VII. Herança Simples

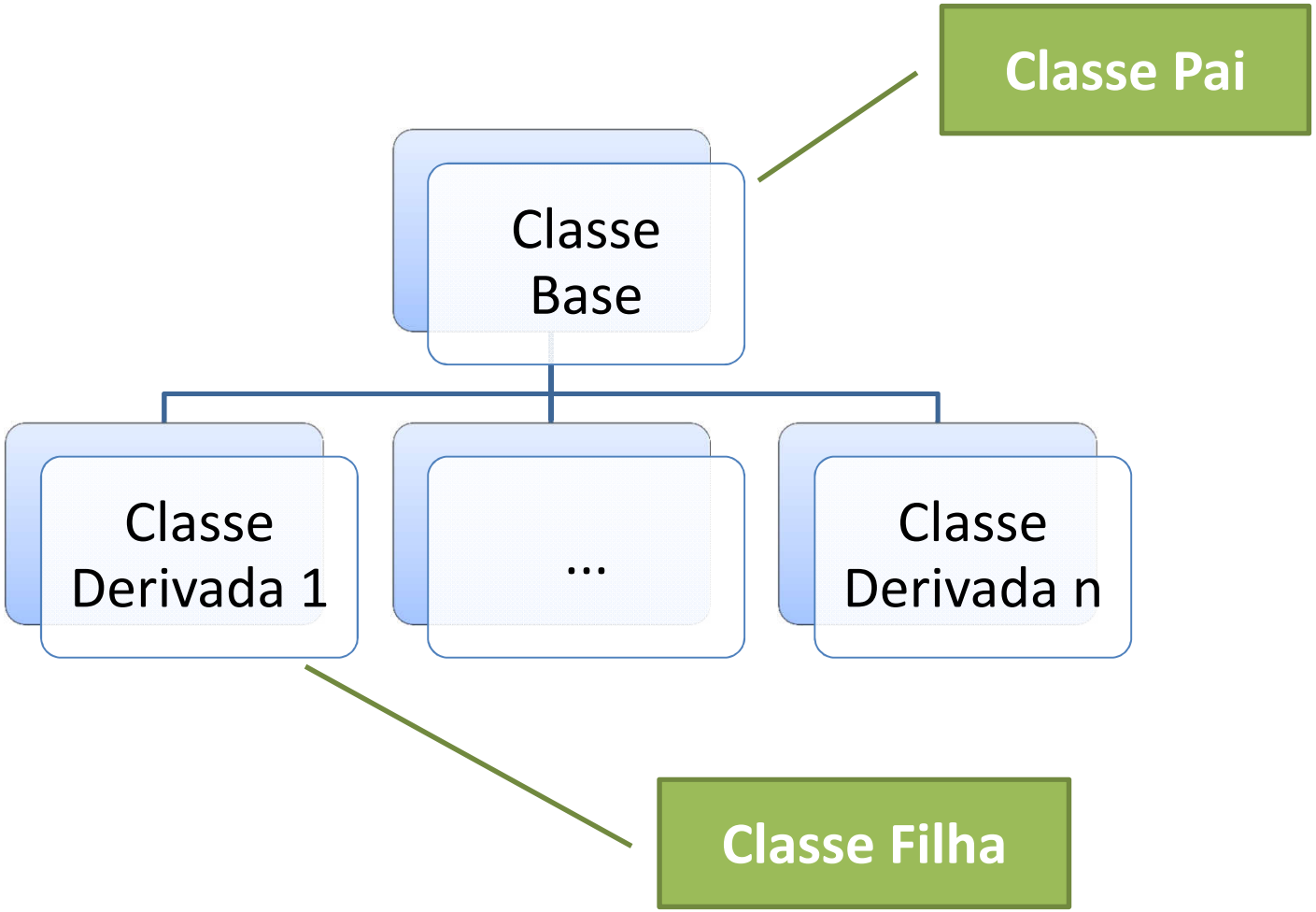
Operador :

