Curso de Ciência da Computação

Algoritmos e Programação de Computadores 2per

Programação Orientada a Objetos POO

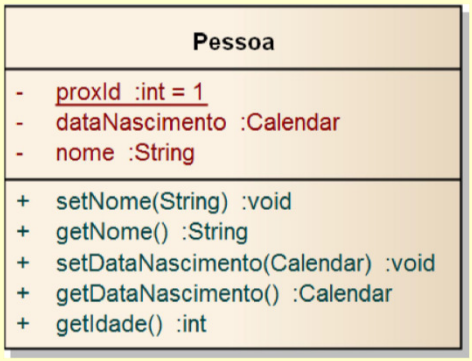
Profa. Fernanda dos Santos Cunha

Atributos de Objetos Relembrando...

- Um **atributo de objeto** é uma característica relevante para o objeto.
  - Dica: ele deve ser assim declarado somente qdo a necessidade de armazenar esta informação for maior que a duração de uma operação
  - Exemplo: nome da pessoa
  - Contraexemplo: idade -> getIdade
- Cada objeto terá sua própria cópia (com respectivo valor) do atributo

Atributos de Classe

- Um **atributo de classe** é uma característica cujo valor é compartilhado por todos os objetos de uma mesma classe.
  - Conhecido também por atributo estático
  - Todos os objetos instanciados acessam o mesmo atributo/ valor
  - Ao lado o "proxId" é um atributo de classe (UML: fica sublinhado)



Revisando a modelagem

```
#ifndef PESSOA_H_INCLUDED
#define PESSOA_H_INCLUDED
#include <iostream>
#include <string>
#include "Date.h"
using namespace std;
class Pessoa{
    static int proxId;
    int id;
    string nome;
    Date dataNascimento;
  public:
    Pessoa();
    ...
};
#endif // PESSOA_H_INCLUDED
```

Inserindo os atributos proxId e id e o construtor Pessoa

Arquivo Pessoa.h

Revisando a modelagem

Arquivo Pessoa.cpp

```
#include <iostream>
#include "Date.h"
#include "Pessoa.h"
using namespace std;

//inicializando o atributo de classe
int Pessoa::proxId = 1;

Pessoa::Pessoa(){
  this->id = proxId++;
  cout << "Pessoa #" << this->id << endl;
}
void Pessoa::setNome(string nome){
  this->nome = nome;
}
...
```

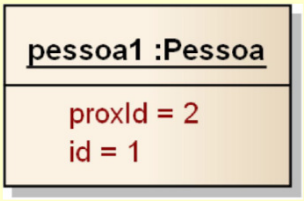
Não usar this para atributo de classe, pois ele é compartilhado!!

Atributos de Classe

- Considerando a criação de 3 objeto do tipo Pessoa:

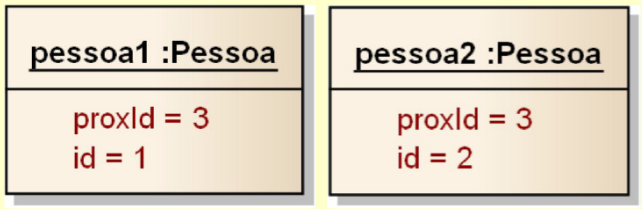
```
Pessoa pessoa1;
Pessoa pessoa2;
Pessoa pessoa3;
```

- Após a instanciação do 1º objeto tem-se:

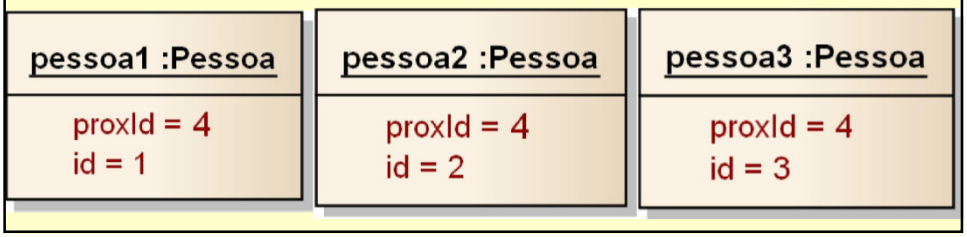


Atributos de Classe

- Após a instanciação do 2º objeto tem-se:



- Após a instanciação do 3º objeto tem-se:

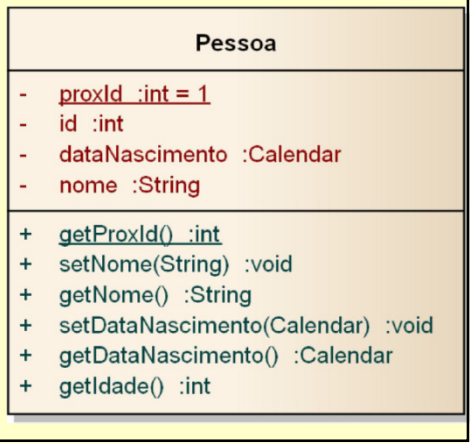


Revisando Operações

- Construtoras
  - Instanciar/criar objetos
- Destrutoras
  - Liberar/destruir objetos
- Modificadoras
  - Modificar o estado (parcial ou total) de objetos (ex: setter's)
- Recuperadoras
  - Recuperar o estado (parcial ou total) de objetos (ex: getter's)

Operação de Classe

- Uma **operação de classe** é um serviço que pode ser invocado diretamente a partir da classe, sem que objetos tenham sido instanciados
  - Conhecido também por operação estática
  - Manipula somente atributos de classe
  - Ao lado o "getProxId" é uma operação de classe (UML: fica sublinhado)



Revisando a modelagem

```
#include <iostream>
#include "Date.h"
#include "Pessoa.h"
using namespace std;

//inicializando o atributo de classe
int Pessoa::proxId = 1;

Pessoa::Pessoa(){
  this->id = proxId++;
  cout << "Pessoa #" << this->id << endl;
}
int Pessoa::getProxId(){
  return proxId;
}
...
```

Arquivo Pessoa.cpp

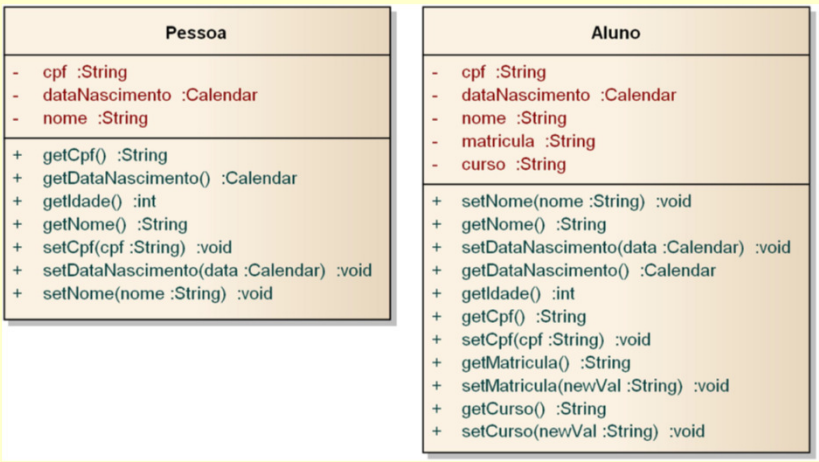
Não usar this para atributo de classe, pois ele é compartilhado!!

Testando...

```
#include <iostream>
#include "Pessoa.h"
using namespace std;
int main(){
      cout << Pessoa::getProxId() <<endl;
      // Atencao a sintaxe: classe:: !!!
      Pessoa alguem;
      cout << alguem.getId()<< endl;
      cout << alguem.getProxId()<< endl;
      cin.get();
      return 0;
}
```

Reanalisando...

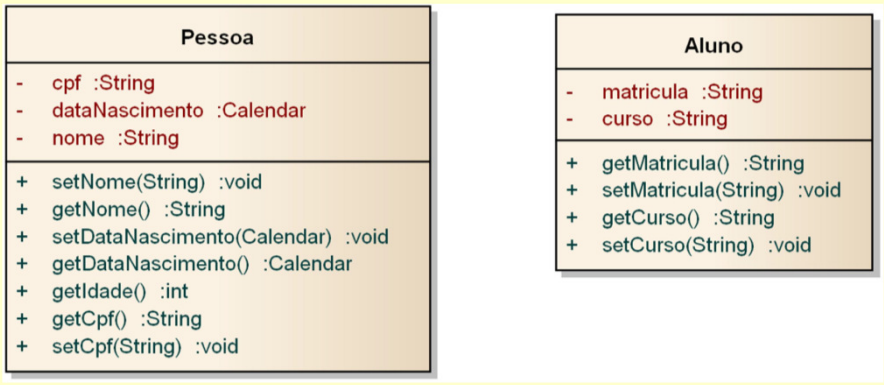
- Olhando estas classes vê-se...



