COMPOSICIÓN Y HERENCIA

Composición y herencia.

Introducción (I)

- Mecanismos de abstracción. Permite compartir código común entre objetos.
- Ventajas:
  - Ahorro de costes en el desarrollo de un programa.
  - Mejora calidad del código.
  - Reduce el número de errores
  - Aumento de la portabilidad y mantenimiento del código.

Composición y herencia.

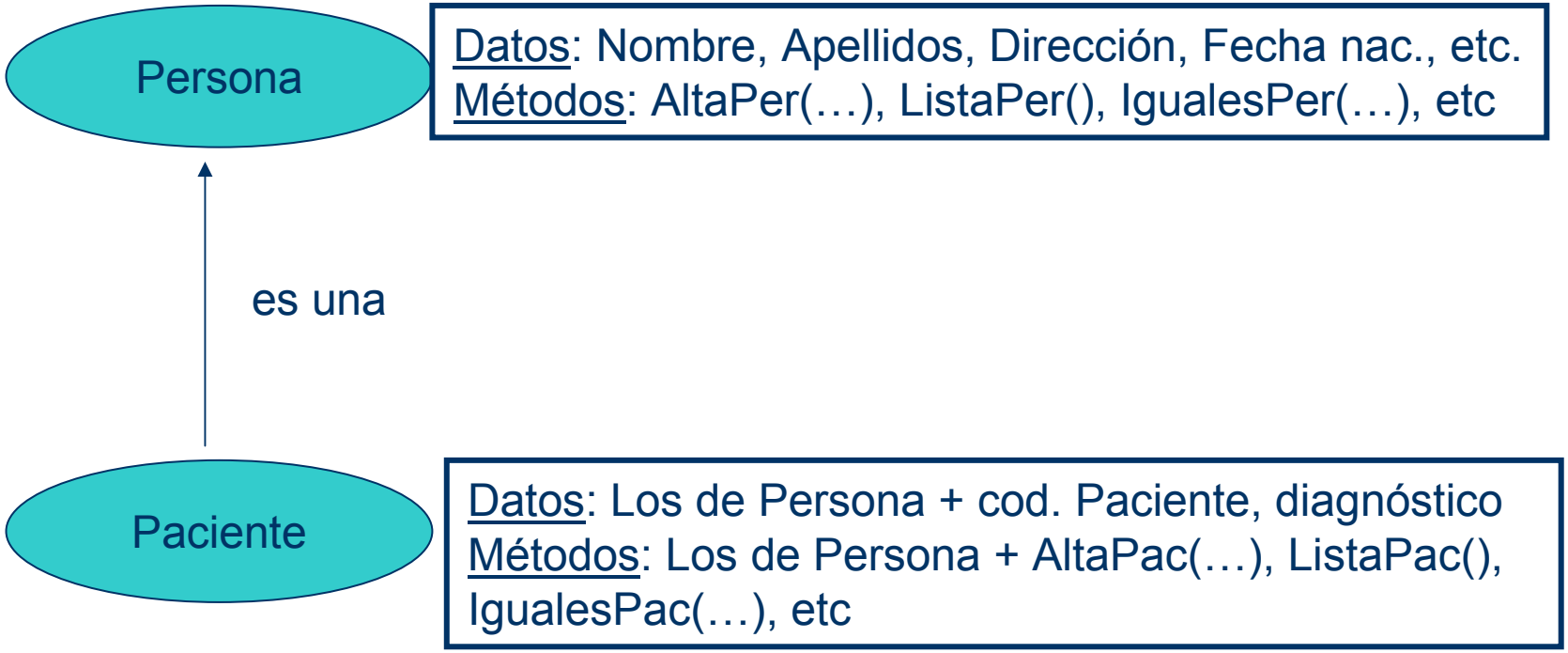
Introducción (II)

- **Composición**:
  - Relación "***tiene-un***".
  - Un objeto de una clase contiene uno o más objetos de otras clases.
- **Herencia**:
  - Relación "***es-un***" entre dos tipos de clases:
    - Clase Base (superclase): Clase general de la cual se hereda.
    - Clase Derivada (subclase): Especialización de una o varias clases base.
  - La clase derivada = los datos y funciones de la clase base + los suyos propios.

