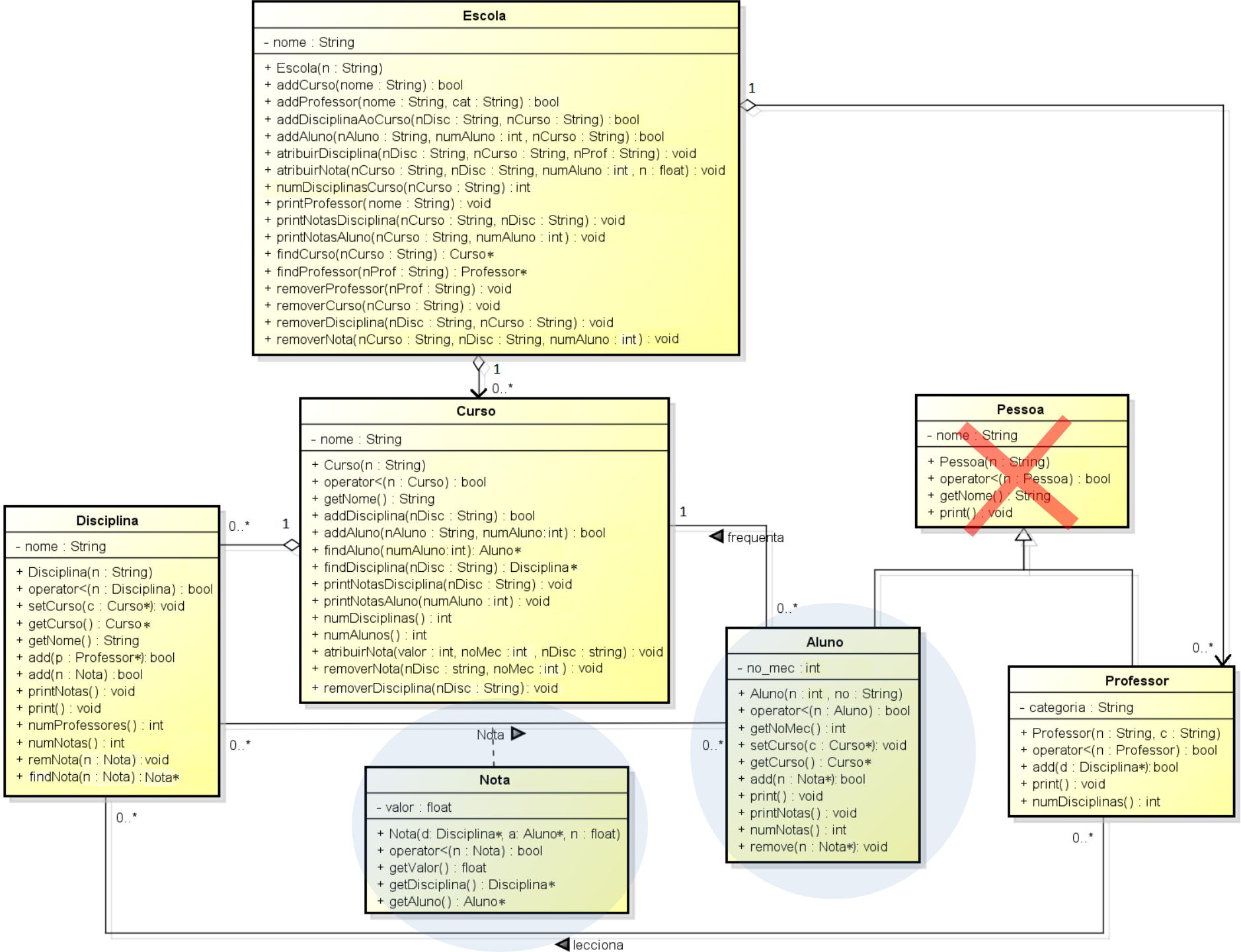
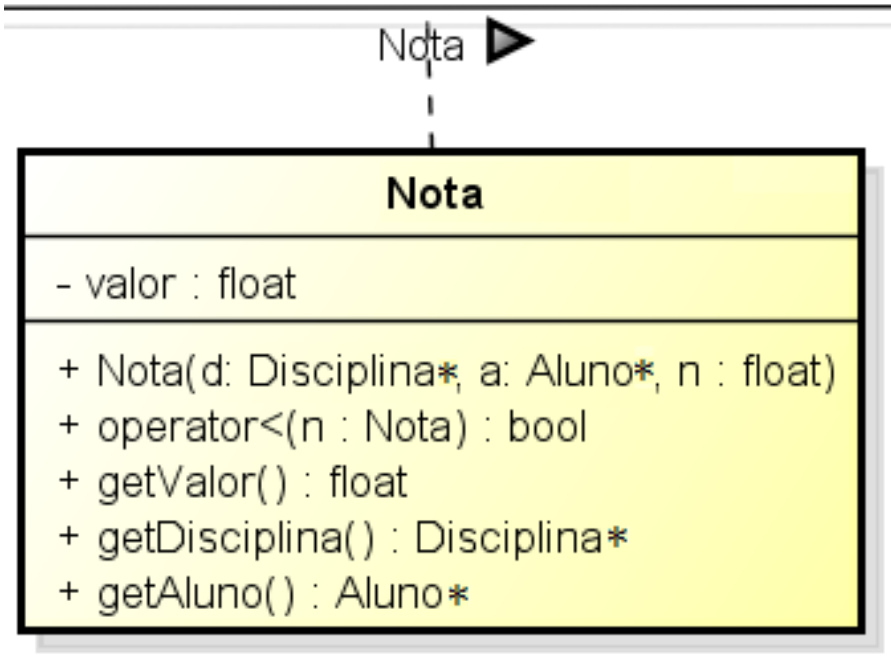
Exercício 21

Para o diagrama do exercício 7, implemente as classes Aluno e Nota e as operações de atribuição de uma nota a um aluno a uma dada disciplina.


