DISEÑO ORIENTADO A OBJETOS

INTRODUCCION A LA O.O

Programación Imperativa

Los algoritmos se expresan mediante procesos o funciones y éstos como una secuencia de tareas a realizar por la computadora.

Programación Modular

Divide el programa en módulos autónomos que se pudieran programar, verificar y modificar individualmente.

Problema: El uso de la Abstracción

Solución: Ocultamiento de la implementación

Tipo de Datos Abstracto

1. Un tipo de dato definido por el programador

2. Un conjunto de operaciones abstractas sobre objetos de ese tipo

3. Encapsulamiento de los objetos de ese tipo, de tal manera que el usuario final del tipo no pueda manipular esos objetos excepto a través del uso de las operaciones definidas.

CARACTERÍSTICAS DEL DISEÑO O.O

- En la programación imperativa, los algoritmos se describen base a procesos.
- La POO encara la resolución de cada problema desde la óptica del objeto.
- El objeto combina atributos del objeto con métodos que actúan sobre los datos.
- Los objetos interactúan entre si a través de mensajes.
- Los métodos en la practica = a procedimientos de la programación imperativa y los mensajes = a la invocación de esos procedimientos.
- Se agrupan las características comunes de objetos en CLASES, a través de un proceso de abstracción.
- Los descendientes de las clases se construyen a través de la subclasificación, donde se heredan los métodos programados y solo se programan las diferencias.
- La POO no se puede desligar de todo el paradigma de orientación a objetos.
    - Paradigma: Conjunto de prácticas y saberes que definen una disciplina científica durante un período específico.
- Principio fundamental del paradigma de POO: construir un sistema de software en base a las entidades de un modelo elaborado a partir de un proceso de abstracción y clasificación.
- El desarrollo de software implica, desde una visión orientada a objetos, una serie de etapas para hacer: requerimientos, análisis, diseño, implementación y prueba.

UML (Lenguaje de Modelo Unificado)

- Lenguaje que permite la visualización, especificación y documentación de sistemas orientados a objetos.
- Es una notación, que aglutina distintos enfoques de orientación a objetos.
- UML 2 define trece (13) diagramas para describir distintas perspectivas del sistema.

ELEMENTOS BASICOS DE LA O.O

1. Objetos

2. Clases

3. Métodos y Mensajes

4. Herencia

OBJETOS

- Unidad atómica que encapsula estado y comportamiento.

Identifica

Un objeto del problema es una entidad, física o conceptual, caracterizada a través de atributos y comportamiento.

El comportamiento determinado por un conjunto de servicios que el objeto puede brindar y un conjunto de responsabilidades que debe asumir.

Abstrae

Un objeto de software es un modelo, una representación de un objeto del problema.

- 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: Los atributos describen la abstracción de características individuales que posee un objeto.
- Comportamientos: Los comportamientos de un objeto representan las operaciones que pueden ser realizadas por un objeto.

Objeto = Estado + Comportamiento + Identidad

- Estado: agrupa los valores instantáneos de todos los atributos de un objeto. Evoluciona con el tiempo.
- Comportamiento: describe las acciones y reacciones de ese objeto.
- Acciones u Operaciones de un objeto: desencadenadas como consecuencia de un estímulo externo, representado en forma de un mensaje enviado por otro objeto.
- Identidad: permite distinguir los objetos de forma no ambigua, independientemente de su estado. Esto 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 bajo la forma de un rectángulo; el nombre del objeto se subraya. O bien usando un nombre genérico mediante el uso de los ':'

