Resumen

Inicial	Acrónimo	Concepto
S	SRP	**Principio de responsabilidad única (*Single responsibility principle*)**la noción de que un objeto solo debería tener una única razón para cambiar.
O	OCP	**Principio de abierto/cerrado (*Open/closed principle*)**la noción de que las “entidades de software … deben estar abiertas para su extensión, pero cerradas para su modificación”.
L	LSP	**Principio de sustitución de Liskov (*Liskov substitution principle*)**la noción de que los “objetos de un programa deberían ser reemplazables por instancias de sus subtipos sin alterar el correcto funcionamiento del programa”. Véase también diseño por contrato.
I	ISP	**Principio de segregación de la interfaz (*Interface segregation principle*)**la noción de que “muchas interfaces cliente específicas son mejores que una interfaz de propósito general”.
D	DIP	**Principio de inversión de la dependencia (*Dependency inversion principle*)**la noción de que se debe “depender de abstracciones, no depender de implementaciones”. La Inyección de Dependencias es uno de los métodos que siguen este principio.

SRP -Principio de Responsabilidad Unica

Cada clase debe tener una única responsabilidad, y que esta debe estar contenida únicamente en la clase.

OCP -Principio Abierto/Cerrado

Una entidad de software debe quedarse abierta para su extensión, pero cerrada para su modificación. (se basan en la herencia para resolver el aparente dilema)

LSP -Principio de Sustitución de Liskov

Objetos de un programa deberían ser reemplazables por instancias de sus subtipos sin alterar el correcto funcionamiento del programa.

Si S es un subtipo de T, entonces los objetos de tipo T en un programa de computadora pueden ser sustituidos por objetos de tipo S (es decir, los objetos de tipo S pueden sustituir objetos de tipo T), sin alterar ninguna de las propiedades deseables de ese programa

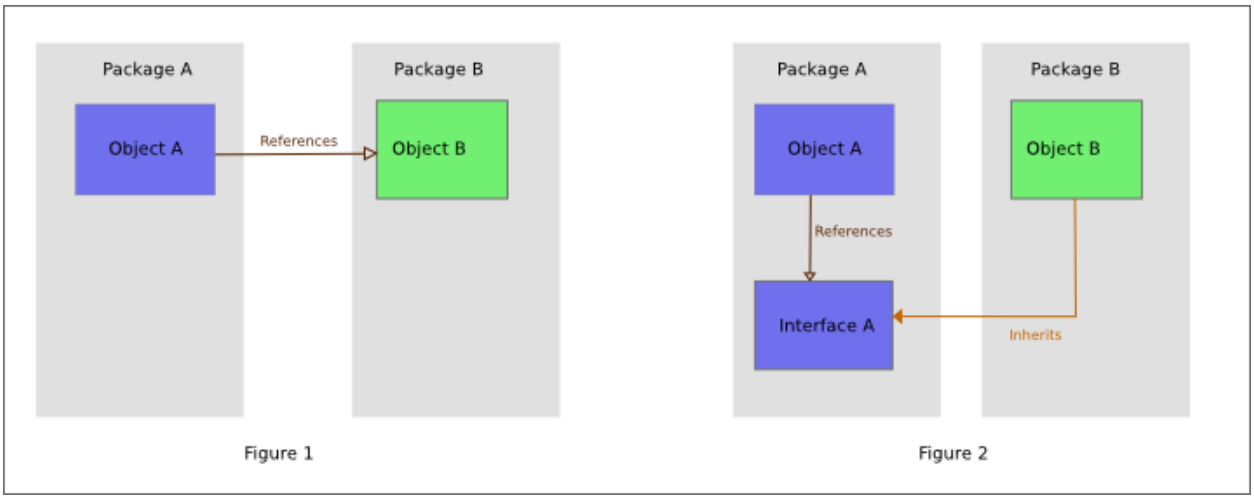
ISP -Principio de Segregación de Interfaz

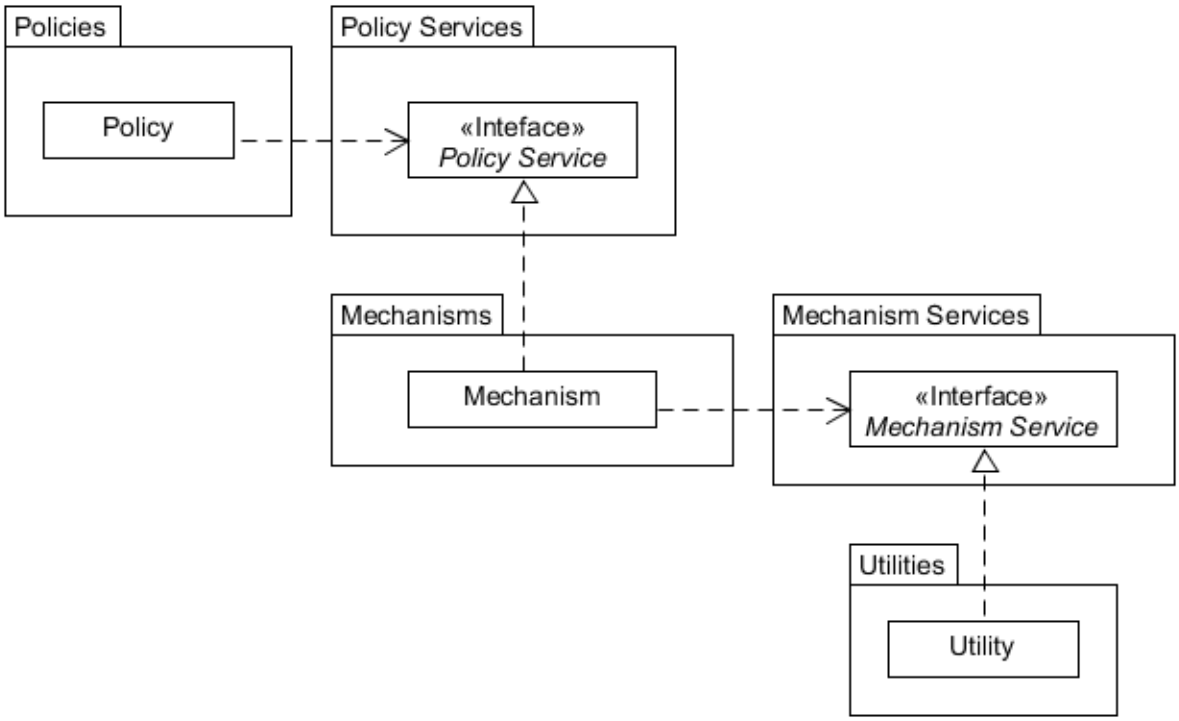
Muchas interfaces cliente específicas son mejores que una interfaz de propósito general

Establece que los clientes de un programa dado solo deberían conocer de este aquellos métodos que realmente usan, y no aquellos que no necesitan usar. Utilizar una interfaz o clase abstracta puede impedir este "arrastre" de dependencias.

DIP -Principio de la inversión de la dependencia

Depender de abstracciones, no depender de implementaciones.




Policies
Policy
Policy Services
«Inteface»
Policy Service
Mechanisms
Mechanism
Mechanism Services
«Interface»
Mechanism Service
Utilities
Utility