Diagrama de Clase

MC Beatriz Beltrán Martínez
MC Miguel Rodríguez Hernández

Otoño 2013

Tipos de diagramas

- **Diagramas de estructura:** mostrar la estructura estática del sistema que se está modelando
  - Incluye: diagramas de clase, componentes y/o objetos.
- **Diagramas de comportamiento:** muestra el comportamiento dinámico entre los objetos y el sistema.
  - Incluye: diagramas de actividades, casos de uso y de secuencia

Diagrama de clase

- Es el más utilizado y más conocido de los diagramas orientados a objetos. Es la fuente de generación de código.
- El diagrama de clase representa clases, sus partes y la forma en la que las clases de los objetos están relacionados con otro.
- Una clase es una definición de un tipo de objeto.

Partes del diagrama de clases

- **Atributos:** describe las características de una clase de objetos.
- **Operaciones:** define el comportamiento de una clase de objetos
- **Estereotipos:** ayuda a entender este tipo de objeto en el contexto de otras clases de objetos con roles similares dentro del diseño del sistema.
- **Asociación:** es un término formal para un tipo de relación.
- **Herencia:** permite organizar las definiciones de la clase para simplificar y facilitar su implementación.

Clases

- Las clases son descripciones de un juego de objetos con características, comportamiento, relaciones y semánticas comunes.
- Se usan para modelar un juego de conceptos o entidades.
  - Se denotan con un rectángulo con compartimentos.

Clases

- En ellos se ponen el nombre, los atributos, las operaciones y además se pueden usar para anotar otras propiedades del modelo como son (reglas del negocio, responsabilidades, excepciones, etc.).
- Pueden tener interfaces para especificar conjuntos de operaciones proporcionadas a su ambiente. Todas las operaciones deben estar asociadas a métodos.
- Pueden tener relaciones de generalización con otras clases.

Atributos

- Son descripciones de características, se usan para modelar información asociada con una entidad, sintaxis:

Nombre_atributo [multiplicidad] : Tipo = Valor_inicial

- La multiplicidad es opcional e indica el número de atributos por instancia de la clase.

Operaciones

- Son descripciones del comportamiento, se usan para modelar los servicios u operaciones asociados con una entidad, esto es, lo que una entidad puede hacer, sintaxis:

Nombre_operación [parámetros:tipo] : Valor_retorno : tipo

Interfaces

- Son clases que definen un juego de operaciones externas accesibles pero sin métodos. Se usan para modelar una serie de operaciones que definen un servicio que puede ser ofrecido por diferentes clases.
- Se representan como clases pero con el estereotipo <<***interface***>>.
- Solo contienen operaciones públicas

Todos los diagramas soportan el Diagrama de Clase

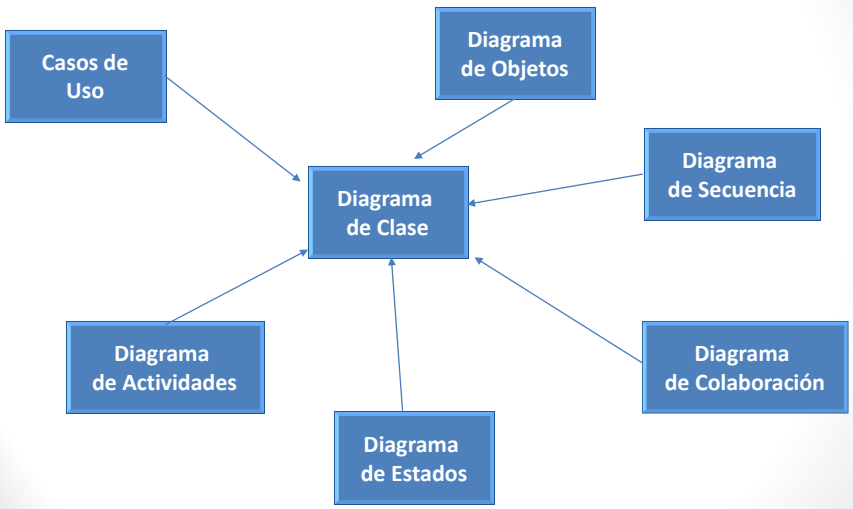


Diagrama de Objetos

- La **clase** define las *reglas;* los objetos expresan los *hechos.*
- La clase define que *puede ser;* el **objeto** describe *que es.*
- Se considera un caso especial del diagrama de clases.
- Puede construirse junto con el de las clases.
- Describe una instancia de un diagrama de clase en un momento en particular.
- Este diagrama contiene objetos y ligas.

