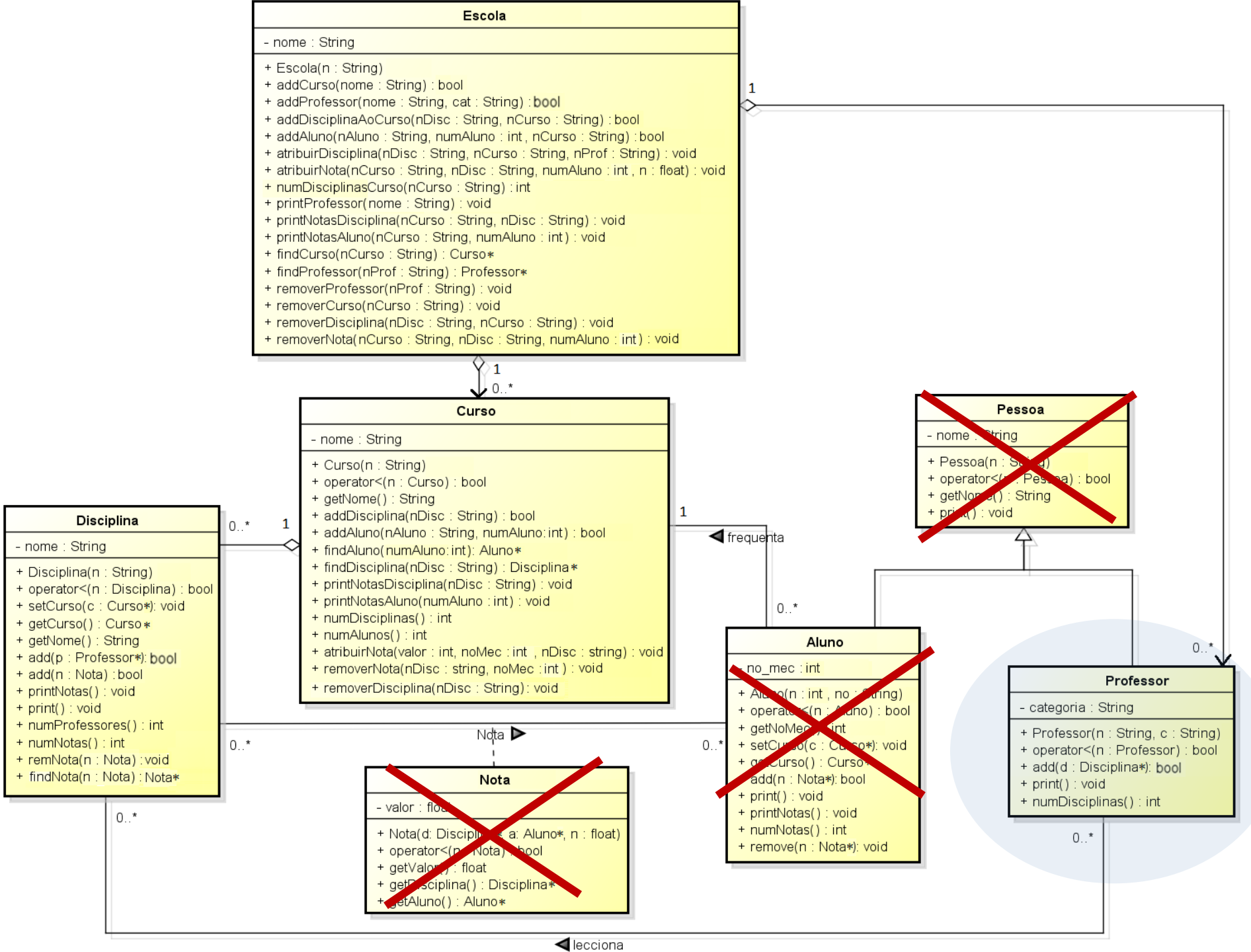
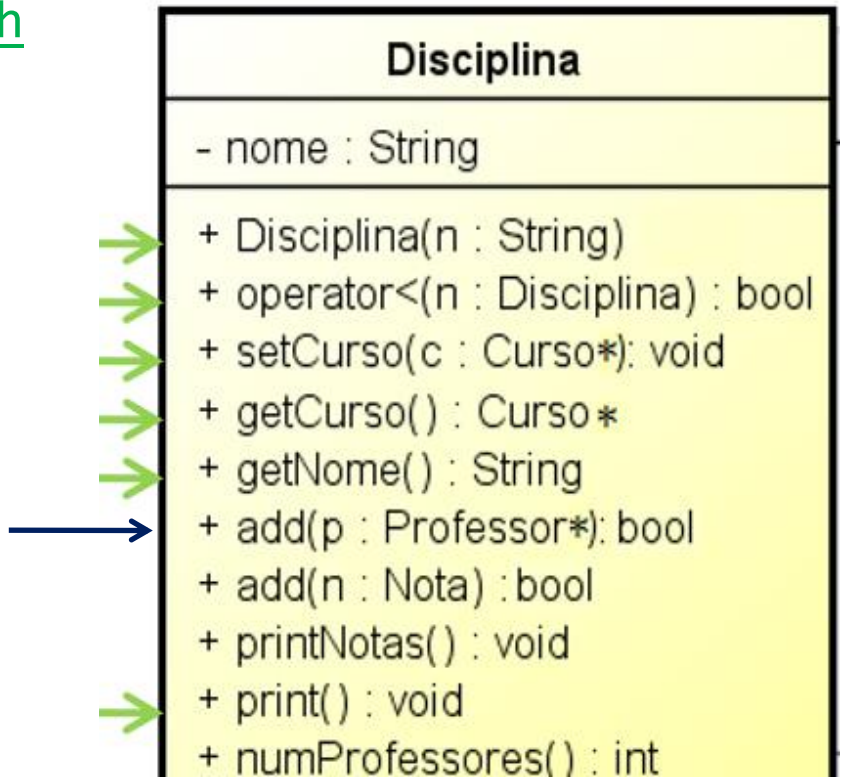
Exercício 20

