Programación Orientada a Objetos

Tarea 4.1. Polimorfismo

José Fidel Argudo Argudo Francisco Palomo Lozano
Inmaculada Medina Bulo Gerardo Aburruzaga García
Pedro Delgado Pérez

Versión 2.0

Tarea 4.1. Cuestiones

Ejercicio 1

Dado el siguiente código:

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

4  struct A {
5    void mostrar(int i) { cout << i << "␣[A-entero]" << endl; }
6    void mostrar(float f) { cout << f << "␣[A-real]" << endl; }
7  };

9  struct B: A {
10   void mostrar(float f) { cout << f << "␣[B-real]" << endl; }
11 };
```

¿Qué mostrarán cada una de las siguientes llamadas a `mostrar`?

```
A a;
1 - a.mostrar(4);
2 - a.mostrar(4.1);
```

```
B b;
3 - b.mostrar(4);
4 - b.mostrar(4.1);
```

Tarea 4.1. Cuestiones

Ejercicio 2

Sea cierta clase base `B` y una derivada `D`. Ambas tienen definido un cierto método `f()`. Diga si el siguiente código es correcto; y, si lo es, a qué método `f()` se llamaría, dependiendo de que `B::f()` sea o no virtual.

```
1  B b, *bp;
2  D d, *dp;

4  bp = &d;
5  bp->f();

7  dp = &b;
8  dp->f();

10 dp = &d;
11 dp->f();
```

Tarea 4.1. Cuestiones

Ejercicio 3

Indique qué enviará exactamente a la salida estándar el siguiente programa al ejecutarse:

```
1  #include <iostream>
2  struct B {
3    B() { std::cout << "Constructor␣de␣B\n"; }
4    virtual ~B() { std::cout << "Destructor␣de␣B\n"; }
5  };

7  struct D: B {
8     D() { std::cout << "Constructor␣de␣D\n"; }
9    ~D() { std::cout << "Destructor␣de␣D\n"; }
10 };

12 int main() {
13   B *pb = new D;
14   delete pb;
15 }
```

¿Qué destructores son virtuales? ¿Cambiaría algo si quitamos la palabra `virtual` del destructor de B?

Tarea 4.1. Cuestiones

Ejercicio 4

¿Cambiaría el comportamiento de la clase `cuadrado` si le quitamos el miembro `area()`?

```
1  class rectangulo {
2  public:
3    rectangulo(double a, double l): ancho(a), largo(l) {}
4    virtual double area() { return ancho * largo; }
5    virtual ~rectangulo() = default;
6  private:
7    double ancho, largo;
8  };

10 class cuadrado: public rectangulo {
11 public:
12   cuadrado (double l): rectangulo(l, l) {}
13   double area() { return rectangulo::area(); }
14 };
```