UML

Diagrama de Clases y de Objetos

Prof. Daniel Riesco

Diagrama de Clase

- Una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica.
- Las clases son graficamente representadas por cajas con compartimentos para:
  - Nombre de la clase, atributos y operaciones / métodos
  - Responsabilidades, Reglas, Historia de Modificaciones, etc.
- Los diseñadores desarrollan clases como conjuntos de compartimentos que crecen en el tiempo agregando incrementalmente aspectos y funcionalidades.

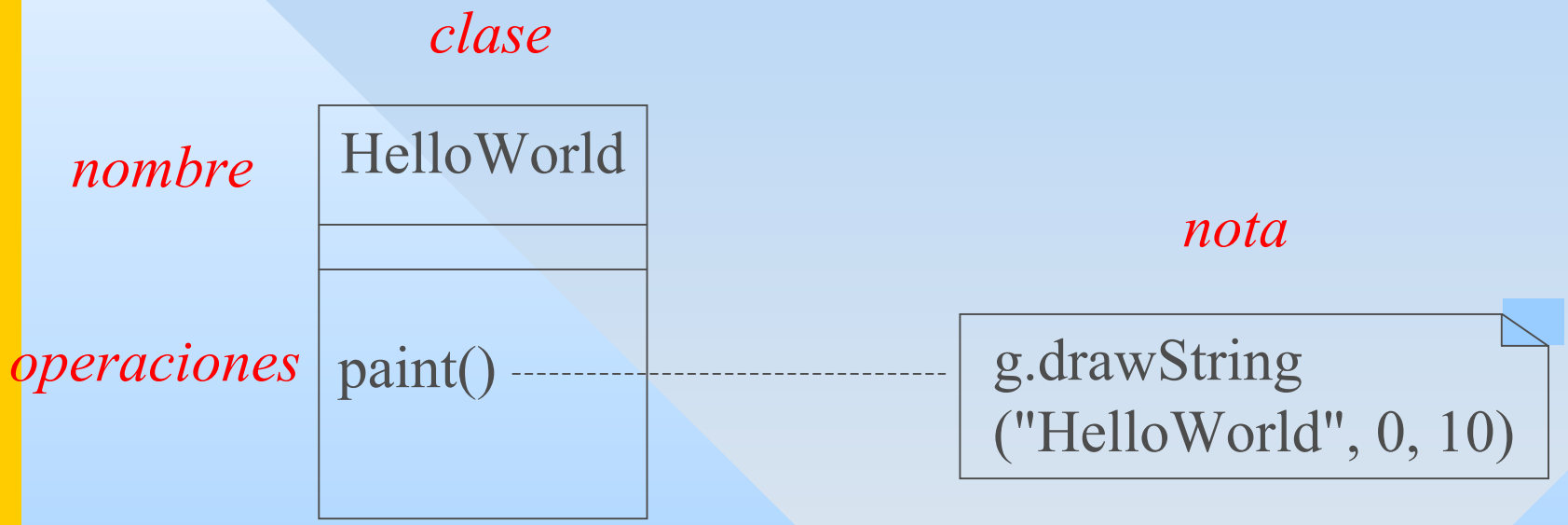
Ejemplo HelloWorld

clase

nombre

operaciones

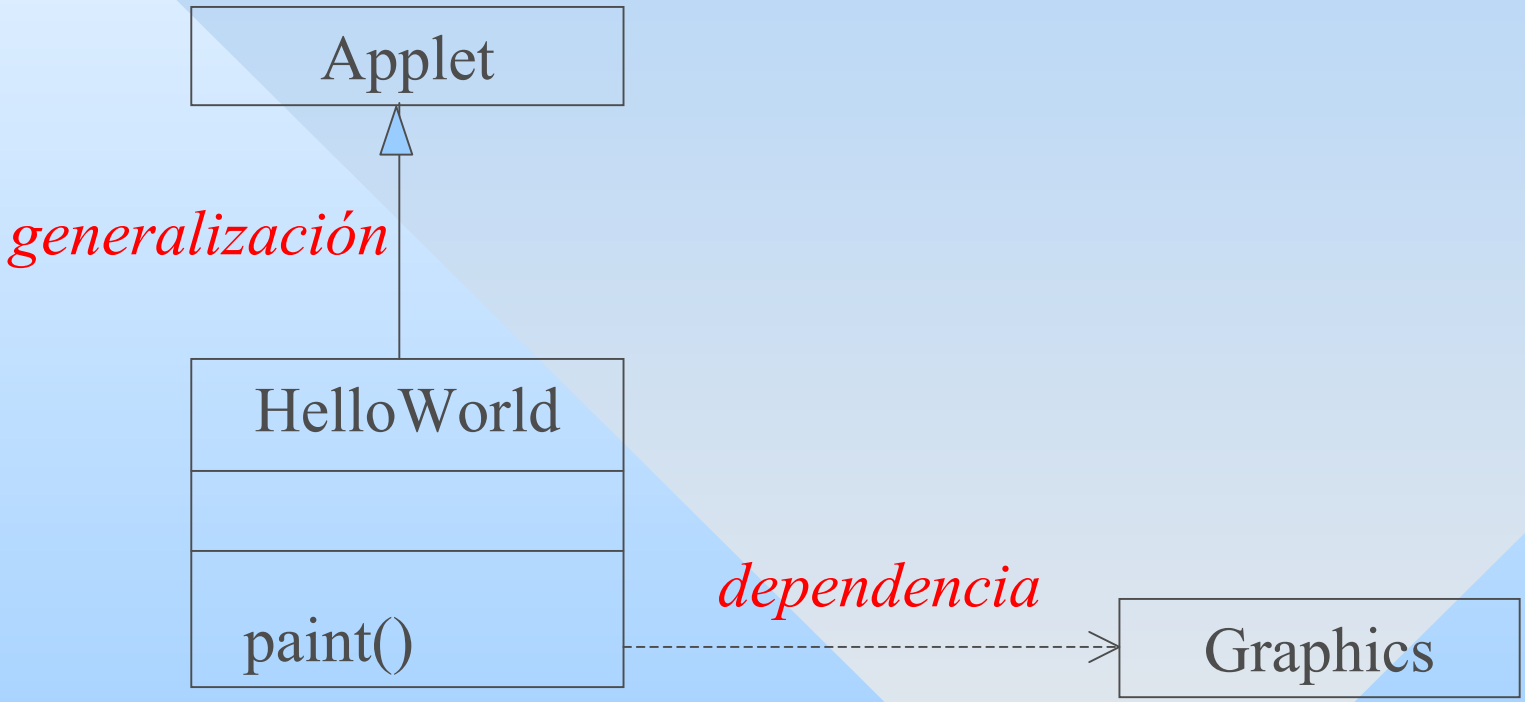
Abstracción para HelloWorld



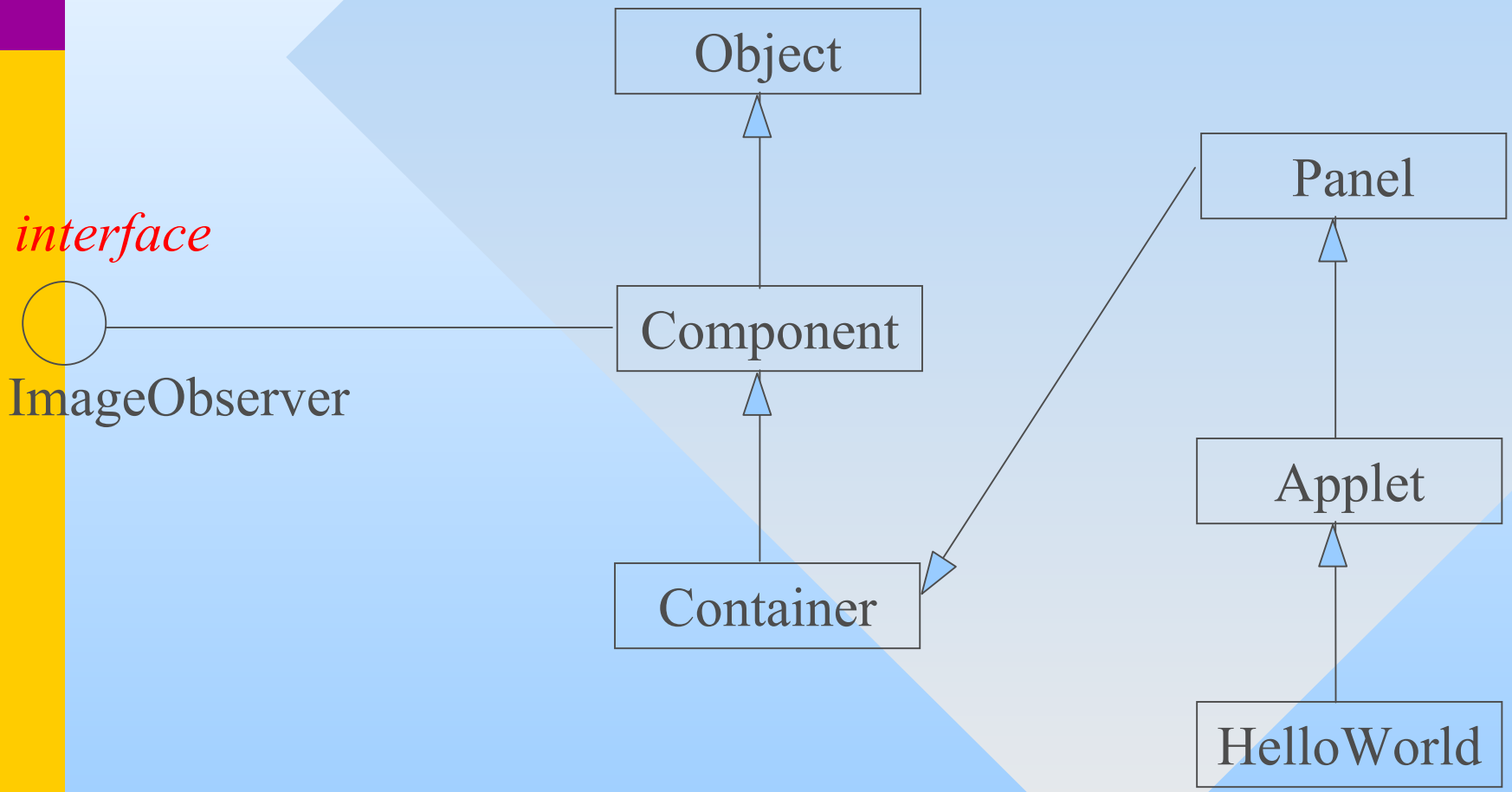
Ejemplo: "Hello, World"

```
import java.awt.Graphics;
class HelloWorld extends java.applet.Applet {
    public void paint (Graphics g) {
        g.drawString ("Hello, World!", 10, 10);
    }
}
```

Diagrama de Clase



Herencia


Object
Panel
interface
Component
ImageObserver
Applet
Container
HelloWorld

Diagramas de Clase

- Muestra un cjto de elementos que son estáticos, como las clases y tipos, junto con sus contenidos y relaciones
- Es un grafo de elementos clasificadores conectados por varias relaciones estáticas
