento     ( &DAELN );

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );
   . . .
}
```



```
void Principal::InicializaDisciplinas ( )
{
   . . .
   Computacao1_2006.setDepartamento  ( &DAELN ) ;
   Introd_Alg_2007.setDepartamento       ( &DAELN ) ;
   Computacao2_2007.setDepartamento  ( &DAELN ) ;
   Metodos2_2007.setDepartamento       ( &DAELN ) ;

   Metodos2_2007.incluaAluno ( &AAA );
   Metodos2_2007.incluaAluno ( &BBB );
   Metodos2_2007.incluaAluno ( &CCC );
   Metodos2_2007.incluaAluno ( &DDD );
   Metodos2_2007.incluaAluno ( &EEE );

   Computacao2_2007.incluaAluno ( &AAA );
   Computacao2_2007.incluaAluno ( &CCC );
   Computacao2_2007.incluaAluno ( &FFF );
}
```



- Está solução é apropriada, pois cada vez que um *aluno* é inserido em uma lista de *disciplina*, é também criado um *elemento_aluno* para tratá-lo naquela lista em específico.

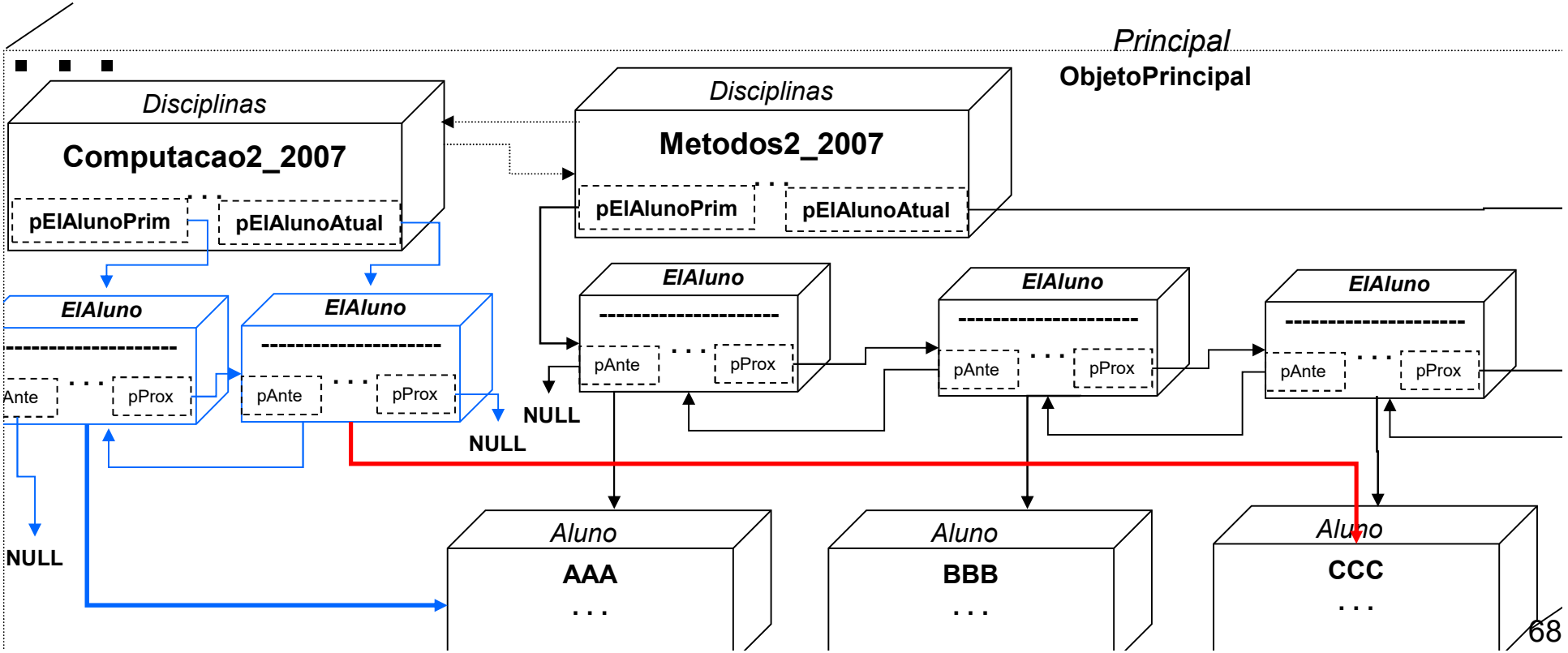


Exercício

- Simule a execução do código

