Universidad Andrés Bello®

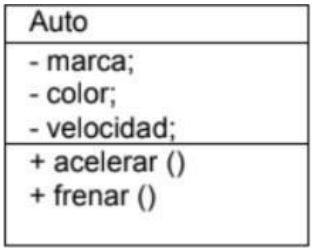
Conectar · Innovar · Liderar

FUNDAMENTOS DE PROGRAMACIÓN EN JAVA

El Paradigma de Orientación a Objeto

Diagramas de Clases

Un diagrama de clases es una herramienta de modelado UML que sirve para representarel diseño de un programa bajo el paradigma de la Programación Orientada a Objetos y que permite visualizar atributos, métodos y las relaciones entre diferentes clases. En estosdiagramas, una clase se representa con un rectángulo que posee tres divisiones: nombrede la clase, atributos de la misma y mensajes que entiende.



En la división superior se encuentra el nombre de la clase. En la segunda parte van los atributos o variables de instancia; las variables de clase van subrayados. En el último cuadro se escriben las operaciones, es decir, los mensajes que puede entender. Lo importante es documentar los mensajes más relevantes y no todos los mensajes de un solo objeto. Se debe tener en cuenta que una clase que no tiene comportamiento, dichode otro modo, no está comunicando qué tipo de rol cumple en la solución, así que o está faltando definir qué es lo que le puede pedir o entonces esa clase no debería estar en el diagrama.

Relaciones

Los tipos más importantes de relaciones estáticas entre clases se indican a continuación.

Asociación

Las relaciones de asociación representan un conjunto de enlaces entre objetos o instancias de clases. Es el tipo de relación más general, y denota básicamente unadependencia semántica. Por ejemplo, una Persona trabaja para una Empresa.

Cada asociación puede presentar elementos adicionales que doten de mayor detalle altipo de relación:

- Rol, o nombre de la asociación, que describe la semántica de la relación en el sentido indicado. Por ejemplo, la asociación entre Persona y Empresa recibe el nombre de trabaja para, como rol en ese sentido.
- Multiplicidad, que describe la cardinalidad de la relación, es decir, especifica cuántas instancias de una clase están asociadas a una instancia de la otra clase. Los tipos de multiplicidad son: Uno a uno, uno a muchos y muchos a muchos.

Herencia

Las jerarquías de generalización/especialización se conocen como herencia. La herencia es el mecanismo que permite a una clase de objetos incorporar atributos y métodos de otra clase, añadiéndolos a los que ya posee. Con la herencia se refleja una relación “es un” entre clases. La clase de la cual se hereda se denomina superclase, y la que hereda subclase.

La generalización define una superclase a partir de otras. Por ejemplo, de las clases Profesor y Estudiante se obtiene la superclase Persona.

La especialización o especificación es la operación inversa, y en ella una clase se descompone en una o varias subclases. Por ejemplo, de la clase empleado se pueden obtener las subclases secretaria, técnico e ingeniero.

Agregación

La agregación es un tipo de relación jerárquica entre un objeto, que representa la totalidad de ese objeto y las partes que lo componen. Permite el agrupamiento físico de estructuras relacionadas lógicamente. Los objetos “son-parte-de” otro objeto completo. Por ejemplo, motor, ruedas, carrocería son parte de automóvil.

Composición

La composición es una forma de agregación, en donde la relación de propiedad es más fuerte, e incluso coinciden los tiempos de vida del objeto completo y las partes que lo componen. Por ejemplo, en un sistema de Máquina de café, las relaciones entre la clase máquina y producto, o entre máquina y depósito de monedas, son de composición.

Dependencia

Una relación de dependencia se utiliza entre dos clases o entre una clase y una interfaz,e indica que una clase requiere de otra para proporcionar alguno de sus servicios.

Interfaces

Una interfaz es una especificación de la semántica de un conjunto de operaciones de unaclase o paquete que son visibles desde otras clases o paquetes. Normalmente, se corresponde con una parte del comportamiento del elemento que la proporciona.