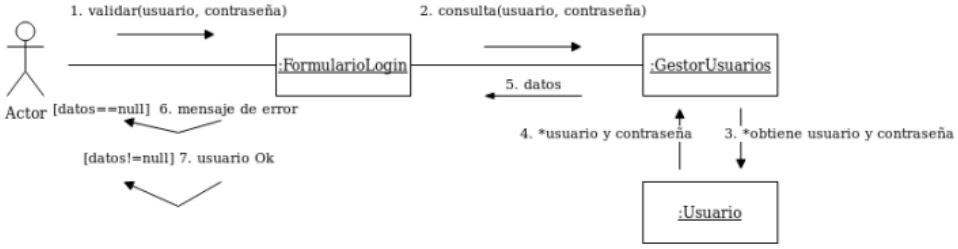
ACTORES

La interacción de objetos es iniciada por el usuario, por eso es necesario, que para modelar un sistema se identifique primero quien o quienes son los usuarios. A partir de la identificación de usuario se define el concepto de actor, que es una entidad externa al sistema, pero necesita intercambiar información con él.

Diagramas de Comunicación

Muestran las interacciones entre objetos en la estructura espacial estática, que permite la colaboración entre objetos. El tiempo no se encuentra representado por lo tanto se numeran los mensajes para indicar el orden de envió.

El * precediendo un mensaje, se utiliza para representar iteración Los [ ], con texto entre ellos, se utilizan para expresar una condición de guarda en un condicional, del tipo if… then...



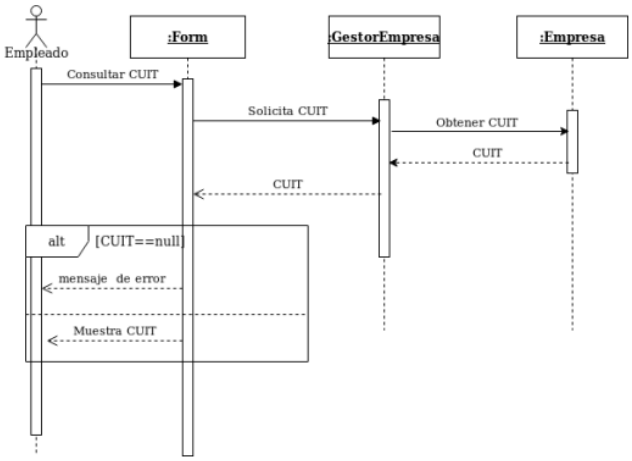
Diagramas de Secuencia

Estos diagramas muestran interacciones entre objetos según un punto de vista temporal.

Objeto: Representado por un rectángulo

Tiempo de vida: Representado por una barra vertical llamada línea de vida de los objetos.

Orden de envío de los mensajes: está dado por la posición sobre el eje vertical.



Diagramas de Secuencia – FRAGMENTOS COMBINADOS

- Fragmento combinado: es una o más secuencias de procesos incluidas en un marco y ejecutadas bajo circunstancias específicas.

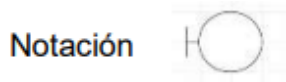
- Fragmento Alternativa (denotado "alt"): modela la elección de una interacción de objetos, a través de una condición de guarda, es decir, modela estructuras condicionales del tipo if…then…else.

- Fragmento de iteración o bucle (denotado "loop"): el fragmento incluye un conjunto de mensajes que se ejecutan múltiples veces, según lo indique la condición de guarda.

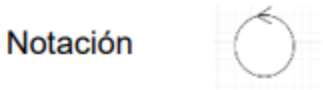
Objetos con Estereotipos (I)

Es común que en las aplicaciones 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 ésta.

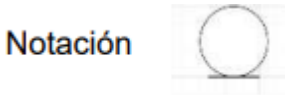
a) **Objeto de Interfaz (Boundary):** representa un elemento con el cual interactúa el usuario.



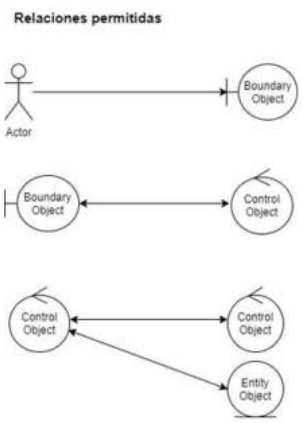
b) **Objeto de Control (Controllers):** se ocupa de organizar y controlar la lógica requerida para alcanzar el objetivo de una determinada funcionalidad. Media entre los objetos de interfaz y los de entidad.