```
void Principal::ListeAlunosDisc ( )
{
   Metodos2_2007.listeAlunos ( );
   printf("\n");
   Metodos2_2007.listeAlunos2 ( );
   printf("\n");

   . . .
}
```

```
void Disciplina::listeAlunos()
{
   ElAluno* paux;
   paux = pElAlunoPrim;
   while ( paux != NULL )
   {
      printf(" Aluno %s matriculado na Disciplina %s \n", paux->getNome(), nome);
      paux = paux->pProx;
   }
}
```



```
void Principal::ListeAlunosDisc ( )
{
   Metodos2_2007.listeAlunos ( );
   printf("\n");
   Metodos2_2007.listeAlunos2 ( );
   printf("\n");

   Computacao2_2007.listeAlunos ( );
   printf("\n");
   Computacao2_2007.listeAlunos2 ( );
   printf("\n");
}
```

```
void Principal::Executar ( )
{

   CalcIdadeProfs ( );
   UnivOndeProfsTrabalham ( );
   DepOndeProfsTrabalham ( );
   ConhecPessoa ( );
   ListeDiscDeptos ( );
   ListeAlunosDisc ( );

}
```

```
void Principal::Executar ( )
{
   CalcIdadeProfs ( );
   UnivOndeProfsTrabalham ( );
   DepOndeProfsTrabalham ( );
   ConhecPessoa ( );
   ListeDiscDeptos ( );
   ListeAlunosDisc ( );

   AAA.setNome ( "Teste" );
   printf ( "O novo nome de AAA é: %s \n", AAA.getNome ( ) );
   Computacao2_2007.listeAlunos ( );
}
```

OK!

Consideração

Uma solução de “clonagem” pode apresentar problemas de desincronização...

```
void Disciplina::incluaAluno (Aluno* pa)
{
   // Aqui é criado um ponteiro para LAluno
   ElAluno* paux;

   // Aqui é criado um objeto LAluno, sendo seu endereço armazenado em aux
   paux = new ElAluno ( );

   // Aqui recebe uma cópia do objeto a..    Isto é uma "clonagem"...
   Aluno* pa2 = new Aluno();
   *pa2 = *pa;
   paux->setAluno ( pa2 );

   if ( ( cont_alunos < numero_alunos ) && ( pa != NULL ) )
   {
      if ( pAlunoPrim = = NULL)
      {
         pAlunoPrim   = paux;
         pAlunoAtual  = paux;
      }
      else
      {
         pAlunoAtual->pProx   =  paux;
         paux->pAnte          =  pAlunoAtual;
         pAlunoAtual          =  paux;
      }
      cont_alunos++;
   }
   else
   {
      printf ( "Aluno não incluído. Turma já lotada em %i alunos \n", numero_alunos );
   }
}
```

```
void Principal::Executar ()
{
   CalcIdadeProfs ( );
   UnivOndeProfsTrabalham ( );
   DepOndeProfsTrabalham ( );
   ConhecPessoa ( );
   ListeDiscDeptos ( );
   ListeAlunosDisc ( );

