Programación Orientada a Objetos

Tema 4:
Análisis y Diseño Orientado a Objetos

Tema 4-1: Ingeniería del Software

Contenidos

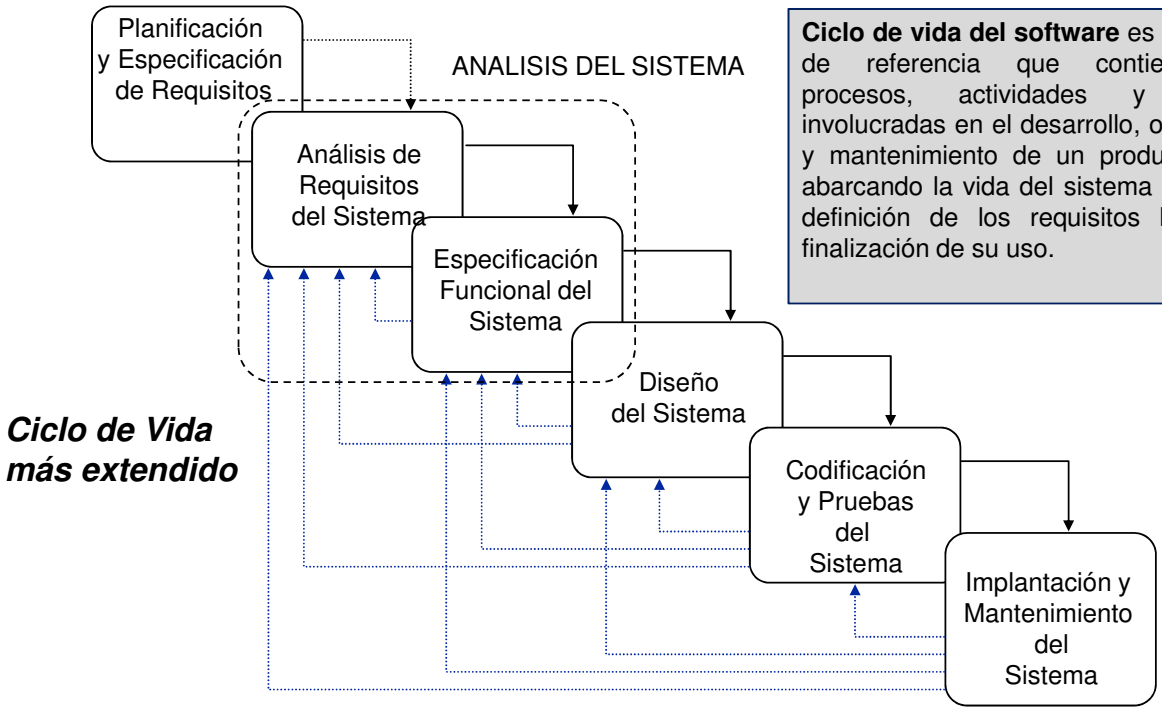
- Tema 4-1: Ingeniería del Software

1. La ingeniería del software
2. Ciclo de vida del software
3. El proceso de desarrollo OO
4. UML
5. NOTACIÓN UML: Diagrama de Casos de Uso
6. NOTACIÓN UML: Diagrama de Clases
7. Ejemplo: Gestión Bancaria
8. Ejemplo: Gestión Académica
9. Ejercicios

INGENIERÍA DEL SOFTWARE

- La **Ingeniería del Software** aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas a los problemas de desarrollo de software.
- **Proceso de ingeniería de software:** Conjunto de etapas parcialmente ordenadas con la intención de lograr un producto software de calidad.
- **Análisis/Diseño Orientado a Objetos:** Es un método de análisis y diseño que examina los requerimientos desde la perspectiva de las clases y objetos encontrados en el vocabulario del dominio del problema.
- **Metodología de Desarrollo:** Es un conjunto integrado de técnicas y métodos (actividades) que permiten obtener de forma homogénea (sistemática) y abierta (a cambios y adaptaciones), cada una de las fases del ciclo de vida del software.

CICLO DE VIDA DEL SW



Ciclo de Vida más extendido

Ciclo de vida del software es el marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto SW, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.

EL PROCESO DE DESARROLLO OO

- **Fases** en que se descompone el proceso de desarrollo OO:

1. **Planificación y Especificación de Requisitos**: Planificación, definición de requisitos, conocer los procesos del dominio, etc.

2. **Construcción**: La construcción del sistema. Se subdivide en las siguientes:
   - **Análisis**: Se analiza el problema a resolver desde la perspectiva de los usuarios y de las entidades externas que van a solicitar servicios al sistema.
   - **Diseño**: El sistema se especifica en detalle, describiendo cómo va a funcionar internamente para satisfacer lo especificado en el análisis.
   - **Implementación**: Se lleva lo especificado en el diseño a un lenguaje de programación.
   - **Pruebas**: Se llevan a cabo una serie de pruebas para corroborar que el software funciona correctamente y que satisface lo especificado en la etapa de Planificación y Especificación de Requisitos.

3. **Instalación**: La puesta en marcha del sistema en el entorno previsto de uso.

EL PROCESO DE DESARROLLO OO

- El desarrollo de un producto software supone un gran esfuerzo que puede durar bastante tiempo por ello es conveniente **dividir el trabajo** en fases e iteraciones.

- **La fase de Construcción** es la que va a consumir la mayor parte del esfuerzo y del tiempo en un proyecto de desarrollo.

- Se adopta un **enfoque iterativo**:
  - Se toma, en cada iteración, un subconjunto de los requisitos (agrupados en casos de uso) y se llevan a su análisis y diseño hasta la implementación y pruebas.
  - El sistema va creciendo incrementalmente en cada ciclo.