Atributos protegidos

Herança

- Processo de criação
  - de novas classes (classe derivada)
  - baseado em classes existentes (classe base)
- A classe derivada herda todas as características da classe base, além de incluir características próprias adicionais
- Reutilização de Código

Herança


Classe Pai
Classe Base
Classe Derivada 1
...
Classe Derivada n
Classe Filha

Ser X Ter

- Um Cliente é uma pessoa ou tem uma pessoa?
- Um Cliente é uma senha ou tem uma senha?
- Uma Conta Corrente é uma conta ou tem uma conta?
- Uma Pessoa é uma data de aniversário, ou tem uma data de aniversário?
- Uma Conta é um saldo ou tem um saldo?
- Uma Reta é um ponto ou tem um ponto?

Herança X Composição

- Herança (SER)
  - Um Cliente é uma Pessoa.
  - Um Funcionário é uma Pessoa.
  - Uma Conta Corrente é uma Conta.
- Composição (TER)
  - Uma Pessoa tem uma Data de Aniversário.
  - Um Cliente tem uma senha.
  - Uma Conta tem um saldo.
  - Uma Reta tem vários pontos.

Exemplo – Classe Pai (Base)

```
class Pessoa {
private:
   string nome;
   Data dtNasc;
public:
   Pessoa(string nome) { this->nome = nome; }
   string getNome();
   void mostra();
};
```

.

Exemplo – Classe Filha

```
class Funcionario {
private:
   string nome;
   Data dtNasc;
   double salario_base;
public:
   Funcionario(string nome) { this->nome = nome; }
   string getNome();
   double getSalario();
   double getSalarioAnual();
   void mostra();
};
```

.

Exemplo – Classe Filha

```
class Funcionario {
private:
   string nome;
   Data dtNasc;
   double salario_base;
public:
   Funcionario(string nome) { this->nome = nome; }
   string getNome();
   double getSalario();
   double getSalarioAnual();
   void mostra();
};
```

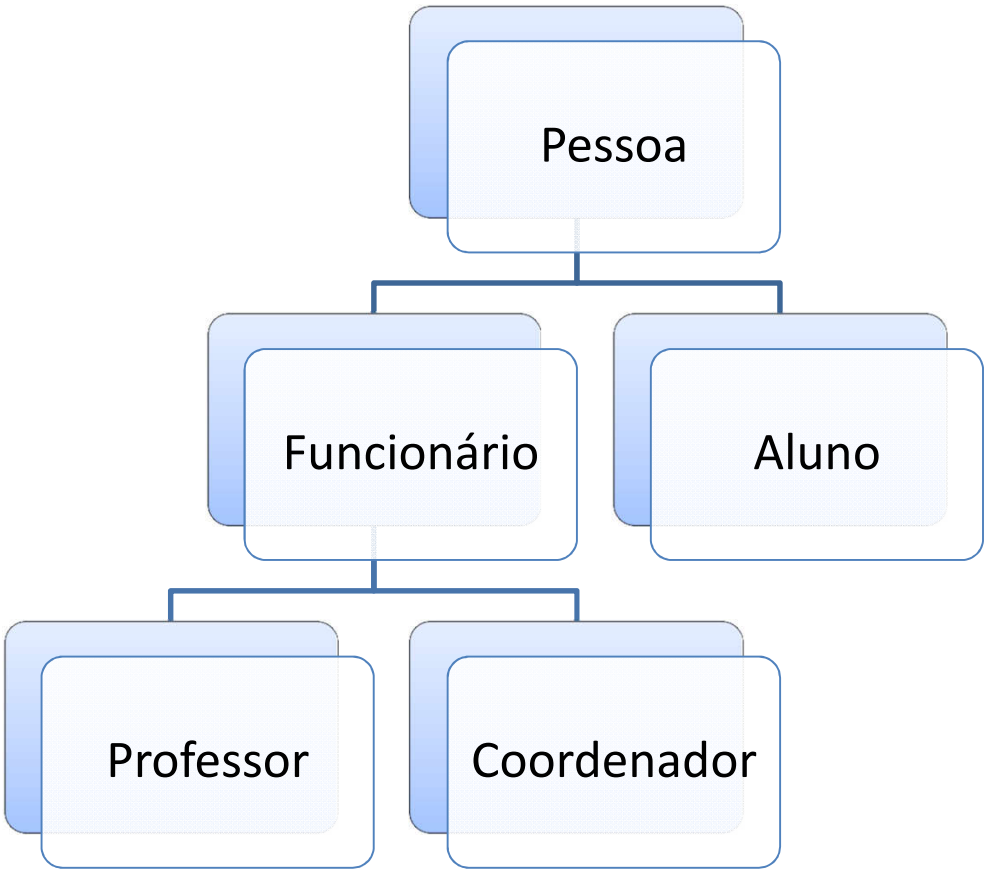
.