Escola
- nome : String
+ Escola(n : String)
+ addCurso(nome : String) : bool
+ addProfessor(nome : String, cat : String) : bool
+ addDisciplinaAoCurso(nDisc : String, nCurso : String) : bool
+ addAluno(nAluno : String, numAluno : int , nCurso : String) : bool
+ atribuirDisciplina(nDisc : String, nCurso : String, nProf : String) : void
+ atribuirNota(nCurso : String, nDisc : String, numAluno : int , n : float) : void
+ numDisciplinasCurso(nCurso : String) : int
+ printProfessor(nome : String) : void
+ printNotasDisciplina(nCurso : String, nDisc : String) : void
+ printNotasAluno(nCurso : String, numAluno : int) : void
+ findCurso(nCurso : String) : Curso*
+ findProfessor(nProf : String) : Professor*
+ removerProfessor(nProf : String) : void
+ removerCurso(nCurso : String) : void
+ removerDisciplina(nDisc : String, nCurso : String) : void
+ removerNota(nCurso : String, nDisc : String, numAluno : int) : void
1
0..*
1
Curso
- nome : String
+ Curso(n : String)
+ operator<(n : Curso) : bool
+ getNome( ) : String
+ addDisciplina(nDisc : String) : bool
+ addAluno(nAluno : String, numAluno: int) : bool
+ findAluno(numAluno: int): Aluno*
+ findDisciplina(nDisc : String) : Disciplina*
+ printNotasDisciplina(nDisc : String) : void
+ printNotasAluno(numAluno : int) : void
+ numDisciplinas( ) : int
+ numAlunos( ) : int
+ atribuirNota(valor : int, noMec : int , nDisc : string) : void
+ removerNota(nDisc : string, noMec : int) : void
+ removerDisciplina(nDisc : String): void
Pessoa
- nome : String
+ Pessoa(n : String)
+ operator<(n : Pessoa) : bool
+ getNome( ) : String
+ print( ) : void
Disciplina
- nome : String
+ Disciplina(n : String)
+ operator<(n : Disciplina) : bool
+ setCurso(c : Curso*): void
+ getCurso( ) : Curso*
+ getNome( ) : String
+ add(p : Professor*): bool
+ add(n : Nota) : bool
+ printNotas( ) : void
+ print( ) : void
+ numProfessores( ) : int
+ numNotas( ) : int
+ remNota(n : Nota) : void
+ findNota(n : Nota) : Nota*
0..*
1
1
frequenta
0..*
0..*
Nota
0..*
0..*
Aluno
- no_mec : int
+ Aluno(n : int , no : String)
+ operator<(n : Aluno) : bool
+ getNoMec( ) : int
+ setCurso(c : Curso*): void
+ getCurso( ) : Curso*
+ add(n : Nota*): bool
+ print( ) : void
+ printNotas( ) : void
+ numNotas( ) : int
+ remove(n : Nota*): void
0..*
Professor
- categoria : String
+ Professor(n : String, c : String)
+ operator<(n : Professor) : bool
+ add(d : Disciplina*): bool
+ print( ) : void
+ numDisciplinas( ) : int
0..*
Nota
- valor : float
+ Nota(d: Disciplina*, a: Aluno*, n : float)
+ operator<(n : Nota) : bool
+ getValor( ) : float
+ getDisciplina( ) : Disciplina*
+ getAluno( ) : Aluno*
0..*
lecciona

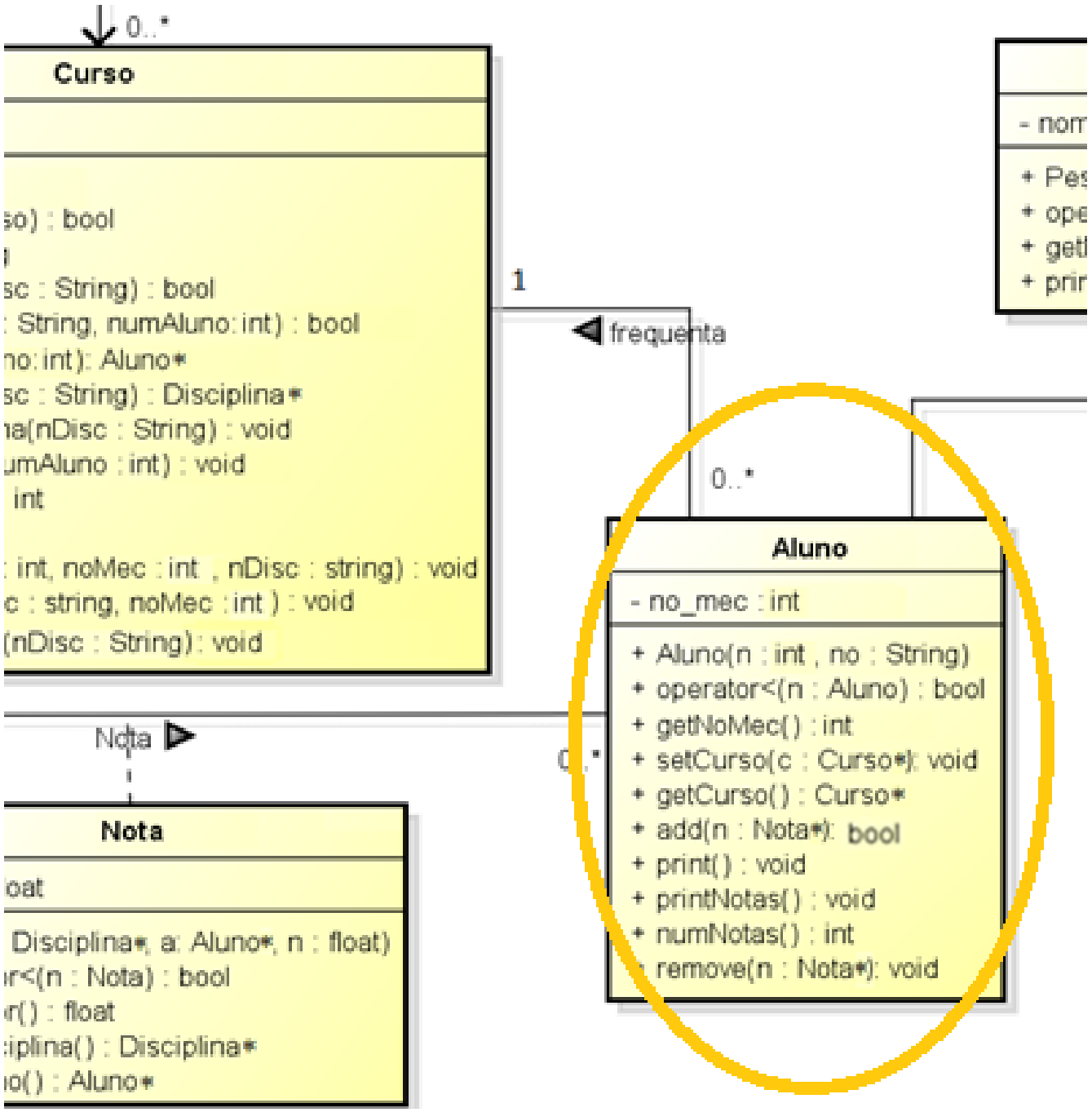
Nota.h




```
#pragma once

class Aluno;
class Disciplina;

class Nota{
   float valor;
   Disciplina *disciplina;
   Aluno *aluno;
public:
   Nota(Disciplina *d, Aluno *a, float n);
   bool operator<(const Nota &outra) const;
   float getValor() const;
   Disciplina *getDisciplina() const;
   Aluno *getAluno() const;
};
```