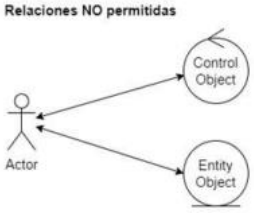


c) **Objeto de Entidad (Entity):** para cada uno de los objetos (entidades) del dominio requeridos para realizar la funcionalidad. Modelan información asociada a algún fenómeno o concepto, como persona o un objeto.



Estereotipos- Relaciones permitidas y NO permitidas





CLASES

Descripción de un conjunto de objetos, ya que consta de comportamientos y atributos que resumen las características comunes del conjunto.

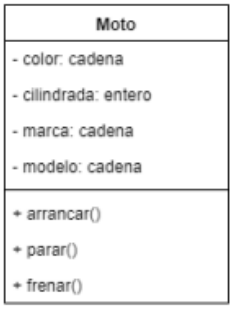
- Una clase abstrae las características de un conjunto de objetos con comportamientos similares.
- La encapsulación de una clase permite la cohesión y presenta distintas ventajas básicas:
  - Se protegen los datos de accesos indebidos
  - El acoplamiento entre las clases se disminuye
  - Favorece la modularidad y el mantenimiento
- Definir clases: colocar código reutilizable en un depósito común en lugar de redefinirlo cada vez que se necesite.
- Cada objeto es instancia de una clase.

Cada clase se representa en un rectángulo con tres compartimientos:

- nombre de la clase

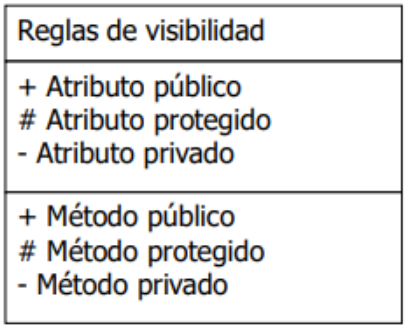
- atributos de la clase

- operaciones de la clase



Clases – VISIBILIDAD

Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos, no obstante, existen distintos niveles de encapsulación también llamados niveles de visibilidad.



Clases – MÉTODOS Y MENSAJES

- Los objetos tienen la posibilidad de actuar. La acción sucede cuando un objeto recibe un mensaje, que es 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 métodos.

Métodos: son el código que se ejecuta para responder a un mensaje.

Mensaje: es la llamada o invocación a un método.

Clases – VARIABLES DE CLASES Y DE INSTANCIA

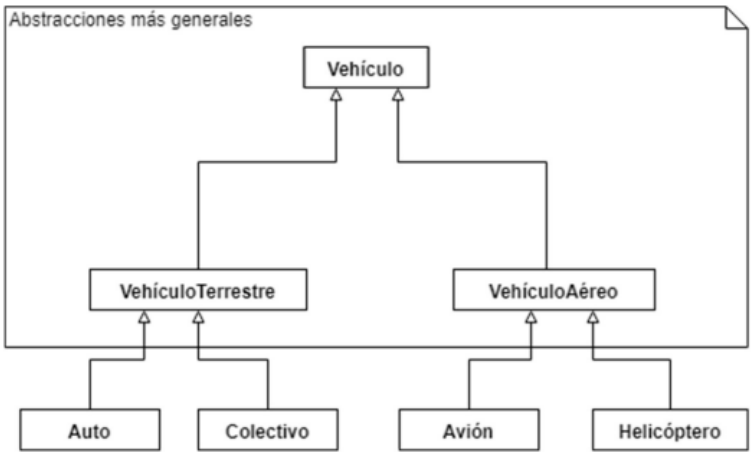
Variables de instancia: Se usan para guardar los atributos de un objeto particular.

Variables de clase: Son aquellos 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.