Paquetes

Los paquetes se usan para dividir el modelo de clases del sistema de información,agrupando clases u otros paquetes según los criterios que sean oportunos. Las dependencias entre ellos se definen a partir de las relaciones establecidas entre los distintos elementos que se agrupan en estos paquetes.

Clases

Una clase se representa como una caja, separada en tres zonas por líneas horizontales. En la zona superior se muestra el nombre de la clase y propiedades generales como el estereotipo. El nombre de la clase aparece centrado y si la clase es abstracta se representaen cursiva. El estereotipo, si se muestra, se sitúa sobre el nombre y entre el símbolo <<
.... >>.

La zona central contiene una lista de atributos, uno en cada línea. La notación utilizada para representarlos incluye, dependiendo del detalle, el nombre del atributo, su tipo y suvalor por defecto, con el formato:

```
visibilidad nombre : tipo = valor-inicial { propiedades }
```

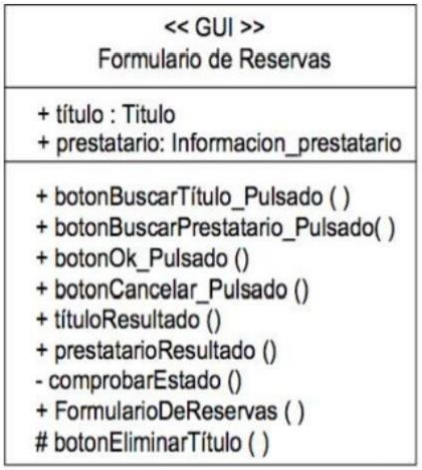
La visibilidad será en general publica (+), privada (-) o protegida (#), aunque puede haberotros tipos de visibilidad dependiendo del lenguaje de programación empleado.

En la zona inferior se incluye una lista con las operaciones que proporciona la clase. Cadaoperación aparece en una línea con formato:

```
visibilidad nombre (lista-de-parámetros): tipo-devuelto {
propiedad }
```

La visibilidad será en general publica (+), privada (-) o protegida (#), aunque como con los atributos, puede haber otros tipos de visibilidad dependiendo del lenguaje de programación. La lista de parámetros es una lista con los parámetros recibidos en la operación separados por comas. El formato de un parámetro es:

```
nombre : tipo = valor-por-defecto
```



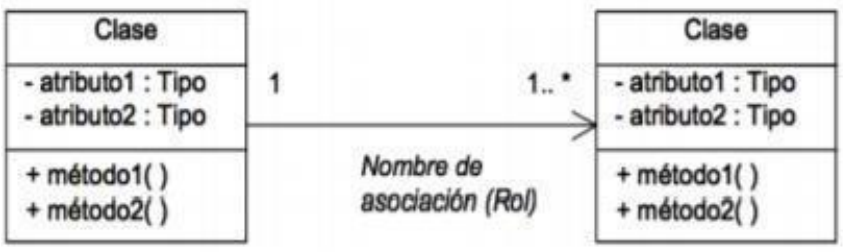
La notación especificada se puede simplificar según el nivel de detalle con el que se quiera trabajar en un momento dado.

Relaciones

Una relación de asociación se representa como una línea continua entre las clasesasociadas. En una relación de asociación, ambos extremos de la línea pueden conectar con la misma clase, indicando que una instancia de una clase está asociada a otras instancias de la misma clase, lo que se conoce como asociación reflexiva. La relación puede tener un nombre y un estereotipo, que se colocan junto a la línea. El nombre suelecorresponderse con expresiones verbales presentes en las especificaciones, y define la semántica de la asociación. Los estereotipos permiten clasificar las relaciones en familiasy se escribirán entre el símbolo << ... >>.

