Resumen de Programación Orientada a Objetos (POO)

**UNIDAD 1: DISEÑO ORIENTADO A OBJETOS**

**INTRODUCCION A O.O**

Programación imperativa: los algoritmos se describen en base a procesos. Así, los algoritmos se expresan mediante procesos o funciones y éstos como una secuencia de tareas a realizar por la computadora.

Programación modular: permite dividir el programa en módulos autónomos que se pudieran programar, verificar y modificar individualmente.

Estas implementaciones tuvieron problemas, como el uso de la abstracción. Pero se logro solucionar gracias al ocultamiento de la implementación.

Tipos de datos abstractos:

1. Tipo de dato definido por el programador.
2. Conjunto de operaciones abstractas sobre objetos de este tipo.
3. Encapsulamiento de los objetos de ese tipo, para que el usuario final no pueda manipular esos objetos excepto a través de uso de operaciones definidas.

**CARACTERISTICAS DEL DISEÑO O.O**

En la programación imperativa tradicional, los algoritmos se describen en base a procesos. La programación orientada a objetos encara la resolución de cada problema desde la óptica del objeto.

El OBJETO combina los datos (atributos del objeto) con los procedimientos u operaciones (métodos) que actúan sobre dichos datos. Los objetos interactúan entre si enviando mensajes.

Los métodos son muy similares a los procedimientos de la programación imperativa tradicional y los mensajes se podrían pensar como invocaciones a los procedimientos.

El programador orientado a objetos puede agrupar características comunes de un conjunto de objetos en clases, a través de un proceso de abstracción.

Los descendientes de estas clases se construyen por medio del mecanismo de subclasificación, permitiendo que sean heredados los métodos programados anteriormente y debiendo programar solamente las diferencias.

POO no se puede desligar de todo el paradigma de orientación a objetos. Paradigma: conjunto de practicas y saberes que definen una disciplina científica durante un periodo especifico.

El principio fundamental del paradigma de POO es construir un sistema de software en base a entidades de un modelo elaborado a partir de un proceso de abstracción y clasificación.

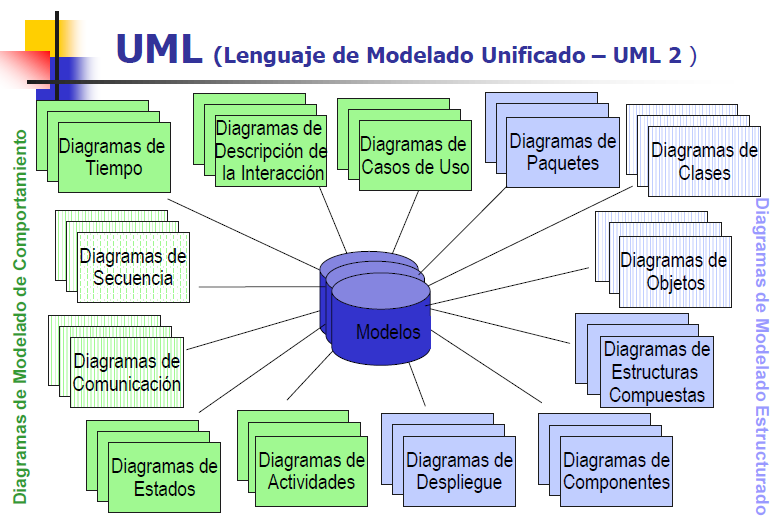
Hay que desarrollar todo el sistema utilizando el paradigma, empezando por el análisis y un diseño O.O.

El desarrollo de software implica una serie de etapas para hacer: requerimientos, análisis, diseño, implementación y prueba.

**UML (Lenguaje de Modelado Unificado)**

UML es un lenguaje que permite la visualización, especificación y documentación de sistemas O.O, permite elaborar diferentes tipos de diagramas, en particular diagramas de clases. No es una metodología sino una notación, que aglutina distintos enfoques de O.O.

El UML2 define 13 diagramas para describir distintas perspectivas del sistema.