   AAA.setNome ( "Teste“ );
   printf ( "O novo nome de AAA é: %s \n", AAA.getNome ( ) );
   Computacao2_2007.listeAlunos ( );
}
```

PROBLEMA!

Qual é o problema ?

Exercício 1

- No destrutor da classe Disciplina, liberar a memória alocada para lista de Alunos (ElAlunos”), liberando/destruindo cada objeto criado dinamicamente.

  - Obs.: usar o comando *delete... Por exemplo: delete* aux*;*

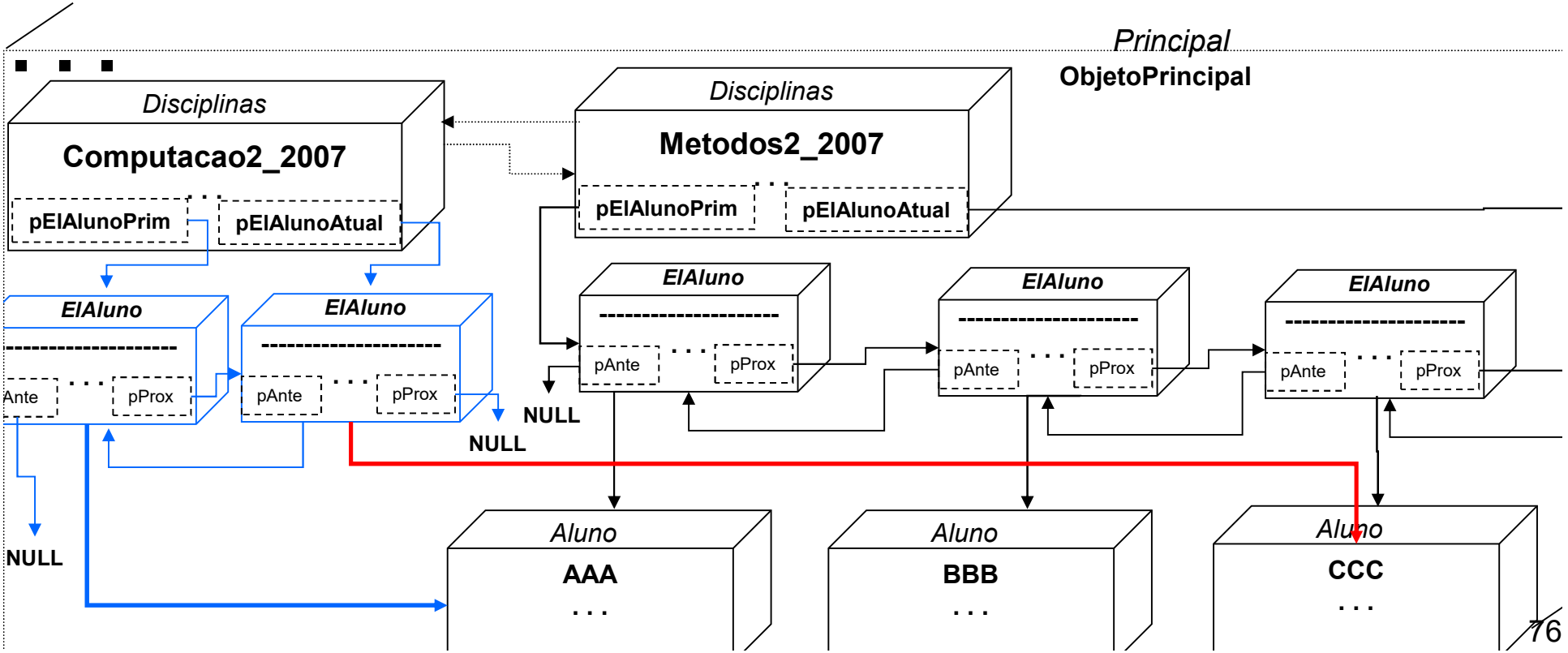
Solução

```
Disciplina::~Disciplina()
{
  ElAluno *paux1, *paux2;

  paux1 = pElAlunoPrim;

  while ( paux1 != NULL )
  {
      paux2 = paux1->pProx;
      delete ( paux1 );
      paux1 = paux2;
  }

  pDeptoAssociado        = NULL;
  pElAlunoPrim           = NULL;
  pElAlunoAtual          = NULL;
  pProx                  = NULL;
  pAnte                  = NULL;
}
```

Para entender melhor a solução, queira simulação em um diagrama de "cubos".

Simule a execução do código:

```
Disciplina::~Disciplina()
{
   ElAluno *paux1, *paux2;
   paux1 = pElAlunoPrim;