Las diferentes propiedades de la relación se pueden representar con la siguientenotación:

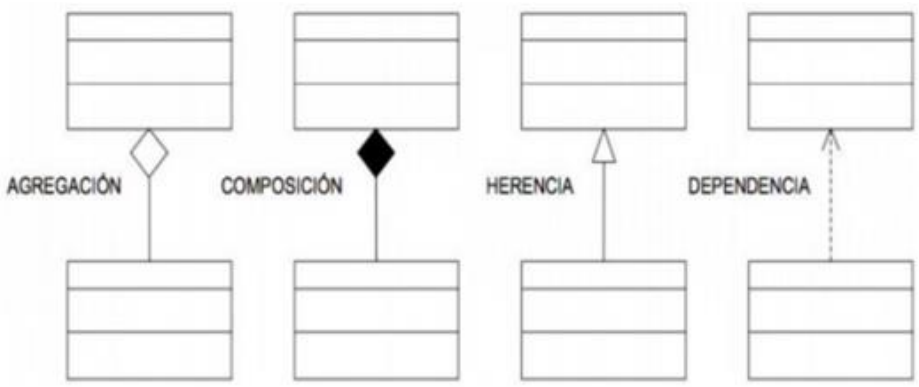
- **Multiplicidad:** La multiplicidad puede ser un número concreto, un rango o una colección de números. La letra "n" y el símbolo "*" representan cualquier número.
- **Orden:** Se puede especificar si las instancias guardan un orden con la palabra clave "{ordered}". Si el modelo es suficientemente detallado, se puede incluir una restricción que indique el criterio de ordenamiento.
- **Navegabilidad:** La navegación desde una clase a la otra se representa poniendo una flecha sin relleno en el extremo de la línea, indicando el sentido de lanavegación.
- **Rol o nombre de la asociación:** Este nombre se coloca junto al extremo de la líneaque está unida a una clase, para expresar de que forma esa clase hace uso de la otra clase con la que mantiene la asociación.



Además, existen notaciones específicas para los otros tipos de relación. De este grupo sedestacan:

- **Agregación:** Se representa con un rombo hueco en la clase cuya instancia es una agregación de las instancias de la otra.
- **Composición:** Se representa con un rombo lleno en la clase cuya instancia contiene las instancias de la otra clase.
- **Dependencia:** Corresponde a una línea discontinua con una flecha apuntando a laclase cliente. La relación puede tener un estereotipo que se coloca junto a la línea,y entre el símbolo: << ... >>.

- **Herencia:** Esta relación se representa como una línea continua con una flecha hueca en el extremo que apunta a la superclase.

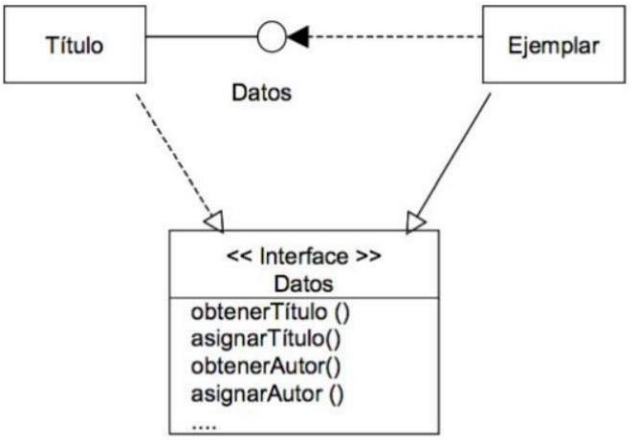


Interfaces

Una interfaz se representa como una caja con compartimentos, igual que las clases. En lazona superior se incluye el nombre y el estereotipo (esto último entre los símbolos <>). La lista de operaciones se coloca en la zona inferior, igual que en las representaciones declases. La zona en la que se listan los atributos estará vacía, o simplemente puede omitirse.

Existe una representación más simple para la interfaz: un circulo pequeño asociado a unaclase con el nombre de la interfaz debajo. Las operaciones de la interfaz no aparecen en esta representación; si se quiere que aparezcan, debe usarse la primera notación.

Entre una clase que implementa las operaciones que una interfaz ofrece y esa interfaz seestablece una relación de realización que, dependiendo de la notación elegida, se representará con una línea continua entre ellas cuando la interfaz se representa como uncírculo y con una flecha hueca discontinua apuntando a la interfaz cuando se represente como una clase.