- Clasificador --> Class, Interface, DataType.
- Clase. Alcance. Referencia. Clase Abstracta.
- Orden: [stereotype] nbre [stringPropiedades]

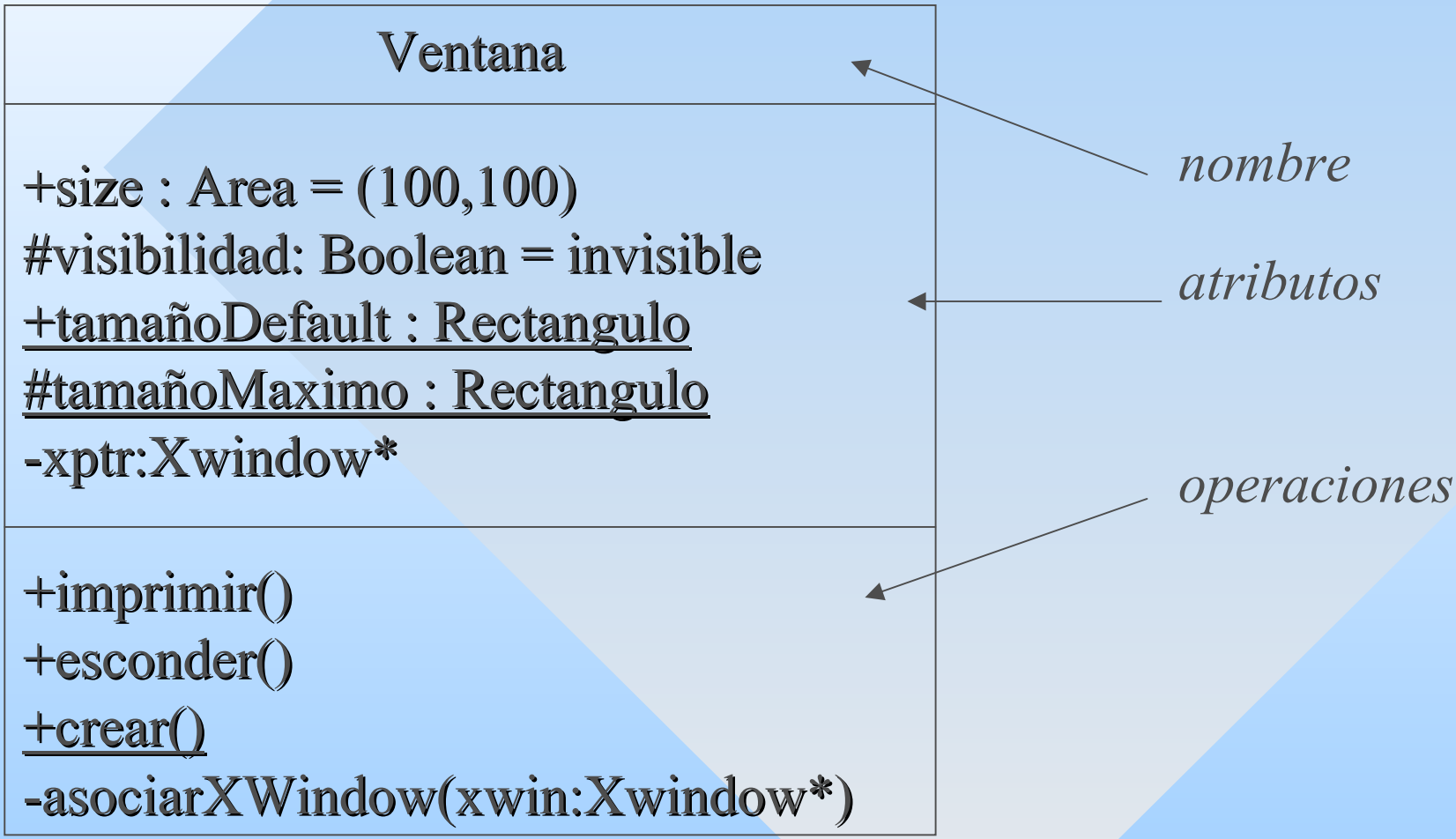
Ejemplo: Clase Dispositivo

- Define e implementa las operaciones para config, transmitir y recibir informac. hacia y desde el puerto serie
- hCom: handler al dispositivo.
- puerta: nombre del puerto serie
- velocidad: velocidad de la comunicación.
- paridad: tipo de paridad
- bitStop: cantidad de bits de stop
- <<constructor>> Dispositivo() crea y abre el dispositivo retornando un handler
- <<query>> RecuperarDispositivo() inf. BD para config., LeerBloque() información del puerto
- <<update>>ConfigurarDispositivo(), GrabBloquePuerto()

Diagramas de Clase

- Atributo:
- visibilidad nbre : exprTipo [= valor] [{prop}]
- visibilidad: public+, protected #, private - (no default)
- prop: {changeable} (default), {frozen}. Multiplicity [].
- Atributos de clase subrayados. Comienzan con minúscula
- Operación:
- visibility nbre (parámetros) [:TipoRetorno] [{prop}]
- prop: {query}, {sequential}, {guarded}, {concurrent}, {abstract}
- parámetros: [in|out|inout] nbre : TipoExp = valorDefault
- Operaciones de clase subrayadas.