**ELEMENTOS BASICOS DE LA O.O**

Los elementos básicos de la O.O son:

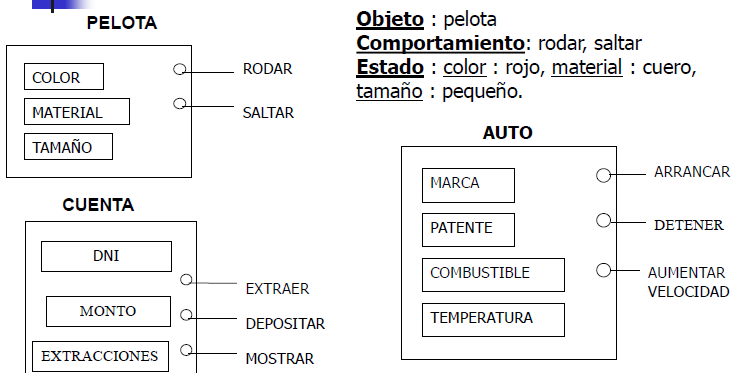
1. Objetos.
2. Clases.
3. Metodos y Mensajes.
4. Herencia.

**OBJETOS**

Un objeto del problema es una entidad, física o conceptual, caracterizada por atributos y comportamientos. El comportamiento queda determinado por un conjunto de servicios que el objeto puede brindar y un conjunto de responsabilidades que debe asumir. Un objeto de software es un modelo, una representación de un objeto del problema. Un objeto de software tiene una identidad y un estado interno y recibe mensajes a los que responde ejecutando un servicio.

Un objeto es una unidad atómica que encapsula estado y comportamiento. La encapsulación en un objeto permite una alta cohesión y un bajo acoplamiento. Los objetos son entidades que tienen atributos (datos) y comportamiento particular (procedimientos).

Atributos de un objeto: describen la abstracción de características individuales que posee un objeto.

Comportamientos: representan las operaciones que pueden ser realizadas por un objeto.

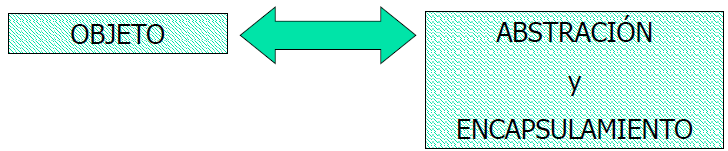
**Objeto = Estado + Comportamiento + Identidad**

El ESTADO agrupa los valores instantáneos de todos los atributos de un objeto. Evoluciona con el tiempo

El COMPORTAMIENTO describe las acciones y reacciones de ese objeto.

Las acciones u operaciones de un objeto se desencadenan como consecuencia de un estimulo externo, representado en forma de un mensaje enviado por otro objeto. **El estado y el comportamiento están relacionados**.

La IDENTIDAD permite distinguir los objetos de forma no ambigua, independientemente de su estado. Permite distinguir dos objetos en los que todos los valores de los atributos son idénticos.



Los objetos informáticos definen una representación abstracta de las entidades de un mundo real o virtual, con el objetivo de controlarlos o simularlos.

En UML, un objeto se representa en un rectángulo; el nombre del objeto se subraya.

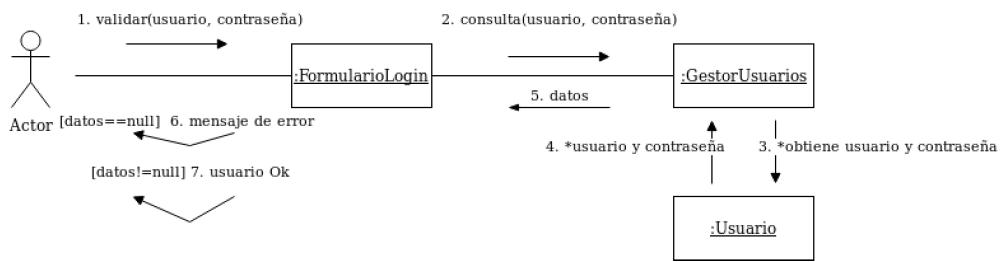
O bien, usando un nombre genérico mediante el uso de ‘:’.

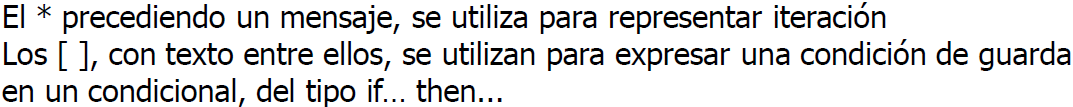
ACTORES

Para modelar un sistema es necesario identificar quien o quienes son sus usuarios. El actor es una entidad externa al propio sistema, pero que necesita intercambiar información con él. Los actores no están restringidos a ser personas físicas, también pueden representar a sistemas externos al sistema que se esté modelando.