Modelando Clases

- La representación de una clase es un rectángulo con 3 divisiones:
  - El del nombre define la clase, (un tipo de objeto).
  - El de los atributos contiene la definición de los datos.
  - El de las operaciones contiene la definición de cada comportamiento soportado por este tipo de objeto.

Ejemplo

La siguiente figura muestra un vuelo de una aerolínea modelado como una clase UML.

Modelando un atributo

- Un atributo describe una pieza de información que un objeto tiene o conoce de sí mismo. Para poder usar esta información se debe asignar un nombre y especificar el tipo de dato.
- El tipo de dato puede ser primitivo o tipo de dato abstracto (definido)
- Cada atributo puede tener reglas que limiten los valores asignados a éste. Se puede usar un valor de default para protegerlo.

Visibilidad de un atributo

- La definición de un atributo debe especificar que otros objetos los pueden ver. La visibilidad es:
  - **Public** (+) permite el acceso a objetos de las otras clases.
  - **Private** (-) limita el acceso a la clase, solo operaciones de la clase tienen acceso.
  - **Protected** (#) permite el acceso a subclases. En el caso de generalización (herencia), las subclases deben tener acceso a los atributos y operaciones de la superclase, sino no pueden heredar.
  - **Package** (~) permite el acceso a los otros objetos en el mismo paquete.

Ejemplo: Especificación de un atributo

Elemento	Ejemplo
Nombre del atributo	compañía
Tipo de dato	compañía:character
Valor de default (si hay)	compañía:character = espacios
Restricciones	compañía:character = espacios {1 a 30}
Caracteres	compañía:character = espacios{1 a 30 alfabéticos, espacios, puntuación, no especiales}
Visibilidad	- compañía: character = espacios {1 a 30 alfabéticos, …….

Modelando una Operación

- Los objetos tienen comportamientos, cosas que puedan hacer y que se les puedan dar a éstos.
- Las operaciones requieren un nombre, argumentos y a veces un valor de retorno.
- Las reglas de privacidad se aplican en la misma forma que para los atributos: Private, Public, Protected y Package.

Ejemplo Especificación de una operación

Elemento	Ejemplo
Nombre	totalOrderAmount
Definir argumentos/ Parámetros, corresponden a una instancia de Order	totalOrderAmount (order: integer)
Definir el tipo de dato de retorno	totalOrderAmount (order: integer) : Dollar
Identificar y describir restricciones	totalOrderAmount (order: integer) : {El total es la suma de cada item (p.u. x cantidad)
Visibilidad	+ totalOrderAmount (order: integer) : {El total es la suma ….

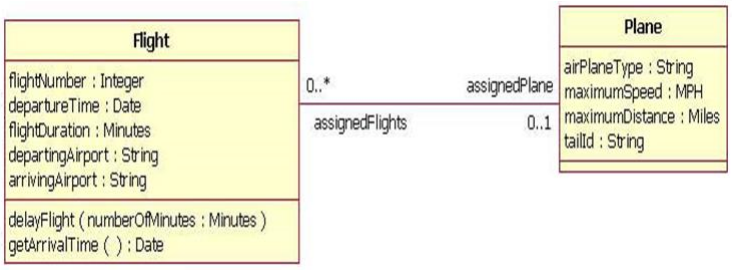
Diagrama de Clases: Asociaciones

- El propósito de la asociación puede expresarse en un nombre, verbo o frase que describa como los objetos de un tipo (clase) se relacionan con objetos de otro tipo (clase).
- Por ejemplo:

  Una persona **tiene** un coche

  Una persona **maneja** un coche
- Multiplicidad: cuantos objetos van a participar en la relación

Asociaciones

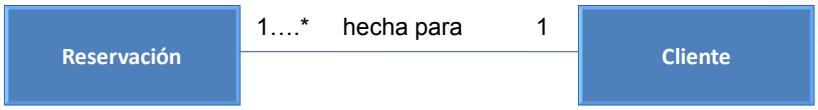


- Se indica el rol y la multiplicidad.
- Un vuelo está asociado con un avión y un avión puede tener asociados ninguno o varios números de vuelo.

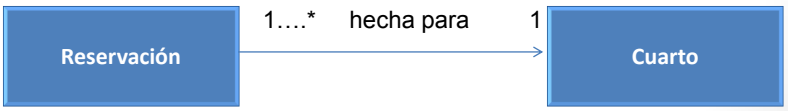
Dirección

- La dirección en las flechas de la asociación determinan en que dirección puede recorrerse una asociación en el momento de la ejecución.
- Una asociación sin flechas significa que se puede ir de un objeto a otro y viceversa.
- Por ejemplo la siguiente el tipo de flecha en la asociación implica que desde el objeto Reservación puedes recuperar (dirigirte hacia) el objeto *Cliente*. También implica que del objeto *Cliente* puedes recuperar el juego de reservaciones para ese cliente.