Para o diagrama do exercício 7, implemente a classe Professor. Implemente as operações de adição de um professor e atribuição de um professor a uma 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..*
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*
0..*
1
frequenta
Pessoa
- nome : String
+ Pessoa(n : String)
+ operator<(n : Pessoa) : bool
+ getNome( ) : String
+ print( ) : void
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
Professor
- categoria : String
+ Professor(n : String, c : String)
+ operator<(n : Professor) : bool
+ add(d : Disciplina*): bool
+ print( ) : void
+ numDisciplinas( ) : int
Nota
Nota
- valor : float
+ Nota(d: Disciplina*, a: Aluno*, n : float)
+ operator<(n : Nota) : bool
+ getValor( ) : float
+ getDisciplina( ) : Disciplina*
+ getAluno( ) : Aluno*
lecciona

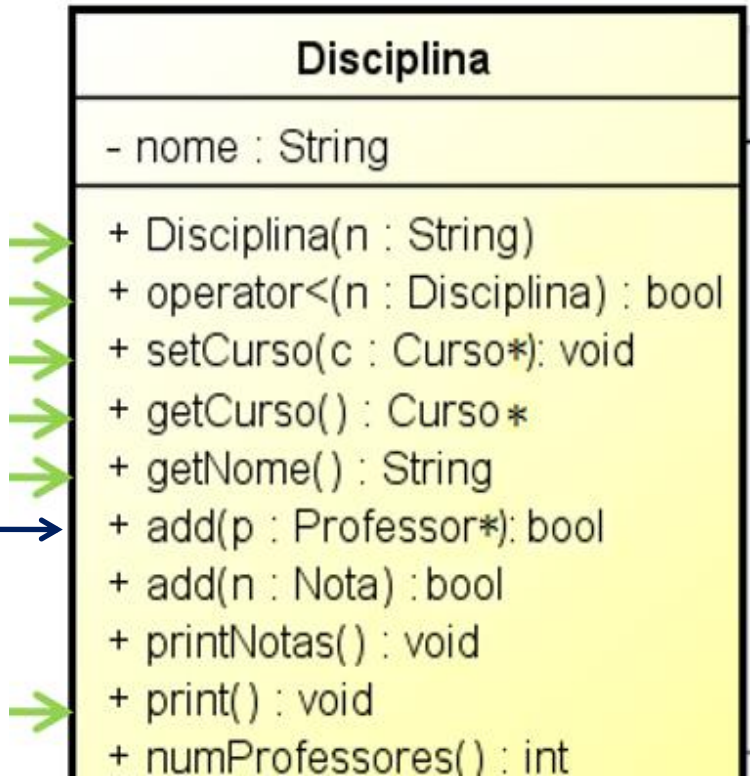
Disciplina.h

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

class Disciplina{
    string nome;
    Curso *curso;
    Colecao<Professor*> professores;
public:
    Disciplina(string n);
    bool operator<(const Disciplina &a) const;
    void setCurso(Curso *c);
    Curso *getCurso() const;
    string getNome() const;
    void print() const;
    bool add(Professor *p);
};
```

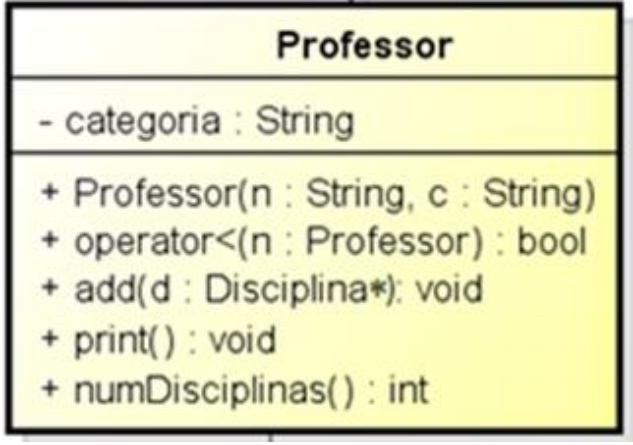


Disciplina.cpp

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

bool Disciplina::add(Professor *p){
   return professores.insert(p);
}
```

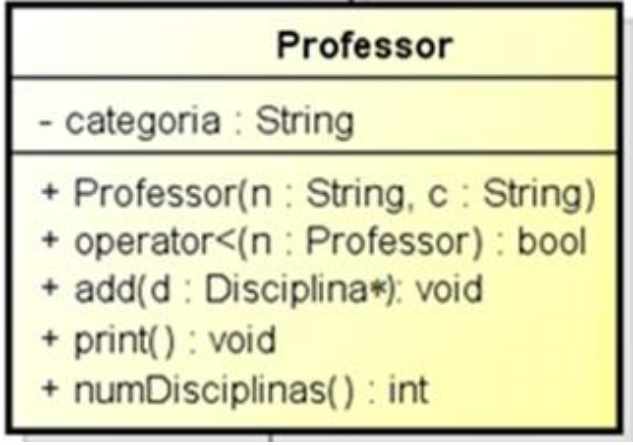


Professor.h




```
#pragma once
#include<string>
#include"Colecao.h"

using namespace std;

class Disciplina;
                //Ainda não temos Pessoa.
class Professor{ //Como Professor não deriva de Pessoa
   string nome;  //o atributo nome tem de ser criado nesta classe
   string categoria;
   Colecao<Disciplina*> disciplinas;
public:
   Professor(string n, string c);
   bool operator<(const Professor &) const;
   bool add(Disciplina *d);
   void print();
   int numDisciplinas() const;
};
```

