ENTORNOS DE DESARROLLO

IES Santiago Hernández
Curso 2017-2018
Ignacio Agudo Sancho

8. Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

8. Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

Diagramas de Clases

- Introducción:
  - Diseño Orientado a Objetos: un sistema se entiende como un conjunto de objetos que tienen propiedades y comportamientos.
  - Objeto: consta de una estructura de datos (atributos) y una colección de métodos u operaciones que manipulan esos datos.
  - Clase: es una plantilla para la creación de objetos.
  - UML (Unified Modeling Language – Lenguaje de Modelado Unificado): es el estándar de lenguaje de modelado basado en diagramas que sirve para expresar modelos (representaciones de la realidad donde se ignoran detalles de menor importancia).

8. Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

Conceptos Orientados a Objetos

- El paradigma OO se basa en el concepto de objeto como aquello que tiene:
  - estado (propiedades más valores),
  - comportamiento (acciones y reacciones a mensajes),
  - identidad (propiedad que lo distingue de los demás)
- La estructura y comportamiento de objetos similares se define en su clase común.

Conceptos Orientados a Objetos

- Clases = estáticas, abstracciones de la esencia de los objetos.
- Objetos = dinámicos, existen en tiempo y espacio y ocupan memoria en la ejecución.
- En definitiva, una clase es un conjunto de objetos que comparten una estructura y comportamiento común.

Conceptos Orientados a Objetos

- El modelo OO tiene una serie de PROPIEDADES:
  - 1. Abstracción
  - 2. Encapsulación
  - 3. Modularidad
  - 4. Jerarquía o herencia
  - 5. Polimorfismo
  - 6. Tipificación
  - 7. Concurrencia
  - 8. Persistencia

Conceptos Orientados a Objetos

- 1. Abstracción:
  - Aislar un elemento de su contexto o del resto de elementos que lo acompañan para obtener una descripción formal.
  - Nos permite componer un conjunto de clases que permitan modelar la realidad o el problema que se quiere resolver.

Conceptos Orientados a Objetos

- 2. Encapsulación:
  - Ocultar todos los detalles de un objeto que no contribuyen a sus características esenciales.
  - Separar el aspecto externo del objeto accesible por otros objetos, del aspecto interno que será inaccesible.
  - Es decir, ocultar los atributos y métodos del objeto a otros objetos, pasando a ser atributos y métodos privados del objeto.

Conceptos Orientados a Objetos

- 3. Modularidad:
  - Es la propiedad de una aplicación o sistema que ha sido descompuesto en módulos o partes más pequeñas coherentes e independientes.
  - Estos módulos se pueden compilar por separado, pero tienen conexiones con los otros módulos.

Conceptos Orientados a Objetos

- 4. Jerarquía o herencia:
  - Se introduce la posibilidad de extender clases que heredan todo el comportamiento y código de la clase extendida.
    - Clase original = padre, base o superclase
    - Clase nueva = hija, derivada o subclase
  - La extensión de una clase es la herencia porque la clase hija hereda todos los métodos y atributos de la clase padre.
  - Cada subclase estará formada por objetos más especializados.

Conceptos Orientados a Objetos

- 5. Polimorfismo:
  - Reunir con el mismo nombre comportamientos diferentes.
  - Un mismo mensaje puede originar conductas diferentes al ser recibido por diferentes objetos.
  - Es decir: dos objetos de dos clases diferentes pueden responder a la llamada a métodos del mismo nombre, cada uno de ellos con distinto comportamiento encapsulado, pero que responden a una interfaz común (marcada a través de la herencia).

Conceptos Orientados a Objetos

- 6. Tipificación:
  - Es la definición precisa de un objeto, de tal forma que objetos de diferentes tipos no puedan ser intercambiados.

Conceptos Orientados a Objetos

- 7. Concurrencia:
  - Es la propiedad que distingue un objeto que está activo de uno que no lo está.
  - El objeto activo está haciendo algo.
  - Se utiliza sobre todo en programación multihilo.

Conceptos Orientados a Objetos

- 8. Persistencia:
  - La existencia del objeto trasciende el tiempo y/o el espacio.
  - El objeto sigue existiendo aunque su creador ya no exista.
  - Se refiere a objetos de clases asociadas a Bases de Datos Orientadas a Objetos o a Bases de Datos Objeto Relacionales.

8. Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

Qué es UML

- Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo del software.
- Para modelar los sistemas, se utilizan diagramas en los que se representan los distintos puntos de vista del modelado.
- En la V.2.5.1 de UML se definen 14 tipos de diagramas, divididos en 3 categorías:
  - De estructura (parte estática del modelo)
  - De comportamiento (parte dinámica del modelo)
  - De interacción (derivados del más general de los diagramas de comportamiento)

Qué es UML

- Diagramas de estructura (parte estática)
- Se centran en los elementos que deben existir en el sistema modelado
- Incluyen:
  - Diagrama de clases
  - Diagrama de estructuras compuestas
  - Diagrama de componentes
  - Diagrama de despliegue
  - Diagrama de objetos
  - Diagrama de paquetes
  - Diagrama de perfiles

Qué es UML

- Diagramas de comportamiento (parte dinámica)
- Se centran en lo que debe suceder en el sistema
- Incluyen:
  - Diagrama de Casos de Uso
  - Diagramas de Interacción
  - Diagrama de Actividad
  - Diagrama de Estado

Qué es UML

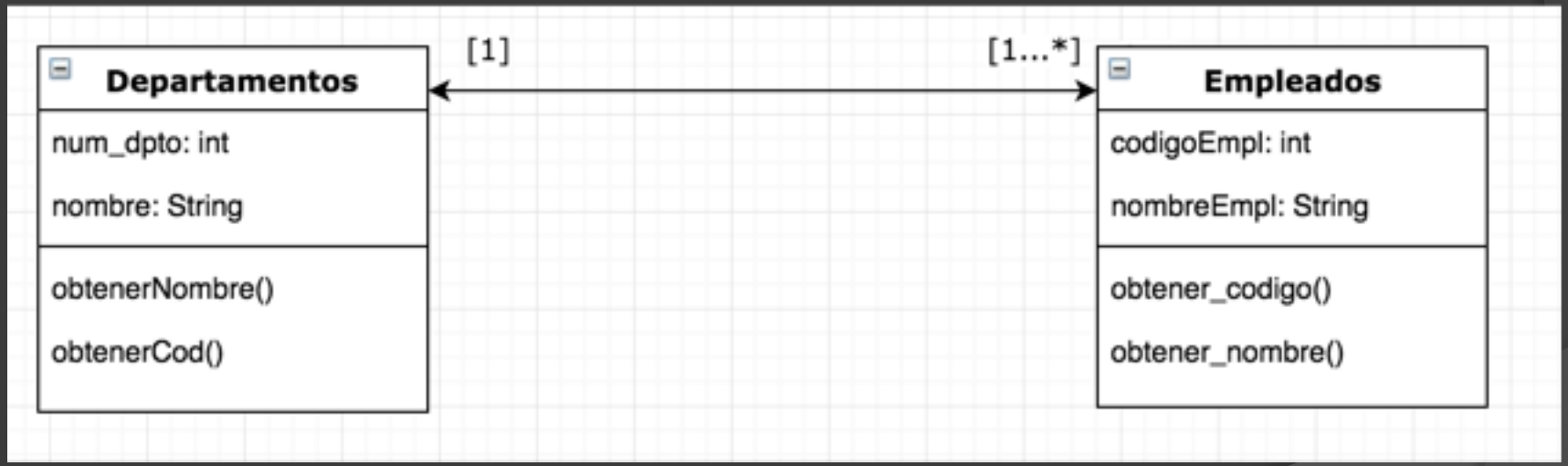
- Diagramas de interacción
- Todos derivados del diagrama de comportamiento más general.
- Se centran en el flujo de control y de datos entre los elementos del sistema modelado.
- Incluyen:
  - Diagrama de Secuencia
  - Diagrama de Comunicación
  - Diagrama de Tiempos
  - Diagrama Global de Interacción

Qué es UML. Tipos de diagramas.

- Cada diagrama representa alguna parte o punto de vista del sistema.
- Los diagramas más utilizados son:
  1. Diagramas de clases
  2. Diagramas de casos de uso
  3. Diagramas de secuencia
  4. Diagramas de estado
  5. Diagramas de actividad

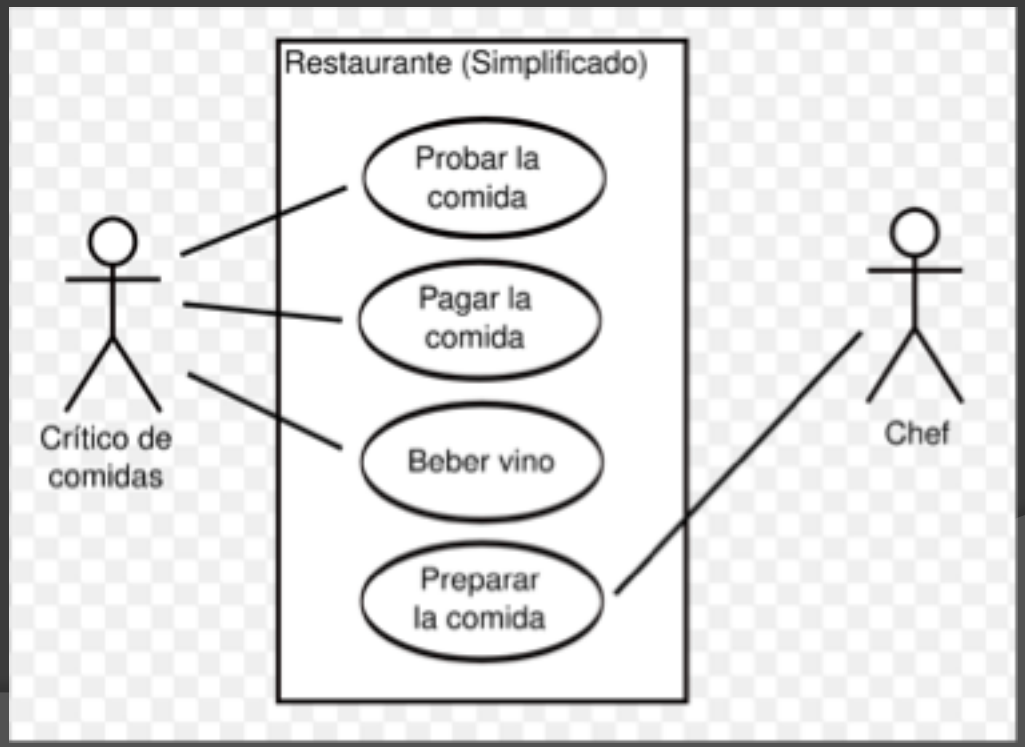
Qué es UML. Tipos de diagramas.

- 1. Diagramas de clases
  - Muestran las clases que componen un sistema y cómo se relacionan unas con otras



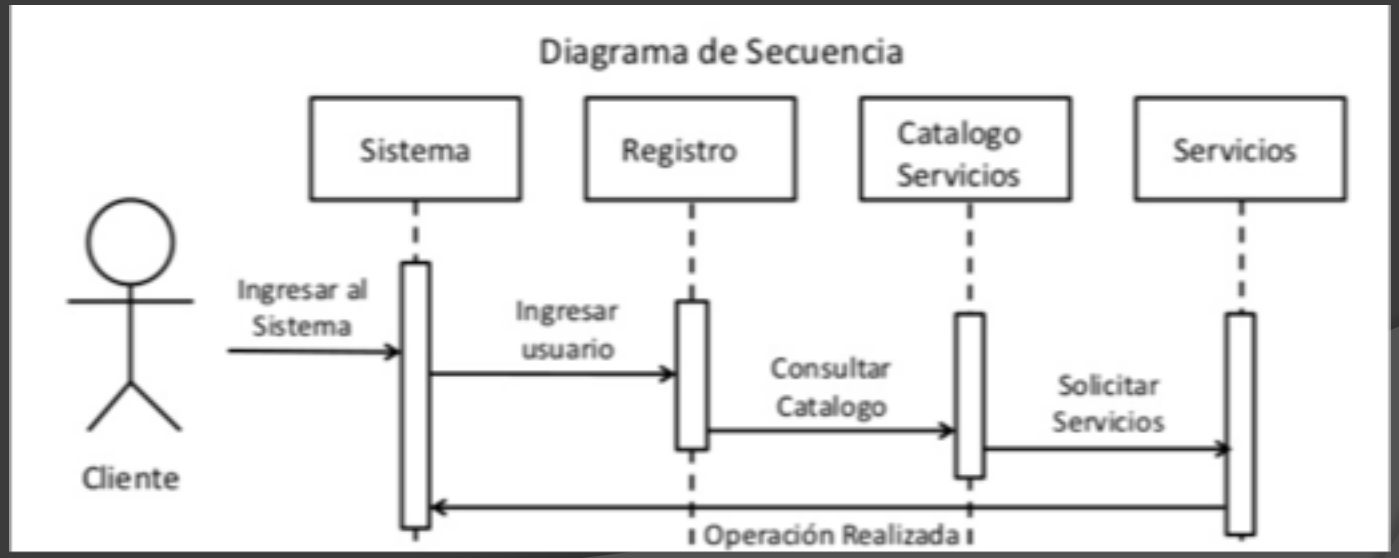
Qué es UML. Tipos de diagramas.

- 2. Diagramas de casos de uso (comportamiento)
  - Muestran un conjunto de actores, las acciones (casos de uso) que se realizan en el sistema, y las relaciones entre ellos.



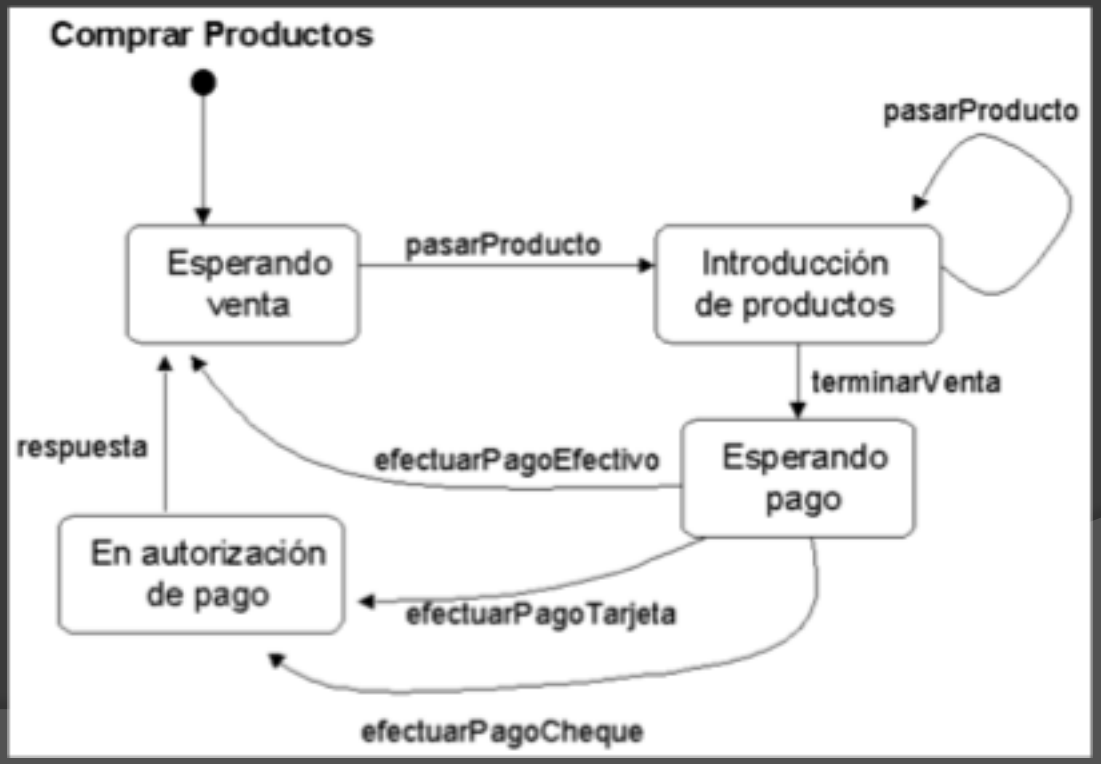
Qué es UML. Tipos de diagramas.

- 3. Diagramas de secuencia
  - Son una representación temporal de los objetos y sus relaciones. Enfatiza la interacción entre los objetos y los mensajes que intercambian entre sí junto con el orden temporal de los mismos.



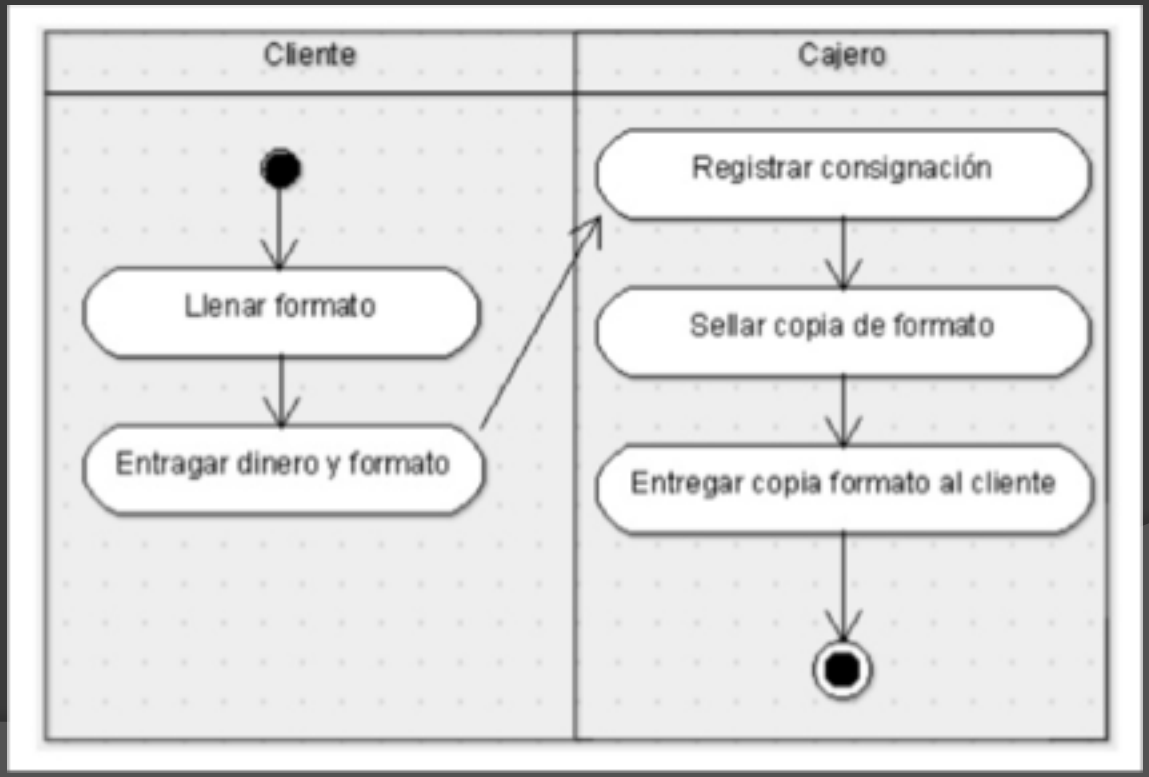
Qué es UML. Tipos de diagramas.

- 4. Diagramas de estado
  - Analizan los cambios de estado de los objetos.
  - Muestran los estados, eventos, transiciones y actividades de los objetos.



Qué es UML. Tipos de diagramas.

- 5. Diagramas de actividad
  - Muestran el flujo de trabajo desde un punto de inicio hasta el punto final, detallando las decisiones que surgen.



8. Diagramas de Clases

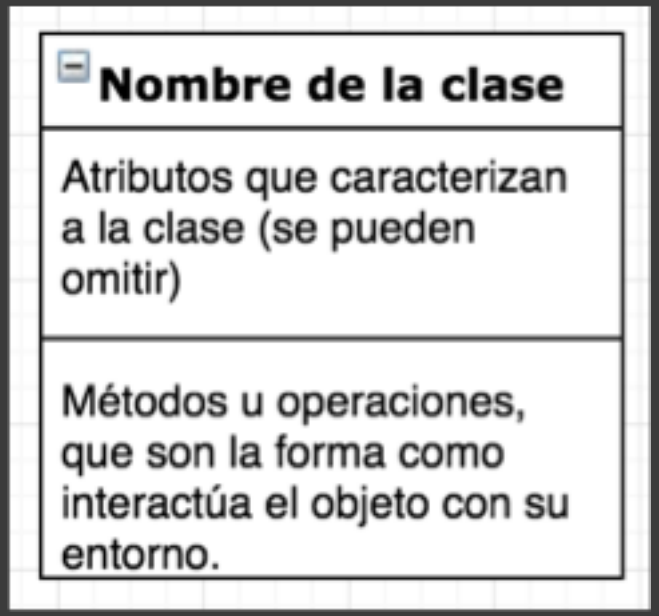
- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

Diagramas de Clases

- Es un tipo de diagrama de estructuras (estático) que describe la estructura de un sistema mostrando sus clases y asociaciones entre ellas.
- Sirve para visualizar las relaciones entre las clases que componen el sistema.
- Está compuesto por:
  - Clases: atributos, métodos y visibilidad.
  - Relaciones: asociación, herencia, agregación, composición, realización y dependencia.

Diagramas de Clases: Las Clases

- Son la unidad básica que encapsula toda la información de un objeto (instancia de una clase).
- Se representa de la siguiente manera:



- NOTA: La visibilidad de los atributos por defecto es private y de los métodos public.

Diagramas de Clases: Las Clases

- LOS ATRIBUTOS
- Representan propiedades de la clase que se encuentran en todas las instancias.
- Pueden representarse mostrando solo el nombre o nombre y tipo, incluso el valor por defecto.
- En UML los tipos básicos son Integer, String y Boolean.
- No obstante, se pueden indicar los tipos de cualquier lenguaje de programación.
- Se indicará la visibilidad con el entorno, la cual está relacionada con el encapsulamiento.

Diagramas de Clases: Las Clases

- LOS MÉTODOS
- Un método es la implementación de un servicio de la clase que muestra un comportamiento común a todos los objetos.
- Definen la forma en la que la clase interactúa con su entorno.

Diagramas de Clases: Las Clases

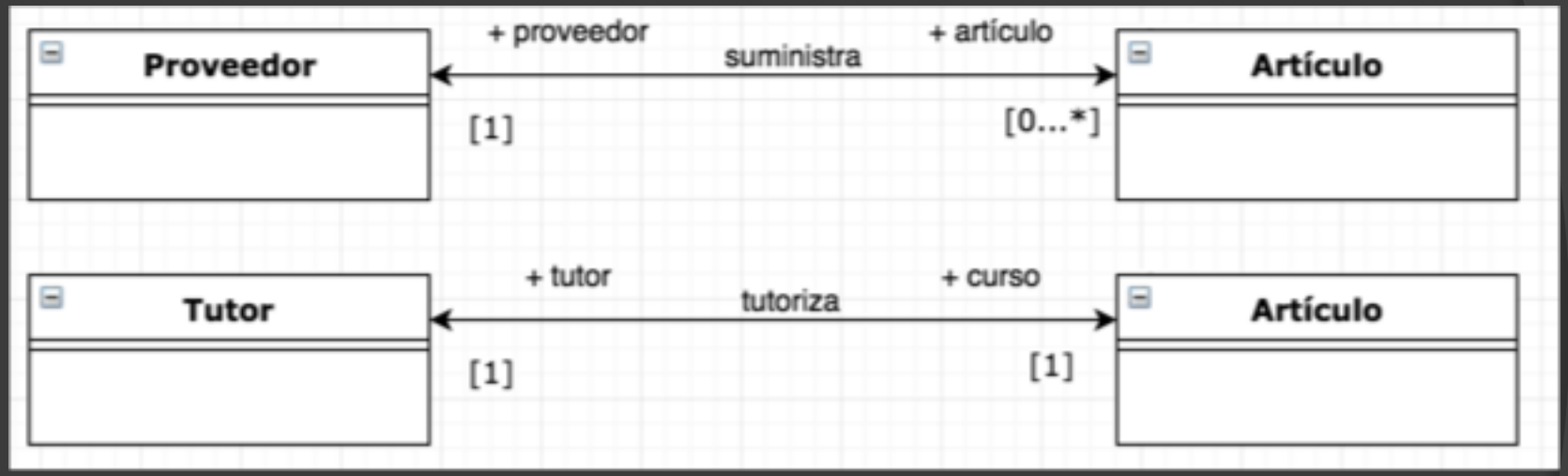
- Tanto los atributos como los métodos, pueden ser
  - public: (+) elemento no encapsulado visible y accesible desde dentro y desde fuera de la clase.
  - private: (-) elemento encapsulado visible y accesible desde dentro de la clase.
  - protected: (#) elemento encapsulado visible y accesible por la clase y las subclases que se deriven.
  - pakage: (~) elemento encapsulado visible solo en las clases del mismo paquete.

Diagramas de Clases: Las Relaciones

- Son asociaciones que representan vínculos existentes entre objetos.
- Tienen un nombre que refleja los elementos de la asociación.
- También tienen cardinalidad llamada multiplicidad, que representa el número de instancias de una clase que se relacionan con las instancias de otra clase (similar al modelo E/R).
- Notación para expresar las multiplicidades:
  - 0…1 1 *
  - 1…* M…N N

Diagramas de Clases: Las Relaciones

- Ejemplo de asociaciones y multiplicidades:



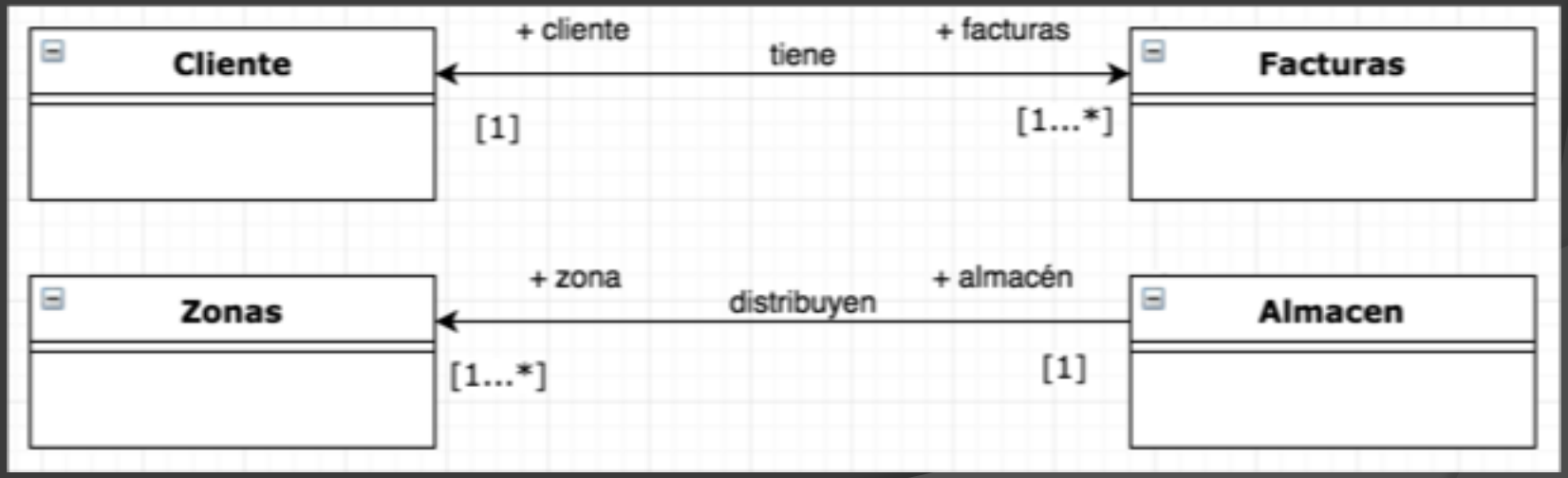
- Un proveedor puede suministrar de 0 a muchos artículos. Un artículo solo puede ser suministrado por un proveedor.
- Un tutor solo puede tutorizar un curso. Un curso solo puede ser tutorizado por un tutor.

Diagramas de Clases: Las Relaciones

- Podemos distinguir los siguientes tipos de relaciones:
  - 1. Asociación
  - 2. Herencia
  - 3. Composición
  - 4. Agregación
  - 5. Realización
  - 6. Dependencia

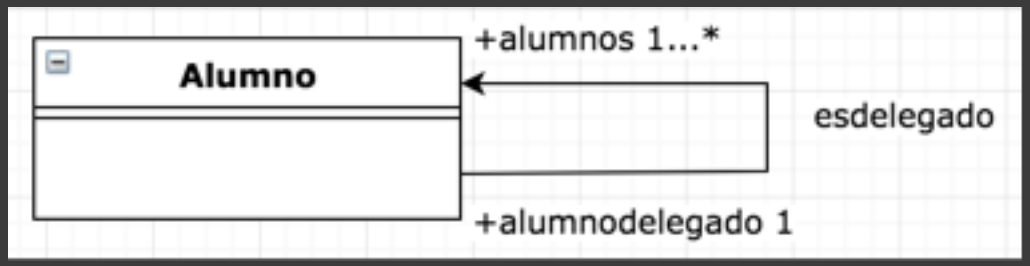
Diagramas de Clases: Las Relaciones

- 1. Asociación
  - Son conexiones entre clases.
  - Pueden ser unidireccionales o bidireccionales.
  - Unidireccional: solamente una de las clases conoce de la existencia de la otra.



Diagramas de Clases: Las Relaciones

- 1. Asociación
  - Una clase puede asociarse consigo misma creando una asociación reflexiva.
  - Estas asociaciones unen entre si instancias de una misma clase.

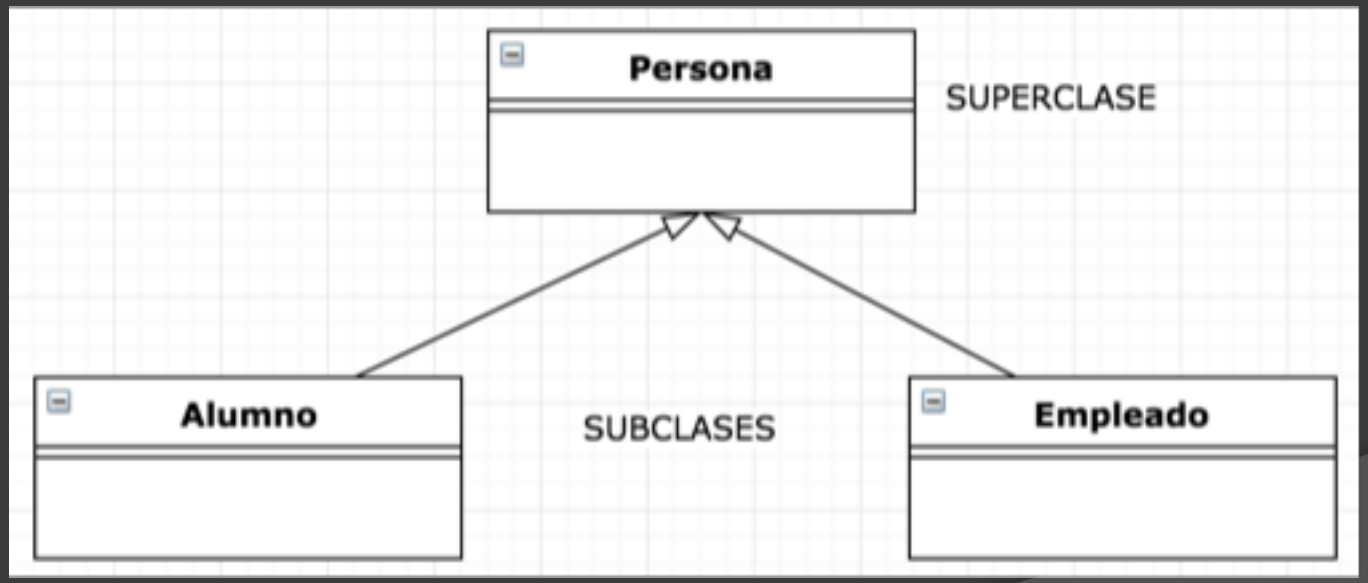


Diagramas de Clases: Las Relaciones

- 2. Herencia (generalización y especialización)
  - Las clases con atributos y operaciones comunes se pueden organizar de forma jerárquica mediante herencia.
  - Superclases y subclases.
  - La generalización define una relación entre una clase más generalizada y una o más versiones refinadas de ella.
  - La superclase generaliza alas subclases y las subclases especializan a la superclase.

Diagramas de Clases: Las Relaciones

- 2. Herencia (generalización y especialización)
  - La representación de la herencia se realiza de la siguiente manera:

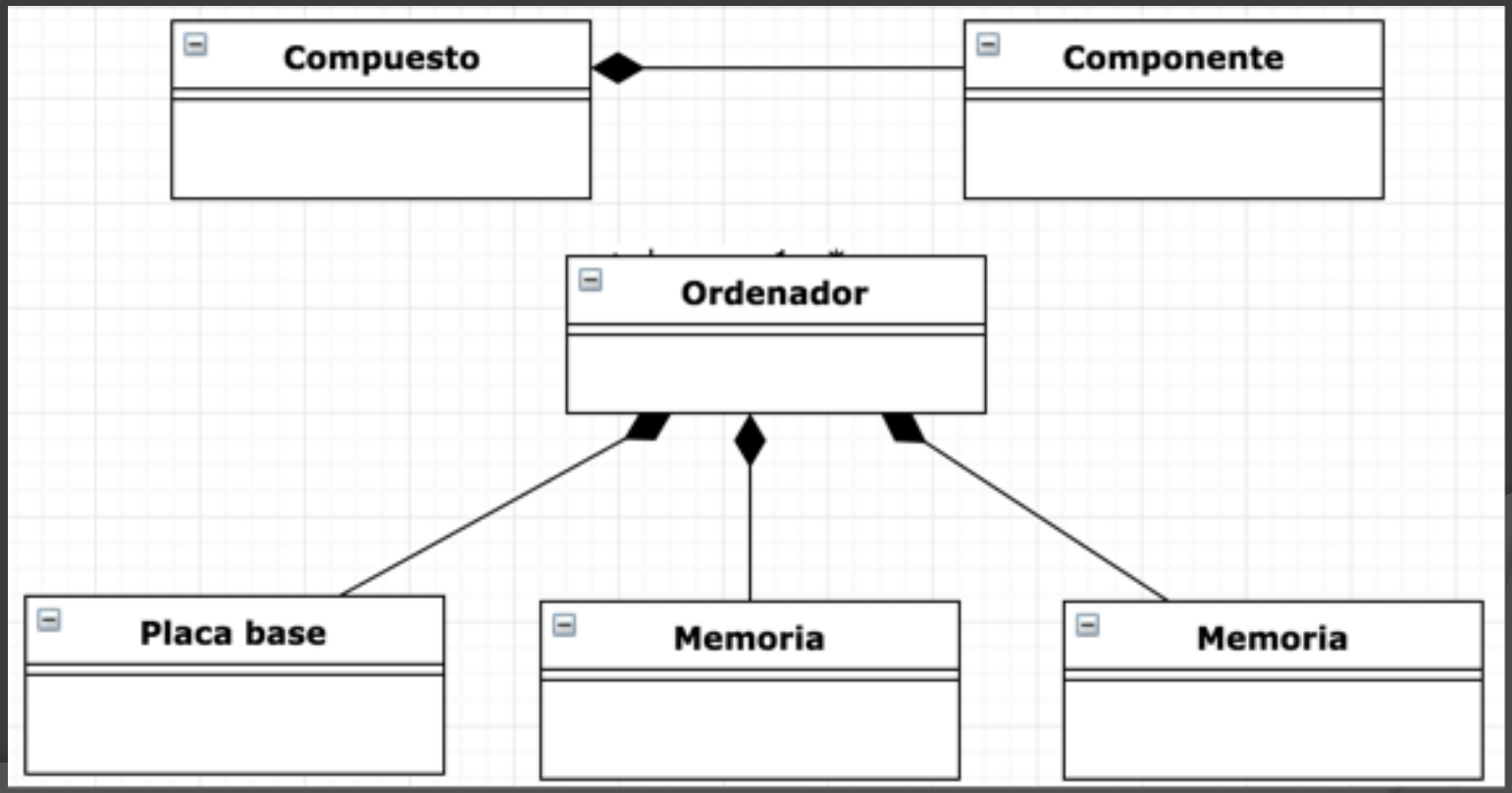


Diagramas de Clases: Las Relaciones

- 3. Composición
  - Un objeto puede estar compuesto por otros objetos.
  - La composición asocia un objeto complejo con los objetos que lo constituyen (sus componentes).
  - Dos tipos:
    - Composición fuerte
      - Los componentes no pueden ser compartidos por varios objetos compuestos. La cardinalidad será 1. Si se elimina el objeto compuesto, se eliminarán sus componentes.
    - Composición débil o agregación
      - Los componentes pueden ser compartidos.

Diagramas de Clases: Las Relaciones

- 3. Composición
  - La representación de la composición fuerte se realiza de la siguiente manera:

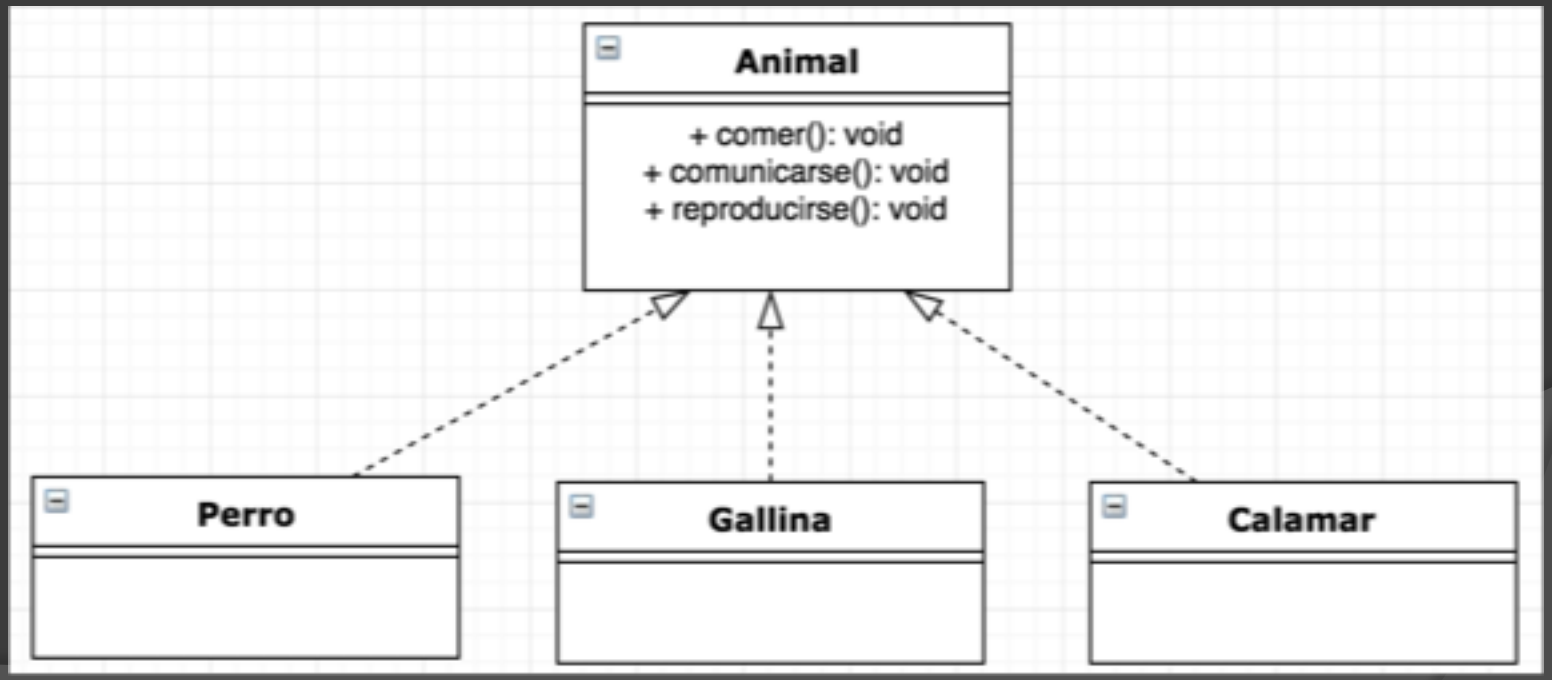


Diagramas de Clases: Las Relaciones

- 4. Agregación
  - Es la composición débil.
  - Los componentes pueden ser compartidos por varios compuestos.
  - Su representación es así:

Diagramas de Clases: Las Relaciones

- 5. Realización
  - Es la relación de herencia entre una clase interfaz y la subclase que implementa esa interfaz.
  - Su representación:



Diagramas de Clases: Las Relaciones

- 6. Dependencia
  - Se establece cuando una clase usa otra clase.
  - La primera necesita a la segunda para su cometido.
  - Las instancias de la clase se crean cuando se necesitan.
  - Un cambio en la clase utilizada puede afectar al funcionamiento de la clase utilizadora, pero no al revés.
  - Su representación:

8. Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

Herramientas para diagramas

- Existen cientos de herramientas CASE que soportan UML.
- Para elegir una, hay que tener claro para qué la vamos a utilizar y cuál es el objetivo que se propone:
  - Para que genere código java
  - Para dibujar modelos
- https://www.draw.io/
- Eclipse Plugin: UML2

Herramientas para diagramas

- Eclipse Plugin: UML2
  - Help → Eclipse Marketplace
  - Buscamos UML2 Designer
  - Instalar la versión correspondiente con nuestra versión de eclipse.
  - Tras instalar, reiniciar Eclipse.

Herramientas para diagramas

- Eclipse Plugin: UML2
- Se han creado nuevas vistas, entre ellas la vista Modeling.
- Ahora vamos a Archivo → Nuevo → UML Project
- Una vez creado el proyecto, botón derecho y elegimos Create Representation.
- Seleccionaremos el Class Diagram y se creará el diagrama.
- Utilizaremos la Paleta de elementos y la pestaña de Propiedades para representar nuestros diagramas de clases.

Herramientas para diagramas

- Ejercicios de Diagramas de Clases

Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

Generar código desde Diagramas

- La generación de código desde Diagramas de Clases es una funcionalidad que nos ofrecen múltiples herramientas.
- Para hacerlo integrado en Eclipse y puesto que hemos visto los plugins necesarios para el desarrollo de los diagramas de clases dentro de este IDE, veremos cómo hacerlo también desde dentro del mismo.

Generar código desde Diagramas

- Para generar código Java en Eclipse es necesario instalar el plugin UML to Java Generator.
- Una vez instalado, desde un modelo ya realizado, pulsamos el botón derecho del ratón y elegimos la opción Run As/Acceleo UML to Java Generation.
- Se muestra la ventana de configuración de la generación.
  - Pondremos nombre del proyecto.
  - Carpeta donde se guardarán los .java y las clases
  - Podemos crear automáticamente getters y setters
- Pulsamos el botón Run y aparecerá el proyecto en el explorador.
- Hacer import java.util.HashSet para corregir errores.

8. Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa

Ingeniería Inversa

- Por un lado es el proceso de analizar un sistema para crear una representación del mismo, pero a un nivel elevado de abstracción.
- Por otro lado es un proceso que recorre hacia atrás el ciclo de desarrollo de software.
- De lo cual se pueden analizar dos tipos:
  - Basada en el código fuente
    - Se conoce el código pero se desconocen aspectos de más alto nivel. La documentación es pobre, inexistente o desactualizada.
  - Basada en el programa ejecutable
    - No hay código fuente disponible, así que los esfuerzos se centran en descubrir ese código fuente.

Ingeniería Inversa

- Algunas herramientas que además de permitirnos hacer diagramas UML también nos permiten realizar ingeniería inversa y realizar un Diagrama de Clases a partir de un proyecto Java son:
  - ArgoUML
  - WhiteStarUML

8. Diagramas de Clases

- Introducción
- Conceptos orientados a objetos
- Qué es UML
- Diagramas de Clases
- Herramientas para el diseño de diagramas
- Generación de código a partir de diagramas de clases
- Ingeniería inversa