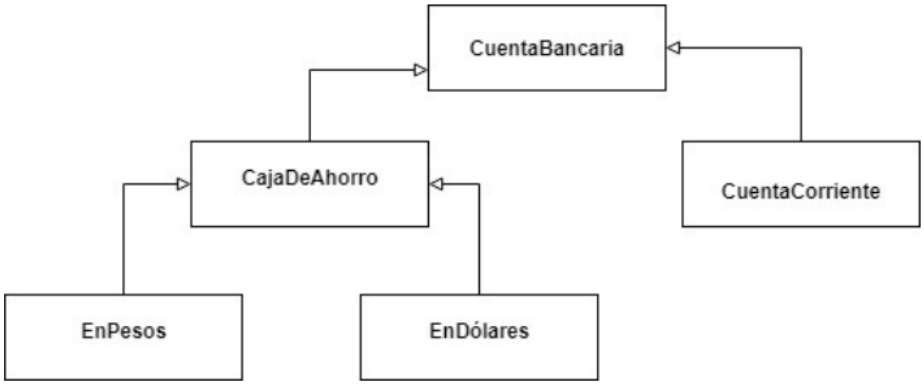
Representa un área de memoria compartida por todos los objetos de la clase

Herencia- GENERALIZACIÓN Y ESPECIALIZACIÓN

Generalización: Factoriza los elementos comunes de un conjunto de clases en una clase general llamada superclase.



Especialización: Captura las particularidades de un conjunto de objetos no discriminadas por las clases ya identificadas.