Dirección



- Supongamos que los requerimientos para un sistema de reservaciones requieren que “desde una reservación”, el sistema pueda recuperar el cuarto.

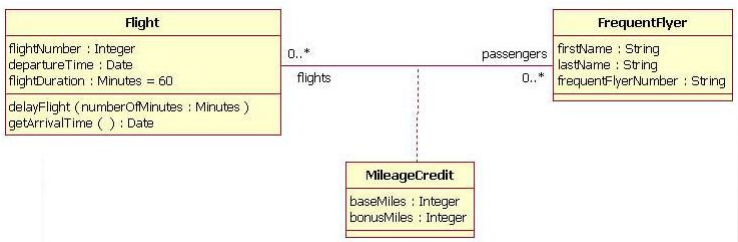


Clase Asociación

- Cuando se modela una asociación entre clases, a veces es necesario incluir otra clase que contiene información valiosa acerca de la relación.
- Se representa como una clase normal solo que la línea que la une con la línea que conecta las asociaciones primarias es punteada.
- La siguiente figura muestra una clase asociación para el ejemplo de los vuelos.

Clase Asociación

- La asociación entre la clase *Flight* y *FrequentFlyer* es a través de una clase llamada *MileageCredit*. Esto significa que debe haber una instancia en esta clase cuando alguna instancia de la clase Flight se asocie con una instancia de la clase *FrequentFlyer*.



Pasos para el diagrama de clases

- Identificar las clases.
- Mostrar los atributos y operaciones (posteriormente)
- Dibujar asociaciones
- Etiquetar asociaciones y en caso necesario los roles
- Indicar multiplicidad
- Dibujar fechas de dirección

Ejercicio

- Supongamos que las personas que de trabajan en una empresa se tienen registradas sus habilidades, esto significa que cualquier empleado puede tener cualesquiera habilidades. ¿Es necesario crear una clase asociación que contenga la información de ambas clases?
- Dibujar las entidades y su asociación.

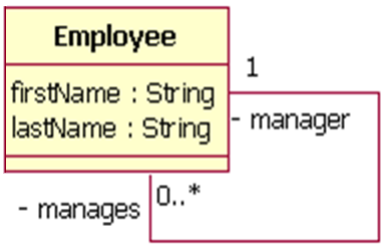
Asociación Reflexiva

- Una clase puede asociarse con sí misma. Una clase Empleado puede relacionarse con sí misma a través del rol gerente/dirige.

- No significa que una instancia está relacionada consigo misma, sino que una instancia de la clase está relacionada con otra instancia de la misma clase.

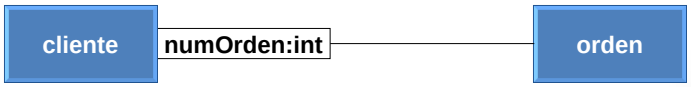
Asociación Reflexiva

- Una instancia de *Employee* puede ser el gerente de otras instancias de Employee. Como el rol ***manages*** tiene una multiplicidad de 0…*, significa que puede no tener otros empleados a quien dirigir. Una instancia de *Employee* tiene 1 sólo gerente o un solo director.



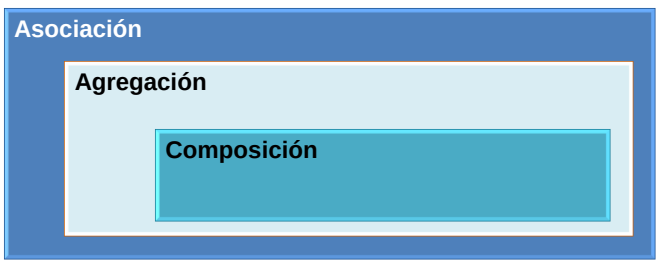
Asociación Cualificada

- Un **cualificador** es un atributo de la clase en el lado opuesto de la asociación, que permite hacer una búsqueda en función a su valor. Por ejemplo "El cliente usa el *numOrden* para buscar una orden".
- Un tipo de objeto usa el cualificador para accesar el otro tipo de objeto.



Agregación y Composición

- Cada agregación es un tipo de asociación.
- Cada composición es una forma de agregación.



Agregación básica

- Es un tipo especial de asociación utilizado para modelar una relación "whole to its parts".
- Por ejemplo, *Coche* es una entidad "whole" y *Llanta* es una parte del *Coche*.
- Una asociación con una agregación indica que una clase es parte de otra clase.
- En este tipo de asociación, la clase hijo puede sobrevivir sin su clase padre.