Composición y herencia.

Introducción (III)

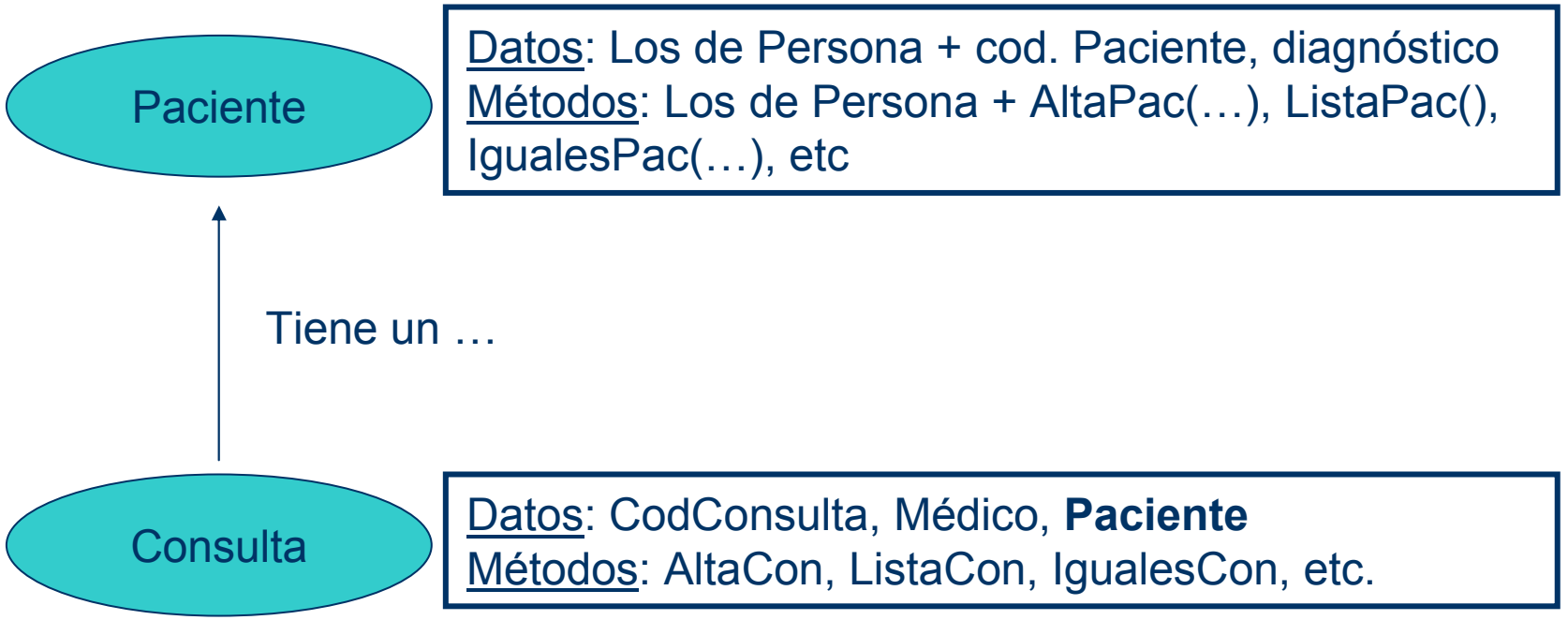
- Ejemplo de herencia:



Composición y herencia.

Introducción (IV)

- Ejemplo de composición:



Composición y herencia.

Composición (I)

- Una clase tiene objetos de otras clases por valor (layering, en C++) o referencia.
- Sólo acceso a parte pública de de los objetos contenidos.
- ¿Cómo y cuando se **construyen** los objetos que contienen?
- ¿Cómo y cuando se **destruyen** los objetos que contienen?

Composición y herencia.