Exemplo – Classe Filha (Derivada)

```
class Pessoa {
private:
   string nome;
   Data dataNascimento;
public:
   Pessoa(string nome) { this->nome = nome; }
   string getNome();
   void mostra();
};

class Funcionario : public Pessoa {
   double salario_base;
public:
   Funcionario(string nome) : Pessoa(nome) {}
   double getSalario();
   double getSalarioAnual();
};
```

Classes Derivadas



PROGRAMANDO HERANÇA

Relação Classe Base X Classe Derivada

Membros	Própria Classe	Classe Derivada	Outros
Privados	X		
Protegidos	X	X	
Públicos	X	X	X

Membros **protected** são semelhantes a membros **private**, exceto pelo fato de serem visíveis a todos os membros de uma classe derivada.

Exemplo – Classe Filha (Derivada)

```
class Pessoa {
protected:
   string nome;
   Data dataNascimento;
public:
   Pessoa();
   string getNome();
   void mostra();
};

class Funcionario : public Pessoa {
   double salario;
public:
   Funcionario(string nome) { this->nome = nome; }
   double getSalario();
   double getSalarioAnual();
};
```

Construtor

- Classe pai com construtor sem parâmetros

```
class Pessoa {
protected:
   string nome;
   Data dtNasc;
public:
   Pessoa() {}
   ...
};
```

```
class Funcionario {
protected:
   double salario;
public:
   Funcionario(double s) {
      salario = s;
   }
   ...
};
```

Construtor

- Classe pai com construtor com parâmetros

```
Pessoa(string n, Data d)
{
   this->nome = n;
   this->dtNasc = d;
}
...
Funcionario(string n, Data d, double s) : Pessoa(n,d)
{
   this->salario = s;
}
```

SOBRESCRITA

Sobrescrita (Override)

- As subclasses podem ter métodos com o mesmo **nome** que a classe pai (sobrecarga).
- As subclasses podem ter métodos com a mesma **assinatura** que a classe pai (sobrescrita).

- Finalidades
  - Reescrita
  - Extensão

Reescrita de Métodos

- Todo fim de ano os funcionários da empresa recebem uma **bonificação**.
  - Funcionário comum = 10%
  - Professor = 15%
  - Coordenador = 20%

Reescrita de Métodos

```
class Funcionario : Pessoa {
   protected: double salario;
   ...
   public: double getBonificacao()
   {
       return this->salario * 0.10;
   }
}
```

```
class Professor: Funcionario {
   ...
   public: double getBonificacao()
   {
       return this->salario * 0.15;
   }
}
```

Main

```
int main()
{
   Funcionario f1;
   Professor p1;
   ...
   f1.setSalario(300);
   p1.setSalario(500);

   aux = f1.getBonificacao(); // 30
   cout << aux;
   aux = p1.getBonificacao(); // 50 ou 75?
   cout << aux;
}
```

Extensão de Métodos

- Como fazer o método mostra() da classe Funcionário acessar o método mostra() da classe Pessoa?

```
public double mostra() {
    Pessoa::mostra();
    cout << salario;
}
```

EXERCÍCIOS

Exercícios

- Modele as seguintes classes com pelo menos um atributo e um método cada:
  - classe Aluno, derivada de Pessoa
  - classe Professor, derivada de Funcionário.
  - classe Coordenador, derivada de Funcionário.

- Teste todas as classes no main().