```
//conteúdo do arquivo ponto2dcirculo.h
#ifndef PONTO2DCIRCULO_H
  #define PONTO2DCIRCULO_H
  class Ponto2D {
    public:
      Ponto2D (float, float);
      void setX (float);
      void setY (float);
      float getX ();
      float getY ();
      void move (float, float);
      void mostraCoordenadas(void);
    protected:
      float x;
      float y;
  };
  class Circulo: public Ponto2D {
    public:
      Circulo(float=0.0, float=0.0);
      void setRaio(float);
      float getRaio(void);
      float area(void);
      bool pontoPertenceAoCirculo (Ponto2D); /*bool é um tipo primitivo
       existente na linguagem C++*/
    private:
      float raio;
      float distanciaAoCentro (float, float); /*função utilitária (auxiliar) ajuda a
       operação das funções membros públicas da classe */
  };
#endif
```

```
//conteúdo do arquivo ponto2dcirculo.cpp
#include <iostream>
#include <cmath>
#include "ponto2dcirculo.h"
using std::cout;
using std::endl;
Ponto2D::Ponto2D (float valorX, float valorY)
{
   setX(valorX);
   setY(valorY);
}
void Ponto2D::setX (float novoX)
{
   x = novoX;
}
void Ponto2D::setY (float novoY)
{
   y = novoY;
}
float Ponto2D::getX ()
{
   return x;
}
float Ponto2D::getY ()
{
   return y;
}
```

```
void Ponto2D::move (float novoX, float novoY)
{
  setX (novoX);
  setY (novoY);
}
void Ponto2D::mostraCoordenadas(void)
{
  cout << "(" << getX() << ", " << getY() << ")" << endl;
}
Circulo::Circulo(float x, float y):Ponto2D(x, y)
{
  setRaio(0);
}
void Circulo::setRaio(float novoRaio)
{
  raio = novoRaio;
}
float Circulo::getRaio(void)
{
  return raio;
}
float Circulo::distanciaAoCentro (float a, float b)
{
  return((float)sqrt(pow(a-getX(),2)+pow(b-getY(),2)));
}
```

```
float Circulo::area()
{
  return((float)3.1415*pow(getRaio(),2));
}

bool Circulo::pontoPertenceAoCirculo (Ponto2D p)
{
  return(getRaio()>=distanciaAoCentro(p.getX(), p.getY())?true:false);
}
```

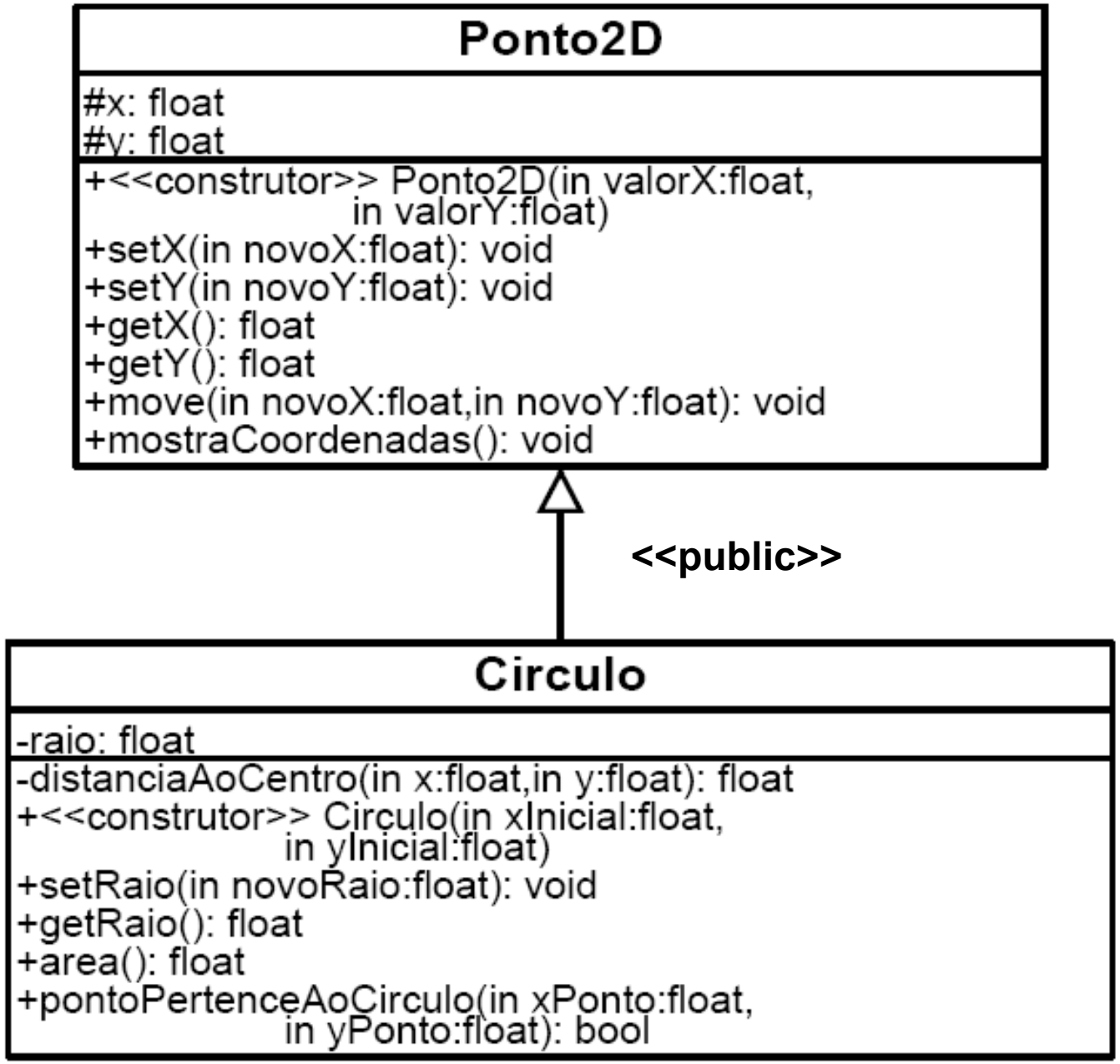
```
//conteúdo do arquivo principalponto2dcirculo.cpp
#include "ponto2dcirculo.h"
#include <iostream>
using std::cin;
using std::cout;
using std::endl;
int main()
{
   Ponto2D p(3.5, 4.1);
   float x, y;
   cout << endl << endl << "O ponto incialmente encontra-se em ";
   p.mostraCoordenadas();
   cout << endl << endl << "Entre com uma nova coordenada x para o ponto: ";
   cin >> x;
   cout << endl << "Entre com uma nova coordenada y para o ponto: ";
   cin >> y;
   p.move (x, y);
   cout << endl << endl << "O ponto encontra-se em ";
   p.mostraCoordenadas();
   cout << endl << endl;
   cout << endl << endl << "Entre com a coordenada x para o centro da
       circunferencia: ";
   cin >> x;
   cout << endl << "Entre com a coordenada y para o centro da circunferencia: ";
```

```
cin >> y;
Circulo c(x, y);
cout << "O centro do circulo encontra-se em ";
c.mostraCoordenadas();
cout << endl << endl << "Entre com uma nova coordenada x para o centro
    da circunferencia: ";
cin >> x;
cout << endl << "Entre com uma nova coordenada y para o centro da
    circunferencia: ";
cin >> y;
c.move (x, y);
cout << "O centro do circulo agora encontra-se em ";
c.mostraCoordenadas();
cout << endl << endl << "Entre com a medida do raio da circunferencia: ";
cin >> x;
c.setRaio(x);
cout << endl << endl << "A area da circunferencia eh " << c.area();
cout << endl << endl << "O ponto";
if (c.pontoPertenceAoCirculo(p))
  cout << " ";
else
  cout << " nao ";
cout << "pertence ao circulo.";
return 0;
}
```

