3o-Parcial-Mayo-2008-con-Solucio...

fluxneon

Programación Orientada a Objetos

2º Grado en Ingeniería Informática

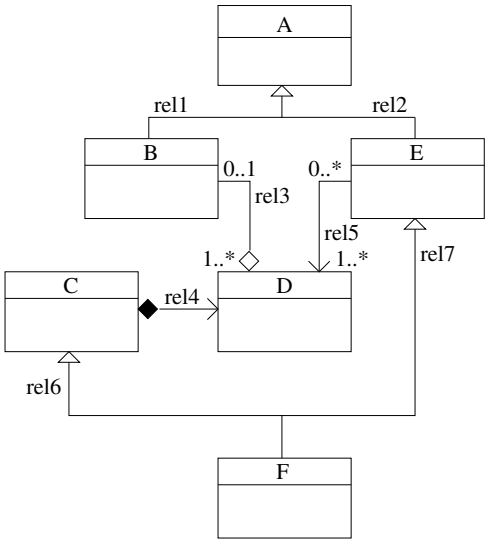
Escuela Superior de Ingeniería
Universidad de Cádiz

Programación Orientada a Objetos

Examen de evaluación continua

Curso 2007-08 **Jueves 29 de mayo de 2008**

1. Dado el siguiente diagrama, defina las clases que aparecen escribiendo exclusivamente los miembros imprescindibles para implementar las relaciones. **5,0 p**




```
#include <set>

class D;

class A {};

class B: public A {
public:
    void rel3(D& d);
private:
    std::set<D*> ds;
};

class D {
public:
    void rel3(B& b);
private:
    B* b;
};

class C {
    D d;
};

class E: public A {
```

```
public:
    void rel5(D& d);
private:
    std::set<D*> ds;
};

class F: public C, public E {};
```

2. Dadas las siguientes definiciones de clases: **5,0 p**

```
#include <iostream>
#include <string>

using namespace std;

class X {
public:
    X(string s = "por omisión") { cout << "Constructor de X: " << s << endl; }
};

class A {
    X x;
public:
    A(): x("A") { cout << "Constructor de A" << endl; }
    void f()    { cout << "Método f() de A" << endl; }
};

class B: virtual public A {
    X x;
public:
    B()      { cout << "Constructor de B" << endl; }
    void f() { cout << "Método f() de B" << endl; }

};

class C: virtual public A {
    X x;
public:
    C()      { cout << "Constructor de C" << endl; }
    void f() { cout << "Método f() de C" << endl; }
};

class D: public B, public C {
    X x;
public:
    D(): x("D") { cout << "Constructor de D" << endl; }
};
```

- ¿Cuántos atributos y cuántos métodos tiene la clase *D*? **0,5 p**
  Tiene cuatro atributos y tres métodos (explicar).
- ¿Hay algún miembro duplicado? **0,5 p**
  No (explicar).
- ¿Cómo se accede a cada uno de los miembros? **0,5 p**
  D d;
  d.A::f(); d.B::f(); d.C::f();
  A los atributos no se puede acceder desde fuera porque son privados.

A continuación, considere el siguiente programa, que incluye dichas definiciones:

```
1  #include "herencia.h"
2
3  int main()
4  {
5      A *pa;
6      B *pb;
7      D d, *pd;
8
9      pd = &d;
10     pa = &d;
11     pa->f();
12     pb = &d;
13     pb->f();
14     d = *pa;
15     pd = (D *)pb;
16     pd->B::f();
17     d.C::f();
18 }
```

- Diga si hay ambigüedades en la función *main*(). En tal caso, resuélvalas. **1,0 p**

  No hay ambigüedades

- Diga si después de resolver las ambigüedades hay errores en la función *main*(). En tal caso, elimínelos. **0,5 p**

  prueba−con−errores.cpp: *In function* `'int main()'`:
  prueba−con−errores.cpp:14: *error*: *no match* **for** `'operator='` *in* `'d = * pa'`
  herencia.h:33: *note*: *candidates are*: *D*& *D*::**operator**=(**const** *D*&)

```
14      // d = *pa;
```

- Diga lo que imprimiría el programa una vez subsanadas las ambigüedades y demás errores. **2,0 p**

  Constructor de X: *A*
  Constructor de A
  Constructor de X: *por omisión*
  Constructor de B
  Constructor de X: *por omisión*
  Constructor de C
  Constructor de X: *D*
  Constructor de D
  Método f() de A
  Método f() de B
  Método f() de B
  Método f() de C