Diagramas de paquetes

1. Paquetes y diagramas de paquetes
2. Paquetes anidados
3. Relaciones de dependencia
4. Generalización de paquetes
5. Dependencias cíclicas
6. Reglas para construir paquetes

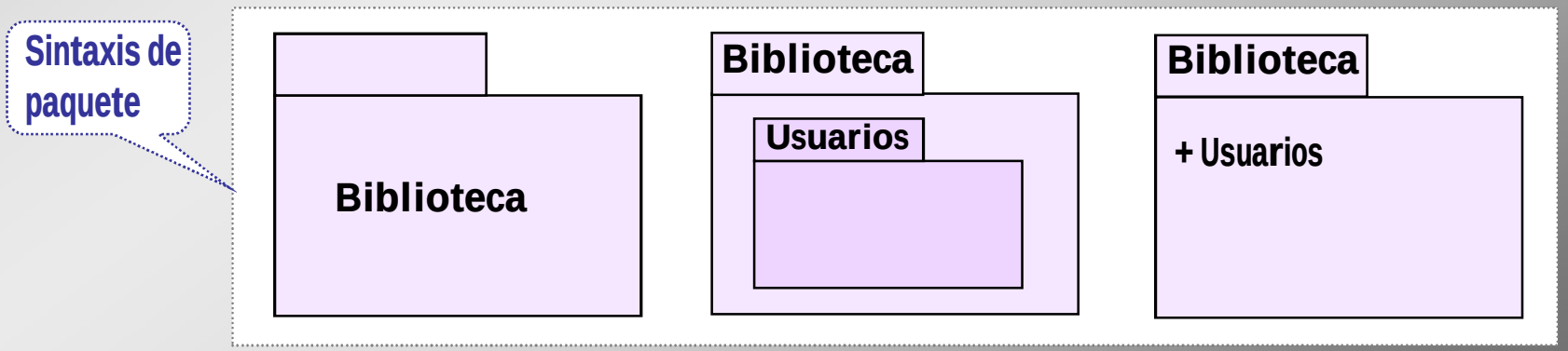
PAQUETES Y DIAGRAMAS DE PAQUETES

¿Qué es un paquete?

Mecanismo de propósito genérico para organizar y encapsular elementos de modelo (incluidos otros paquetes) y diagramas

- Proporciona un espacio de nombres dentro del cual todos los nombres son únicos
- Cada elemento es exclusivo de un paquete
- Cada paquete tiene un nombre que lo distingue de otros paquetes
  - Nombre simple
  - Nombre cualificado

    Formado por el nombre del paquete precedido por los nombres de los paquetes en los que se encuentra, separados por dos puntos dobles (Biblioteca::Usuarios)

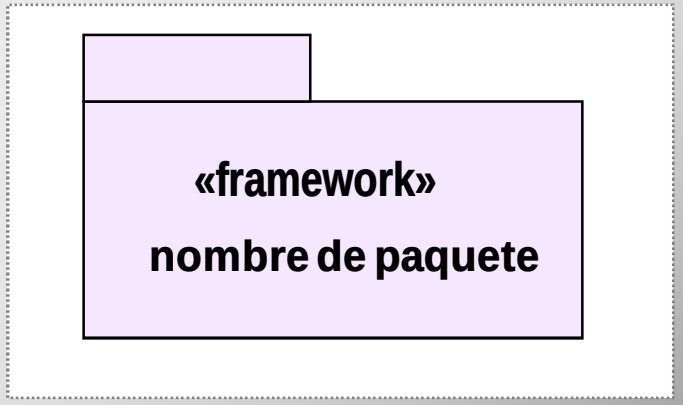


PAQUETES Y DIAGRAMAS DE PAQUETES

Estereotipos

UML proporciona dos estereotipos estándar para adaptar la semántica de los paquetes a fines específicos

Estereotipo	Semántica
«framework»	Un paquete que contiene elementos de modelo que especifican una arquitectura reutilizable
«modelLibrary»	Un paquete que contiene elementos que están pensados para ser reutilizados por otros paquetes