Composición (II)

Ejemplo 6.1

```
1  #include <iostream>
2
3  using namespace std;
4
5  class A {
6    public:
7      A() {
8        cout << "A()" << endl;
9      };
10
11     ~A() {
12       cout << "~A()" << endl;
13     };
14 };
15
```

```
16 class B {
17   public:
18     B() {
19       cout << "B()" << endl;
20     };
21
22     ~B() {
23       cout << "~B()" << endl;
24     };
25
26   private:
27     A a;
28 };
29
30 int main(void) {
31   B b;
32
33   cout << "Llama destructor" << endl;
34   b.~B();
35   cout << "Fin" << endl;
36
37   return 0;
38 }
```

A()
B()

~B()
~A()

~B()
~A()

Composición y herencia.

Inicialización de los objetos miembro (I)

- Sólo en los constructores.
- Entre lista parámetros del constructor y {:

  *Constructor (parametros): **obj1(...), obj2(...), ...{***
- Si no se da un inicializador → constructor por defecto.

Ejemplo 6.2

```
1  #ifndef __TLINEA__
2  #define __TLINEA__
3
4  #include <iostream>
5
6  using namespace std;
7
8  #include "tcoordenada.h"
9
10 class TLinea {
11   friend ostream& operator<<(ostream &, const TLinea &);
12
13   public:
14     TLinea();
15     TLinea(const TCoordenada &, const TCoordenada &);
16     TLinea(const TLinea &);
17     ~TLinea();
18     float Longitud(void);
19
20   private:
21     TCoordenada p1, p2;
22 };
23
24 #endif
```

Composición y herencia.

Inicialización de los objetos miembro (II)

Ejemplo 6.3

```
1  #include "tlinea.h"
2
3  TLinea::TLinea() {
4    p1.x = 0;
5    p1.y = 0;
6    p1.z = 0;
7
8    p2.x = 0;
9    p2.y = 0;
10   p2.z = 0;
11 }
12
13 TLinea::TLinea(const TCoordenada & a, const TCoordenada & b) {
14   p1.x = a.x;
15   p1.y = a.y;
16   p1.z = a.z;
17
18   p2.x = b.x;
19   p2.y = b.y;
20   p2.z = b.z;
21 }
22
23 TLinea::TLinea(const TLinea & l) {
24   p1.x = l.p1.x;
25   p1.y = l.p1.y;
26   p1.z = l.p1.z;
27
28   p2.x = l.p2.x;
29   p2.y = l.p2.y;
30   p2.z = l.p2.z;
31 }
32
33 TLinea::~TLinea() {
34   p1.x = 0;
35   p1.y = 0;
36   p1.z = 0;
37
38   p2.x = 0;
39   p2.y = 0;
40   p2.z = 0;
41 }
42
43 float
44 TLinea::Longitud(void) {
45   return Distancia(p1, p2);
46 }
47
48 ostream&
49 operator<<(ostream &s, const TLinea &obj) {
50   s << "(" << obj.p1 << ", ";
51   s << obj.p2 << ")";
52
53   return s;
54 }
```

Recordad: se puede acceder a la parte privada de TCoordenada puesto que TLinea se declaró amiga (friend) de aquella.

Composición y herencia.

Inicialización de los objetos miembro (III)

Ejemplo 6.4

```
1  TLinea::TLinea() {
2    // No hace nada
3  }
4
5  TLinea::TLinea(const TCoordenada & a, const TCoordenada & b): p1(a), p2(b) {
6    // No hace nada
7  }
8
9  TLinea::TLinea(const TLinea & l): p1(l.p1), p2(l.p2) {
10   // No hace nada
11 }
12
13 TLinea::~TLinea() {
14   // No hace nada
15 }
```

Se llama al constructor por defecto

Se llama al constructor copia

Ejemplo 6.5

```
1  #include <iostream>
2
3  using namespace std;
4
5  #include "tcoordenada.h"
6  #include "tlinea.h"
7
8  int
9  main(void)
10 {
11   TCoordenada p1;
12   TCoordenada p2(0, 3, 4);
13   TCoordenada p3(p2);
14
15   TLinea l1, l2(p1, p2);
16
17   cout << l1 << endl;
18   cout << l1.Longitud() << endl;
19   cout << l2 << endl;
20   cout << l2.Longitud() << endl;
21
22   return 0;
23 }
```

((0, 0, 0), (0, 0, 0))

0

((0, 0, 0), (0, 3, 4))

5

Composición y herencia.