- Conseguimos **disminuir el grado de complejidad** que se trata en cada ciclo, y se obtiene una parte del sistema funcionando que se puede contrastar con el usuario / cliente.

EL PROCESO DE DESARROLLO OO

- **Planificación y Especificación de Requisitos:**
- Estudiar la especificación de requisitos para descubrir las secuencias típicas de acciones desde la perspectiva del usuario. Estas acciones son los denominados **casos de uso**.
- Un **caso de uso** es una secuencia típica de acciones en un sistema, desde el punto de vista del usuario, que muestra cómo el sistema interacciona con el exterior y que se obtiene como resultado del uso del sistema.
- Los casos de uso son descritos en un documento en el que se detallan los siguientes puntos de cada uno:
  - ✓ Nombre del caso de uso
  - ✓ Actores participantes
  - ✓ Tipo de caso (importancia del mismo – primario, secundario)
  - ✓ Descripción del caso de uso

Universidad de Alcalá
Departamento de Ciencias de la Computación

EL PROCESO DE DESARROLLO OO

- **Análisis:**
- Se intenta llegar a una **buena comprensión del problema** por parte del equipo de desarrollo, sin entrar en cómo va a ser la solución en cuanto a detalles de implementación.
- Trabajamos con los modelos de casos de uso construidos en la fase anterior, ampliándolos y refinándolos.
- Se construye un **Modelo de Objetos Conceptual** o Modelo de Análisis mediante un **diagrama de clases,** compuesto de **clases y relaciones** entre las clases.
- En el **Modelo de Objetos Conceptual** se tiene una representación de conceptos (objetos - clases) del mundo real, es una primera aproximación al modelo de diseño.
- Se deberán **identificar los conceptos más importantes del sistema** (objetos físicos, roles de una persona, etc.), **los atributos de los mismos y las relaciones** existentes entre ellos. Por ejemplo en un sistema bancario se pueden identificar conceptos como cuenta, cliente, tarjeta de crédito, saldo, recibo, etc.

EL PROCESO DE DESARROLLO OO

DIFERENCIAS

MODELO DE CASOS DE USO	MODELO DE ANÁLISIS
• Descrito en el lenguaje del cliente	• Descrito en el lenguaje del desarrollador
• Vista externa del sistema	• Vista interna del sistema
• Utilizado fundamentalmente como contrato entre el cliente y los desarrolladores sobre qué debería y que no debería hacer el sistema	• Utilizado fundamentalmente por los desarrolladores para comprender cómo debería darse forma al sistema, es decir, como debería ser diseñado e implementado
• Captura la funcionalidad del sistema desde el punto de vista del usuario	• Esboza como llevar a cabo la funcionalidad dentro del sistema, incluida la funcionalidad significativa para la arquitectura

Universidad de Alcalá
Departamento de Ciencias de la Computación

EL PROCESO DE DESARROLLO OO

- **Diseño:**
- En la fase de Diseño se crea una **solución a nivel lógico** para satisfacer los requisitos, basándose en el conocimiento reunido en la fase de análisis.
- Las **tareas** que se realizan en esta fase son las siguientes:
  - ✓ Definir el Diagrama de Clases de Diseño detallado.
  - ✓ Definir las estructuras de datos necesarias para almacenar la información que utiliza el sistema.
  - ✓ Definir la Interfaz de Usuario e Informes.
- Los **diagramas de clases** definidos en la fase anterior se pueden **refinar** con la especificación de atributos y operaciones para cada una de las clases y las relaciones con otras clases (generalización, agregación, composición, uso, etc.). Con la información obtenida en los casos de uso, se pueden derivar las operaciones y asignarse a las clases existentes.

- El **Unified Modeling Language (UML)** define un lenguaje de modelado orientado a objetos común para visualizar, especificar, construir y documentar los componentes de un sistema software OO.
- El UML no es una metodología, sino una **notación** que trata de posibilitar el intercambio de modelos de software.
- Un **modelo** es una simplificación de la realidad creada para comprender mejor un sistema.
- Un **proceso de desarrollo de software** debe ofrecer un conjunto de modelos que permitan expresar el producto desde cada una de las perspectivas de interés.

- Los modelos de UML se utilizan para representar las distintas fases o etapas que se plantean en una **metodología de desarrollo software**. Ejemplos de metodologías: Métrica 3 y el Proceso Unificado.
- UML utiliza **modelos orientados a objetos**: Representación de un sistema a partir de los objetos o entidades que lo constituyen, con atributos y operaciones, y relaciones con otros objetos.
- UML es un lenguaje de modelado visual, utiliza **diagramas**, para la representación de los sistemas. Los diagramas se utilizan para visualizar un sistema desde diferentes perspectivas, de forma que un diagrama es una proyección de un sistema.

- ***Diagramas para modelar el Comportamiento del Sistema:***
- **Diagrama de Casos de Uso**: Muestra un conjunto de casos de uso y actores y sus relaciones.
- **Diagrama de Secuencia**: Diagrama de interacción con la relación temporal de los mensajes y los objetos.
- **Diagrama de Colaboración**: Diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes.
- **Diagrama de Estados**: Muestra una máquina de estados, que consta de estados, transiciones, eventos y actividades. Vista dinámica del sistema.
- **Diagrama de Actividades**: Muestra el flujo de actividades dentro de un sistema.
- ***Diagramas para modelar la Estructura del Sistema:***
- **Diagrama de Clases**: Muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones.
- **Diagrama de Objetos**: Muestra un conjunto de objetos y sus relaciones.
- **Diagrama de Componentes**: Muestra la organización y las dependencias entre un conjunto de componentes.
- **Diagrama de Despliegue**: Representa la infraestructura de un sistema en tiempo de ejecución.