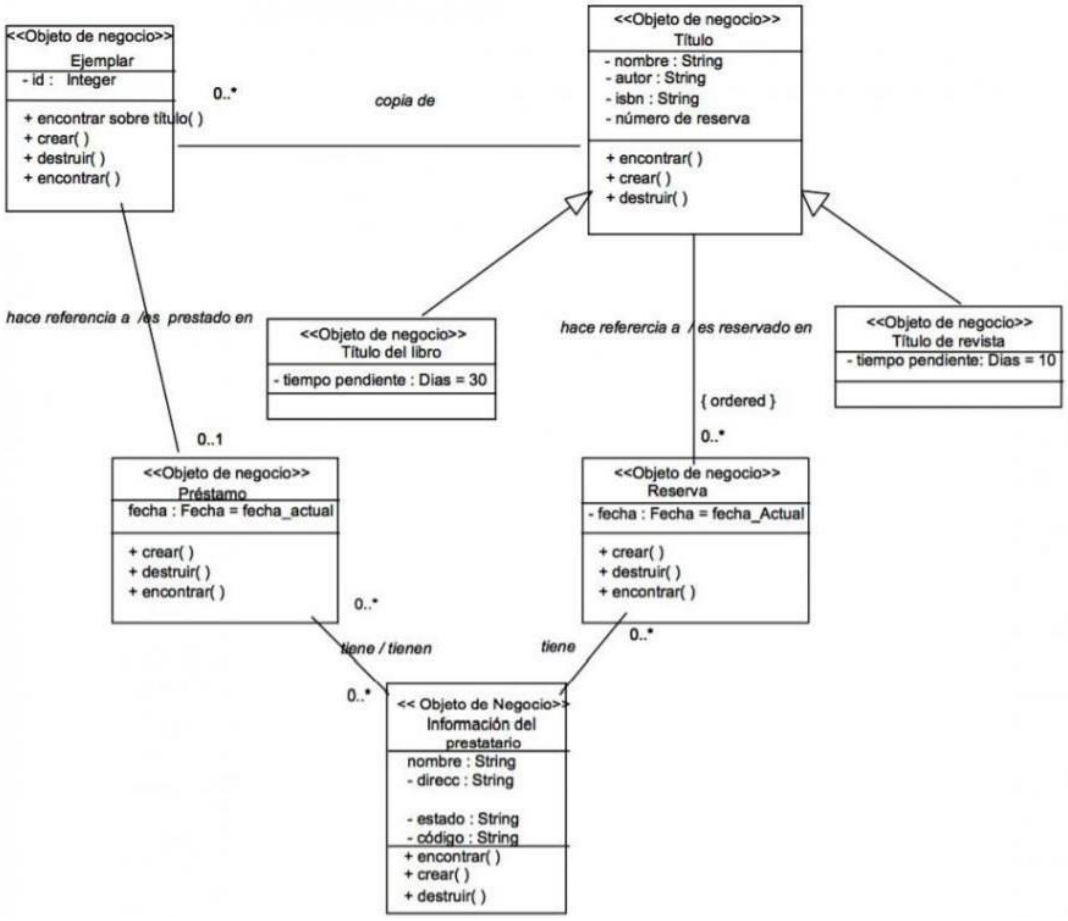


Paquetes

Los paquetes se representan mediante un icono con forma de carpeta y las dependenciascon flechas discontinuas entre los paquetes dependientes.

Ejemplo 1: Desarrolle un diagrama de clases para un sistema encargado de la gestión de préstamos y reservas de libros y revistas de una biblioteca.

Dependiendo del momento del desarrollo el diagrama estará más o menos detallado. Así, el diagrama tendría la siguiente estructura en el proceso de análisis:



Anexo: Referencias

1.- Lenguaje Java y Entorno de Desarrollo
Referencia: http://www.jtech.ua.es/j2ee/2006-2007/doc/sesion01-apuntes.pdf

2.- Diagramas de clases
Referencia: https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml