Programação Orientada a Objetos

Prof. Delano M. Beder

Roteiro 04

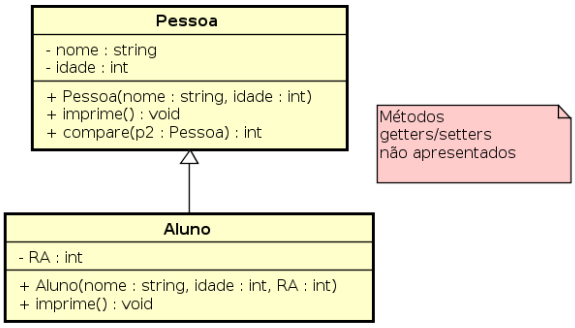
Representar **Pessoas** com as seguintes informações (nome e idade) e as seguintes operações:

- Construtor que inicializa os atributos da classe
- Operações **getNome()**, **setNome()**, **getIdade()** e **setIdade()**
- Operação **imprime** que imprime as informações de uma pessoa
  Exemplo: Nome: Fulano da Silva, Idade: 18 anos
- Operação **compare** que recebe como parâmetro uma 2ª pessoa **p2** e:
  Retorna negativo se **p2** é "maior" (mais velho) que o objeto que está executando a operação
  Retorna zero se **p2** é "igual" (mesma idade) ao objeto que está executando a operação
  Retorna positivo se **p2** é "menor" (mais novo) que o objeto que está executando a operação

Representar **Alunos** com as seguintes informações (nome, idade e RA) e as seguintes operações:

- Construtor que inicializa os atributos da classe
- Operações **getNome()**, **setNome()**, **getIdade()**, **setIdade()**, **getRA()** e **setRA()**
- Operação **imprime** que imprime as informações de um aluno
  Exemplo: Nome: Fulano da Silva, Idade: 18 anos, RA: 927450

Paradigma Orientado a Objetos



classe Pessoa com os atributos (nome e idade) e as operações:

- `Pessoa (string nome, int idade)`

  Exemplo:
  Pessoa p1("Fulano da Silva", 18)

- `void setNome(string), string getNome(), void setIdade(int)` e `int getIdade()`

- `void imprime()`
  Imprime as informações de uma pessoa

  Exemplo:
  Pessoa p1("Fulano da Silva", 18)
  p1.imprime() => Nome: Fulano da Silva, Idade: 18 anos

- `int compare(Pessoa p2)`
  Retorna negativo se p2 é "maior" (mais velho) que o objeto que está executando a operação
  Retorna zero se p2 é "igual" (mesma idade) ao objeto que está executando a operação

Retorna positivo se p2 é "menor" (mais novo) que o objeto que está executando a operação

Exemplo:
Pessoa p1(“Fulano da Silva”, 18)
Pessoa p2(“Sincrano Andrade”, 19)
Pessoa p3(“Beltrano dos Santos”, 18)

p1.compare(p2) => retorna negativo
p1.compare(p3) => retorna 0
p2.compare(p1) => retorna positivo

classe Aluno com os atributos (RA) e as operações:

- `Aluno (string nome, int idade, int RA)`

  Exemplo:
  Aluno a1(“Fulano da Silva”, 18, 927450)

- `void setNome(string), string getNome(), void setIdade(int)` e `int getIdade(), void setRA(int)` e `int getRA()`

- `void imprime()`
  Imprime as informações de um aluno

  Exemplo:
  Aluno a1(“Fulano da Silva”, 18, 927450)
  a1.imprime() => Nome: Fulano da Silva, Idade: 18 anos, RA: 927450

1. Crie um projeto (C++) denominado Heranca

2. Implementação da abstração Pessoa

Nova classe C++ denominada Pessoa

Dois arquivos: Pessoa.h (Cabeçalho) e Pessoa.cpp (Código-fonte)

2.1 Arquivo Pessoa.h

```
#ifndef PESSOA_H
#define PESSOA_H

#include <string>
#include <iostream>
using namespace std;

class Pessoa {
public:
  Pessoa(string nome, int idade);
  int getIdade() const;
  void setIdade(int idade);
  string getNome() const;
  void setNome(string nome);
  void imprime() const;
  int compare(const Pessoa& p) const;
private:
  string nome;
  int idade;
};

#endif /* PESSOA_H */
```

2.2 Arquivo Pessoa.cpp

```
#include "Pessoa.h"

Pessoa::Pessoa(string nome, int idade) :
nome(nome), idade(idade) {
}

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

void Pessoa::setNome(string nome) {
  this->nome = nome;
}

int Pessoa::getIdade() const {
  return idade;
}

void Pessoa::setIdade(int idade) {
  this->idade = idade;
}

void Pessoa::imprime() const {
  cout << "Nome: " << nome << endl;
  cout << "Idade: " << idade << endl;
}

int Pessoa::compare(const Pessoa& p) const {
  return idade - p.idade;
}
```

3. Arquivo main.cpp

```
#include "Pessoa.h"

int main(int argc, char** argv) {

  Pessoa p1("Fulano da Silva", 18);
  Pessoa p2("Sincrano Andrade", 19);
  Pessoa p3("Beltrano dos Santos", 18);

  p1.imprime();

  cout << "Nome: " << p2.getNome() << endl;
  cout << "Idade: " << p2.getIdade() << endl;

  p3.imprime();

  cout << p1.compare(p2) << endl;
  cout << p1.compare(p3) << endl;
  cout << p2.compare(p1) << endl;

  return 0;
}
```

4. Compile e execute (verifique a saída impressa)

5. Implementação da abstração Aluno

Nova classe C++ denominada Aluno

Dois arquivos: Aluno.h (Cabeçalho) e Aluno.cpp (Código-fonte)

5.1 Arquivo Aluno.h

```
#ifndef ALUNO_H
#define ALUNO_H

#include "Pessoa.h"

class Aluno : public Pessoa {
public:
  Aluno(string nome, int idade, int RA);
  int getRA() const;
  void setRA(int RA);
  void imprime() const;
private:
  int RA;
};

#endif /* ALUNO_H */
```

5.2 Arquivo Aluno.cpp

```
#include "Aluno.h"

Aluno::Aluno(string nome, int idade, int RA) :
Pessoa(nome, idade), RA(RA) {
}

int Aluno::getRA() const {
  return RA;
}

void Aluno::setRA(int RA) {
  this->RA = RA;
}

void Aluno::imprime() const {
  Pessoa::imprime();
  cout << "RA: " << RA << endl;
}
```

6. Atualize o arquivo main.cpp

```
#include "Pessoa.h"
#include "Aluno.h"

int main(int argc, char** argv) {

  Pessoa p1("Fulano da Silva", 18);
  Pessoa p2("Sincrano Andrade", 19);
  Pessoa p3("Beltrano dos Santos", 18);

  p1.imprime();

  cout << "Nome: " << p2.getNome() << endl;
  cout << "Idade: " << p2.getIdade() << endl;

  p3.imprime();

  cout << p1.compare(p2) << endl;
  cout << p1.compare(p3) << endl;
  cout << p2.compare(p1) << endl;

  Aluno a1("Aluno teste", 18, 927450);
  a1.imprime();

  cout << p1.compare(a1);

  return 0;
}
```

7. Compile e execute (verifique a saída impressa)

8. Fim

Exercício de fixação

- Implemente a classe **Professor** (subclasse de **Pessoa**) com o atributo **float salario**
- Implemente os métodos **float getSalario()** e **void setSalario(float valor)**
- Implemente (sobreponha) o método **void imprime()**
- Atualize o **main.cpp** e faça alguns testes com a classe **Professor**