NOTACIÓN UML: Diagrama de Casos de Uso

- Un Diagrama de Caso de Uso muestra la relación entre **Actores** y los **Casos de Uso** del sistema.

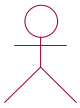
- Estos conceptos permiten definir:
  1. que elementos externos al sistema interactúan con él (Actor)
  2. que funciones deben ser realizadas por el sistema (Caso de Uso)

- Los casos de uso describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el **punto de vista de un usuario**; permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.

NOTACIÓN UML: Diagrama de Casos de Uso

- Un **Caso de Uso** es un concepto que representa una **unidad funcional** coherente, proporcionada por el sistema y que se manifiesta con un intercambio de mensajes entre el sistema y los interlocutores exteriores (llamados actores). Se representan gráficamente mediante una elipse que contiene el nombre del caso de uso.

- Un **actor** representa un **rol** (o conjunto de roles) que un usuario puede representar al interactuar con el sistema. Su representación gráfica es la figura de un hombre dibujado con unas líneas simples.

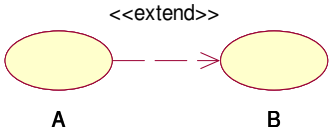


Actor

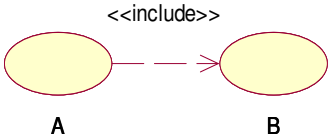
NOTACIÓN UML: Diagrama de Casos de Uso

- El Diagrama de Casos de Uso representa las relaciones entre los actores y los casos de uso, además de poder expresar las relaciones entre casos de uso si es que las hubiera. Las relaciones entre casos de uso pueden ser de dos tipos:

  - La relación **extiende:** que significa que un caso de uso A aumenta el comportamiento de un caso B.



  - La relación de **inclusión:** que significa que el caso de uso A incorpora el comportamiento del caso de uso B como parte de su propio comportamiento.

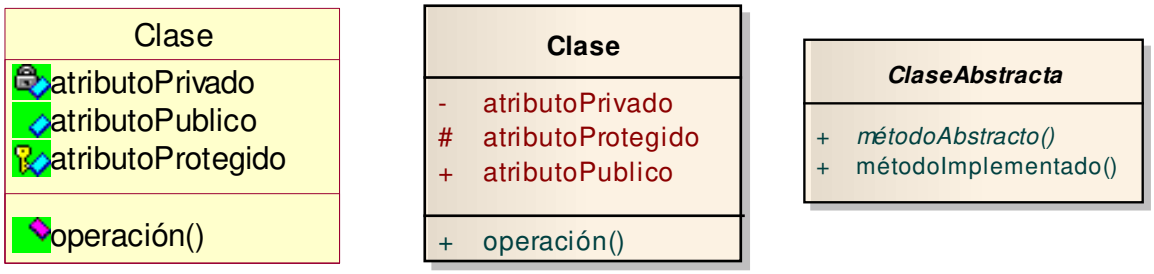


NOTACIÓN UML: Diagrama de Clases

- Un **Diagrama de Estructura Estática** (conocido más popularmente como **Diagrama de Clases**) muestra la estructura estática del modelo del sistema, es decir, todo aquello que "exista" en el sistema, mostrando su estructura interna así como sus relaciones entre los diferentes elementos.

- En un diagrama de clases, los elementos que nos vamos a encontrar son: las clases y las relaciones entre clases.

  - **Clase:** Nombre + Atributos + Operaciones (métodos)



NOTACIÓN UML: Diagrama de Clases

  - **Interfaces:** Representan un conjunto de operaciones que especifican los servicios que puede brindar una clase o componente y nunca debe especificar sus implementaciones.

  - Las **asociaciones** pueden estar formadas por un número indeterminado de clases pero las más comunes y utilizadas son las asociaciones binarias, es decir, aquellas relaciones entre dos clases. En los extremos de la relación especificaremos la cardinalidad y también podrán aparecer los atributos que representan la asociación.

NOTACIÓN UML: Diagrama de Clases

- **Agregación** (relación del tipo todo/parte) entre clases se expresa mediante un rombo vacío adyacente a la clase que representa la totalidad y de dicho rombo parten las asociaciones al resto de clases que forman dicha agregación.

- Es una asociación que indica que la clase adyacente al rombo vacío consta en parte o totalmente de instancias de la clase asociada, pero estas clases no comparten un ciclo de vida de forma que pueden existir sin necesidad del otro.

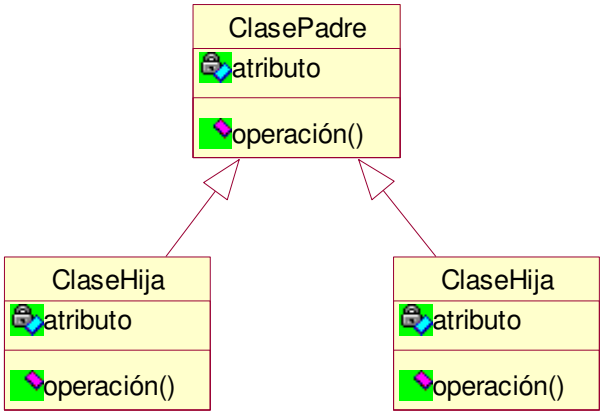
NOTACIÓN UML: Diagrama de Clases

- **Composición** (relación de pertenencia) es representada de igual forma que la agregación pero con el interior del rombo pintado de negro.

- Asociación que indica que la clase adyacente al rombo relleno está compuesta de 1 a varias instancias de la clase asociada, necesita que existan sus clases asociadas para que tenga sentido, al igual que la asociada necesita que la clase del rombo exista, comparten ciclo de vida de manera que en el momento que una deje de existir la asociada también dejará de existir.