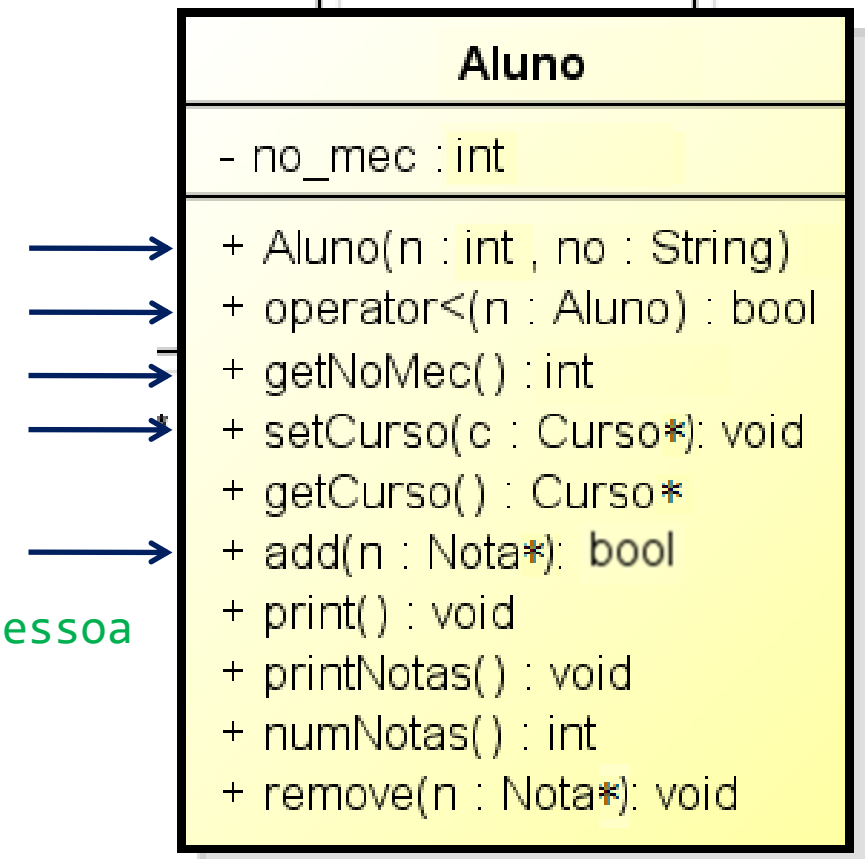
Nota.cpp

```
#include"Nota.h"
#include"Aluno.h"

Nota::Nota(Disciplina *d, Aluno *a, float n){
   disciplina=d;
   aluno=a;
   valor=n;
}

float Nota::getValor() const{ return valor; }

Aluno *Nota::getAluno() const{ return aluno; }

Disciplina *Nota::getDisciplina() const{return disciplina;}

bool Nota::operator<(const Nota &outra) const{
   return( *aluno < *(outra.aluno) );
}
```


0..*
Curso
so) : bool
sc : String) : bool
: String, numAluno: int) : bool
no: int): Aluno*
sc : String) : Disciplina*
na(nDisc : String) : void
umAluno : int) : void
int
: int, noMec : int , nDisc : string) : void
c : string, noMec : int) : void
(nDisc : String): void
1
frequenta
0..*
Aluno
- no_mec : int
+ Aluno(n : int , no : String)
+ operator<(n : Aluno) : bool
+ getNoMec( ) : int
+ setCurso(c : Curso*): void
+ getCurso( ) : Curso*
+ add(n : Nota*): bool
+ print( ) : void
+ printNotas( ) : void
+ numNotas( ) : int
remove(n : Nota*): void
Nota
Nota
oat
Disciplina*, a: Aluno*, n : float)
r<(n : Nota) : bool
r( ) : float
iplina( ) : Disciplina*
o( ) : Aluno*