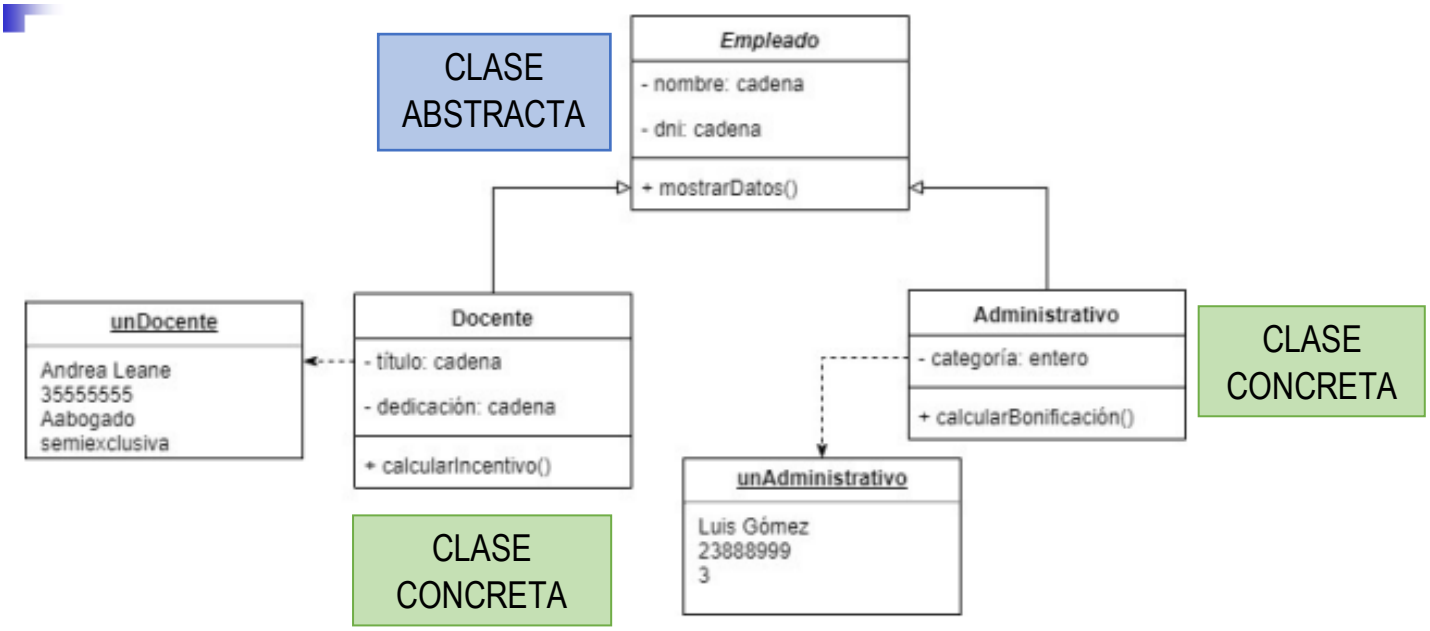


- Las clases se ordenan según una jerarquía, la superclase es una abstracción de sus subclases.
- Los árboles de clases no crecen a partir de la raíz. Se determinan partiendo de las hojas, ya que los niveles superiores son abstracciones construidas para ordenar y comprender.
- Una subclase identifica el comportamiento de un conjunto de objetos que hereda características de la clase padre y adiciona algunas especificas que ésta no posee.

Herencia- CLASE CONCRETA Y CLASE ABSTRACTA

Clase Abstracta: No existe un objeto que sea instancia directa de ella, pero si existe una subclase de ella que es instanciable. Con cursiva.

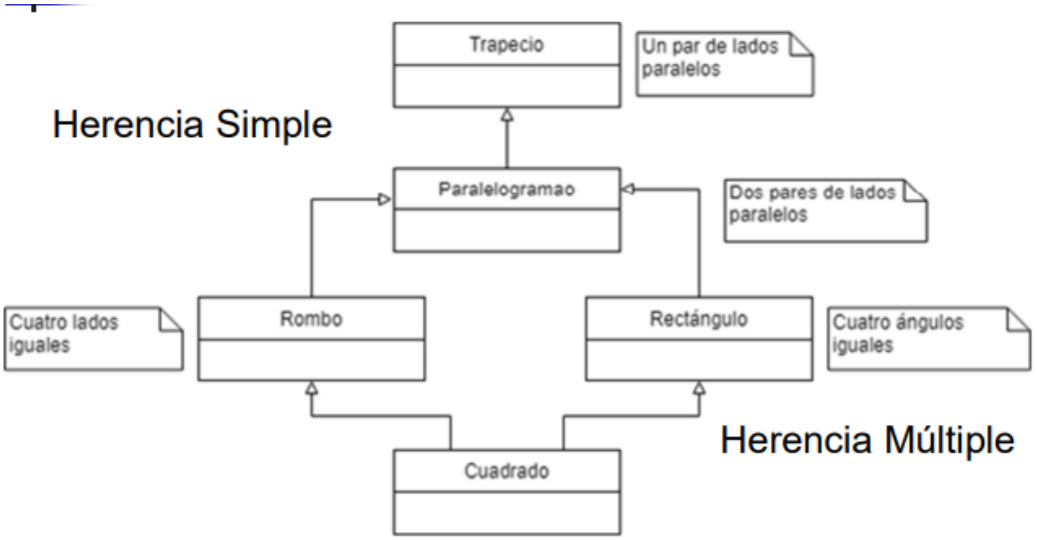
Clase Concreta: Es aquella que es instanciable, existe al menos un objeto de esa clase.



Herencia- TIPOS DE HERENCIA

Herencia Simple: Cuando una clase hereda de una sola clase padre. Es soportada por todos los lenguajes orientados a objetos.

Herencia Múltiple: Cuando una clase hereda de más de una sola clase. Es soportada solo por algunos lenguajes.



RELACIONES ENTRE CLASES

Expresan cómo se comunican los objetos de esas clases entre sí y muestran el acoplamiento de las clases.

Acoplamiento entre clases: Número de clases con las que una clase concreta está relacionada (acoplada). Una clase está acoplada si sus objetos lo están. Un objeto está acoplado con otro si uno de ellos actúa sobre el otro.