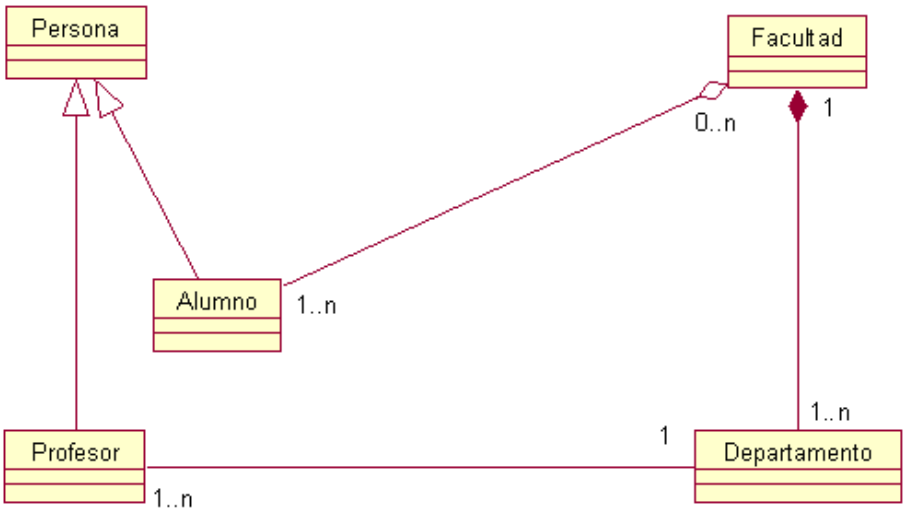
NOTACIÓN UML: Diagrama de Clases

- **Herencia (o generalización)** se representa mediante un triángulo unido a la clase padre por un vértice y del cual salen las relaciones a las clases hijas.



NOTACIÓN UML: Diagrama de Clases

- **Ejemplo**: En este ejemplo se pueden apreciar las relaciones que hay entre una facultad, sus departamentos, sus profesores y sus alumnos:



NOTACIÓN UML: Diagrama de Clases

1. **Realizar un análisis sintáctico-gramatical de la documentación existente:**
   - ✓ Utilizar la documentación de los casos de uso.
   - ✓ Subrayar cada nombre (sustantivo) o cláusula nominal.
2. **Decidir qué objetos se admiten como objetos del sistema.**
   - ✓ A partir de los nombres subrayados, proponer varios objetos potenciales.
3. **Identificación de relaciones:**
   - ✓ Las relaciones se obtienen analizando la estructura de la información del sistema.
   - ✓ Expresiones como: “es”, “tiene”, “consta de”, en la descripción del sistema, sugieren la existencia de relaciones entre objetos.
4. **Identificación de atributos:**
   - ✓ Los atributos se obtienen de la lista de objetos candidatos y de la descripción del sistema.
   - ✓ Los objetos descartados por simples serán atributos.
5. **Identificación de operaciones:**
   - ✓ Las operaciones de los objetos se derivan de los verbos que aparecen en la descripción del sistema.
   - ✓ Los parámetros de las operaciones se derivan de la información intercambiada por los objetos que interactúan.

Ejemplo: Gestión Bancaria

- Se desea desarrollar una aplicación de gestión bancaria. Especificaciones:

  - El sistema debe ser capaz de **gestionar una serie de productos asociados a los clientes del banco**. Los productos que gestiona el banco son: cuentas bancarias, fondos de inversión y carteras de valores.

  - Las cuentas deben tener: número de cuenta, fecha de apertura, saldo y tipo de interés y los datos de sus clientes.

  - El banco tiene dos tipos de cuentas: corrientes y a plazo.

  - Las cuentas corrientes pueden tener tarjetas de crédito asociadas. Solo éstas cuentas pueden tener el resto de productos asociados.

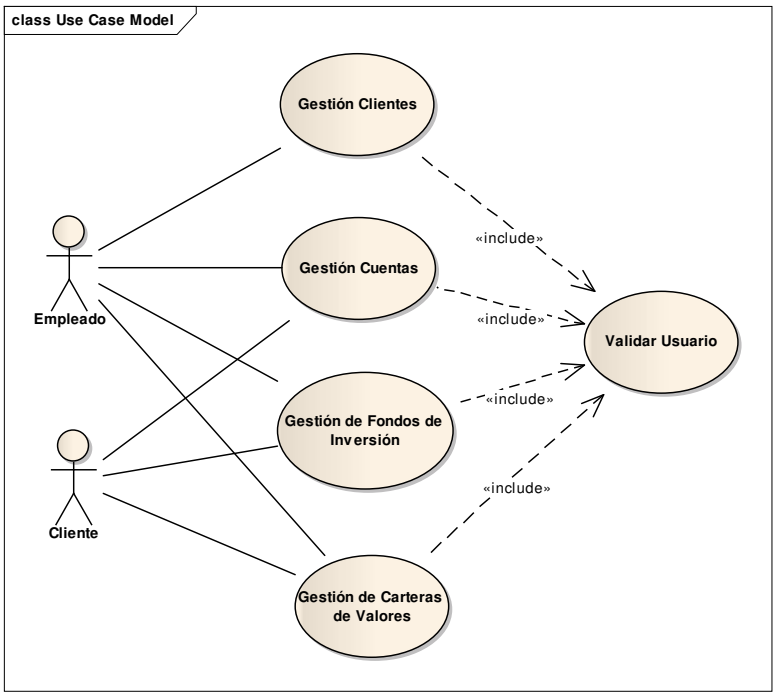
  - Las cuentas a plazo deben tener el número de meses que estará abierta.