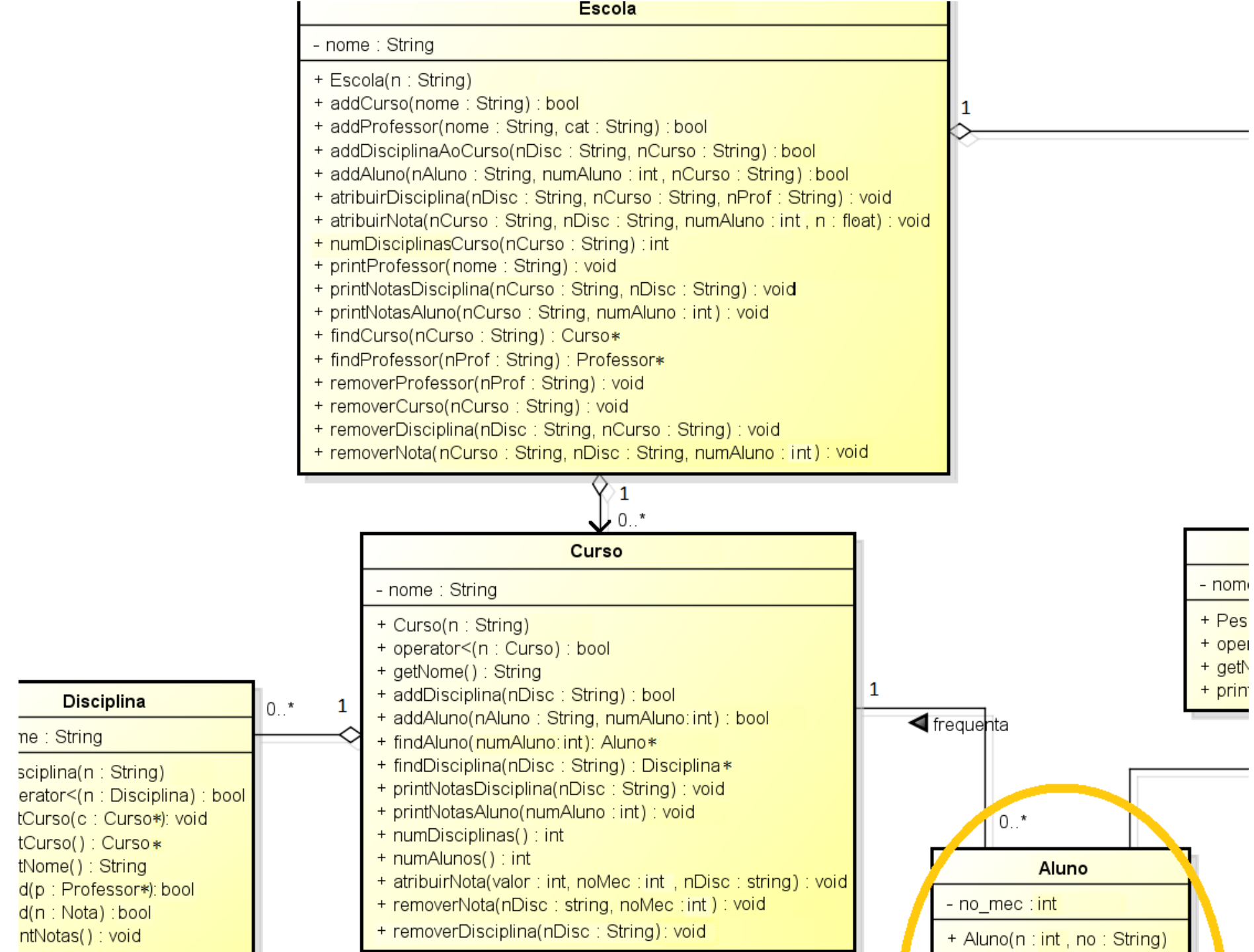
Aluno.h

```
#pragma once
#include<string>
#include "Colecao.h"
using namespace std;
class Curso;
class Nota;
class Aluno{
   string nome; //ainda não existe cl. Pessoa
   int no_mec;
   Curso *curso;
   Colecao<Nota*> notas;
public:
   Aluno(string nom, int num);
   bool operator<(const Aluno &outro) const;
   int getNoMec() const;
   string getNome() const; //enquanto não existe a classe Pessoa
   void setCurso(Curso *c);
   bool add(Nota *n);
};
```