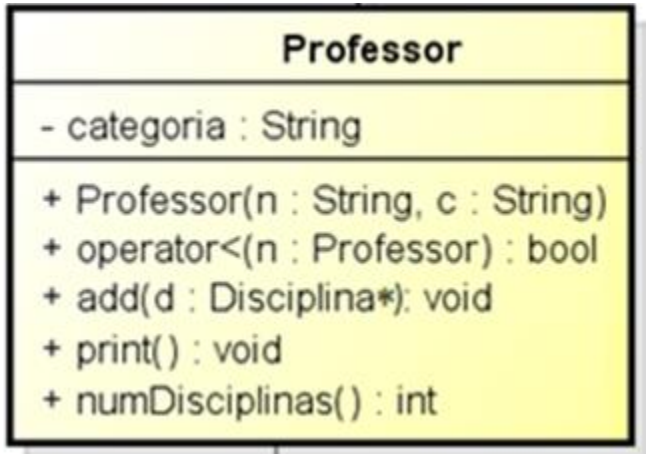
Professor.cpp




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

Professor::Professor(string n, string c){ nome=n; categoria=c;}

bool Professor::operator<(const Professor &p) const{
   return nome<p.nome;
}

void Professor::print(){
   cout << nome << " (" << categoria << ")" << endl;
   cout << "Disciplinas que leciona: \n";
   Colecao<Disciplina*>::iterator it;