Ejemplo: Gestión Bancaria

- De los clientes y los empleados se debe almacenar la siguiente información: DNI, nombre, dirección y teléfono. De los clientes el tipo y de los empleados su identificador único en el banco.
- De los empleados necesitamos saber en qué sucursal trabajan. Cada sucursal tendrá un identificador y una dirección.
- Los fondos de inversión deben tener un nombre, importe, rentabilidad y la fecha de apertura y vencimiento.
- Las carteras de valores están compuestos por los valores asociados, almacenando el nombre del valor, el número de títulos y el precio de cotización.
- Las tarjetas de crédito deben almacenar el tipo (Visa, MasterCard, etc.), el número, el titular y la fecha de caducidad.

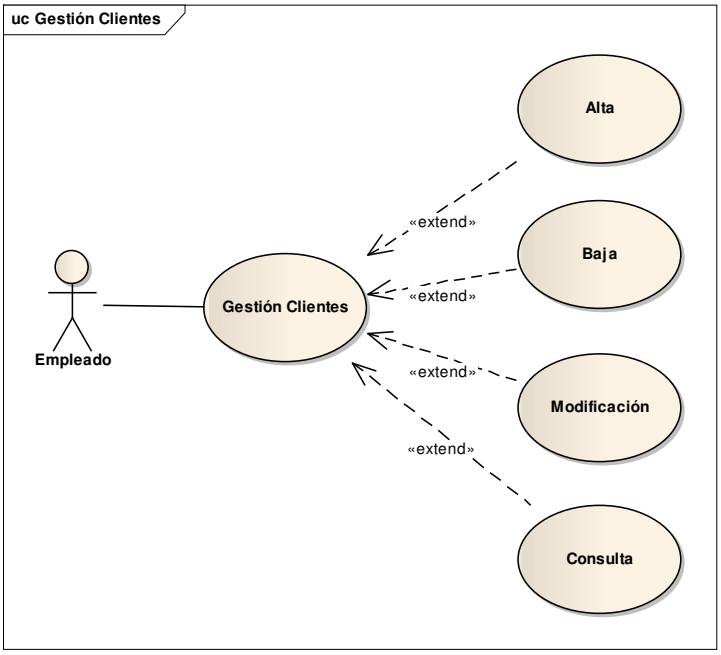
Ejemplo: Gestión Bancaria

- Casos de uso principales del banco:



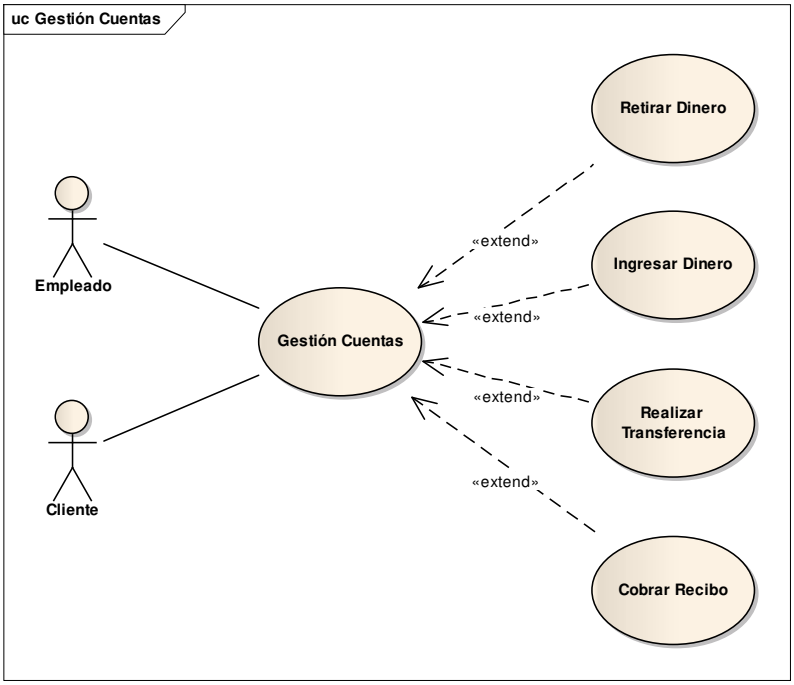
Ejemplo: Gestión Bancaria

- Subcaso de uso del empleado: Gestión de Clientes.



Ejemplo: Gestión Bancaria

- Subcaso de uso: Gestión de Cuentas.



Ejemplo: Gestión Bancaria

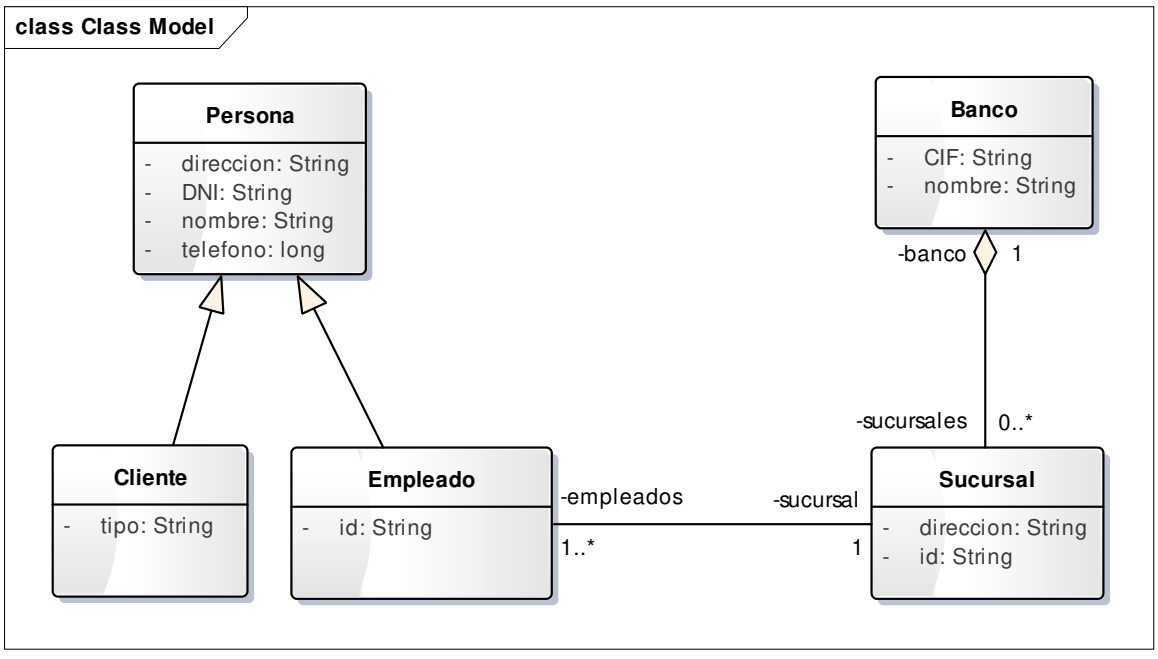
- Ejemplos de descripción de algunos de los casos de uso:

Caso de Uso: Alta Cliente
Actores: Empleado
Tipo: Secundario
Descripción: 1. El cliente proporciona todos los datos al empleado. 2. El empleado comprueba los datos y da de alta al cliente en el sistema.

Caso de Uso: Realizar transferencia
Actores: Empleado y Cliente
Tipo: Secundario
Descripción: 1. El cliente proporciona los datos de su cuenta y la cuenta destino. 2. El empleado comprueba los datos y realiza la transferencia.

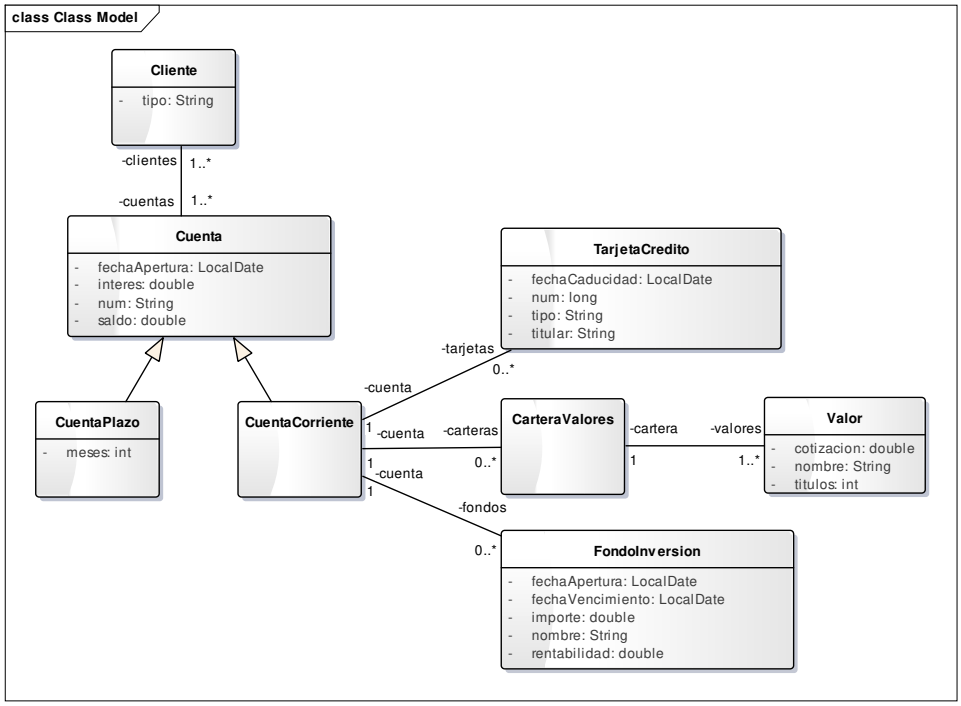
Ejemplo: Gestión Bancaria

- Diagrama de clases 1.



Ejemplo: Gestión Bancaria

- Diagrama de clases 2.



Ejemplo: Gestión Académica

- Se desea desarrollar una aplicación para la gestión académica de una universidad. Especificaciones:

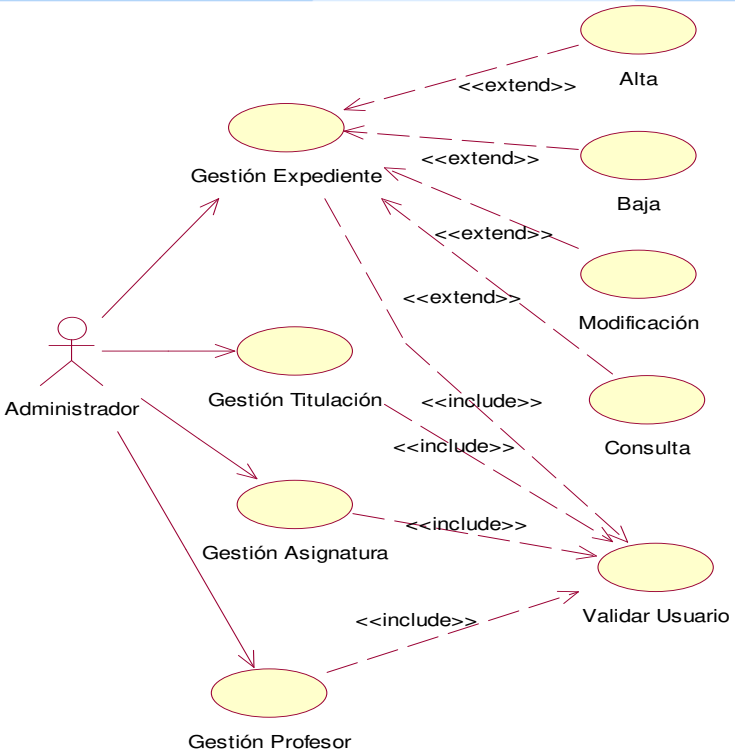
  - El sistema debe ser capaz de **gestionar todos los expedientes académicos** de los alumnos dando la posibilidad de realizar las operaciones típicas de altas, bajas, modificaciones y consultas de los datos del mismo. Debemos dar un número de expediente único en el sistema y una fecha de apertura.