Herencia (I)

- Mecanismo que permite la reutilización de código.
- Definición de una clase nueva (**clase derivada**) a partir de una que ya existe (**clase base**).
- Jerarquía de clases.
- Una clase deriva de:
  - Una clase base: **Herencia simple**
  - Múltiples clases base: **Herencia múltiple**

Composición y herencia.

Herencia (II)

- Tipos de herencia:
  - **Pública**: La más habitual. Se hereda todo tal cual.
  - **Protegida**: Para control de herencia de la parte pública a un primer nivel.
  - **Privada**: Alternativa a la composición. Todo lo heredado queda como privado en la clase derivada.
- No acceso a la parte privada de la clase base desde las clases derivadas.

Composición y herencia.

Herencia (III)

- Cuadro resumen relación tipos de herencia y modos de acceso en una clase base.

ACCESOS / HERENCIAS	Público	Protegido	Privado
Pública	Público	Protegido	No accesible
Protegida	Protegido	Protegido	No accesible
Privada	Privado	Privado	No accesible

Composición y herencia.

Herencia (IV)

- Ejemplo: Clase TCoordenadaV

Ejemplo 6.6

```
1  #ifndef __TCOORDENADAV__
2  #define __TCOORDENADAV__
3
4  #include "tcoordenada.h"
5
6  class TCoordenadaV: public TCoordenada {
7    friend ostream& operator<<(ostream &, const TCoordenadaV &);
8
9    public:
10     TCoordenadaV();
11     TCoordenadaV(int, int, int);
12     TCoordenadaV(int, int, int, int);
13     TCoordenadaV(const TCoordenadaV &);
14     ~TCoordenadaV();
15
16   private:
17     int valor;
18 };
19
20 #endif
```

Composición y herencia.

Herencia (V)

Ejemplo 6.7

```
1  #include "tcoordenadav.h"
2
3  TCoordenadaV::TCoordenadaV(): TCoordenada() {
4    valor = 0;
5  }
6
7  TCoordenadaV::TCoordenadaV(int a, int b, int c): TCoordenada(a, b, c) {
8    valor = 0;
9  }
10
11 TCoordenadaV::TCoordenadaV(int a, int b, int c, int v):
12                 TCoordenada(a, b, c) {
13   valor = v;
14 }
15
16 TCoordenadaV::TCoordenadaV(const TCoordenadaV &obj): TCoordenada(obj) {
17   valor = obj.valor;
18 }
19
20 TCoordenadaV::~TCoordenadaV() {
21   // Se invoca automáticamente al destructor de TCoordenada
22   valor = 0;
23 }
24
25 ostream&
26 operator<<(ostream &s, const TCoordenadaV &obj) {
27   s << (TCoordenada) obj << ": " << obj.valor;
28
29   return s;
30 }
```

Conversión de tipo (casting)

Composición y herencia.

Herencia (VI)

Ejemplo 6.8

```
1  #include <iostream>
2
3  using namespace std;
4
5  #include "tcoordenada.h"
6  #include "tcoordenadav.h"
7
8  int
9  main(void)
10 {
11    TCoordenadaV p1;
12    TCoordenadaV p2(10, 20, 30);
13    TCoordenadaV p3(40, 50, 60, -100);
14    TCoordenadaV p4(p3);
15
16    cout << p1 << endl;
17    cout << p2 << endl;
18    cout << p3 << endl;
19    cout << p4 << endl;
20
21    return 0;
22 }
```

```
(0, 0, 0): 0
(10, 20, 30): 0
(40, 50, 60): -100
(40, 50, 60): -100
```