DIAGRAMAS DE COMUNICACIÓN

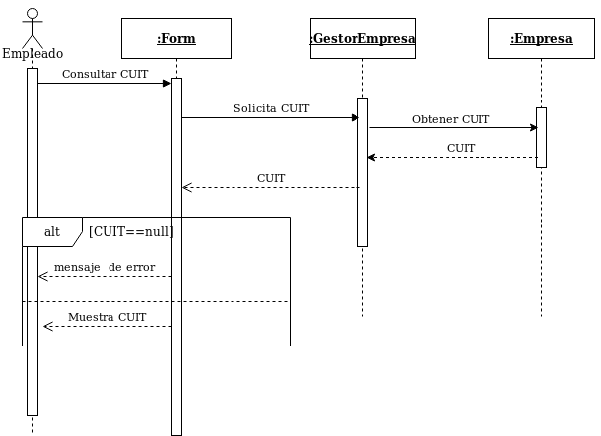
Muestran las interacciones entre objetos en la estructura espacial estática, permite la colaboración entre objetos. El tiempo no se representa de manera explícita, por lo que los mensajes se numeran para indicar el orden de los envíos.





DIAGRAMAS DE SECUENCIA

Muestran interacciones entre objetos según un punto de vista temporal. Un objeto se representa por un rectángulo y el tiempo de vida se representa por una barra vertical llamada línea de vida de los objetos. El orden de envio de los mensajes esta dado por la posición sobre el eje vertical.

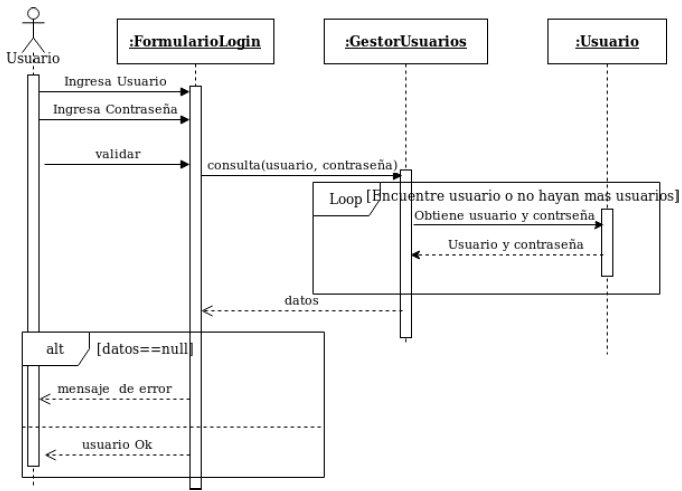


DIAGRAMAS DE SECUENCIA – Fragmentos combinados.

Permiten agregar un grado de lógica de procedimientos a los diagramas. Un fragmento combinado es una o mas secuencias de procesos incluidas en un marco y ejecutadas bajo circunstancias específicas.

Fragmento *ALTERNATIVA* (“alt”): modela la elección de una interacción de objetos, a traves de una condición de guarda, modela estructuras condicionales del tipo **if… then… else**.

Fragmento de ITERACION o BUCLE (“loop”): incluye un conjunto de mensajes que se ejecutan multiples veces, según lo que indique la condición de guarda.

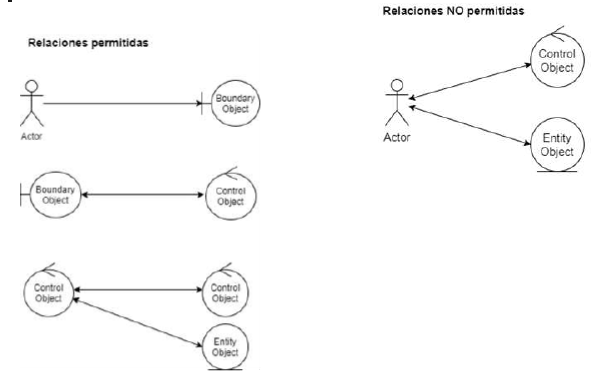
Ejemplo:

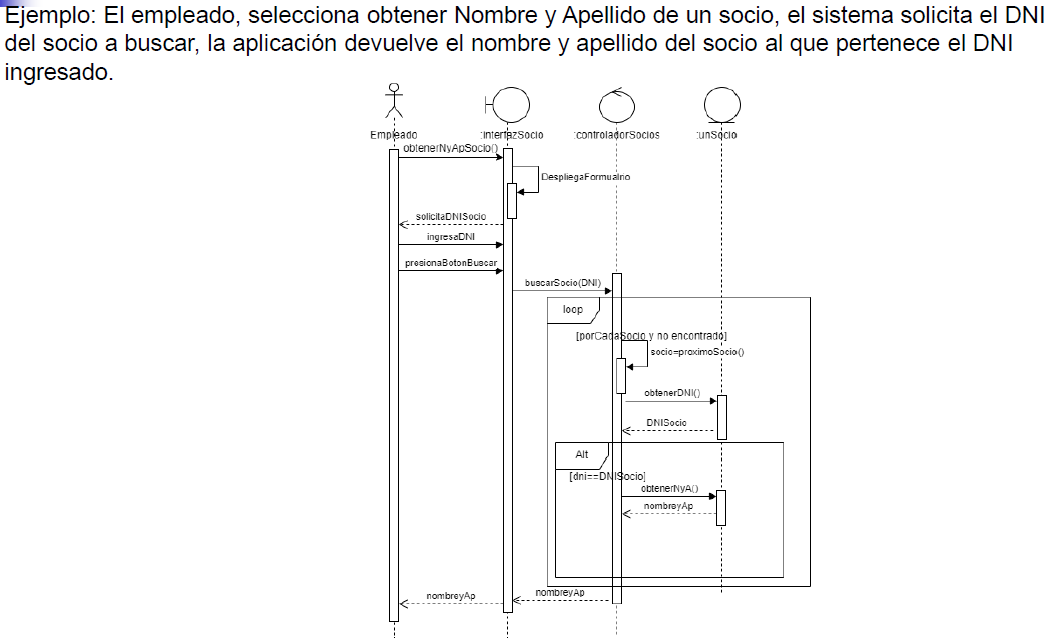
OBJETOS CON ESTEREOTIPO

En aplicaciones es común que usen otros objetos que no pertenecen al dominio del problema, algunos surgen para presentar una interfaz al usuario (actor) de la aplicación, otros para controlar la lógica interna de la aplicación.

1. Objeto de INTERFAZ (Boundary): representa un elemento con el cual interactúa el usuario (actor). Representan una abstracción de una ventana, un formulario, etc.
2. Objeto de CONTROL (Controllers): organizar y controlar la lógica requerida para alcanzar el objetivo de una determinada funcionalidad. Media entre los objetos de interfaz y los de entidad.
3. Objeto de ENTIDAD (Entity): para cada uno de los objetos (entidades) del dominio requeridos para realizar la funcionalidad. Modelan informacion asociada a algun fenomeno o concepto, como persona o un objeto

ESTEREOTIPOS- RELACIONES PERMITIDAS Y NO PERMITIDAS



Ejemplo de diagrama de secuencia con estereotipos – Busqueda de información

CLASES

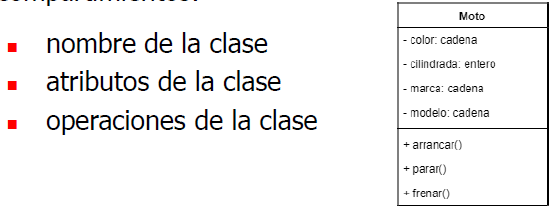
Una CLASE abstrae las características de un conjunto de objetos con comportamientos similares.

La ENCAPSULACION de una clase permite la cohesión y presenta distintas ventajas:

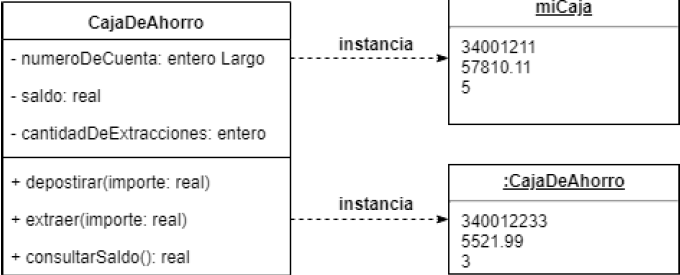
* Se protegen los datos de accesos indebidos.
* El acoplamiento entre las clases se disminuye.
* Favorece la modularidad y el mantenimiento.

Una CLASE es una descripción de un conjunto de objetos, ya que consta de comportamientos y atributos que resumen las características comunes del conjunto. (Desde el punto de vista del diseño de un sistema de software, una clase es un patrón que establece los atributos y el comportamiento de un conjunto de objetos. En la implementación de un sistema de software una clase es un módulo de software que puede construirse, verificarse y depurarse con cierta independencia a los demás.)