PAQUETES Y DIAGRAMAS DE PAQUETES

Diagrama de paquetes

Diagrama de UML usado para describir la estructura de un sistema en base a agrupaciones lógicas y dependencias entre éstas

- Los elementos básicos de un diagrama de paquetes: **paquetes y sus relaciones**

Se usan para

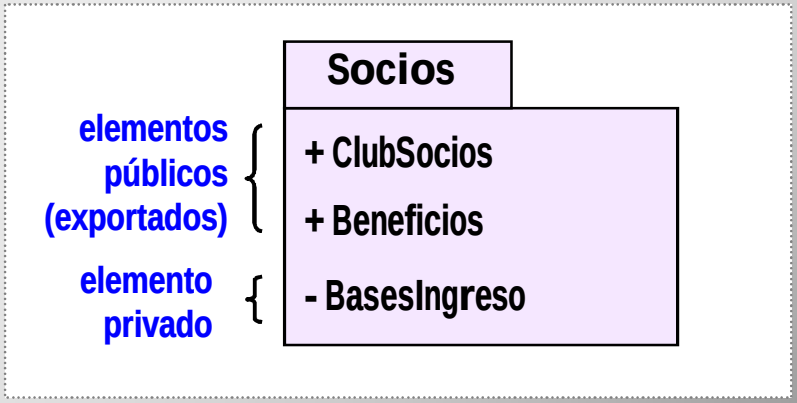
- **Agrupar elementos relacionados semánticamente**
- **Definir un "límite semántico" en el modelo**
- **Proporcionar unidades para trabajo en paralelo y gestión de la configuración**

PAQUETES Y DIAGRAMAS DE PAQUETES

Visibilidad del contenido de un paquete

La visibilidad de un elemento indica si es visible o no a los clientes del paquete

Visibilidad	Símbolo	Semántica
Público	+	Los elementos con visibilidad pública son visibles a elementos fuera del paquete; se exportan por el paquete
Privado	-	Los elementos con una visibilidad privada sólo son visibles por otros elementos de ese paquete

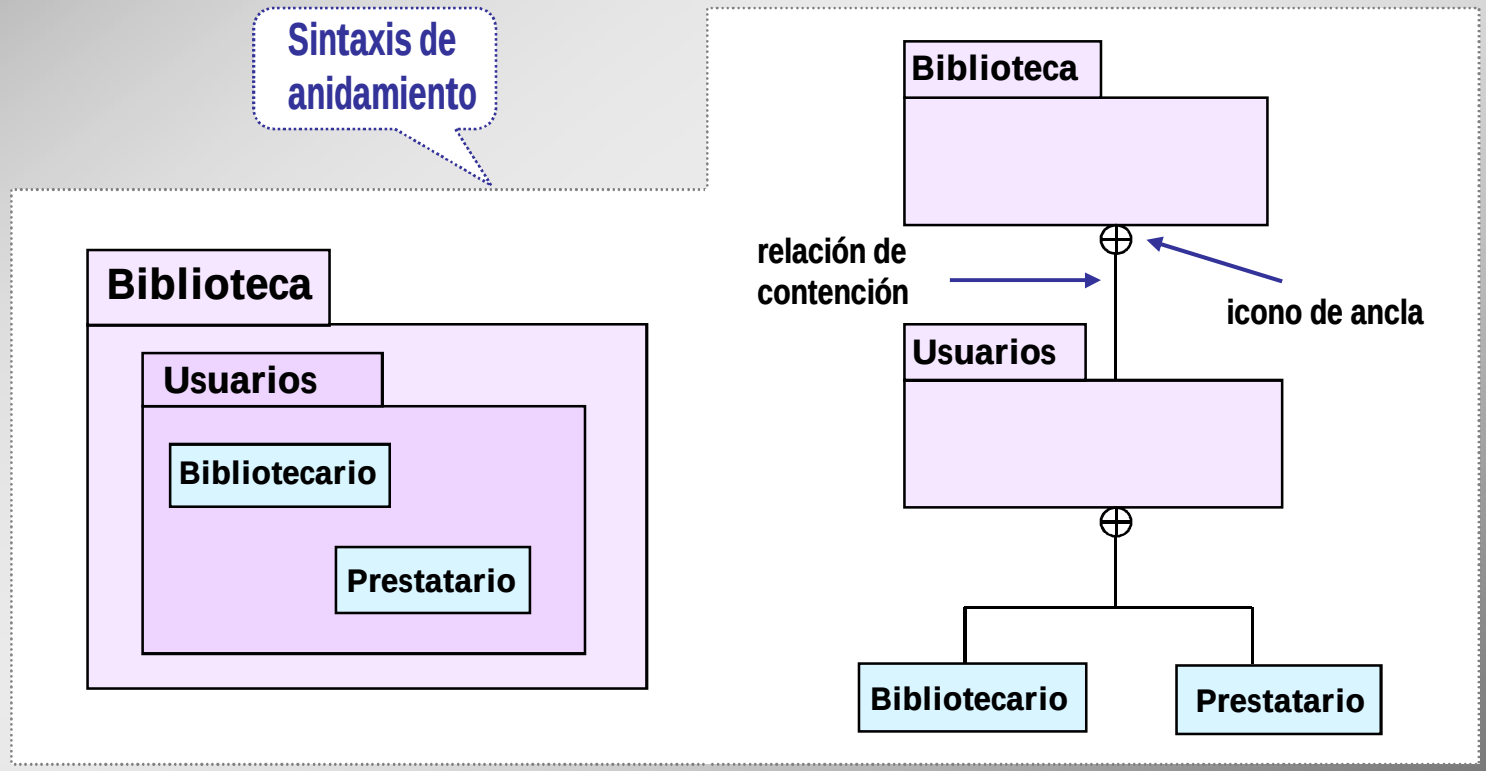


PAQUETES ANIDADOS

Los paquetes pueden estar anidados dentro de otros paquetes en cualquier nivel

- Los paquetes anidados tienen acceso al espacio de nombres de su paquete propietario
- El paquete contenedor debe utilizar nombres cualificados para acceder a los contenidos de sus paquetes
- Suelen ser suficientes dos o tres niveles de anidamiento

  Más de eso, el modelo puede ser difícil de leer y navegar



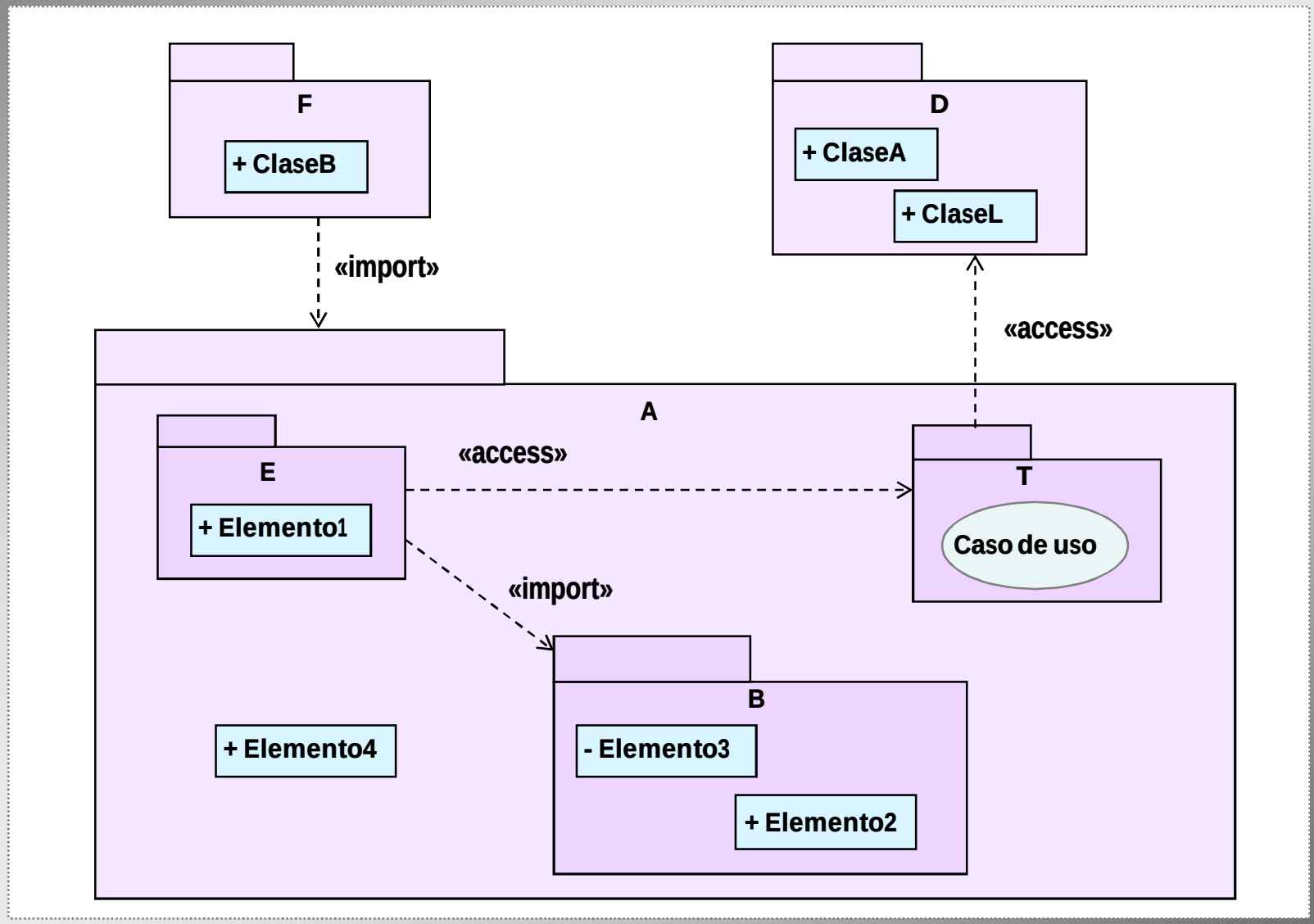
RELACIONES DE DEPENDENCIA

Los paquetes pueden estar relacionados entre sí por una dependencia

Tipo de dependencia	Semántica
«use»	El paquete destino utiliza un elemento público del paquete origen. Por defecto, cuando no hay estereotipo hay una dependencia de este tipo
«import»	Los elementos públicos del espacio de nombres del paquete origen se añaden como elementos públicos al espacio de nombres del destino
«access»	Los elementos públicos del espacio de nombres del paquete origen se añaden como elementos privados al espacio de nombres del destino

RELACIONES DE DEPENDENCIA

Ejemplo de relaciones de dependencia

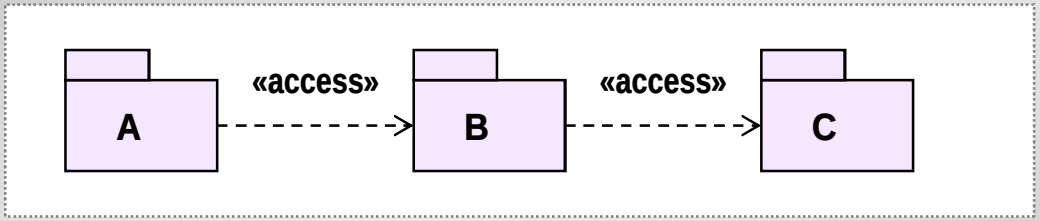


RELACIONES DE DEPENDENCIA

Transitividad

Si existe una relación entre un elemento A y un elemento B y una relación entre el elemento B y el elemento C, entonces existe una relación implícita entre el elemento A y el elemento C

- La relación «import» es **transitiva**
- La relación «access» **no** es **transitiva**



La ausencia de transitividad significa que

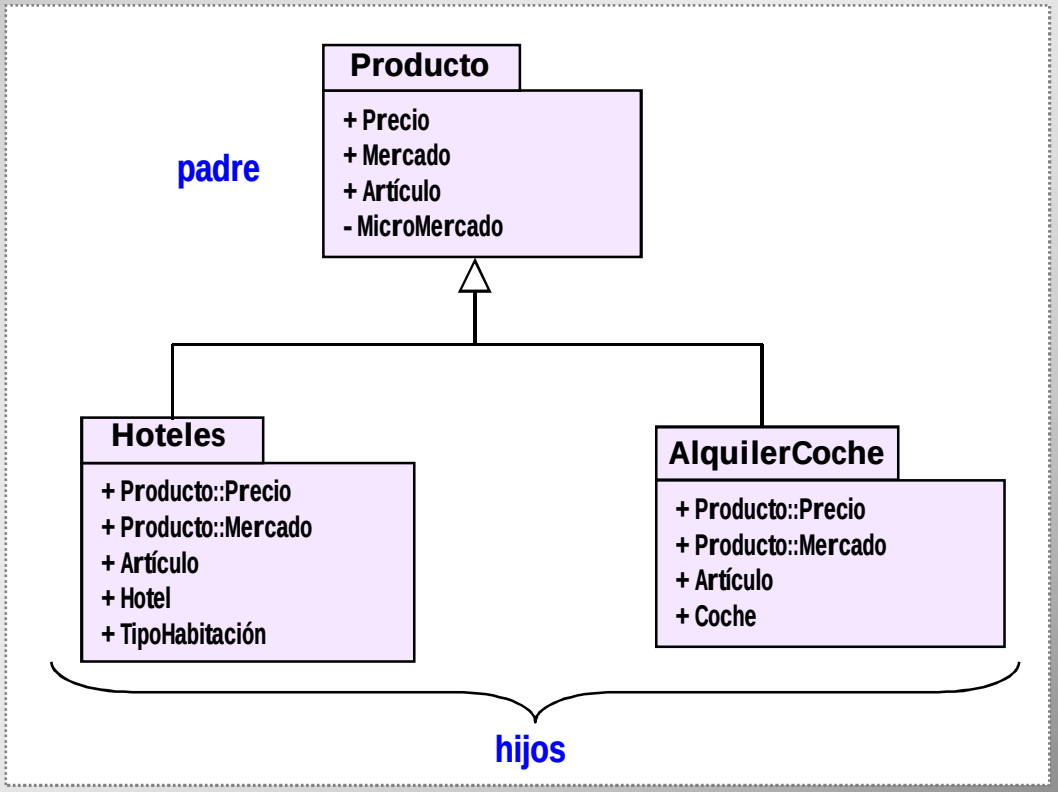
- Los elementos públicos en el paquete C se convierten en elementos privados en el paquete B
- Los elementos públicos en el paquete B se convierten en elementos privados en el paquete A
- Los elementos en el paquete A no pueden acceder a elementos del paquete C

GENERALIZACIÓN DE PAQUETES

Los paquetes hijo más especializados heredan los elementos públicos de su paquete padre.

Los paquetes hijo pueden

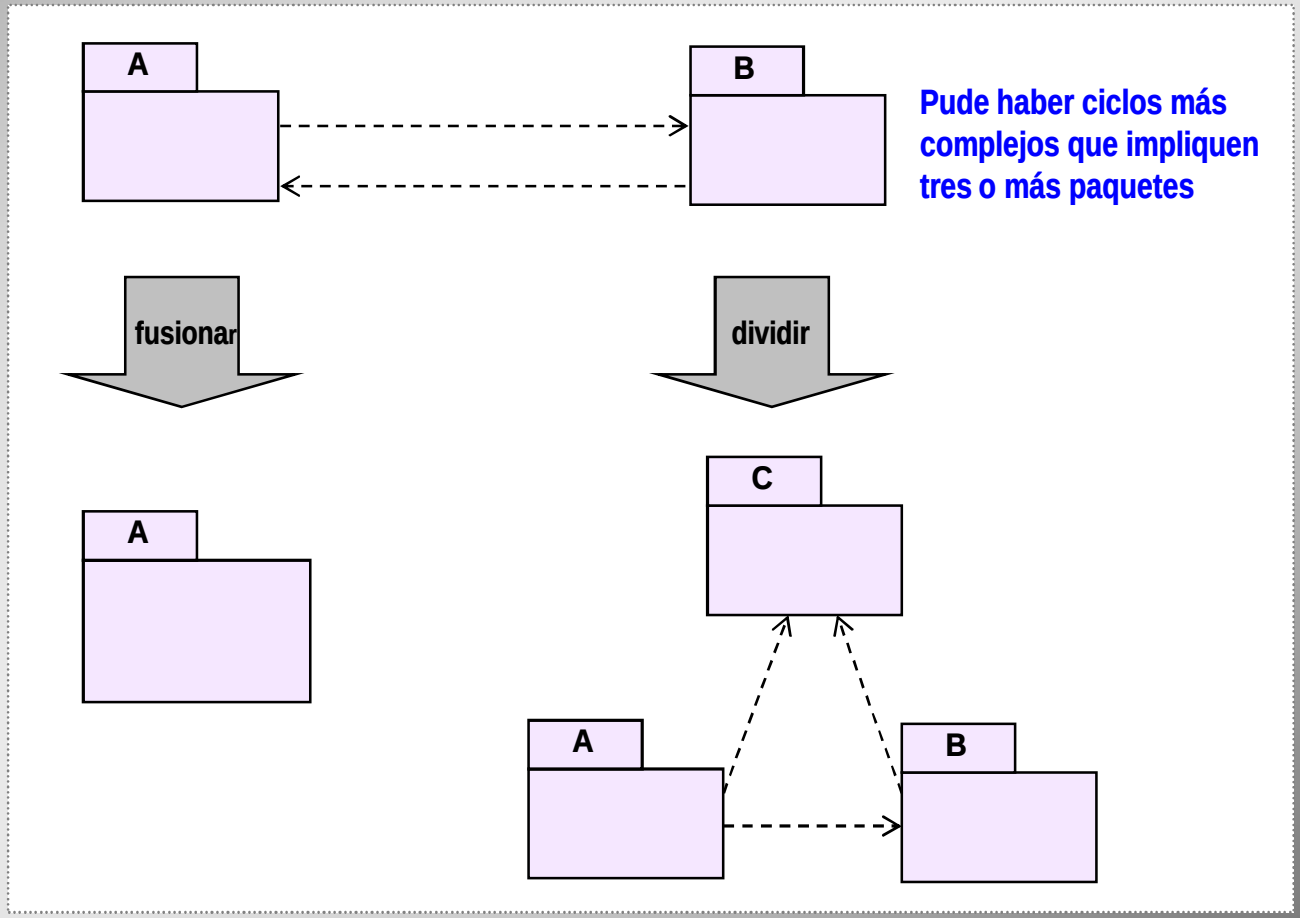
- Añadir elementos nuevos
- Anular elementos al proporcionar un elemento alternativo con igual nombre



DEPENDENCIAS CÍCLICAS

Situación en la que un paquete A depende de alguna forma de otro paquete B y viceversa

Formas de evitarlas



REGLAS PARA CONSTRUIR PAQUETES

Un paquete bien estructurado debe

- Ser lo más cohesivo posible
- Estar poco acoplado con otros paquetes
- Contener un grupo de clases estrechamente relacionadas
- Poseer un conjunto equilibrado de elementos
- Evitar dependencias cíclicas con otros paquetes