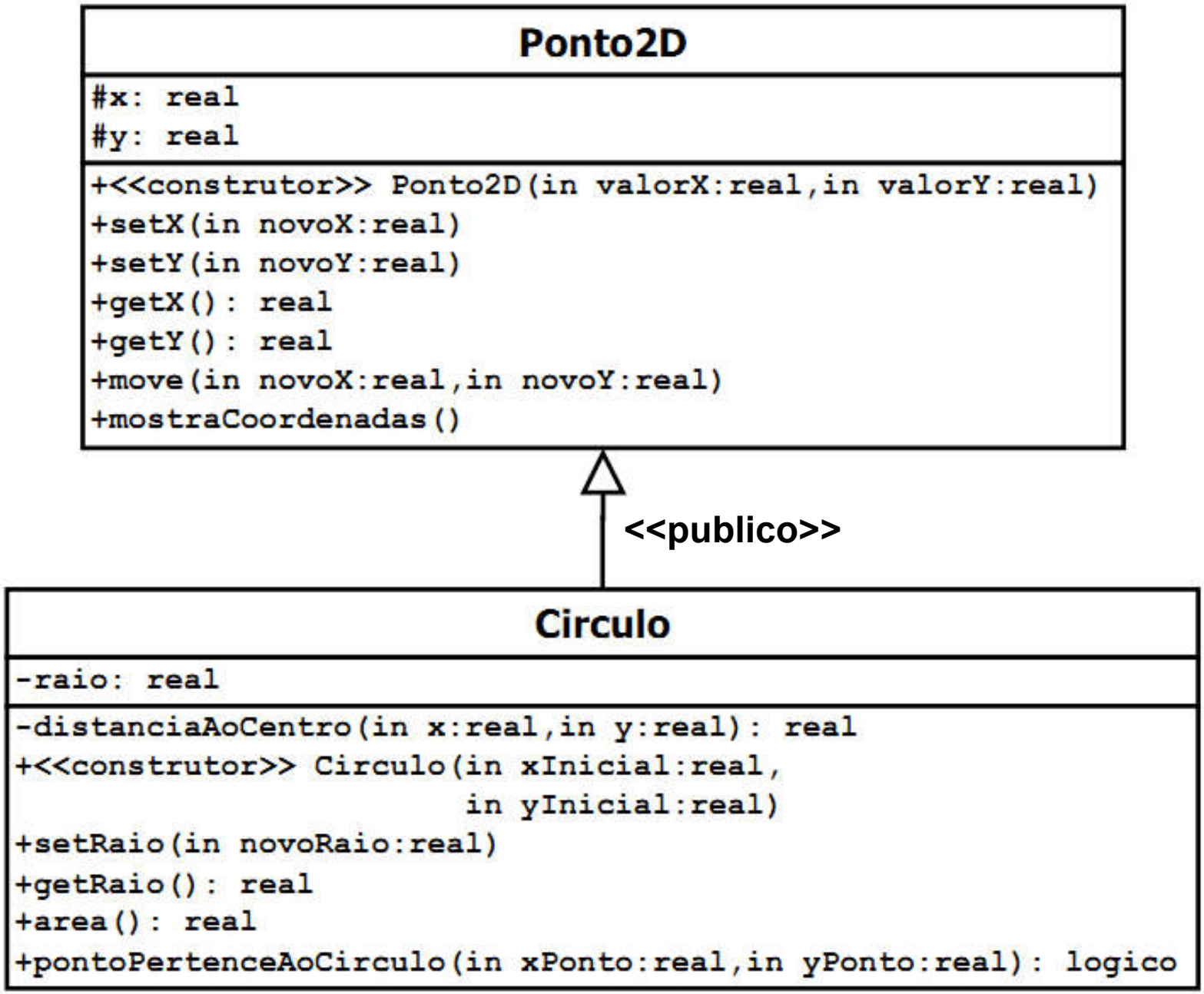
Exercício:

Construa um diagrama de classes em UML para representar as classes **Ponto2D** e **Circulo**, definidas no exercício anterior.