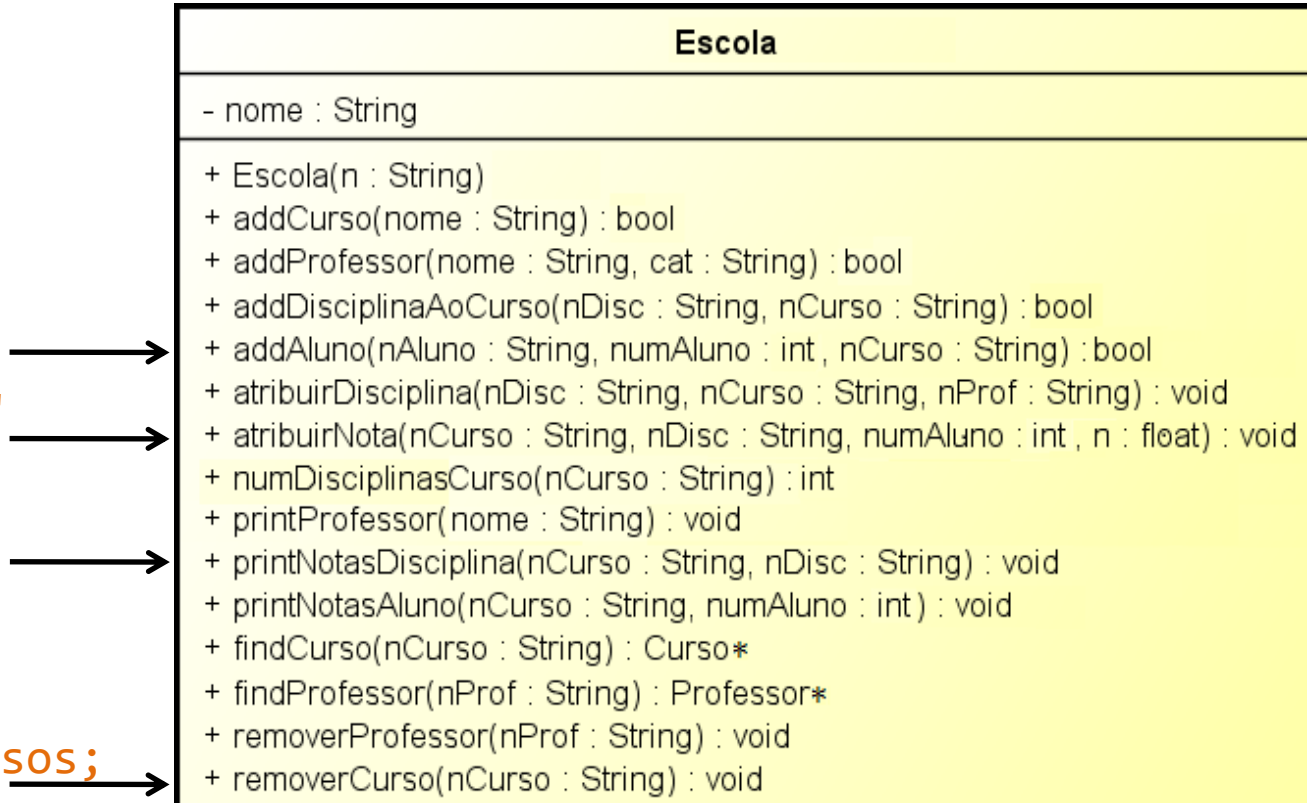
Aluno.cpp

```
#include"aluno.h"

Aluno::Aluno(string nom, int num){
   nome=nom; no_mec=num;
   curso=NULL;
}

int Aluno::getNoMec() const{ return no_mec; }

string Aluno::getNome() const{ return nome; }

void Aluno::setCurso(Curso *c){ if(curso==NULL) curso=c; }

bool Aluno::operator<(const Aluno &outro) const{
   return no_mec<outro.no_mec;
}

bool Aluno::add(Nota *n){ return notas.insert(n); }
```


Escola
- nome : String
+ Escola(n : String)
+ addCurso(nome : String) : bool
+ addProfessor(nome : String, cat : String) : bool
+ addDisciplinaAoCurso(nDisc : String, nCurso : String) : bool
+ addAluno(nAluno : String, numAluno : int , nCurso : String) : bool
+ atribuirDisciplina(nDisc : String, nCurso : String, nProf : String) : void
+ atribuirNota(nCurso : String, nDisc : String, numAluno : int , n : float) : void
+ numDisciplinasCurso(nCurso : String) : int
+ printProfessor(nome : String) : void
+ printNotasDisciplina(nCurso : String, nDisc : String) : void
+ printNotasAluno(nCurso : String, numAluno : int) : void
+ findCurso(nCurso : String) : Curso*
+ findProfessor(nProf : String) : Professor*
+ removerProfessor(nProf : String) : void
+ removerCurso(nCurso : String) : void
+ removerDisciplina(nDisc : String, nCurso : String) : void
+ removerNota(nCurso : String, nDisc : String, numAluno : int) : void
1
1
0..*
Curso
- nome : String
+ Curso(n : String)
+ operator<(n : Curso) : bool
+ getNome( ) : String
+ addDisciplina(nDisc : String) : bool
+ addAluno(nAluno : String, numAluno: int) : bool
+ findAluno(numAluno: int): Aluno*
+ findDisciplina(nDisc : String) : Disciplina*
+ printNotasDisciplina(nDisc : String) : void
+ printNotasAluno(numAluno : int) : void
+ numDisciplinas( ) : int
+ numAlunos( ) : int
+ atribuirNota(valor : int, noMec : int , nDisc : string) : void
+ removerNota(nDisc : string, noMec : int) : void
+ removerDisciplina(nDisc : String): void
Disciplina
ne : String
sciplina(n : String)
erator<(n : Disciplina) : bool
tCurso(c : Curso*): void
tCurso( ) : Curso*
tNome( ) : String
d(p : Professor*): bool
d(n : Nota) : bool
ntNotas( ) : void
0..*
1
1
frequenta
0..*
Aluno
- no_mec : int
+ Aluno(n : int , no : String)
- nom
+ Pes
+ oper
+ getN
+ print

Escola.h

```
#pragma once
#include<string>
#include"Colecao.h"
#include"Curso.h"
#include"Professor.h"
using namespace std;

class Escola{
   string nome;
   Colecao<Curso> cursos;
   Colecao<Professor> professores;
public:
   ...
   bool addAluno(string nAluno, int numAluno, string nCurso);
   void atribuirNota(string nCurso, string nDisc,
                                          int numAluno, float n);
   void printNotasDisciplina(string nCurso, string nDisc);
   void removerCurso(string nCurso);
};
```

