Linguagem de Programação C++

Universidade Federal de Santa Catarina

Departamento de Engenharia Elétrica, CTC

Prof. Eduardo Augusto Bezerra

Profa. Cristiane Woszezenki

Herança

A herança está relacionada às hierarquias e as relações entre os objetos. No dia a dia, quando se fala de herança se refere à transferência de propriedades de um pai aos seus filhos, ou seja, aquilo que é do pai passa a ser do filho.

É comum ainda o dito popular “saiu ao pai”, que significa que o filho tem as mesmas características do pai. De uma maneira geral as pessoas sabem que o filho saiu ao pai mas não é ele, ou seja não são a mesma pessoa. E que o filho apresenta determinadas características diferentes de seu pai.

Na análise orientada a objeto, herança é o mecanismo em que uma classe filha compartilha automaticamente todos os métodos e atributos de sua classe pai. A herança permite implementar classes descendentes com métodos e atributos que se diferenciam da classe pai.

Herança é a propriedade de podermos criar classes que se ampliam a partir de definições básicas. De classes mais simples e genéricas para classes mais complexas e específicas.

Exemplo:

- Um determinado processador possui todas as características de sua versão anterior preservadas, porém com mais memória cache. A memória cache já existia mas foi ampliada.

- Uma placa mãe nova apresenta uma nova interface de comunicação, que é uma novidade não existindo em modelos anteriores.

O conceito de herança permite a criação de uma classe derivada, ou seja, dada uma classe base, pode-se criar uma classe derivada que herda todos os atributos e métodos da classe base.

Protótipo para herança

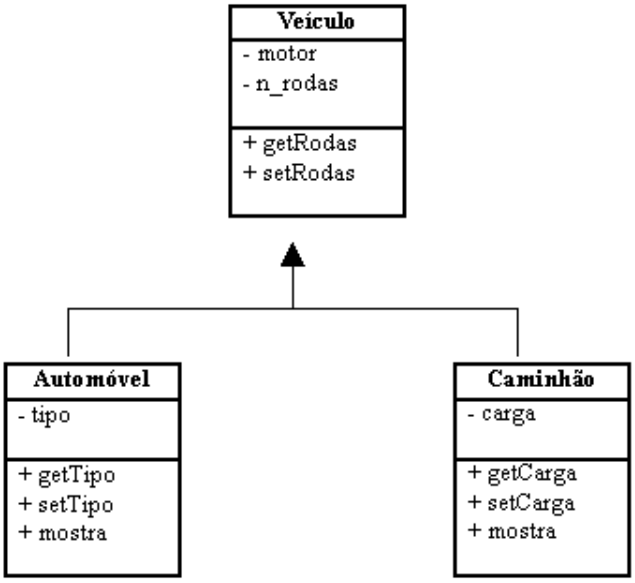
```
class Base {
   // Definição dos atributos e métodos da classe Base
};

class Derivada: public Base {
   // Definição dos atributos e métodos da classe Derivada
};
```

Observe a linha, `class Derivada: public Base`

nesta linha, informa-se que a classe Derivada é herdeira da classe Base.

A palavra **public**, é o especificador de herança e define a forma como a classe Derivada pode acessar os atributos e métodos da classe Base. O termo **public** indica que queremos que as funções e variáveis **public** na classe Base permaneçam **public** na classe Derivada. Em lugar do termo **public**, poderíamos optar por **private** ou **protected**. Em qualquer desses casos todas as variáveis e funções públicas da classe base seriam convertidas para a classe derivada (veja Seção *Especificador de Herança*). O uso de **public** nesses casos é o padrão.

Exemplo de herança em C++




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

// Classe Base
class Veiculo{
private:
   int motor;
   int n_rodas;
public:
   void setNRodas(int);
   int getNRodas(void);
};

void Veiculo::setNRodas(int rodas){
   n_rodas = rodas;
}

int Veiculo::getNRodas(void ){
   return n_rodas;
}

//Primeira classe derivada

class Caminhao: public Veiculo{
private:
   int carga;
public:
   int getCarga(void);
   void setCarga(int);
   void mostra(void);
};

void Caminhao::setCarga(int c){
   carga = c;
}
```

```
int Caminhao::getCarga(void){
   return carga;
}

void Caminhao::mostra(void){
   cout << "Caminhao: n rodas = " << getNRodas( ) << "carga = " << getCarga( );
}

// Segunda classe derivada

class Automovel : public Veiculo {
private:
   string tipo;
public:
   void setTipo(string);
   string getTipo(void);
   void mostra(void);
};

void Automovel::setTipo(string t){
   tipo = t;
}

void Automovel::getTipo(void){
   return tipo;
}

void Automovel::mostra(void){
   cout << "Automovel: n rodas = " << getNRodas( ) << "tipo = " << getTipo( );
}

int main( ){
   Caminhao FNM;
   Automovel Fusca;

   FNM.setNRodas(12);
   FNM.setCarga(3000);
   FNM.mostra( );

   Fusca.setNRodas(4);
   Fusca.setTipo(SEDAN);
   Fusca.mostra( );

   return 0;
}
```

ESPECIFICADOR DE HERANÇA

O especificador de herança altera a forma como se processa a herança. Pode-se usar os especificadores public, protected e private.

Protótipo:

```
class Derivada: public Base {};

class Derivada: protected Base {};

class Derivada: private Base {};
```

O acesso aos membros da classe Base vai depender do especificador de herança (public, protect e private) e dos especificadores de acesso na classe pai (public, protect e private). A tabela mostra o acesso herdado. Na primeira

coluna o especificador de acesso utilizado na classe base, na segunda coluna o especificador de herança e na terceira coluna o acesso efetivamente herdado.

Observe na primeira linha da tabela, que se o atributo é public e o especificador de herança é public, o acesso herdado é public. Se o atributo é protected e o especificador de herança é private, o acesso herdado é private.

Tipo de acesso na classe base	Especificador de herança	Acesso herdado
public	public	public
protected	public	protected
private	public	inacessível
public	protected	protected
protected	protected	protected
private	protected	inacessível
public	private	private
protected	private	private
private	private	inacessível

Exemplo de uso do especificador de acesso:

```
class A {
   public:
      int x;
   protected:
      int y;
   private:
      int z;
};

class B: public A {
   int X() {return x;}; //ok x é público
   int Y() {return y;}; //ok y é protegido
   int Z() {return z;}; //Erro não tem acesso a z
};

class C: private A {
   int X() {return x;}; //ok x é privado
   int Y() {return y;}; //ok y é privado
   int Z() {return z;}; //Erro não tem acesso a z
};
```

Siga esse link para o código fonte de exemplos de herança.