Notación: Una Clase

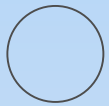


Una **clase** es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

Diagramas de Clase

- Clases parametrizadas.
  - Template no es directamente utilizable.
  - Parámetros, nbre [: expTipo]
- Utilidad: es una agrupación de variables globales y procedimientos en la forma de declaración de clase.
- Metaclase: clase cuyas instancias son clases.
- Objeto: Subrayado con valores particulares.

Notación: Interface



Iortografía

- Una interface es una colección de operaciones que especifican un servicio de una clase o componente, es decir, un comportamiento externamente visible de ese elemento.
- Se especifican las operaciones externamente visibles sin especificación de la estructura interna.

Modelado de Clases

- Una Responsabilidad es un contrato u obligación de una clase.
- Modelado del Vocabulario
  - Identificar los conceptos que usan los usuarios (Tarjetas CRC - casos de uso)
  - Para cada abstracción, identificar el conjunto de responsabilidades. Cada clase debe estar bien definida y un buen reparto de responsabilidades.
  - Proporcionar atributos y operaciones necesarios para cumplir con dichas responsabilidades.
- Clases muy grandes (varias responsabilidades) -> dificil de cambiar y no reutilización.
- Clases muy pequeñas -> modelo dificil de entender.

Modelado de Clases

- Modelado de conceptos que no son software.
  - Se modela como una clase
  - Para distinguirla de clases del sistema se usa un nuevo bloque de construcción con estereotipos.
  - Si es un hardware que tiene software -> nodo.
- Modelado de tipos primitivos.
  - Se modela como una clase con el estereotipo adecuado.
  - Si se necesita especificar el rango, se usa restricciones.
- Una clase debe:
  - proporcionar una abstracción bien definida de algo del dominio del problema o de la solución
  - Contiene un conjunto pequeño de responsabilidades
  - Muestra una clara distinción entre la implementación y la especificación de la abstracción.
  - Ser sencilla, entendible, extensible y adaptable.

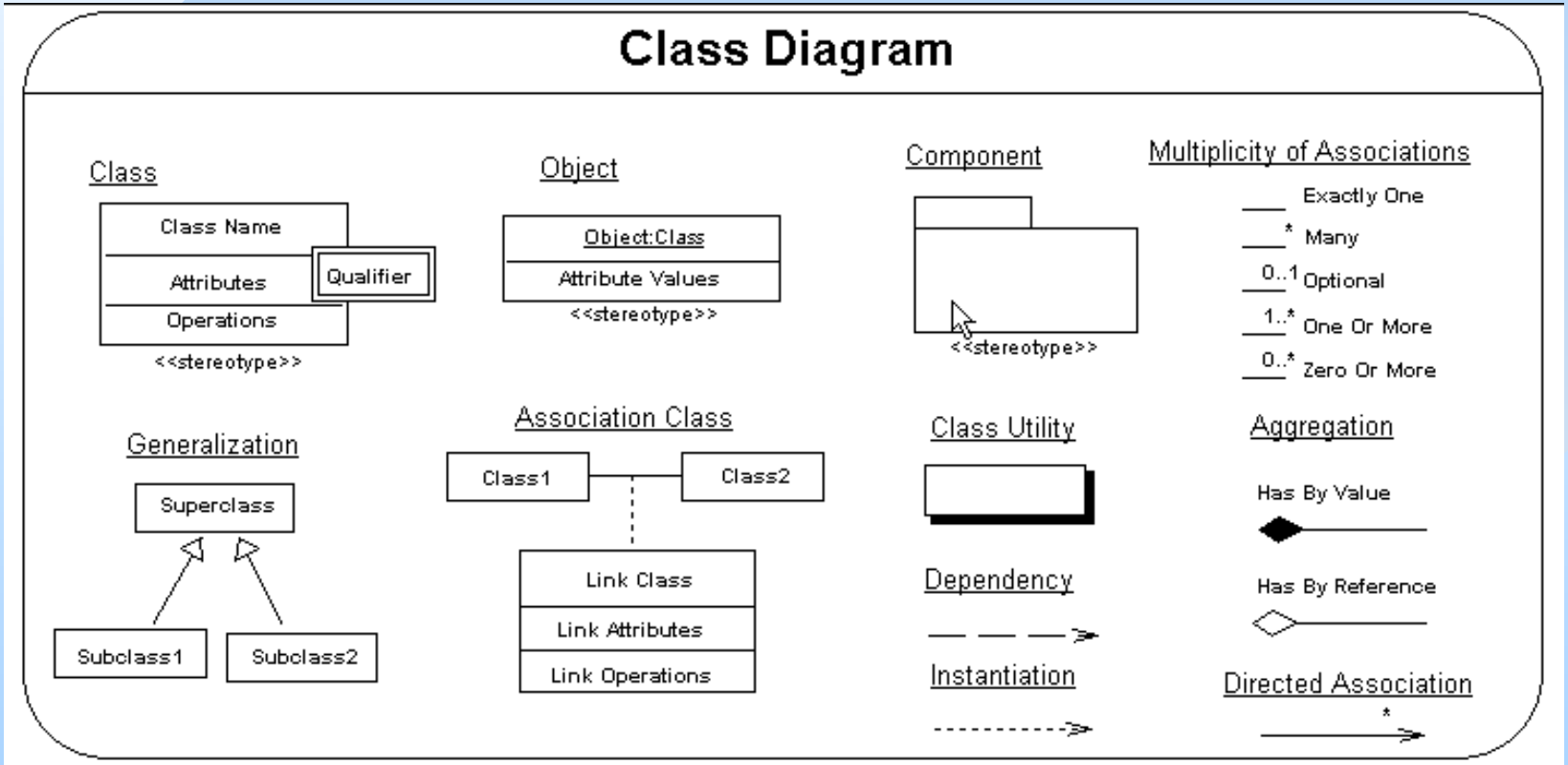
Diagramas de Clase: Relaciones

- Final de asociación:
  - multiplicidad
  - orden {unordered} (default), {ordered}
  - navegación
  - agregación. Diamante lleno es composición.
  - Nombre del rol
  - cambiable (default) {frozen} {addOnly}
  - visibilidad
- Asociación binaria. Opcional {or}. Asociación n-aria.
- Clase asociación: es una asociación que tiene propiedades de una clase.
- Composición. Tpo de vida. Multiplicidad del contenedor=1
- Generalización. Discriminador.
  - Restricción: {ovelapping},{disjoint},{complete},{incomplete}
- Relación de dependencia
  - <<amport>> <<access>> <<become>>, <<bind>>

Modelado de Relaciones

- Dependencia
  - Parámetro de una operación. Si se muestra la signatura no es necesario la relación de dependencia
- Generalización
  - Dado un cjto de clases, se busca responsabilidades, atributos y operaciones comunes.
  - Se elevan a una clase más general (nueva o no). No crear demasiados niveles.
- Asociación.
  - Relación estructural. Relación bilateral. Diferencia con dependencia y Generalización.
  - Equilibradas. Ni muy profundas (<=5 niveles) ni muy anchas.
  - Herencia múltiple se puede reemplazar por agregación.