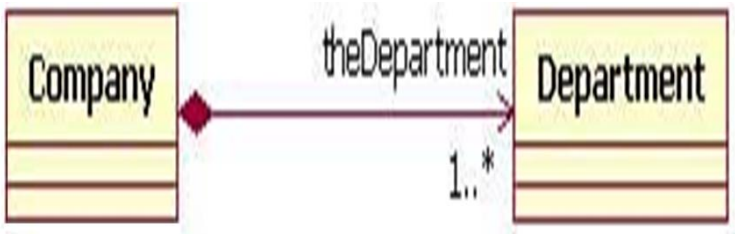
Agregación básica

- Para representar una relación de agregación, se dibuja una línea sólida de la clase padre (total) a la clase hijo (parte), y con un diamante en el lado de la clase padre. Una llanta puede existir sin automóvil.

Agregación / Composición

- En este caso el ciclo de vida de una instancia de la clase hijo depende del ciclo de vida de una instancia de la clase padre.
- A diferencia de la agregación básica, para representarla el diamante no es hueco.
- Una instancia de la clase *Company* debe tener al menos una en la clase *Departamento*.
- En este tipo de relaciones, si una la instancia *Company* se elimina, automáticamente la instancia *Departamento* también se elimina.
- Otra característica importante es que la clase hijo solo puede relacionarse con una instancia de la clase padre.

Agregación / Composición



Ejercicios Agregación y Composición

Hacer los diagramas de asociación indicando si existe agregación / composición. Anotar la multiplicidad.

1. Jugadores de basketball y equipos
2. Libro y capítulos del libro
3. Motor y automóvil
4. Líneas de un pedido y el pedido
5. En una empresa se llevan a cabo proyectos, estos proyectos están formados por una o más actividades y a su vez cada actividad tiene 1 o más tareas específicas. Cada tarea es asignada a un empleado y los empleados pueden o no tener asignadas tareas.

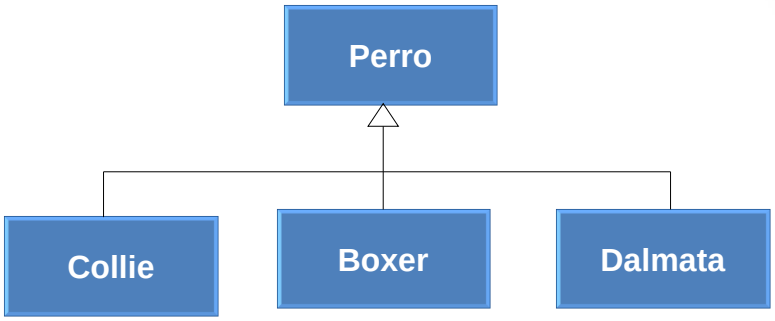
Generalización

- Son asociaciones entre elementos más generales y elementos más específicos, en los cuales éstos últimos son consistentes totalmente con los primeros, por lo que heredan las características proporcionadas por lo elementos generales y además pueden aumentar información.
- Este tipo de relación también se conoce como herencia.
- En una generalización no hay multiplicidad ni roles.
- La visibilidad "**protected**" permite que solo objetos de la misma clase o subclase vean el elemento.

Elementos de la generalización

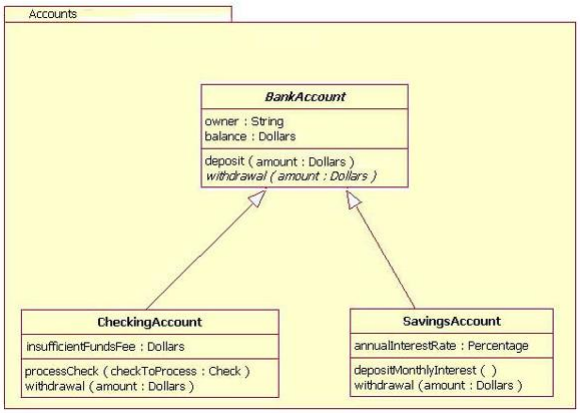
- Para dibujarla, hay que definir:
  - **Superclase:** es una clase que contiene alguna combinación de atributos, operaciones y asociaciones que son comunes a dos o más tipos de objetos que comparten el mismo propósito.
  - **Subclase:** es una clase que contiene una combinación de atributos, operaciones y asociaciones que son únicas a un tipo de objeto definido por una superclase.
  - La superclase es reutilizada por la subclase.