   for(it=disciplinas.begin(); it!=disciplinas.end(); it++)
        cout<<'\t'<<(*it)->getNome()<<" de "
            <<(*it)->getCurso()->getNome()<<endl;
}
```

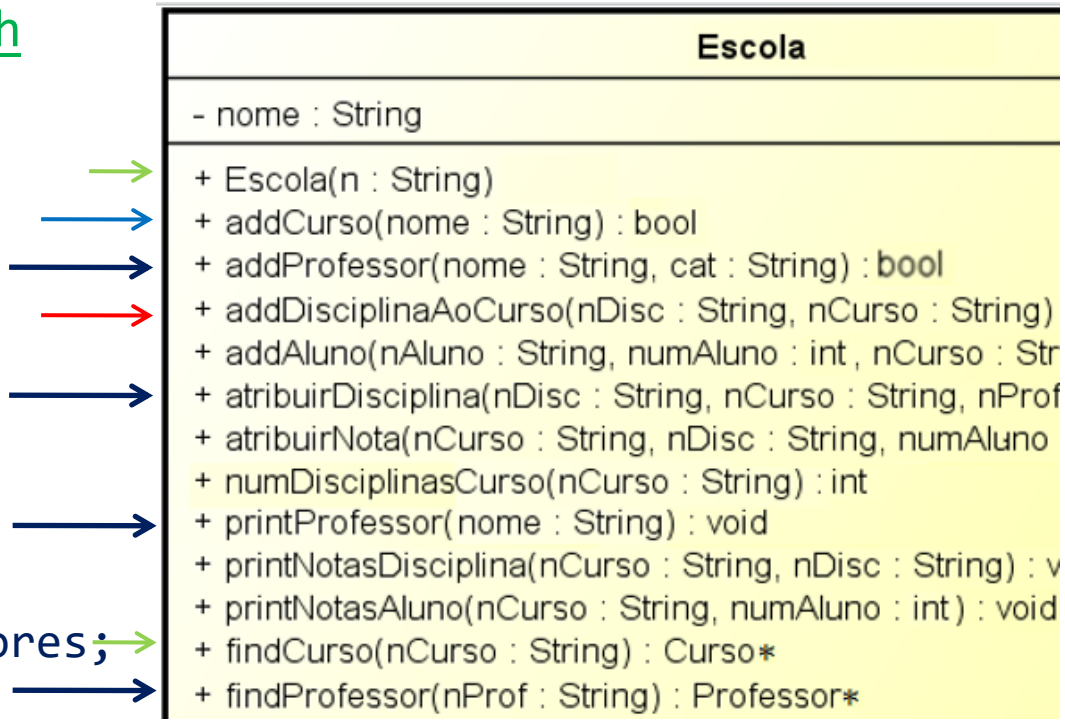
Professor.cpp (continuação)

```
...

bool Professor::add(Disciplina *d){
   return disciplinas.insert(d);
}

int Professor::numDisciplinas() const{
   return disciplinas.size();
}
```



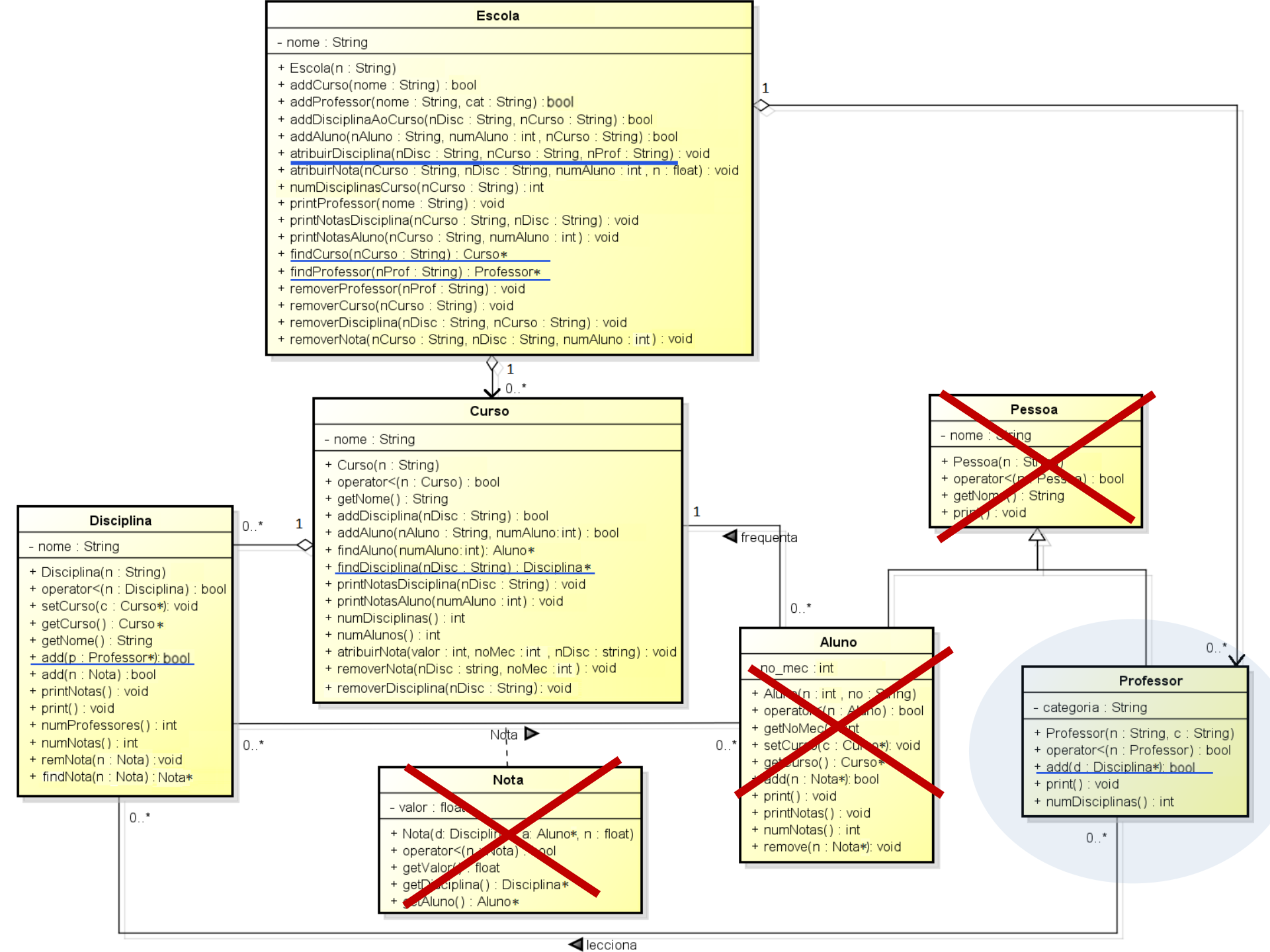
Escola.h

```
#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:
   Escola(string n);
   bool addCurso(string nome);
   bool addDisciplinaAoCurso(string nDisc, string nCurso);
   int numDisciplinasDoCurso(string nCurso);
   void printProfessor(string nProf);
   bool addProfessor(string nome, string cat);
   void atribuirDisciplina(string nDisc,string nCurso,string nProf);
   Professor *findProfessor(string nProf);
   Curso *findCurso(string nCurso);
};
```



Escola.cpp

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

void Escola::printProfessor(string nProf){
   Professor *f=findProfessor(nProf);
   if(f!=NULL) f->print();
   else cout << "Professor " << nProf << " nao existe.\n";
}

Professor *Escola::findProfessor(string nProf){
       Professor d(nProf, "");
       return professores.find(d);
}

bool Escola::addProfessor(string nome, string cat){
       Professor p(nome, cat);
       return professores.insert(p);
}
```


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..*
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
1
frequenta
0..*
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
0..*
0..*
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
Pessoa
- nome : String
+ Pessoa(n : String)
+ operator<(n : Pessoa) : bool
+ getNome( ) : String
+ print( ) : void
0..*
Professor
- categoria : String
+ Professor(n : String, c : String)
+ operator<(n : Professor) : bool
+ add(d : Disciplina*): bool
+ print( ) : void
+ numDisciplinas( ) : int
0..*
0..*
leciona

Escola.cpp (continuação)

```
...

void Escola::atribuirDisciplina(string nDisc,
                                string nCurso, string nProf){
   Curso *fc=findCurso(nCurso);
   if(fc!=NULL){
        Disciplina *fd=fc->findDisciplina(nDisc);
        if(fd!=NULL){
               Professor *fp=findProfessor(nProf);
               if(fp!=NULL){
                      fp->add(fd);
                      fd->add(fp);
               }else cout << "Professor "<<nProf<< " nao existe.\n";
        }else cout<< "Disciplina "<<nDisc<< " nao existe no curso: "
                  << fc->getNome() << endl;
   }else cout << "Curso " << nCurso << " nao exite.\n";
}
```

main.cpp

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

void main(){
   Escola e("ESTiG");
   ...
   e.addProfessor("Rui", "Prof. Adjunto");
   e.addProfessor("Pedro", "Assistente");
   e.atribuirDisciplina("POO","IG","Rui");
   e.atribuirDisciplina("PI","EI","Rui");
   e.atribuirDisciplina("PI","EI","Pedro");

   e.printProfessor("Rui");
   cout<<endl;
   e.printProfessor("Pedro");
}
```