   while ( paux1 != NULL )
   {
        paux2 = paux1->pProx;
        delete ( paux1 );
        paux1 = paux2;
   }
   . . .
}
```



Outra solução

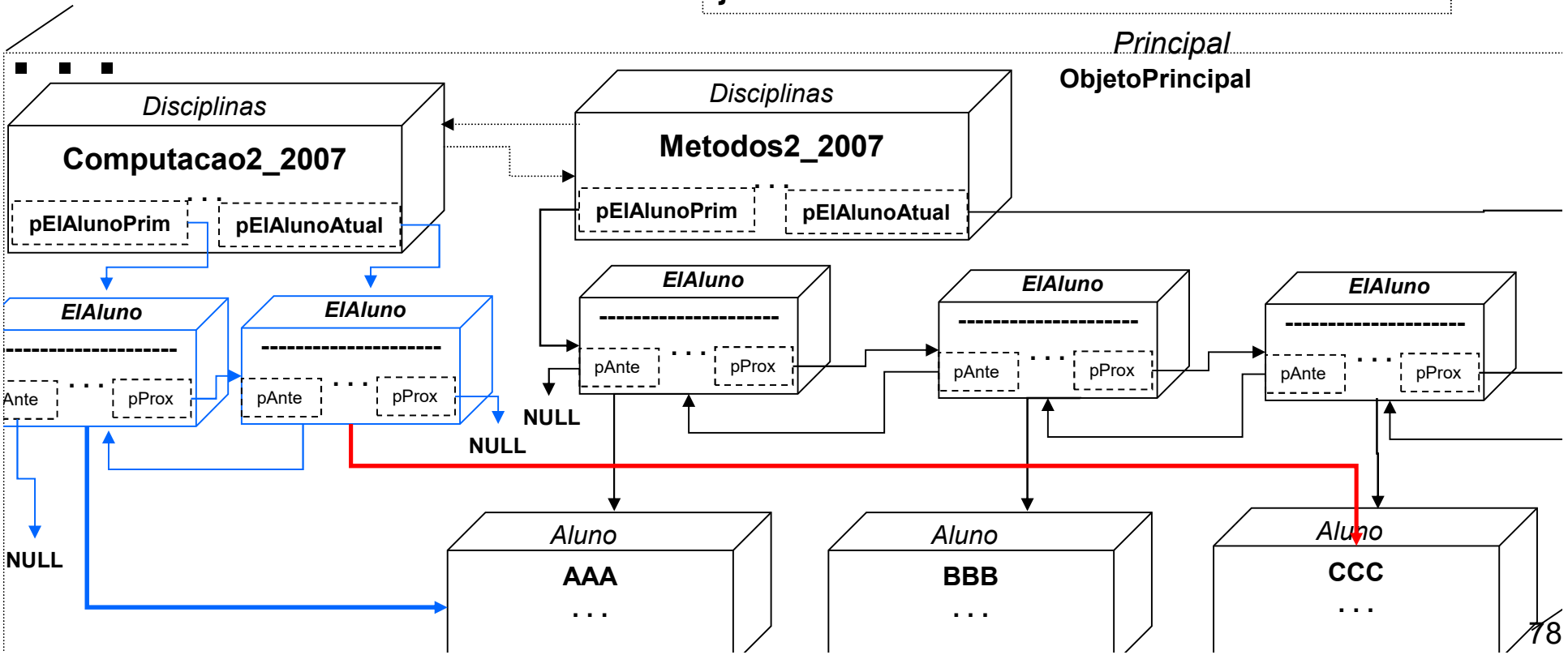
```
Disciplina::~Disciplina()
{
    ElAluno* paux;          // ponteiro para ElAluno
    paux = pElAlunoPrim;

    while ( NULL != pElAlunoPrim )
    {
        pElAlunoPrim = pElAlunoPrim->pProx;
        delete paux;
        paux = pElAlunoPrim;
    }

    pDeptoAssociado     = NULL;
    pElAlunoAtual       = NULL;
}
```

- No destrutor da classe Disciplina, liberar a memória alocada para lista de Alunos (“LAlunos”), liberando-destruindo cada objeto criado dinamicamente.

- Obs.: usar o comando *delete*... *Por exemplo:* *delete* aux*;*

Simule a execução do código

```
Disciplina::~Disciplina()
{
    ElAluno* paux;          // ponteiro para ElAluno
    paux = pElAlunoPrim;

    while ( NULL != pElAlunoPrim )
    {
        pElAlunoPrim = pElAlunoPrim->pProx;
        delete paux;
        paux = pElAlunoPrim;
    }
    pDeptoAssociado    = NULL;
    pElAlunoAtual      = NULL;
}
```



Exercício 2

- Cada Departamento deve ser capaz de armazenar uma lista de disciplinas.

- A classe *Disciplina*, entretanto, não deverá possuir um ponteiro para o Próximo. Isto deverá estar em uma classe associada chamada *ElDisciplina relacionada a Disciplina*...

Exercício 3

- Elaborar uma solução para o armazenar as notas (1ª parcial, 2ª parcial e final) e número de faltas de cada aluno em cada disciplina.