  - La información mínima que se debe guardar de un **alumno** son el DNI, nombre, dirección, la titulación en la que está matriculado, así como las asignaturas que está cursando actualmente. También debemos almacenar su historial académico, donde deben aparecer todas las asignaturas cursadas y sus respectivas notas y convocatoria.

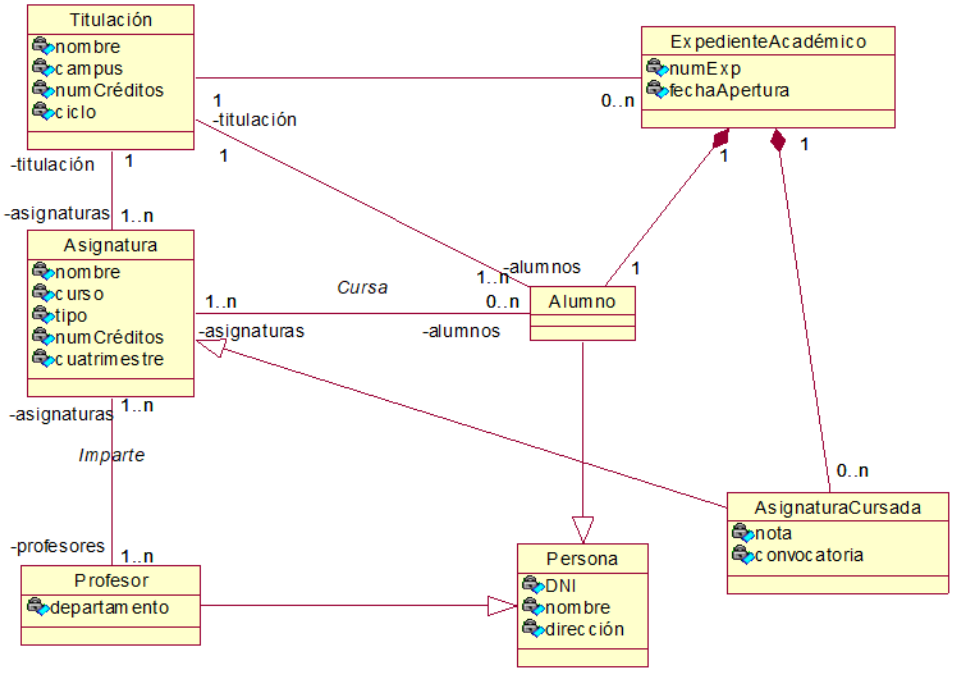
Ejemplo: Gestión Académica

- Se debe realizar la **gestión de las distintas titulaciones** que existen en la universidad teniendo en cuenta que una titulación sólo se da en un campus determinado y los datos que podemos consultar son el nombre, el número de créditos, si es de primer o segundo ciclo, etc.
- Se tienen que **gestionar las asignaturas** que se imparten en una titulación, teniendo en cuenta que una asignatura solo se puede dar en un único curso. Algunos de los datos que se pueden consultar de una asignatura son: el nombre, número de créditos, cuatrimestre en el que se imparte y su tipo (obligatoria, troncal, optativa).
- Se debe guardar la información de **los profesores** que imparten las distintas asignaturas de la titulación. Se debe almacenar como mínimo su DNI, nombre, dirección y departamento al que pertenece. También se podrá consultar las distintas asignaturas que imparte.

Ejemplo: Gestión Académica



Ejemplo: Gestión Académica



BIBLIOGRAFÍA

Análisis y diseño estructurado y orientado a objetos de sistemas informáticos.
Amescua, Antonio.
McGraw-Hill, 2003.

Programación orientada a objetos. Segunda Edición.
Joyanes Aguilar, Luis.
McGraw-Hill, 1998.

Utilización de UML en Ingeniería del Software con Objetos y componentes.
Stevens, Perdita.
Addison Wesley, 2002.

El Proceso Unificado de Desarrollo de Software.
Jacobson, Ivar; Booch, Grady, Rumbaugh, James.
Addison Wesley, 2000.

El Lenguaje Unificado de Modelado.
Booch, Grady, Rumbaugh, James; Jacobson, Ivar.
Addison Wesley, 2001.

Ingeniería de software orientado a objetos.
Bruegge, Brend; Dutoit, Allen H.
Prentice Hall, 2002.

Ingeniería de software. 9ª Edición.
Sommerville, Ian.
Pearson Education, 2012.