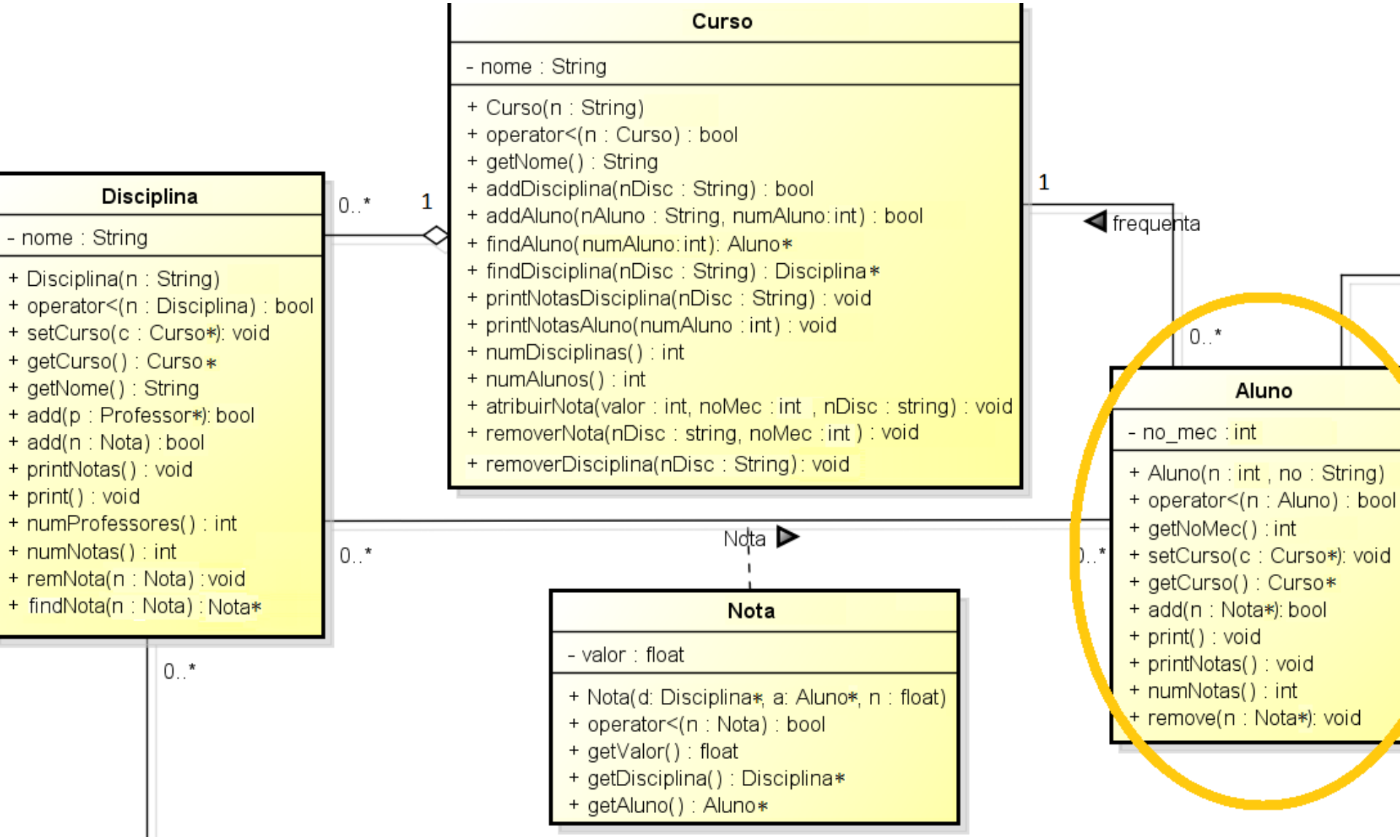


Escola.cpp

```
#include<iostream>
#include"Escola.h"
...
bool Escola::addAluno(string nAluno, int numAluno, string nCurso){
   Curso *f=findCurso(nCurso);
   if(f != NULL) return f->addAluno(nAluno, numAluno);
   else {cout<<"Curso: "<<nCurso << " nao existe.\n"; return false;}
}
void Escola::printNotasDisciplina(string nCurso, string nDisc){
   Curso *fc=findCurso(nCurso);
   if(fc != NULL) fc->printNotasDisciplina(nDisc);
   else cout << "Curso " << nCurso << " nao existe.\n";
}
void Escola::atribuirNota(string nCurso, string nDisc,
                                              int numAluno, float n){
   Curso *fc=findCurso(nCurso);
   if(fc != NULL) fc->atribuirNota(n, numAluno, nDisc);
   else cout << "Curso " << nCurso << " nao existe.\n";
}
```