... DUPLICAÇÃO DE CÓDIGO !!! ☹

Reanalisando

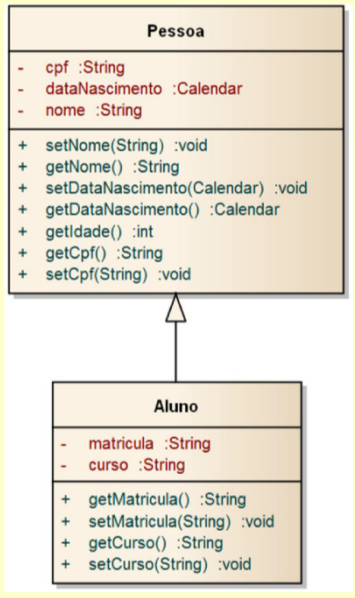
- Melhorou assim? Sim !!



=> Mas falta algo ainda... Como sinalizar que um aluno é uma pessoa ??? Deve haver uma ligação entre as classes

Generalização

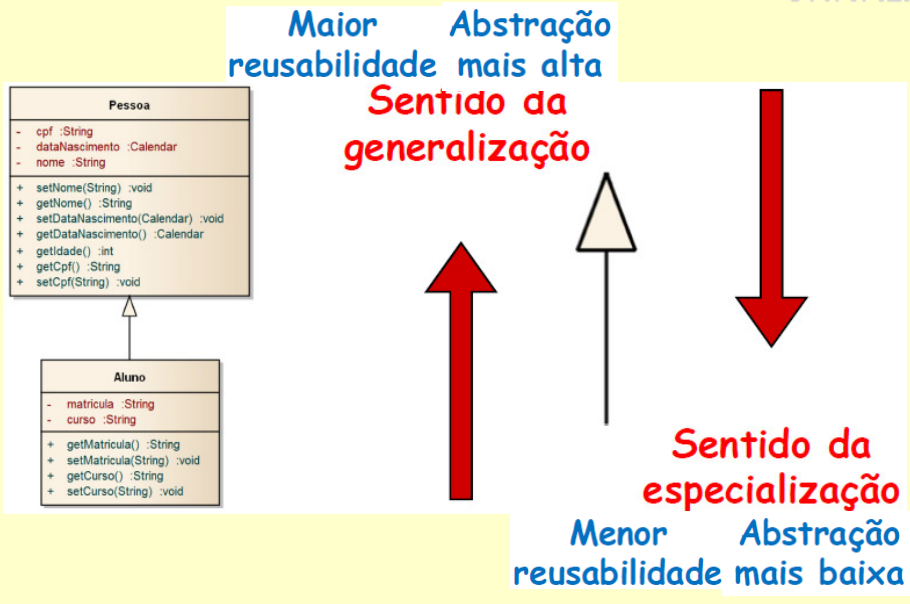
- Relacionamento que indica **Herança**
- O sentido do relacionamento é da classe **mais especializada** para a **mais genérica**
- Aluno é a **subclasse** (mais especializada)
- Pessoa é a **superclasse** (mais genérica)



Generalização

- A subclasse **herda** todos os atributos e operações da superclasse
- Cada objeto (instância) da subclasse é também uma instância indireta da superclasse
- A superclasse tem um nível de abstração maior do que a subclasse

Generalização/Especialização



Generalização em C++

Arquivo Aluno.h

```
#ifndef ALUNO_H_INCLUDED
#define ALUNO_H_INCLUDED
#include <iostream>
#include <string>
#include "Pessoa.h"
using namespace std;

class Aluno : public Pessoa { // indicacao de heranca
    string matricula, curso;
  public:
    void setMatricula(string matricula);
    string getMatricula();
    void setCurso(string curso);
    string getCurso();
};
#endif // ALUNO_H_INCLUDED
```

Generalização em C++

Arquivo Aluno.cpp

```
#include <iostream>
#include <string>
#include "Aluno.h"
Aluno::Aluno(){
  cout << "Tipo Aluno" <<endl; }

void Aluno::setMatricula(string matricula){
  this->matricula = matricula; }

string Aluno::getMatricula(){
  return this->matricula; }

void Aluno::setCurso(string curso){
  this->curso = curso; }

string Aluno::getCurso(){
  return this->curso; }
```

Testando...


```
#include <iostream>
#include "Aluno.h"
using namespace std;
int main(){
      cout << Aluno::getProxId()<< endl; // Atencao!!
      Aluno alguem;
      alguem.setNome("Fernanda");
      alguem.setMatricula("15474");
      alguem.setCurso("Computacao");
      cout << alguem.getId()<<endl; // definido em Pessoa
      cout << alguem.getNome()<<endl;
      cout << alguem.getMatricula()<<endl; //def. em Aluno
      cout << alguem.getCurso()<<endl;
      cin.get();
      return 0;
}
```