Linguagem de Programação C++



Linguagem de Programação C++



Linguagem de Programação C++

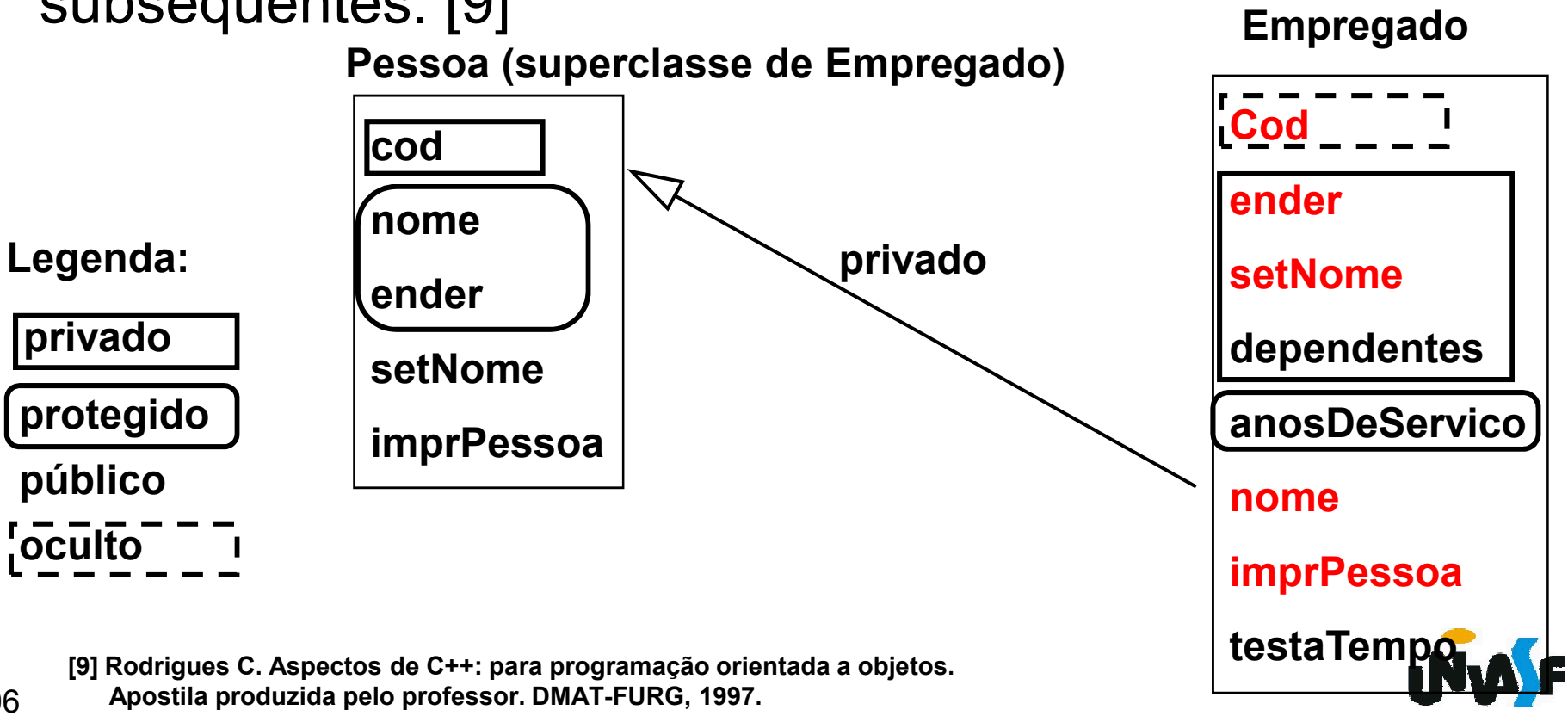
É possível efetuar uma redeclaração individual do modo de acesso, mudando explicitamente o status de um membro de classe derivada.