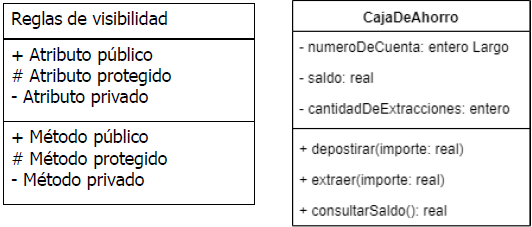
Definir clases significa colocar codigo reutilizable en un deposito común en lugar de redefinirlo cada vez que se necesite.

Cada objeto es una instancia de una clase.

Cada clase se representa en un rectángulo con 3 comportamientos:

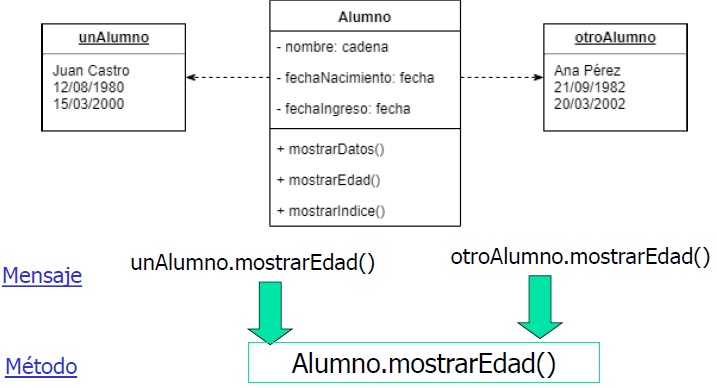


CLASES – VISIBILIDAD

Los atributos de una clase no deben ser manipulables directamente por el resto de los objetos, por lo que existen distintos NIVELES DE ENCAPSULACION o NIVELES DE VISIBILIDAD.

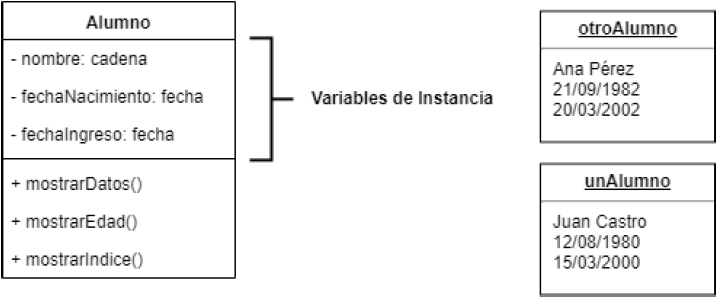
CLASES – METODOS Y MENSAJES

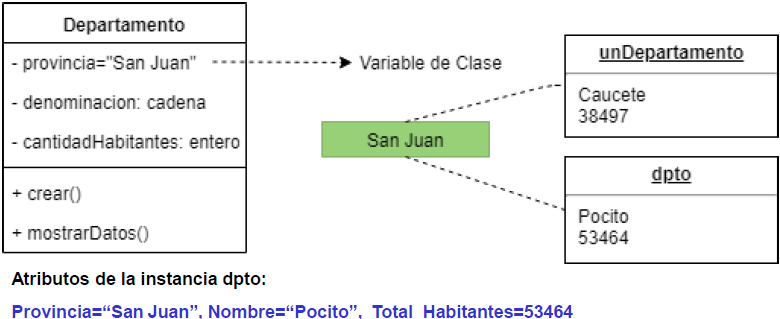
Los objetos tienen la posibilidad de actuar. La acción sucede cuando un objeto recibe un MENSAJE, una solicitud que pide al objeto que se comporte de manera determinada. Cada objeto recibe, interpreta y responde a mensajes enviados por otros objetos.

Los comportamientos u operaciones que caracterizan un conjunto de objetos residen en la clase y se llaman METODOS. Son el codigo que se ejecuta para responder a un mensaje, y el mensaje es la llamada o invocación a un método.

CLASES – VARIABLES DE CLASES Y DE INSTANCIA

VARIABLES DE INSTANCIA (miembros de dato): se usan para guardar los atributos de un objeto particular.



VARIABLES DE CLASES: atributos que tienen el mismo valor para cada objeto de la clase. Si el valor de la variable de clase es cambiado para una instancia, el mismo cambia para todas las instancias de la clase y subclase. Representa un área de memoria compartida por todos los objetos de la clase.