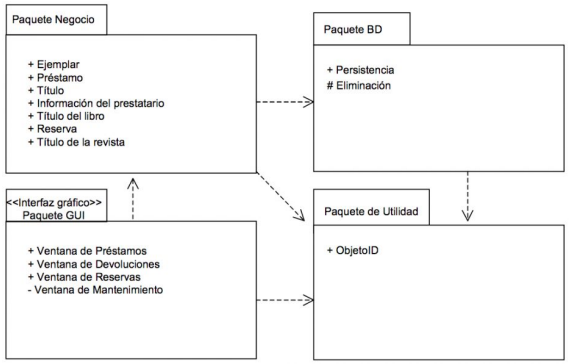


UML Diagrama de paquetes

Ángel González M.

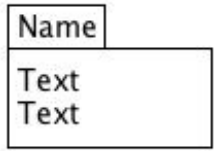
Que es un paquete

Un paquete es un mecanismo utilizado para agrupar elementos de UML

Permite organizar los elementos modelados con UML, facilitando de ésta forma el manejo de los modelos de un sistema complejo

Define un espacio de nombres: Dos elementos de UML pueden tener el mismo nombre, con tal y estén en paquetes distintos

En este sentido, son similares a los namespaces en C++ o a los paquetes en Java



El diagrama de paquetes

Permiten dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales

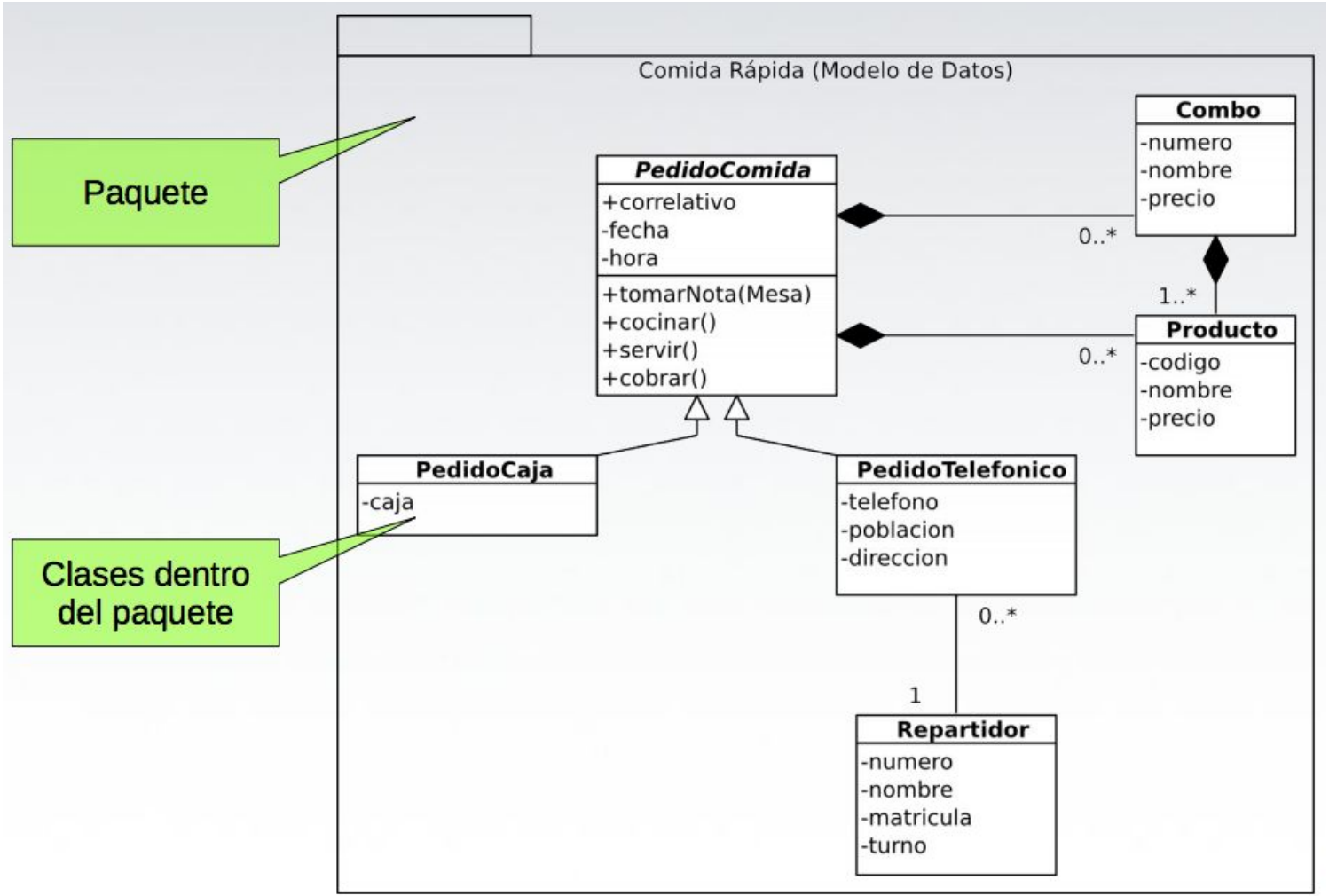
En general, pueden tener una interfaz (métodos de clases e interfaces exportadas) y una realización de éstas interfaces (clases internas que implementan dichas interfaces)

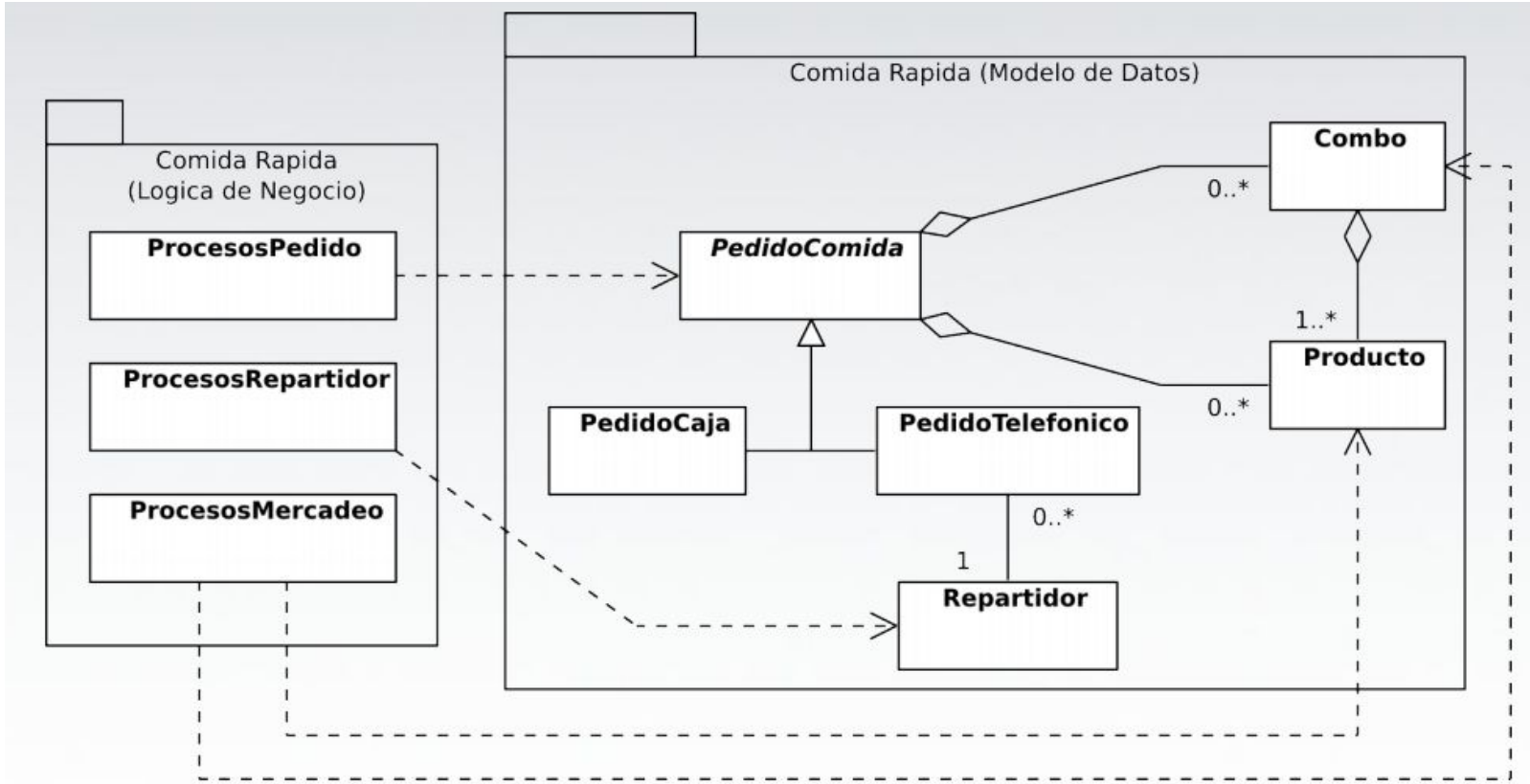
Se pueden utilizar para plantear la arquitectura del sistema a nivel macro

Los paquetes pueden estar anidados unos dentro de otros, y unos paquetes pueden depender de otros paquetes

Se pueden utilizar para plantear la arquitectura del sistema a nivel macro

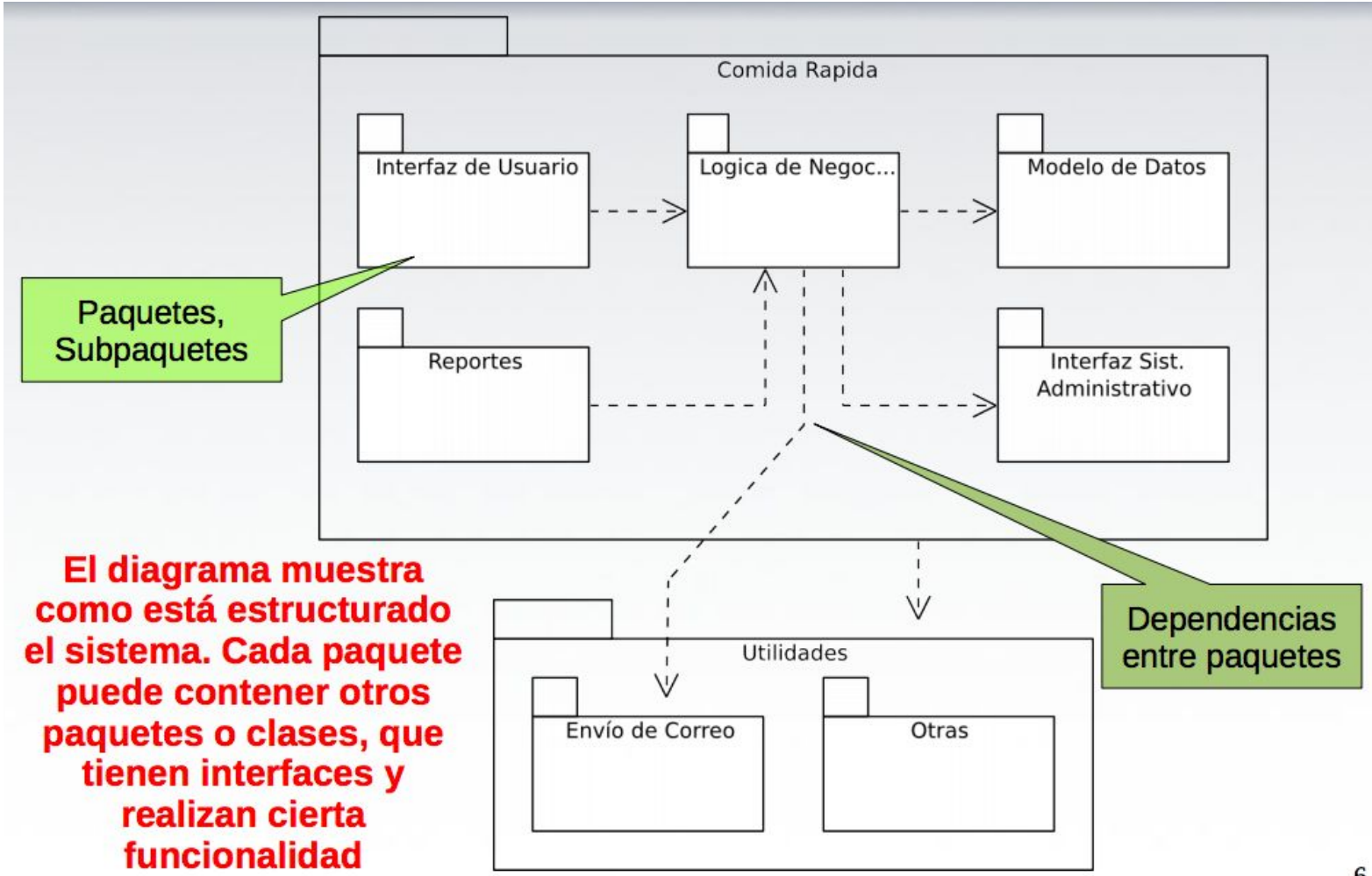
Ejemplo

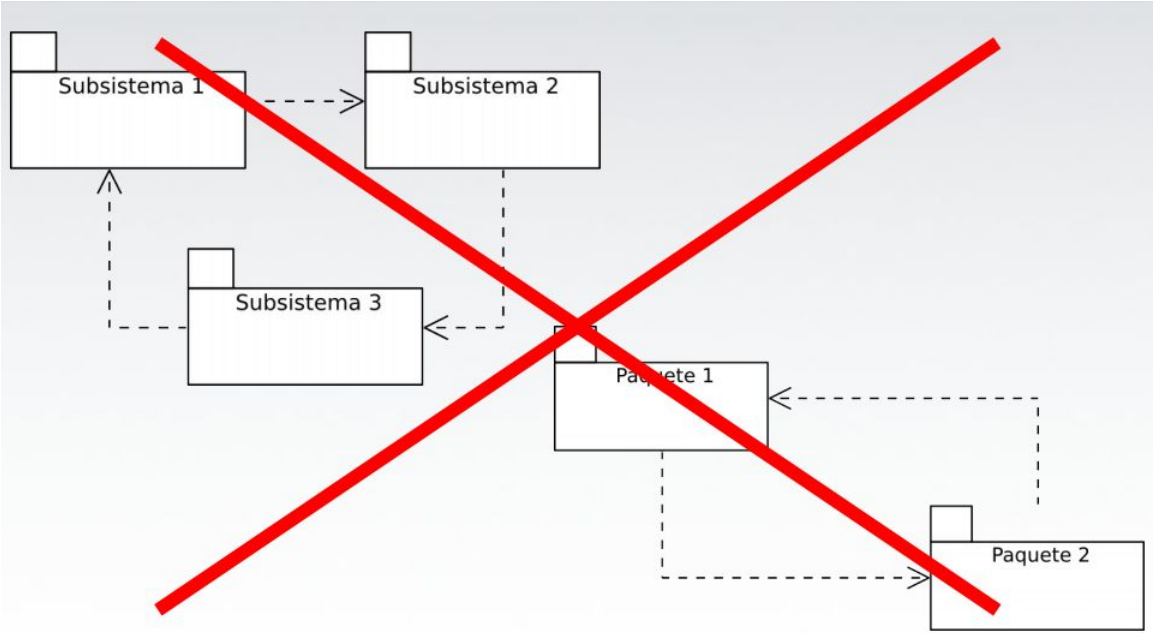

Comida Rápida (Modelo de Datos)
Paquete
Clases dentro del paquete
PedidoComida
+correlativo
-fecha
-hora
+tomarNota(Mesa)
+cocinar()
+servir()
+cobrar()
Combo
-numero
-nombre
-precio
Producto
-codigo
-nombre
-precio
PedidoCaja
-caja
PedidoTelefonico
-telefono
-poblacion
-direccion
Repartidor
-numero
-nombre
-matricula
-turno
0..*
1..*
0..*
0..*
1



También se pueden mostrar algunas clases dentro de los paquetes, así como las relaciones de dependencia de estas clases con otras clases o paquetes

Ejemplo


Comida Rapida
Interfaz de Usuario
Logica de Negoc...
Modelo de Datos
Reportes
Interfaz Sist. Administrativo
Utilidades
Envío de Correo
Otras
Paquetes, Subpaquetes
Dependencias entre paquetes
El diagrama muestra como está estructurado el sistema. Cada paquete puede contener otros paquetes o clases, que tienen interfaces y realizan cierta funcionalidad



Es importante evitar las dependencias circulares, esto aplica en general a paquetes y a clases, aunque en lo que respecta a clases muchas veces no se puede evitar

Relaciones entre paquetes

Entre paquetes pueden existir relaciones de **dependencia** y **generalización**.

Las dependencias entre paquetes denotan que algún elemento de un paquete depende de los elementos en otro paquete.

Existen diferentes tipos de relaciones de dependencia entre paquetes:

- **Importación**: Modelado como una dependencia estereotipada con **<<import>>**.
- **Acceso**: Modelado como una dependencia estereotipada con **<<access>>**.
- **Combinación**: Modelado como una dependencia estereotipada con **<<merge>>**.
- **Exportación**: Modelado implícitamente a través de la visibilidad pública en los elementos del paquete. No se exporta explícitamente a algún paquete.

Relación de importación (por defecto) <<import>>

La importación de paquetes o import se define como "una relación entre un espacio de nombres de importación y un paquete, lo que indica que el espacio de nombres importador **agrega los nombres de los miembros del paquete a su propio espacio de nombres**". Por defecto, una dependencia entre dos paquetes sin etiqueta se interpreta como una relación de este tipo.

Relación de acceso <<access>>

Tanto la importación de paquetes como el **acceso** de paquetes indican que el paquete origen, paquete fuente o paquete importador, tiene acceso al contenido del paquete destino, es decir, el contenido público del destino se añade al espacio de nombres del origen.

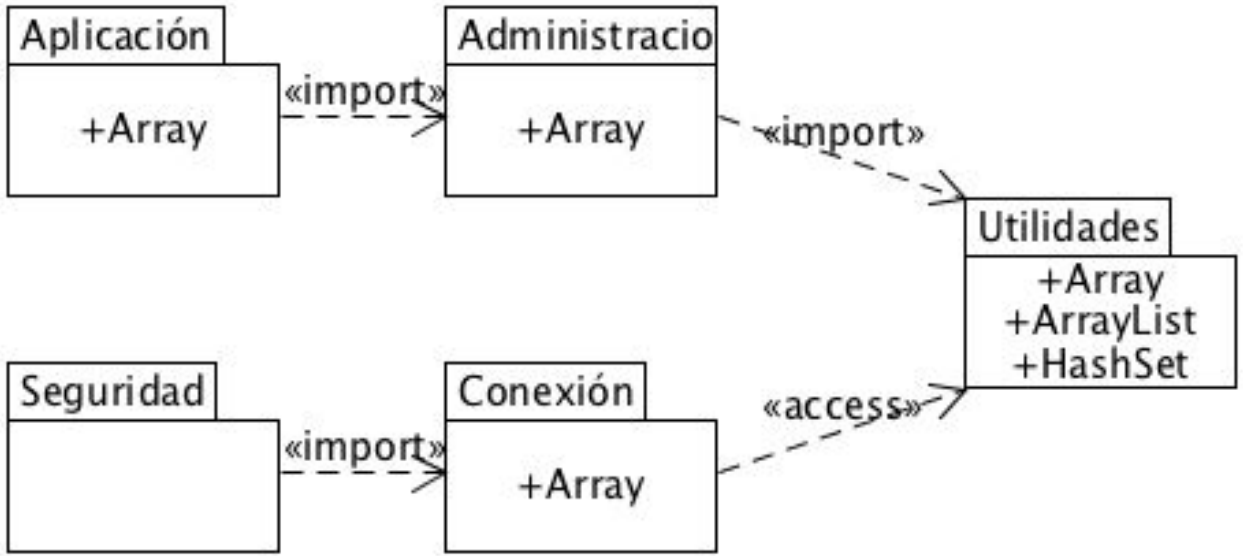
La diferencia radica en que:

- la **importación <<import>>** de paquetes: añade los elementos públicos del paquete destino al espacio de nombres público del origen,
- mientras que el **acceso <<access>>** de paquetes añade los elementos públicos del destino al espacio de nombres privado del origen.
- Ambas relaciones son transitivas.

Ejemplo import y access

En este caso Administración importa Array del paquete de utilidades y Aplicación podrá seguir usando Array.

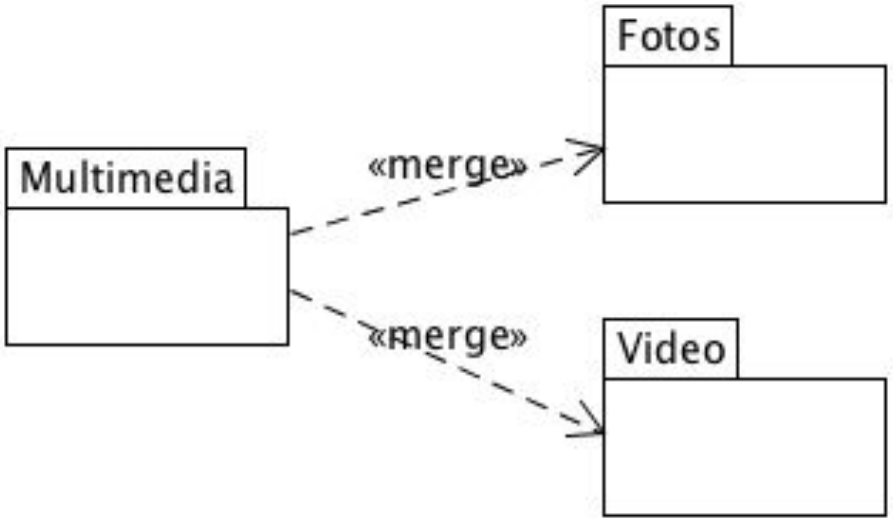
En el caso de conexión, accede a Array de la clase de utilidad y seguridad no podrá acceder a Array



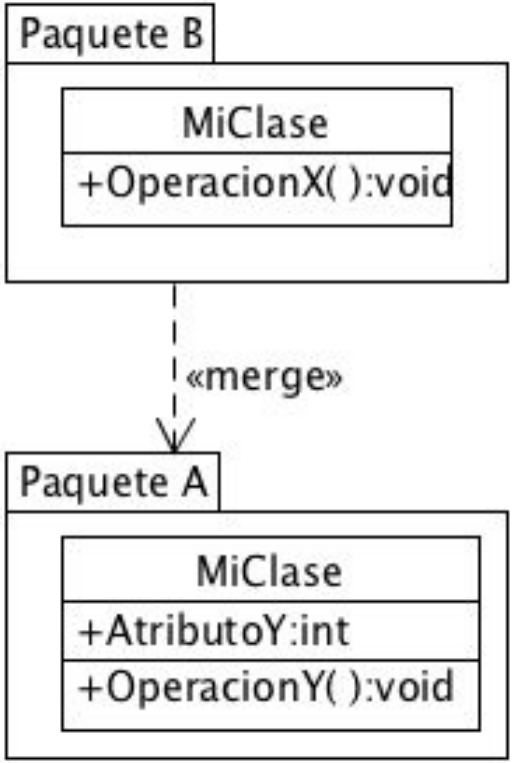
Relación de combinación <<merge>>

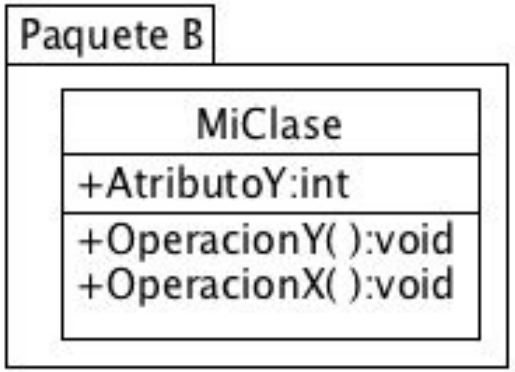
Una relación de combinación o fusión entre paquetes especifica que el contenido del paquete origen (receptor) se extiende con el contenido del paquete destino. La combinación de paquetes o merge se define como "una relación dirigida entre dos paquetes, que indica que el contenido de los dos paquetes se va a combinar. Es muy similar a la generalización en el sentido de que el elemento fuente añade conceptualmente las características del elemento destino para sus propias características lo que resulta en un elemento que combina las características de ambos". En esta relación, si existe un elemento tanto en el paquete fuente y el paquete de destino, después la definición del elemento de origen se ampliará para incluir la definición del elemento de destino.

Ejemplo de merge



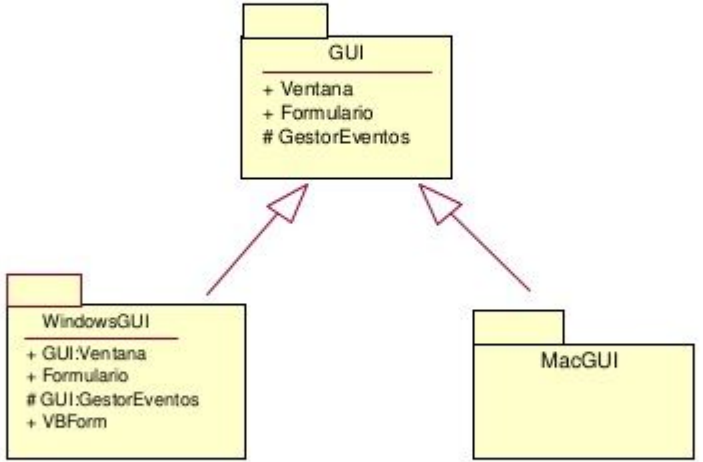
Ejemplo merge





Relación de generalización

Una generalización es una relación entre un clasificador más general (superclase) y un clasificador más específico (subclase). Cada instancia del clasificador específico es también una instancia indirecta del clasificador general. La generalización entre paquetes es similar a la generalización entre clases, los paquetes hijos heredan los elementos del paquete padre. La generalización entre paquetes suele utilizarse para especificar familias de paquetes.



Visibilidad de los elementos

Los paquetes controlan la visibilidad de los elementos que contienen.

Visualmente se representa la visibilidad de los elementos anteponiendo a su nombre uno de los símbolos: +, para los públicos, -, para los privados, y #, para los protegidos.

- +: El elemento es público. Se encuentra disponible a otros elementos del paquete contenedor o uno de sus paquetes anidados, y a los paquetes que importan el paquete contenedor.
- -: El elemento es privado. No disponibles fuera del paquete contenedor.
- #: El elemento está protegido. No son posibles el resto de visibilidades.