7.F. Polimorfismo.

3. Conversión de objetos.

Como ya has visto, en principio no se puede acceder a los **miembros específicos** de una **subclase** a través de una **referencia** a una **superclase**. Si deseas tener acceso a todos los métodos y atributos específicos del objeto **subclase** tendrás que realizar una **conversión explícita** (**casting**) que convierta la referencia más general (**superclase**) en la del tipo específico del objeto (**subclase**).

Para que puedas realizar conversiones entre distintas clases es obligatorio que exista una relación de **herencia** entre ellas (una debe ser clase derivada de la otra). Se realizará una **conversión implícita o automática** de **subclase** a **superclase** siempre que sea necesario, pues un objeto de tipo **subclase** siempre contendrá toda la información necesaria para ser considerado un objeto de la **superclase**.

Ahora bien, la conversión en sentido contrario (de **superclase** a **subclase**) debe hacerse de forma **explícita** y según el caso podría dar lugar a errores por falta de información (atributos) o de métodos. En tales casos se produce una **excepción** de tipo `ClassCastException`.

Por ejemplo, imagina que tienes una clase **A** y una clase **B**, **subclase** de **A**:

```
class ClaseA {

    public int atrib1;

}

class ClaseB extends ClaseA {

    public int atrib2;

}
```

A continuación declaras una variable referencia a la clase **A** (**superclase**) pero sin embargo le asignas una referencia a un objeto de la clase **B** (**subclase**) haciendo uso del **polimorfismo**:

```
A obj;  // Referencia a objetos de la clase A

obj= new B (); // Referencia a objetos clase A, pero apunta realmente a objeto clase B (polimorfismo)
```

El objeto que acabas de crear como **instancia de la clase B** (**subclase** de **A**) contiene más información que la que la referencia **obj** te permite en principio acceder sin que el compilador genere un error (pues es de clase **A**). En concreto los objetos de la clase **B** disponen de **atrib1** y **atrib2**, mientras que los objetos de la clase **A** sólo de **atrib1**. Para acceder a esa información adicional de la clase especializada (**atrib2**) tendrás que realizar una **conversión explícita** (**casting**):

```
// Casting del tipo A al tipo B (funcionará bien porque el objeto es realmente del tipo B)

System.out.printf ("obj.atrib2=%d\n", ((B) obj).atrib2);
```

Sin embargo si se hubiera tratado de una **instancia de la clase A** y hubieras intentado acceder al miembro **atrib2**, se habría producido una **excepción** de tipo `ClassCastException`:

```
A obj;  // Referencia a objetos de la clase A

obj= new A ();  // Referencia a objetos de la clase A, y apunta realmente a un objeto de la clase A

// Casting del tipo A al tipo B (puede dar problemas porque el objeto es realmente del tipo A):

// Funciona (la clase A tiene atrib1)

System.out.printf ("obj.atrib2=%d\n", ((B) obj).atrib1);
```

```
// ¡Error en ejecución! (la clase A no tiene atrib2). Producirá una ClassCastException.
```

```
System.out.printf ("obj.atrib2=%d\n", ((B) obj).atrib2);
```