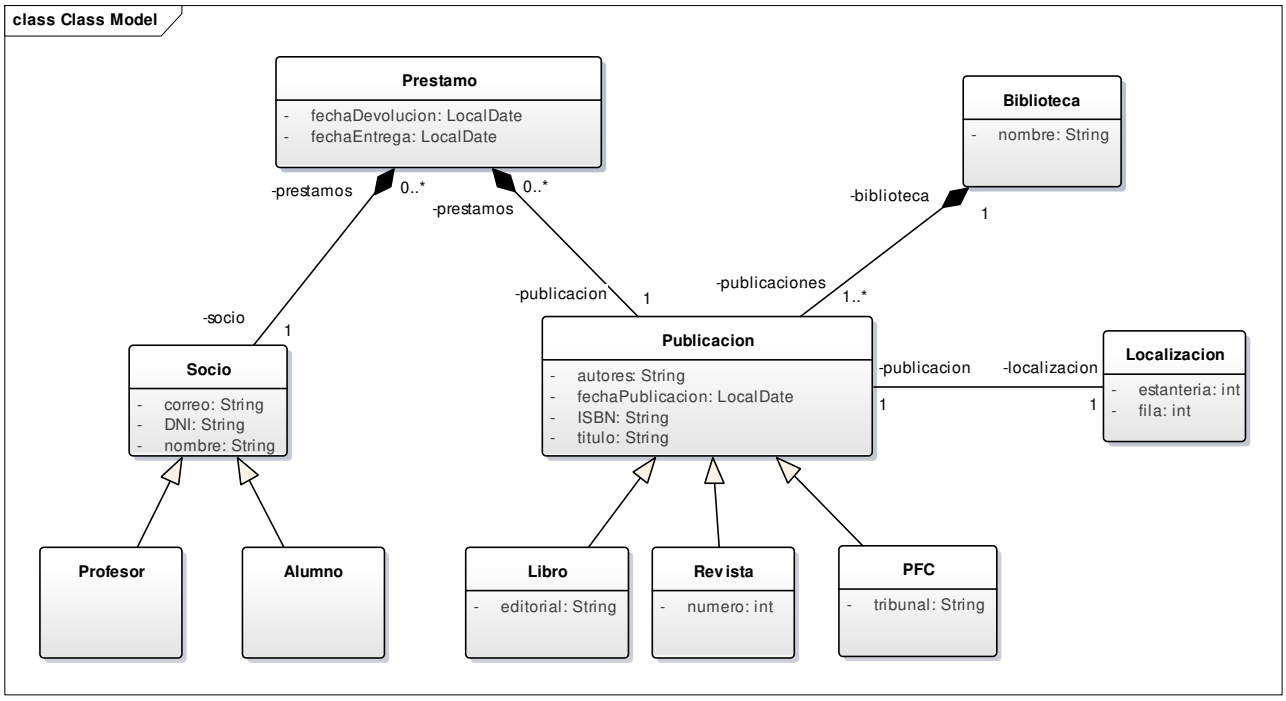
- www.uml.org
- es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

Ejercicios UML

Ejercicio: Biblioteca

- Se desea desarrollar una aplicación para el manejo de un sistema bibliotecario de una universidad. Especificaciones:
  - La biblioteca contiene tres tipos de publicaciones: libros, revistas y proyectos fin de carrera. Todas las publicaciones posen la siguiente información: isbn, título, autores y fecha de publicación. Los libros incorporan la editorial, las revistas incorporan el número y los proyectos incorporan el tribunal que lo evaluó. Cada publicación en la biblioteca necesita una localización que se compondrá del número de estantería y fila que ocupa.
  - Para poder realizar un préstamo en la biblioteca se necesita ser socio de la misma, todos sus socios tienen DNI, nombre y correo electrónico, tendremos dos tipos distintos de socios: profesores y alumnos. Cada uno de los préstamos que realiza la biblioteca deben quedar reflejados en el sistema debiendo contener los siguientes datos: publicación, socio, fecha de entrega y fecha de devolución.

Ejercicio: Biblioteca



Ejercicio: Aeropuerto

- Se desea realizar una aplicación para la gestión de la información de un aeropuerto. Especificaciones:

  - La clase central de la aplicación será Aeropuerto, de la cual nos interesa registrar su nombre y su CIF, y mantener listas de otros objetos asociados, que se presentarán a continuación.

  - El aeropuerto consta de un cierto número de terminales de las cuales hay que registrar su código identificativo, la lista de compañías aéreas que operan en este terminal, y la lista de infraestructuras que disponen.

  - En una terminal habrá dos tipos diferentes de infraestructuras: puertas (de embarque o de llegada) y mostradores de facturación. De cualquier infraestructura guardaremos un coste base que sirve como referencia al aeropuerto para facturar a la compañía aérea el uso de las infraestructuras aeroportuarias. Además, de las puertas guardaremos un código identificativo y el coste suplementario de la pasarela de acceso a aeronaves, que será cero en caso de que la puerta no disponga de este elemento. De los mostradores de facturación guardaremos su número.

Ejercicio: Aeropuerto

- En el aeropuerto habrá un conjunto de compañías aéreas operando, de las cuales interesa mantener su información. De las compañías aéreas guardaremos su nombre comercial, el CIF, la terminal en la que opera (supondremos que una compañía aérea sólo opera en una terminal), el rango de mostradores de facturación que tienen asignados y también se mantendrá una lista de los vuelos que la compañía opera en este aeropuerto.

- En el aeropuerto también interesa llevar registro de todos los vuelos de salida y de llegada que se van efectuando a lo largo del tiempo. De un vuelo registraremos la compañía aérea que lo opera, la fecha/hora prevista de salida o llegada, el código identificativo del vuelo y la puerta que le sea asignada. Además, cada vuelo mantendrá una lista de su pasaje. Concretamente, cada uno de los elementos de estas listas será una reserva de vuelo individual efectuada por un pasajero. Estas reservas se irán añadiendo a la lista a medida que se vayan creando. Finalmente, también habrá que registrar el estado del vuelo, que podrá ser: "programado", "confirmado", "efectuado" (una vez el avión haya despegado o aterrizado), o bien "cancelado".

Universidad de Alcalá
Departamento Ciencias de la Computación

Ejercicio: Aeropuerto

- De los pasajeros se guardará el nombre y el DNI, y se mantendrá una lista de las reservas de vuelo que han efectuado. Una reserva de vuelo contendrá la información del vuelo en el que está asociada, así como la información del pasajero. Adicionalmente, la reserva mantendrá el valor de su estado, que puede ser: "no confirmada" o "confirmada". Una reserva se confirma cuando el pasajero efectúa la operación de check-in.

Ejercicio: Aeropuerto