Elementos del Diagrama de Clase



Modelo Conceptual / del Dominio

- Un ***Modelo Conceptual /Dominio*** es el conjunto de diagramas de estructura estático con clases, atributos y asociaciones, pero no operaciones.
- Construcción del Modelo Conceptual / Dominio
  - Representa un aspecto de la realidad
  - Ayuda a los Ingenieros de Sofware a gestionar la complejidad
  - Es más simple que la realidad
- Un Modelo Conceptual / Dominio debería:
  - Organizar Datos dentro de Objetos y Clases
  - Estructurar Datos vía herencia y asociaciones
  - Especificar comportamiento e interfaces públicas
  - Describir el comportamiento global
  - Describir Restricciones

Estático: Diagrama de Clase

- Utilizado para la estructura estática del modelo conceptual / Dominio
- El diagrama de clase describe
  - Tipos de objetos en la aplicación / dominio
  - Relaciones estáticas entre objetos
- El diagrama de clase contiene
  - Clases: Objetos, Atributos, and Responsabilidades
  - Paquetes: Agrupación de clases
  - Subsistemas: Agrupación de clases/paquetes

Modelado de Clases

Dado un sistema de la vida real, ¿cómo decide que clases usar?

- Los términos usados por usuarios y desarrolladores para describir el sistema son clases candidatas.
- Para cada clase, ¿cuáles son sus responsabilidades? ¿están balanceadas entre las clases?
- ¿Qué atributos y operaciones necesita cada clase para llevar a cabo sus responsabilidades?

Identificación de Sustantivos: Un ejemplo de una biblioteca

Una biblioteca contiene libros y revistas. Puede haber varias copias de un libro. Algunos de los libros son reservados sólo para préstamos a corto plazo. Todos los otros pueden ser prestados a cualquier miembro de la biblioteca por tres semanas. Los miembros de la biblioteca pueden normalmente solicitar hasta seis items de una vez, pero miembros del staff pueden solicitar hasta doce items a la vez. Solamente miembros del staff pueden obtener prestado revistas.

El sistema debe conservar la pista de cuando los libros y revistas son prestados y retornados forzando las reglas de la biblioteca.

Identificación de Sustantivos: Un ejemplo de una biblioteca

Una biblioteca contiene libros y revistas. Puede haber varias copias de un libro. Algunos de los libros son reservados sólo para préstamos a corto plazo. Todos los otros pueden ser prestados a cualquier miembro de la biblioteca por tres semanas. Los miembros de la biblioteca pueden normalmente solicitar hasta seis items de una vez, pero miembros del staff pueden solicitar hasta doce items a la vez. Solamente miembros del staff pueden pedir prestamos de revistas.

El sistema debe conservar la pista de cuando los libros y revistas son prestados y retornados forzando las reglas de la biblioteca.

Clases Candidatas

Biblioteca	Nombre del Sistema
Libro	
Revista	
Copia	
PréstamosACortoPlazo	evento
MiembroDeBiblioteca	
Semana	medida
Item	libro o revista
Tiempo	término abstracto
MiembroDelStaff	
Sistema	término general
Regla	término general

Relaciones entre Clases

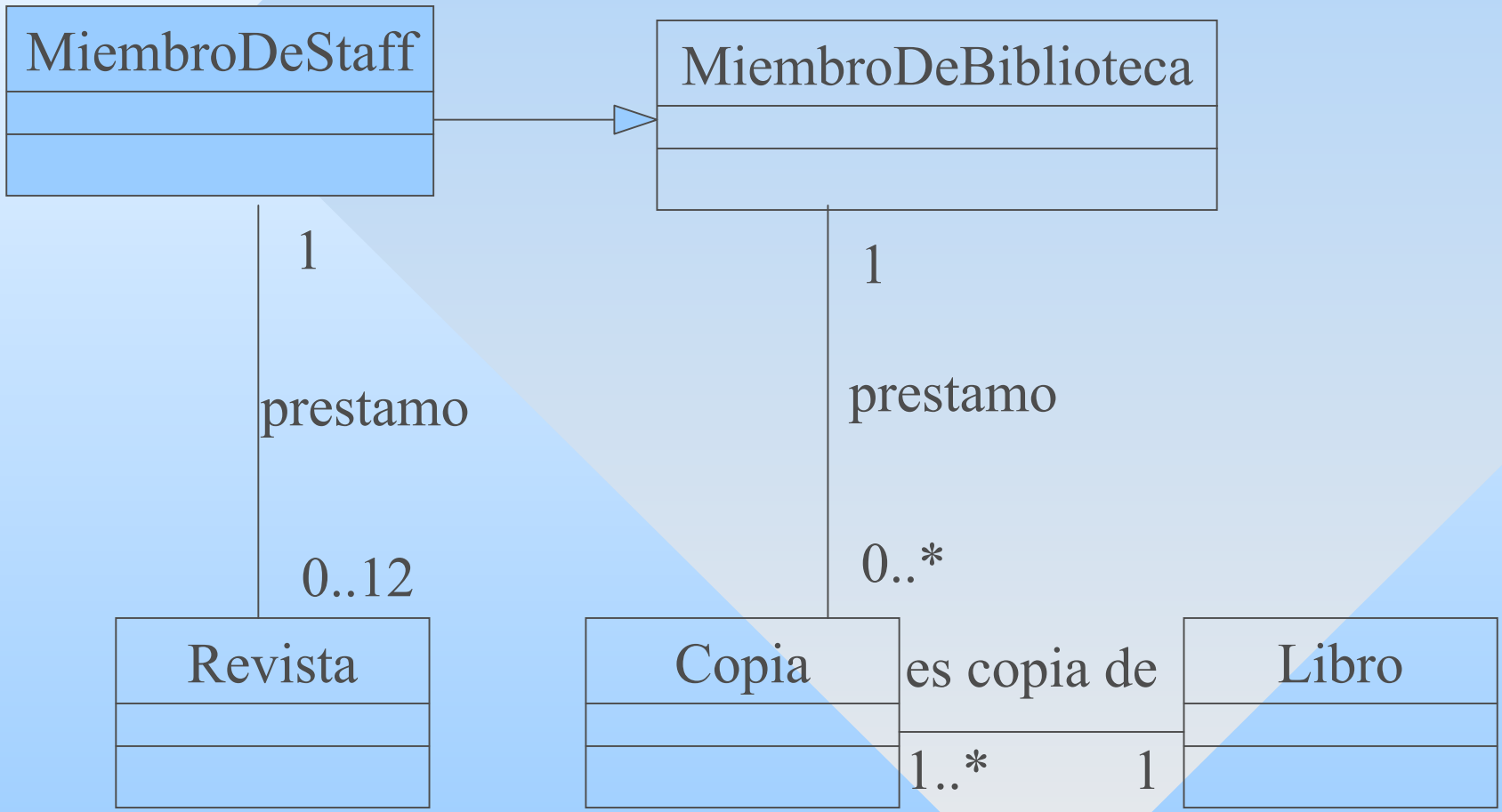
Libro	es un	Item
Revista	es un	Item
Copia	es una copia de	Libro
MiembroDeBiblioteca		
Item		
MiembroDeStaff	es un	MiembroDeBiblioteca

¿Es el Item necesario?

Operations

MiembroDeBiblioteca	pide prestado	Copia
MiembroDeBiblioteca	devuelve	Copia
MiembroDeStaff	pide prestado	Revista
MiembroDeStaff	devuelve	Revista

Diagrama de Clase



Generalización y Association

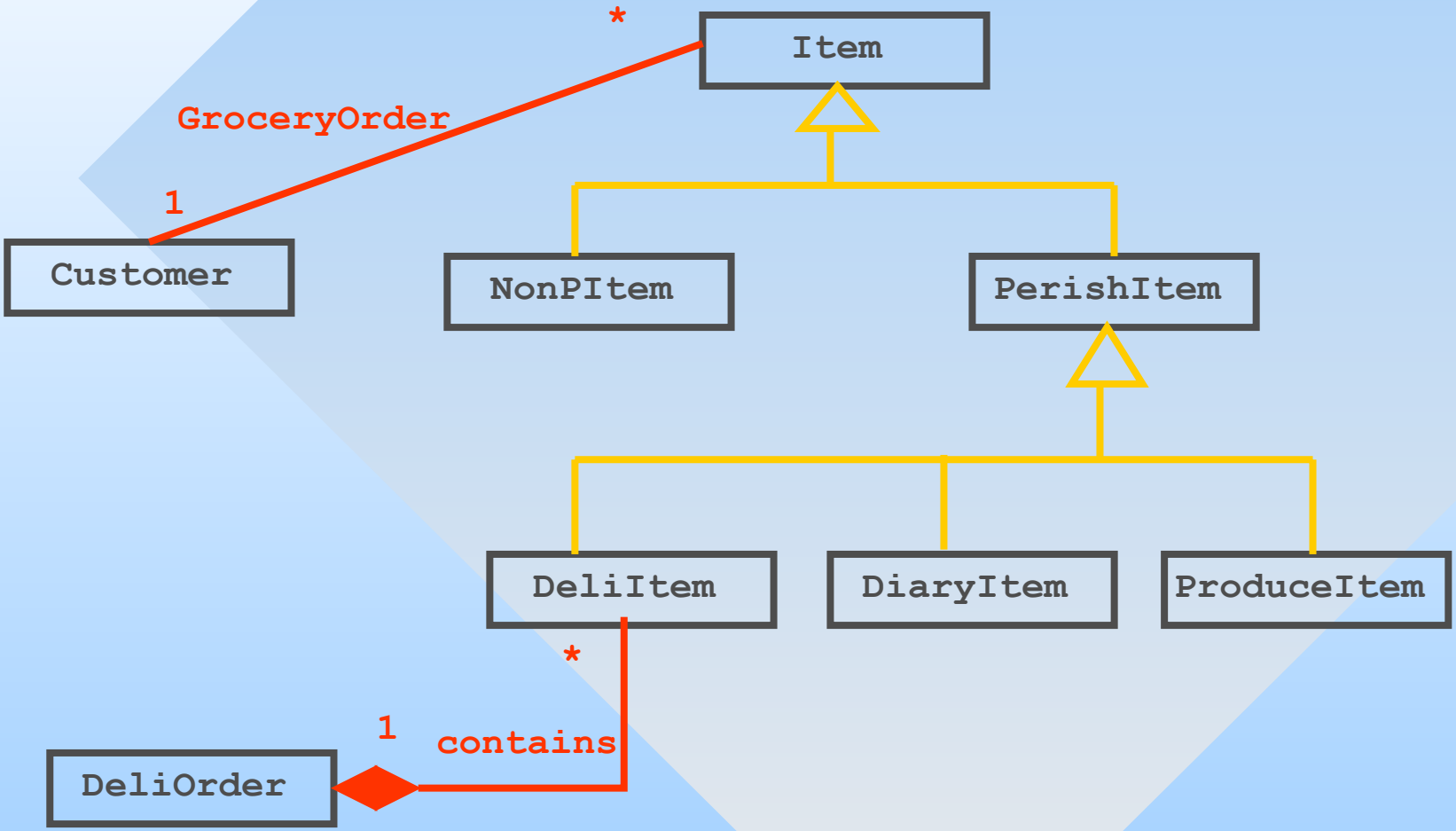


Diagrama de Clase

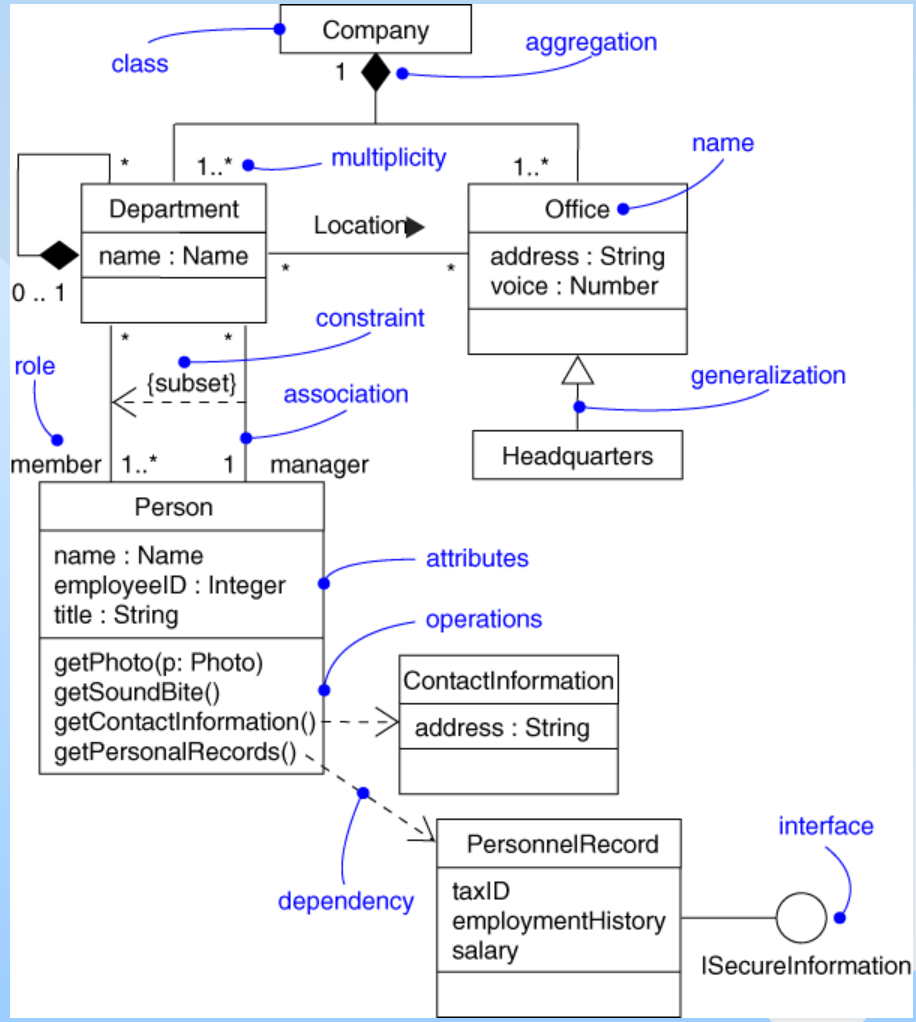
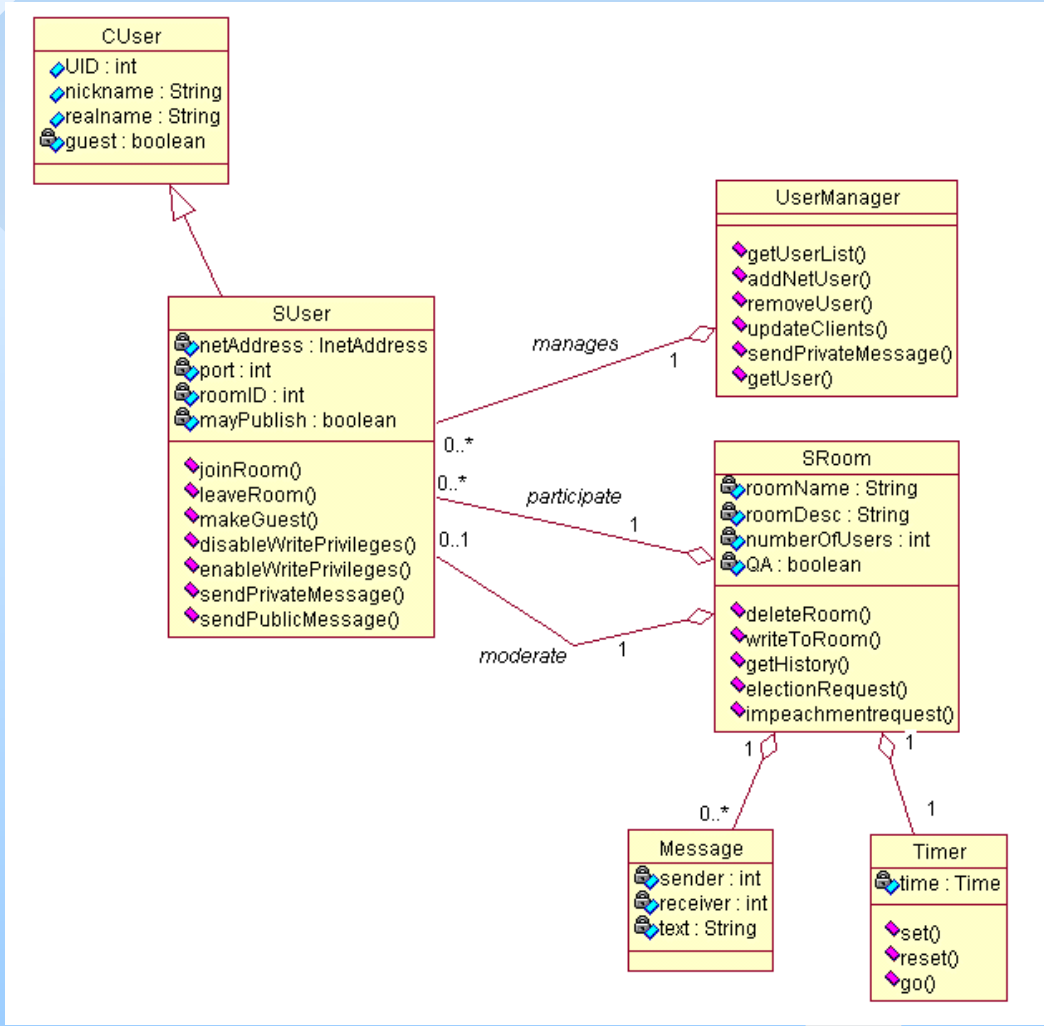
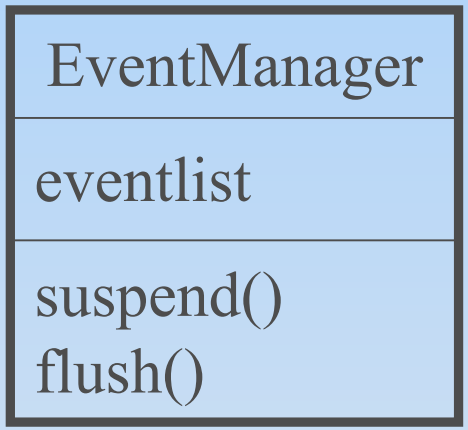


Diagrama de Clase



Clase Activa

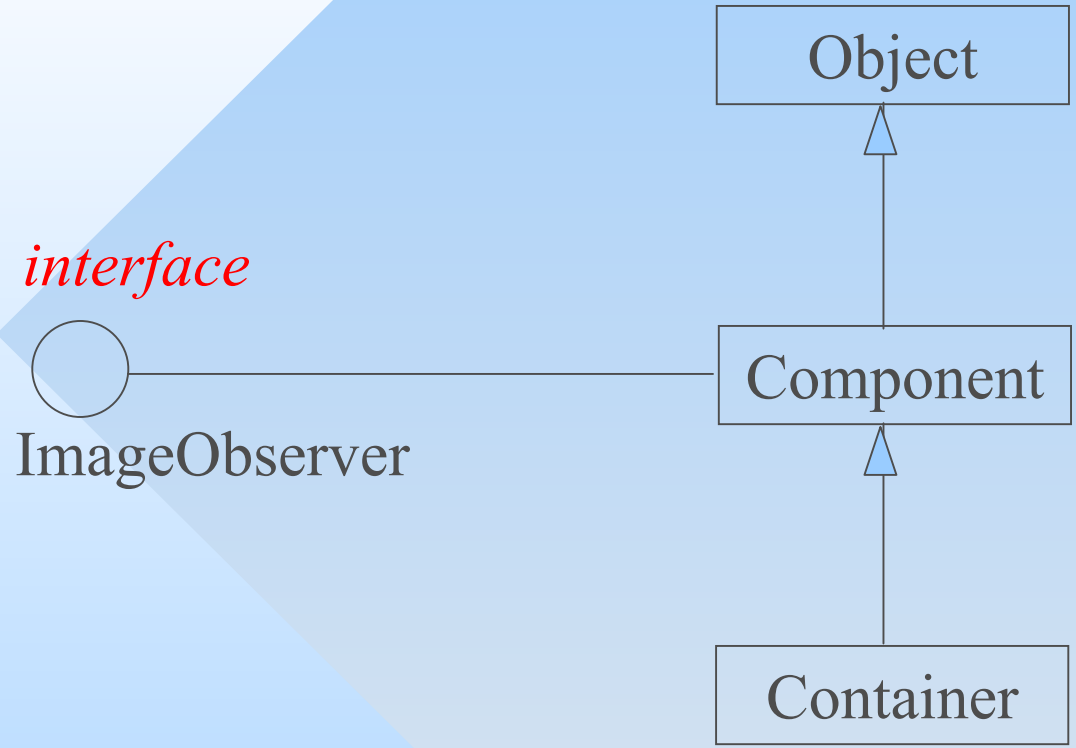


Una clase activa es una clase cuyos objetos poseen uno o más procesos o threads y por lo tanto pueden inicial una actividad de control.

Objeto:

- –{active}, posee el thread,es capaz de iniciar actividad de control
- –pasivo, mantiene los datos pero no inicia la actividad

Interface



Una interface es una colección de operaciones que se usa para especificar un servicio de una clase o componente.

Ejemplo: Patrón Iterator

- Provee una forma de acceder a elementos de un objeto agregado sin exponer la representación subyacente
  - Ej: una clase Lista
    - Querer recorrer la lista en varias formas
      - Hacia adelante
      - Hacia atrás
      - filtrada
      - ordenada
      - ...

Motivación para los iteradores

- No aglutinar la interface Lista con varios recorridos
    - Aún si se hace, no se puede anticipar todos los posibles recorridos
- Quere más de un recorrido sobre la misma lista.
- Los Iteradores mueven la responsabilidad para acceder y recorrer desde los agregados al objeto iterador.

Ejemplo de Iterador (1)

```
class List {
      size() {}
      add() {}
      remove() {}
}


interface ListIterator {
      getFirst();
      getNext();
}
```

Ejemplo de Iterador (2)

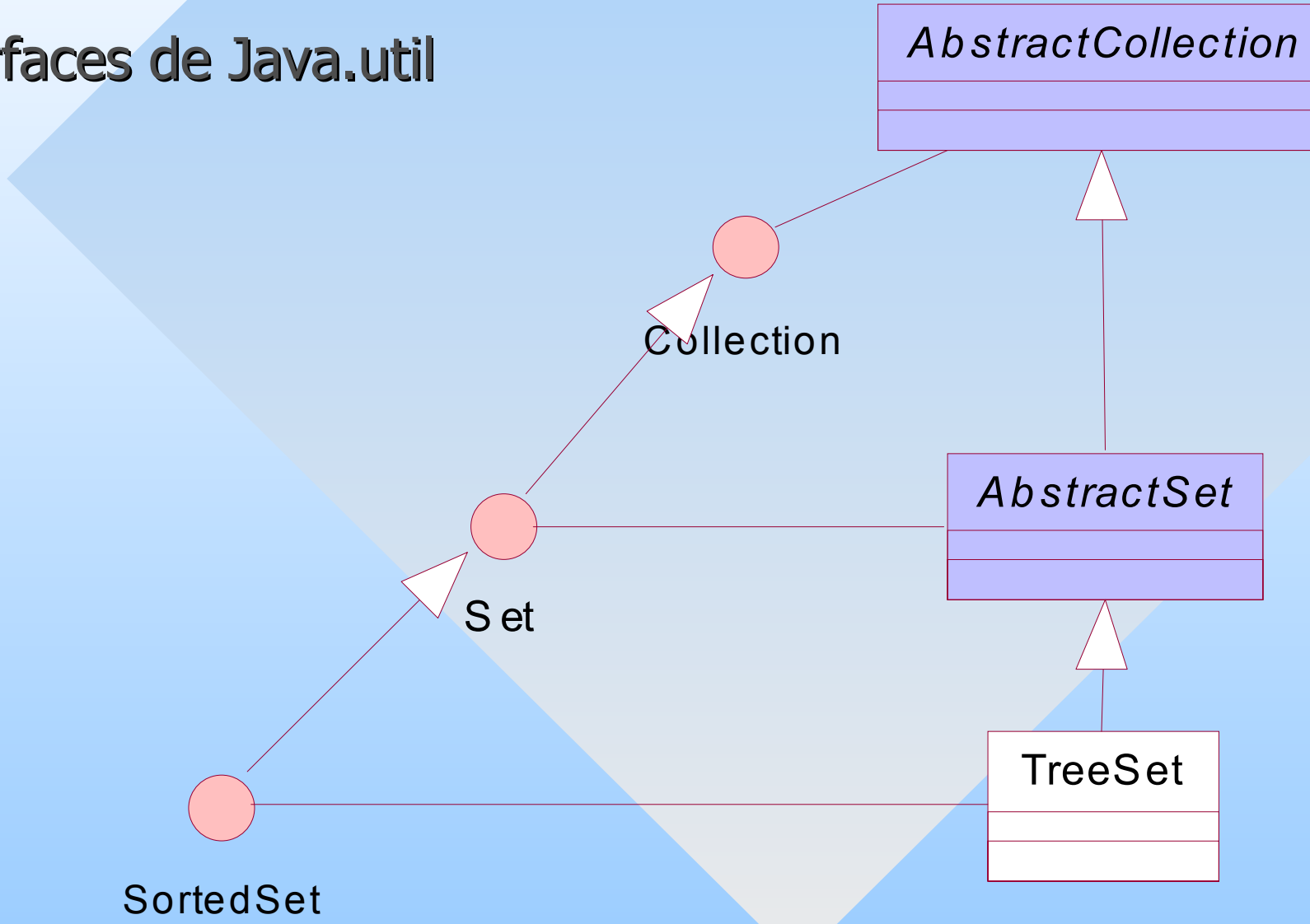
```
class FilteredListIterator implements ListIterator {
      List.Node curr;
      FilteredListIterator(List list, Filter f) {}

      getFirst() {
            curr = list.head;
            while (curr != null) {
                  if (f.accepts(curr.data))
                        break;
                  curr = curr.next;
            }
            return curr;
      }

      getNext() {}
}
```

Otras característias del patrón Iterador

- Los Iteradores proveen una interface común para el acceso al objeto
  - Pueden usar la misma interface para listas implementadas como arrays y listas implementadas como listas encadenadas.
  - Es más facil cambiar las implementaciones de la estructura de datos
- Hay varios ejemplos en java.util de JDK 1.2

Interface: otro ejemplo

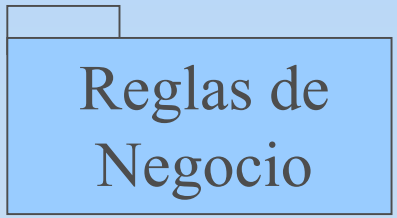
Interfaces de Java.util



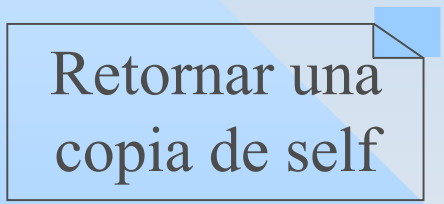
Gerenciamiento del Modelo Paguetes y Organización

- Los paquetes:
  - Son una agrupación de elementos del modelo
  - Son cohesivos (límite bien definido alrededor de un conjunto de elementos relacionados).
  - Poco acoplados (exportando sólo aquellos elementos que otros paquetes necesitan, e importando solo lo necesario y suficiente
  - Pueden contener paquetes subordinados (anidados: aconsejable 2 a 3 niveles) y otros elementos del modelo.
  - Forman un espacio de nombres (p'::A, p"::A).
- El sistema completo es un simple paquete (anónimo).
- Todo diagrama y elementos del modelo UML pueden estar organizados en paquetes
- De la propiedad: Jerarquía de paq. es un árbol
- Del uso: es un grafo.

Paquete & Nota



Un **paquete** es un mecanismo de propósito general para organizar elementos dentro de grupos.

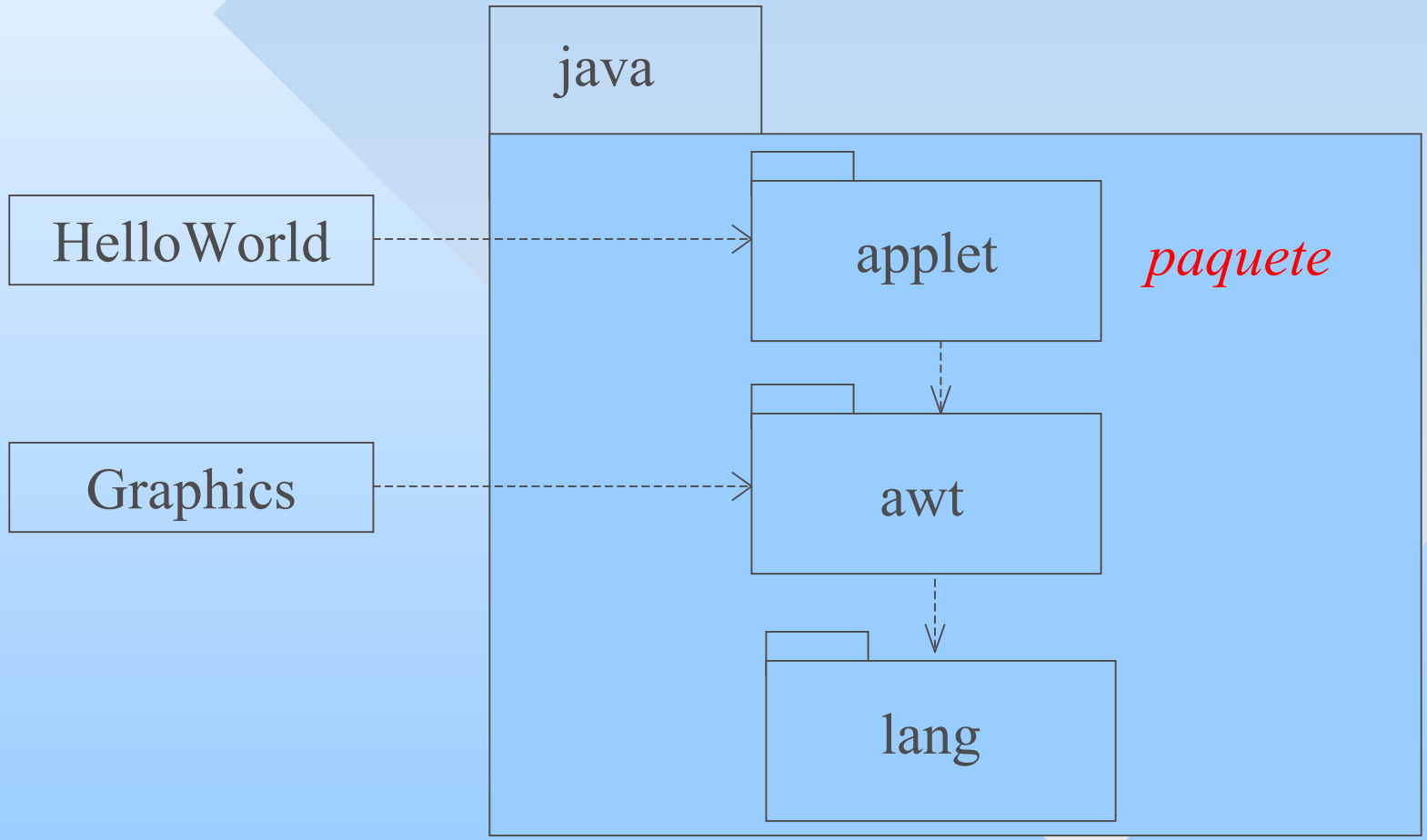


Una **nota** es un símbolo para mostrar restricciones y comentarios adjuntos a un elemento o una colección de elementos.

Relaciones entre Paquetes

- Importación. Relación de dependencia con:
  - <<import>> añade el contenido del destino al espacio de nombres del origen.
  - <<access>> no lo añade. Se deben calificar los nombres.
  - La dependencia de importación no es transitiva.
- Exportación. Se especifica el elemento con su visibilidad.
  - Visibilidad: +, #, -.
- Generalización.
  - Heredan los elementos públicos y privados.
  - Pueden redefinir elementos y añadir nuevos.
  - Un paquete especializado puede usarse en cualquier lugar que se utilice un paquete más general.
- Estereotipos: facade (vista), framework, stub, subsystem, system.

Empaquetando Clases



Paquetes

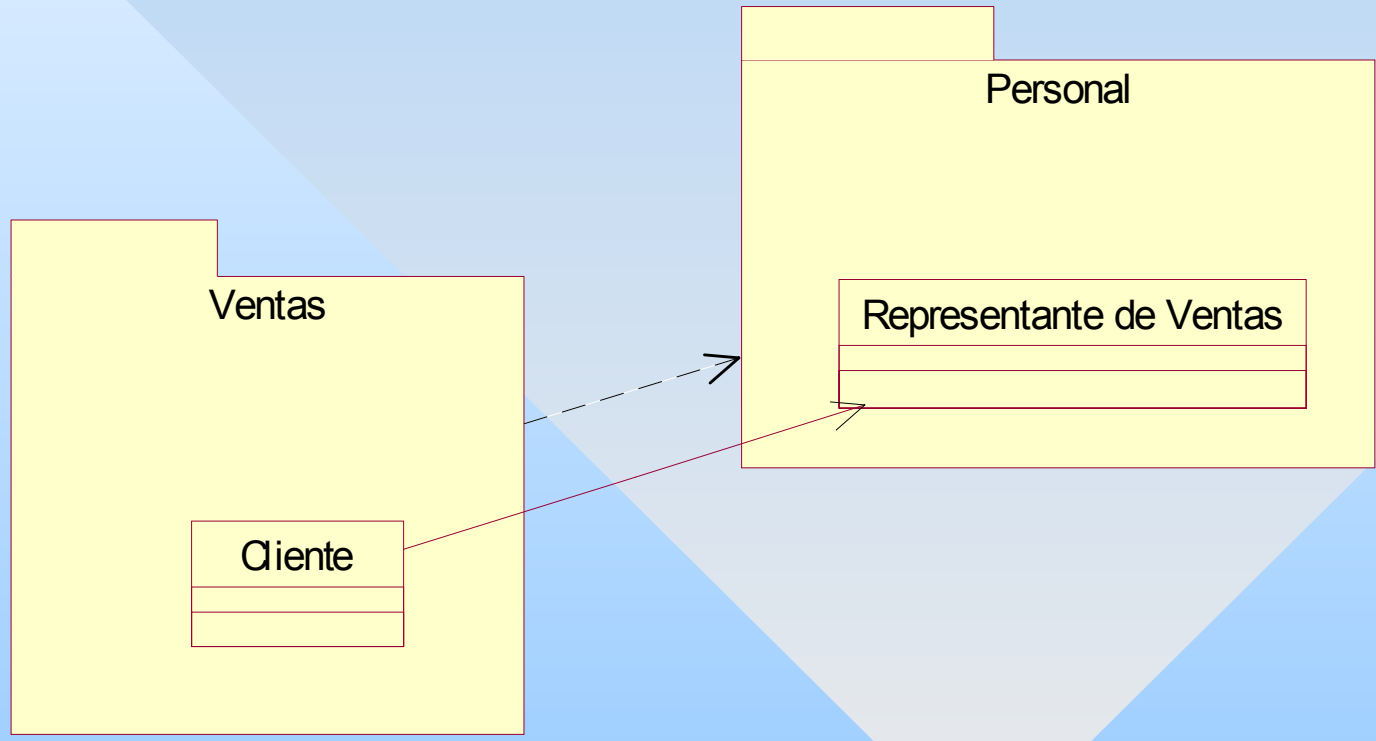
- Los paquetes ofrecen un mecanismo general para la partición de los modelos y la agrupación de los elementos de modelado.
- Cada paquete corresponde a un subconjunto del modelo. Contiene clases, objetos, relaciones, componentes y diagramas.
- La arquitectura del sistema viene dada en forma de paquetes y por las relaciones de dependencia entre ellos.
- Un paquete puede contener a otros, sin límite de anidamiento
- Cada elemento pertenece a sólo un paquete
- Una clase de un paquete puede aparecer en otro paquete por la importación a través de una relación de dependencia entre paquetes

Paquetes (Cont.)

- Las importaciones entre paquetes
  - se representan por medio de una relación de dependencia estereotipada y orientada del cliente al proveedor
  - Al menos una clase del paquete cliente usa los servicios ofrecidos por al menos una clase del paquete proveedor
- Una clase depende de otra si accede a un valor del proveedor, invoca a una operación o referencia al proveedor como argumento en alguna operación
- Todas las clases no son necesariamente visibles desde el exterior del paquete
- El operador "::" permite designar una clase definida en un contexto distinto del actual
- Un paquete encapsula a la vez que agrupa

Paquetes (Cont.)

- Cada elemento de un paquete se incluye como visible o no desde el exterior del paq.



Técnicas

- Los paquetes no tienen identidad (no hay instancias, son invisibles para el sistema en ejecución).
- Se pueden utilizar como unidades básica para un SCM, para grupos de desarrollo diferentes.
- Los paquetes se pueden utilizar para modelar las vistas arquitectónicas.
- Vista es una proyección de la organización y estructura de un sistema, centrada en un aspecto particular del sistema.
  - Descomposición de un sistema en paquetes casi ortogonales.
    - Vista de diseño, procesos, implementación, despliegue, caso de uso.
  - Los paquetes contienen todas las abstracciones pertinentes para esa vista.
    - Todos los componentes del modelo pertenecen al paquete vista de implementación.
    - Existirán dependencias entre los elementos de las distintas vistas.

Modelos

- Los modelos permiten visualizar, especificar, construir y documentar un sistema.
- Los sistemas bien estructurados son cohesivos funcional, lógica y físicamente, construidos a partir de subsistemas débilmente acoplados.
- Un modelo bien estructurado proporciona una simplificación de la realidad desde un punto de vista bien definido y relativamente independiente.

Puente Grua: Funcionalidades

- Controlar el puente a través de un PLC.
  - Dar directivas al pte grúa con un Controlador Lógico Programable.
  - Comunicación por la puerta serial de un PC.
  - Protocolo con fuertes componentes de tiempo.
- Modelizar el recinto de materiales.
  - Recinto Configurable: largo, ancho y alto.
  - Grilla. Cada pto representa pozos, tolvas y cintas de descarga.
  - Control del relieve (pto en situación crítica) y flujo de materiales.
  - Infor. de c/pto: material depositado, altura del mismo, etc.
  - Establecer prioridades para no para no parar la cadena productiva
- Interpretar consignas de trabajo: manuales y automáticas
- Generar alarmas ante la ocurrencia de eventos anormales: fallas mecánicas o eléctricas deben ser conocidas de inmediato por el operador del sistema.

Componentes del Sistema

- Interfaz del Sistema de Grúa Robotizada, permite
  - la comunicación del operador con el sist. que controla el pte grúa.
  - definir el recinto de materiales
  - la def y reconocimiento de alarmas, y la especif. de las consignas.
- Sistema de Control de Comunicaciones, permite
  - la parametrización de la comunicación : Computadora/PLC
- RDBMS,
- generador de información gerencial, obtener infor. sobre
  - producción de la planta,
  - el funcionamiento (alarmas reconocidas, histórico de alarmas, tiempo ocioso, etc.),
  - el costo financiero de material en stock y minimización de stock (just in time).
- un sistema de grúa robotizado
  - encargado de la gestión del puente grúa

Diagramas de Objetos

- Muestra instancias compatibles con un diagrama de clase particular.
- Incluye sus objetos y los valores de sus datos.
- Snapshot del estado detallado del sistema en un pto del tpo.

Clases & Objetos