

# **UTILIZACIÓN AVANZADA DE CLASES**

**ADOLFO SANZ DE DIEGO**

**JUNIO DE 2011**

# 1 ¿DISEÑO INCORRECTO?

- ¿Cómo sabemos si nuestro diseño es incorrecto?
  - las clases son difíciles de cambiar
  - las clases son difíciles de reutilizar
  - las clases son difíciles de usar
  - las clases tienen código repetido
  - las clases dejan de funcionar sin motivo aparente

## 2 MEJORARLO: PRINCIPIOS SOLID

S	SRP	(The <b>S</b> ingle Responsibility Principle o <b>Principio de</b> <b>Responsabilidad</b> <b>Única</b> )	Las clases se diseñan sólo para un propósito.	Tom DeMarco	1
---	-----	---	--	----------------	---

---

○	OCP	(The Open/Closed Principle o Principio Abierto/Cerrado)	Las clases debe permitir ser extendidas, sin necesidad de ser modificadas.	Bertrand Meyer	1
---	-----	---	---	-------------------	---

---

L	LSP	(The Liskov Substitution Principle o Principio de Sustitución de Liskov)	Las clases padre siempre deben poder ser sustituidas por sus clases hijas y viceversa.	Barbara Liskov	1
---	-----	--	--	----------------	---

---

I	ISP	(The Interface Segregation Principle o <b>Principio de Segregación de Interfaces</b> )	Las clases no deben ser forzados a depender de interfaces con métodos que no utilizan.	Robert C. Martin	1
---	-----	--	--	------------------	---

---

<b>D</b>	DIP	(The <b>D</b> ependency Inversion Principle o <b>Principio de</b> <b>Inversión de</b> <b>Dependencias)</b>	Los clientes deben delegar la gestión de las instancias de objetos a las librerías.	Martin Fowler	2
----------	-----	--	--	------------------	---

# 3 ¿CÓMO UTILIZAR NUESTRAS CLASES?

- **Colaboración**
  - Cuando 2 o más clases colaboran de distintas formas (ver teoría POO)
- **Herencia**
  - Cuando una clase hija extiende una clase padre.
- **Error común**
  - Aplicar herencia para todo.



# 4 HERENCIA

- En Java no hay herencia múltiple pero una clase puede implementar varias Interfaces.
- Las **Interfaces** son clases que sólo definen métodos (comportamientos) y que obligan a las clases hijas a implementarlos.
- Las **clases abstractas** son clases que no se pueden instanciar, osea que sólo se pueden heredar. Implementan varios métodos, y definen otros como abstractos, obligando a las clases hijas a implementarlos.

# 5 CLASES PADRE Y CLASES HIJAS

- El operador **this** hace referencia a la instancia concreta del objeto.
- El operador **super** hace referencia a la clase padre.
- Los métodos no marcados como final y/ como private, se pueden sobrescribir en las clases hijas (**polimorfismo**).