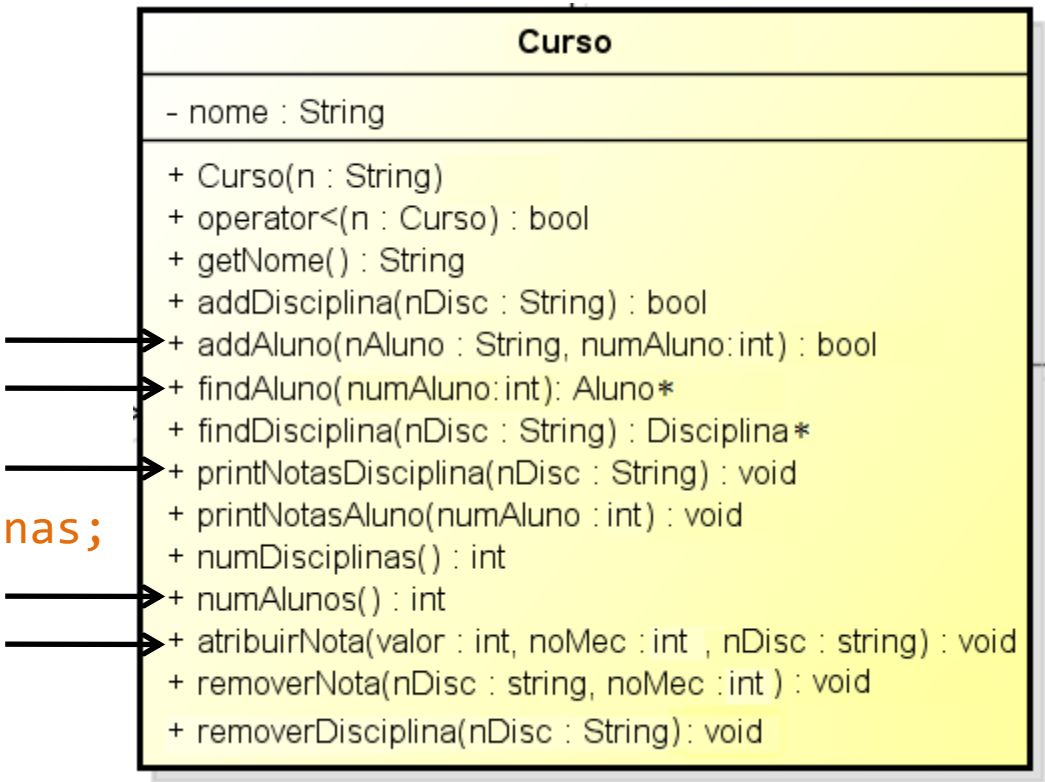
Escola.cpp (continuação)

```
...

void Escola::removerCurso(string nCurso){
   Curso *fc=findCurso(nCurso);
   if(fc != NULL){
        if(fc->numAlunos()>0) cout << "Impossivel remover o curso "
                              << nCurso<<": contem alunos inscritos!\n";
        else if(fc->numDisciplinas()>0)
                cout << "Impossivel remover o curso " << nCurso
                        <<": encontram-se alocadas disciplinas!\n";
             else{
                Curso c(nCurso);
                cursos.erase(c);
                cout << "Curso " << nCurso <<" eliminado!\n";
             }
   }else cout << "Curso " << nCurso << " nao existe.\n";
}
```


Curso
- nome : String
+ Curso(n : String)
+ operator<(n : Curso) : bool
+ getNome( ) : String
+ addDisciplina(nDisc : String) : bool
+ addAluno(nAluno : String, numAluno:int) : bool
+ findAluno(numAluno:int): Aluno*
+ findDisciplina(nDisc : String) : Disciplina*
+ printNotasDisciplina(nDisc : String) : void
+ printNotasAluno(numAluno : int) : void
+ numDisciplinas( ) : int
+ numAlunos( ) : int
+ atribuirNota(valor : int, noMec : int , nDisc : string) : void
+ removerNota(nDisc : string, noMec :int) : void
+ removerDisciplina(nDisc : String): void
Disciplina
- nome : String
+ Disciplina(n : String)
+ operator<(n : Disciplina) : bool
+ setCurso(c : Curso*): void
+ getCurso( ) : Curso*
+ getNome( ) : String
+ add(p : Professor*): bool
+ add(n : Nota) : bool
+ printNotas( ) : void
+ print( ) : void
+ numProfessores( ) : int
+ numNotas( ) : int
+ remNota(n : Nota) : void
+ findNota(n : Nota) : Nota*
Nota
- valor : float
+ Nota(d: Disciplina*, a: Aluno*, n : float)
+ operator<(n : Nota) : bool
+ getValor( ) : float
+ getDisciplina( ) : Disciplina*
+ getAluno( ) : Aluno*
Aluno
- no_mec : int
+ Aluno(n : int , no : String)
+ operator<(n : Aluno) : bool
+ getNoMec( ) : int
+ setCurso(c : Curso*): void
+ getCurso( ) : Curso*
+ add(n : Nota*): bool
+ print( ) : void
+ printNotas( ) : void
+ numNotas( ) : int
+ remove(n : Nota*): void
0..*
1
1
frequenta
0..*
0..*
0..*
Nota

Curso.h

```
#pragma once
#include"Colecao.h"
#include"Disciplina.h"
#include"Aluno.h"

class Curso{
   string nome;
   Colecao<Disciplina> disciplinas;
   Colecao<Aluno> alunos;
public:
   Curso(string n);
   ...
   int numAlunos() const;
   bool addAluno(string nAluno, int numAluno);
   Aluno *findAluno(int numAluno);
   void printNotasDisciplina(string nDisc);
   void atribuirNota(float v, int numAluno, string nDisc);
};
```



Curso.cpp

```
#include"Curso.h"
#include<iostream>
...

int Curso::numAlunos() const {return alunos.size();}

bool Curso::addAluno(string nAluno, int numAluno){
       Aluno a(nAluno, numAluno);
       a.setCurso(this);
       return alunos.insert(a);
}

Aluno *Curso::findAluno(int numAluno){
   Aluno a("", numAluno);
   return alunos.find(a);
}
```