Vamos analisar o exemplo:

```
class Pessoa
{
  long int cod; //privado – obs. Evitar utilizar a especificação implícita
  protected:
    char nome[30];
    char ender[50];
  public:
    void setNome(novoNome[30]);
    void imprPessoa ();
};
class Empregado:Pessoa //recepção default: private – obs. anterior
{
  public:
    char Pessoa::nome[30]; //este atributo seria privado por default
    void Pessoa::imprPessoa (); //este método seria privado por default
    int testaTempo(void);
  private:
    int dependentes;
  protected:
    int anosDeServico;
};
```

Linguagem de Programação C++

Assim, por exemplo, o campo nome, de objetos da classe Empregado, será visível fora da classe. Porém, continuara privado nas classes derivadas de empregado. Ou seja, a especificação explícita individual do modo de recepção de um campo privativo **não** se transmite às derivações subsequentes. [9]



[9] Rodrigues C. Aspectos de C++: para programação orientada a objetos. Apostila produzida pelo professor. DMAT-FURG, 1997.

Linguagem de Programação C++

Em nosso estudo dos conceitos e princípios da OO vimos que uma análise do mundo real nos remete a perceber que em alguns sistemas temos a necessidade de uma subclasse com mais de uma superclasse.

A linguagem C++ possibilita a definição de uma subclasse que possui mais de uma superclasse, em outras palavras, possibilita a **herança múltipla**.

Vamos imaginar o seguinte exemplo: temos duas classes Embalagem e Rotulo, ambas com membros específicos. Pretende-se obter uma classe EmbalagemRotulada, que associa ao recipiente uma descrição.

A sintaxe é:

class EmbalagemRotulada: public Embalagem, public Rotulo

Os objetos desta classe podem receber mensagens de Embalagem e Rotulo.

UNIVASF

Linguagem de Programação C++

Devemos ter ciência de que **problemas** ocorrerão se mais de uma superclasse oferecer funções membros de mesmo nome. Neste caso, será necessário redeclará-las.

Por exemplo, imagine que ambas as superclasses apresentam uma função membro denominada apagar e sem parâmetros, este problema pode ser resolvido da seguinte forma **(obs. Este problema se mantém mesmo que os métodos tenham conjuntos de parâmetros distintos)**:

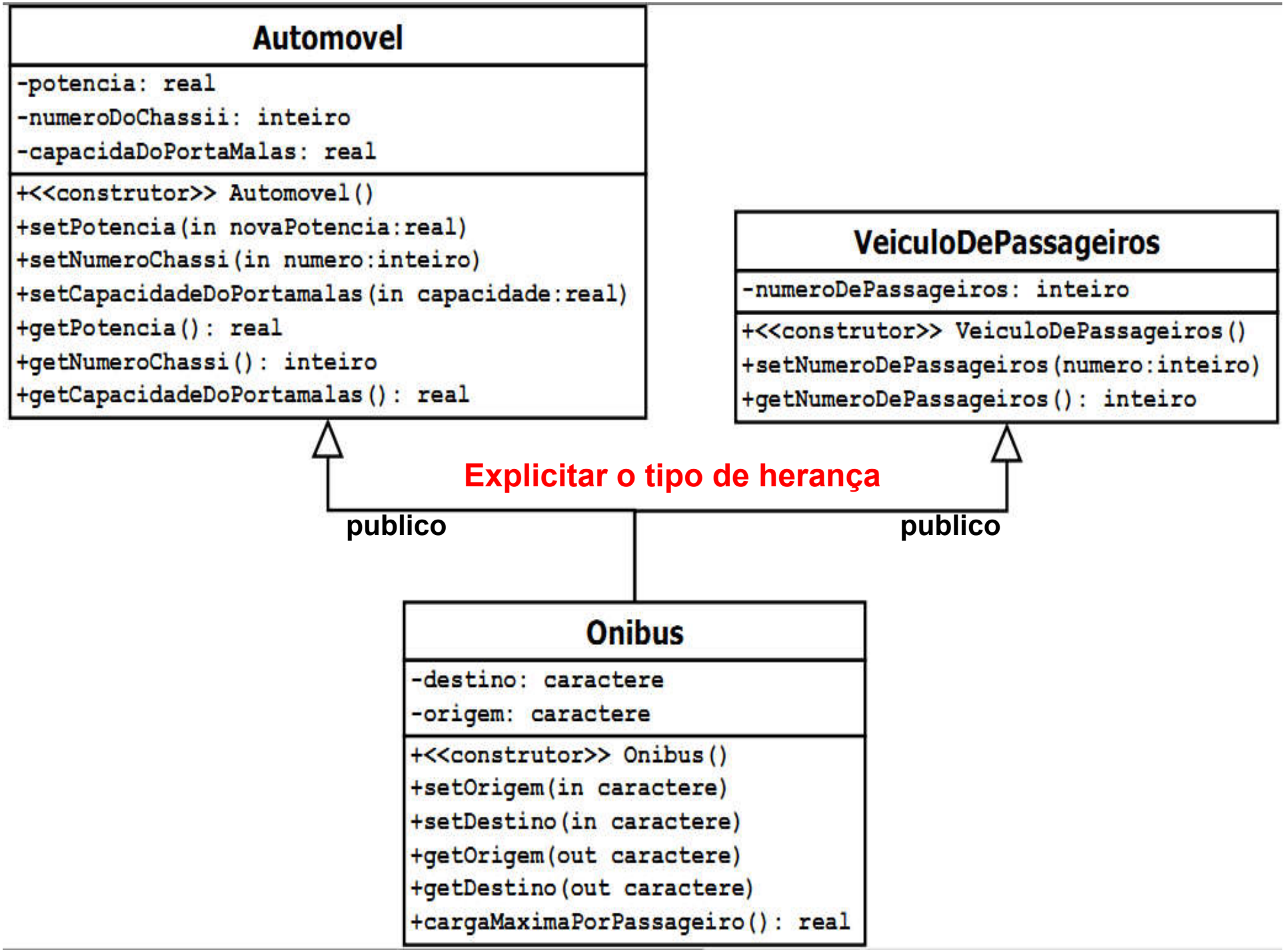
```
class EmbalegemRotulada: public Embalagem, public Rotulo
{
  public:
    void apagar(void);
}
...
void EmbalegemRotulada::apagar(void)
{
  Embalagem::apagar();
  Rotulo::apagar();
}
```