Herencia



Paquetes

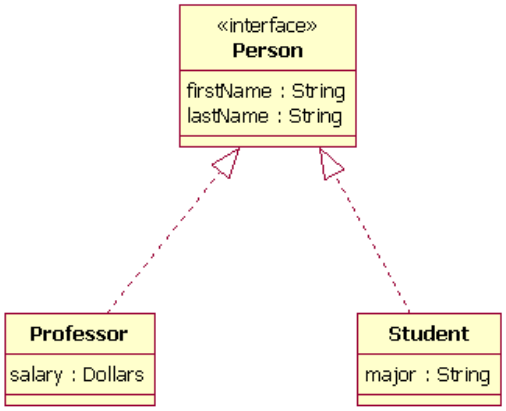
- Es un elemento organizador que proporciona UML al dividir el sistema en paquetes lo hace más fácil de entender.



Interfaces

- Una clase tiene una instancia de su tipo, mientras que una interface debe tener al menos una clase para implantarla. En UML, una interface es considerada como una especialización de una clase.
- Una interface se dibuja como una clase, pero en el compartimento superior del rectángulo aparece un texto o una inicial que indica que se trata de una interface y no de una clase.
- Una **interface** no es una clase.

Interfaces



Ejemplo interface

En el diagrama anterior las clases *Professor* y *Student* implementan a la interface *Person* y no heredan de ésta, podemos deducirlo a partir de:

1. El objeto *Person* de acuerdo a la simbología del diagrama está como una interface y *Professor* y *Student* están como clases.
2. No se trata de herencia ya que la línea con la flecha está punteada y no sólida.

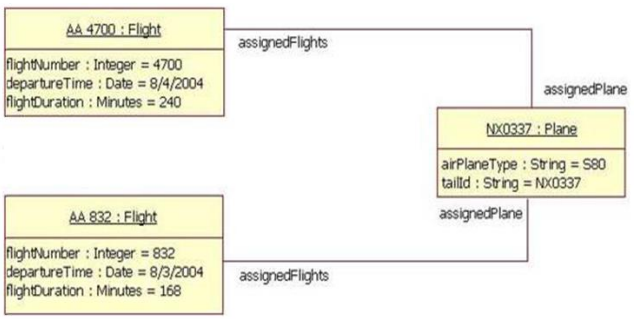
Instancias

- Cuando se modela la estructura de un sistema, a veces es útil mostrar ejemplos de las instancias de las clases.
- UML proporciona el elemento *instance especification*, que muestra información importante utilizando un ejemplo.
- La notación es la misma que la de una clase, solo que en el espacio superior el nombre se forma con:

  nombre de la instancia : nombre de la clase

Instancias

- Además de mostrar las instancias es muy útil mostrar sus relaciones, el ejemplo muestra dos instancias de la clase *Flight*, ya que el diagrama de clase indica que la relación entre la clase *Plane* y la clase *Fight* es 0 a muchos:



Roles

- Se puede incluir el rol de las clases, el siguiente ejemplo de los roles jugados por la clase *Employee* (de la asociación reflexiva), mostramos que la relación es entre un *Employee* jugando el papel de gerente y un *Employee* jugando el rol de miembro del equipo.

Caso de estudio

- **Establecimiento del problema: para el sistema de control de inventario:**

  "Nuestro sistema está diseñado para inventariar y embarcar únicamente productos identificados. Estos productos pueden ser comprados directamente de proveedores y revenderlos, o podemos empacarlos juntos para crear un producto especial. Los clientes colocan órdenes para uno o más ítems pero nosotros detectamos en el sistema clientes que hayan o no hayan comprado. Cada ítem corresponde a un producto. Identificamos cada producto usando un número serial único. El cliente puede preguntar sobre el estatus de su orden utilizando el número de orden.

Caso de Estudio

Los embarques de productos de los proveedores se reciben y se colocan en el inventario. Cada producto es asignado a una ubicación con lo que se puede encontrar fácilmente cuando se surten las órdenes. Cada ubicación tiene un identificador único. Las órdenes para los clientes se embarcan a medida que los productos están disponibles, por lo que puede haber más de un envío para satisfacer una sola orden de compra, pero puede ser que un solo embarque contenga productos de múltiples órdenes. Los ítems que no se entregaron son colocados en una backorder con una referencia a la orden original".

Construyendo el diagrama de clase

1. Identificar las clases, nombrarlas y definirlas con lo que sabes que son parte del modelo.
2. Identificar, nombrar y definir las asociaciones entre pares de clases. Tener cuidado con clases reflexivas, asignar multiplicidad.
3. Evaluar cada asociación para determinar si debe ser una agregación y cada agregación para ver si debe ser una composición
4. Evaluar las clases para posible generalización (herencia).