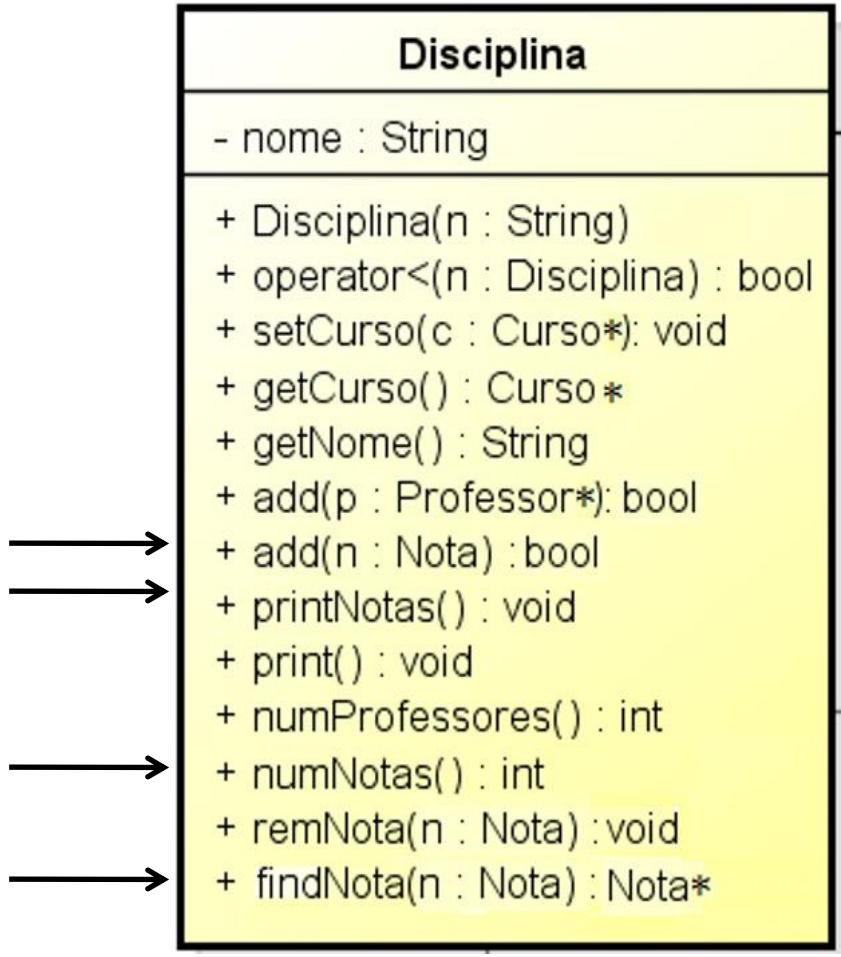
Curso.cpp (continuação)

```
void Curso::printNotasDisciplina(string nDisc){
   Disciplina *fd=findDisciplina(nDisc);
   if(fd != NULL) fd->printNotas();
   else cout << "Disciplina " << nDisc << " nao existe no Curso "
             << nome << endl;
}
```

```
void Curso::atribuirNota(float v, int numAluno, string nDisc){
  Disciplina *fd = findDisciplina(nDisc);
  if (fd != NULL){
    Aluno *fa = findAluno(numAluno);
    if (fa != NULL){
        Nota n(fd, fa, v);
        if(fd->add(n)) fa->add(fd->findNota(n));
        else cout << "Nota nao lancada!\n";
    }else cout<<"Aluno "<<numAluno<<" nao esta matric. no curso "<<nome<<endl;
  }else cout<<"Disciplina "<<nDisc<<" nao existe no Curso "<<nome<<endl;
}
```

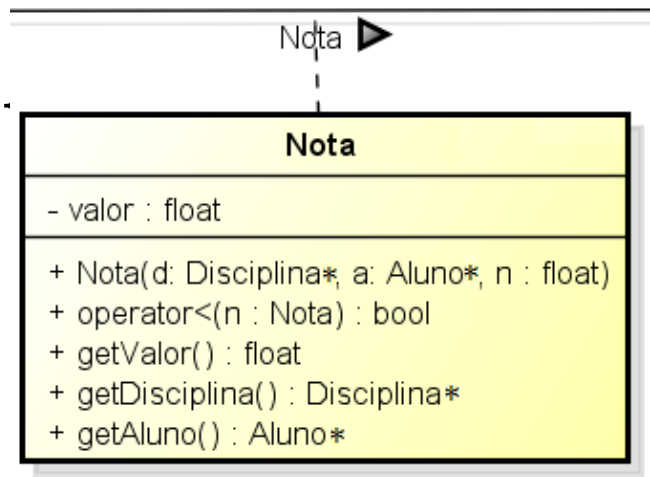
Disciplina.h

```
...
#include"Nota.h"
using namespace std;
class Curso;
class Professor;
class Disciplina{
   string nome;
   Curso *curso;
   Colecao<Professor*> professores;
   Colecao<Nota> notas;
public:
   ...
   bool add(const Nota &n);
   int numNotas() const;
   void printNotas();
   Nota* findNota(const Nota &n);
};
```



Disciplina.cpp

```
#include<iostream>
#include "Disciplina.h"
#include "Professor.h"
#include "Aluno.h"
#include "Curso.h"
...
bool Disciplina::add(const Nota &n){
   return notas.insert(n);
}
int Disciplina::numNotas() const{
   return notas.size();
}
Nota* Disciplina::findNota(const Nota &n)
      return notas.find(n);
}
```



Disciplina.cpp (continuação)

```
...

void Disciplina::printNotas(){
   cout << "Pauta de " << nome << " do curso "
        << getCurso()->getNome()<<":\n";
   Colecao<Nota>::iterator it;
   for(it=notas.begin(); it!=notas.end(); it++)
       cout<<'\t'<<it->getAluno()->getNoMec()<<" "
           <<it->getAluno()->getNome()<<" "
           <<it->getValor()<<" valores.\n";
}
```

```
...
void main(){
   Escola e("ESTiG");
   ...
   e.addAluno("Alberto",1234,"EI");
   e.addAluno("Ana",4321,"EI");
   e.addAluno("Paulo",4458,"IG");

   e.atribuirNota("EI","PI",1234,9.5f);
   e.atribuirNota("EI","PI",4321,5.2f);
   e.atribuirNota("IG","POO",4458,17.6f);

   e.printNotasDisciplina("EI","PI");
   cout<<endl;
   e.printNotasDisciplina("IG","POO");
   cout<<endl;
   e.removerCurso("EI");
   e.addCurso("EE");
   e.removerCurso("EE");
   e.addDisciplinaAoCurso("AM", "EE");
}
```

main.cpp