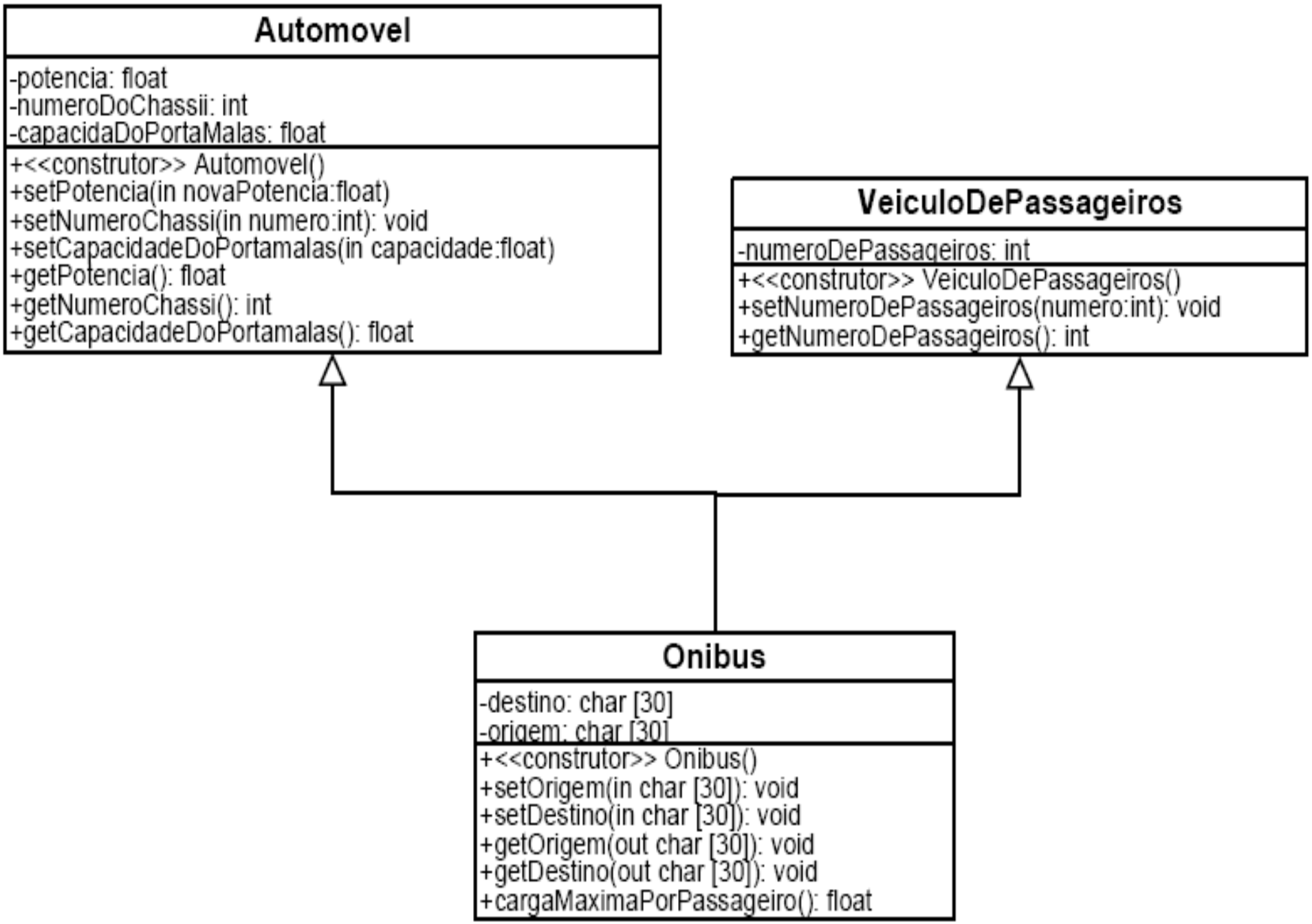
Exercício:

Na oportunidade, em que vimos o conceito de herança múltipla, trabalhamos o exemplo:

De um determinado sistema no qual constavam as classes Automovel e VeiculoDePassageiros, surgindo então a necessidade de definir a classe Onibus.

Sabendo-se que um objeto da classe Onibus é uma instância de Automovel e também é uma instância de VeiculoDePassageiros. Construa um diagrama de classes UML para representar as classes retro aludidas.


Automovel
-potencia: real
-numeroDoChassii: inteiro
-capacidaDoPortaMalas: real
+<<construtor>> Automovel()
+setPotencia(in novaPotencia:real)
+setNumeroChassi(in numero:inteiro)
+setCapacidadeDoPortamalas(in capacidade:real)
+getPotencia(): real
+getNumeroChassi(): inteiro
+getCapacidadeDoPortamalas(): real
VeiculoDePassageiros
-numeroDePassageiros: inteiro
+<<construtor>> VeiculoDePassageiros()
+setNumeroDePassageiros(numero:inteiro)
+getNumeroDePassageiros(): inteiro
Explicitar o tipo de herança
publico
publico
Onibus
-destino: caractere
-origem: caractere
+<<construtor>> Onibus()
+setOrigem(in caractere)
+setDestino(in caractere)
+getOrigem(out caractere)
+getDestino(out caractere)
+cargaMaximaPorPassageiro(): real


Automovel
-potencia: float
-numeroDoChassii: int
-capacidaDoPortaMalas: float
+<<construtor>> Automovel()
+setPotencia(in novaPotencia:float)
+setNumeroChassi(in numero:int): void
+setCapacidadeDoPortamalas(in capacidade:float)
+getPotencia(): float
+getNumeroChassi(): int
+getCapacidadeDoPortamalas(): float
VeiculoDePassageiros
-numeroDePassageiros: int
+<<construtor>> VeiculoDePassageiros()
+setNumeroDePassageiros(numero:int): void
+getNumeroDePassageiros(): int
Onibus
-destino: char [30]
-origem: char [30]
+<<construtor>> Onibus()
+setOrigem(in char [30]): void
+setDestino(in char [30]): void
+getOrigem(out char [30]): void
+getDestino(out char [30]): void
+cargaMaximaPorPassageiro(): float

Linguagem de Programação C++

Com base neste diagrama implemente, em C++, as classes em questão e construa um programa que se utilize da classe Onibus explorando sua interface.