Según el acoplamiento se pueden distinguir distintos tipos de relaciones:

- Asociación
- Agregación
- Generalización/ Especialización/ Herencia

ASOCIACIÓN

- Conexión entre dos clases; refleja una conexión que existe en el ámbito de una aplicación.
- La comunicación puede ser tanto uni como bidireccional.
- Puede tener nombre y se representa por una línea continua entre las clases asociadas.
- No es contenida por las clases, ni subordinada a las clases asociadas.
- Una asociación puede verse como una abstracción de los vínculos que existen entre objetos instancias de las clases asociadas.

En UML, el número de instancias (cardinalidad) que participan en la relación, y se anota en cada extremo de la relación, y se llama **multiplicidad**.

Los valores de multiplicidad más comunes son:

- **1** uno y sólo uno
- **0..1** cero o uno (0 es una relación opcional)
- **m..n** de m a n (m y n enteros naturales)
- ***** de cero a varios
- **0..*** de cero a varios
- **1..*** de uno a varios

La asociación tiene dos variantes:

Clase asociación: Una dupla de objetos, se relaciona con una única instancia de la clase asociación. No importa la multiplicidad en ambos extremos.

Clase que modela la asociación: Para una dupla de objetos existe más de un objeto asociado a la clase.

AGREGACIÓN – REUTILIZACIÓN Y EXTENSIÓN

Reutilización: Uso de clases u objetos desarrollados y probados en un determinado contexto, para incorporar esa funcionalidad en una aplicación diferente a la de origen. La forma más simple de reutilizar una clase es simplemente haciendo una nueva clase que la contenga. Esto es posible a través de la composición.

Extensión: Aprovechar las clases desarrolladas para una aplicación, utilizándolas para la construcción de nuevas clases, en la misma u otra aplicación.

AGREGACIÓN

- Consiste en definir como atributos de una clase a objetos de otras clases ya definidas.
- La agregación es una relación no simétrica (todo/parte) en la que una de las clases cumple un papel predominante respecto de la otra.
- La agregación declara una dirección a la relación todo/parte.
- Gráficamente se representa colocando un rombo del lado de la clase agregado.
- Se representa con un rombo vacío.

Dentro de la agregación se tienen dos casos:

Agregación: Un objeto contiene como partes a objetos de otras clases, pero de tal modo que la destrucción del objeto contenedor no implica la destrucción de sus partes. Los objetos contenidos pueden existir independientemente del objeto contenedor.

Composición: Un objeto de una clase contiene como partes, a objetos de otras clases y estas partes están físicamente contenidas por el agregado.

-Los objetos agregados no tienen sentido fuera del objeto resultante.

- Los tiempos de vida de los objetos continente y contenido están estrechamente acoplados

- La destrucción del objeto continente implica la destrucción de sus partes.

- Las composiciones generan una relación de existencia entre el todo y cada una de sus partes.

- La multiplicidad del lado agregado puede tomar el valor 0 o 1. El valor 0 se refiere a un atributo no explicitado.

DIAGRAMA DE CLASES

Es un diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas.

CONCEPTOS CLAVE

Encapsulamiento: - Término formal que describe al conjunto de métodos y de datos de un objeto de manera tal que el acceso a los datos se permite solamente a través de los métodos propios de la clase a la que pertenece el objeto.

- La comunicación entre los distintos objetos se realiza solamente a través de mensajes explícitos.

Abstracción: - La orientación a objetos fomenta que los programadores y usuarios piensen en las aplicaciones en términos abstractos.

- A partir de un conjunto de objetos, se piensa en los comportamientos comunes de los mismos para situarlos en superclases, las cuales constituyen un depósito para elementos comunes y reutilizables.

Polimorfismo: - Capacidad que tienen objetos de clases diferentes, relacionados mediante la herencia, a responder de forma distinta a una misma llamada de un método.

-Fomenta la extensibilidad del software

- Software escrito para invocar comportamiento polimórfico se escribe en forma independiente del tipo de los objetos a los cuales los mensajes son enviados.

- Nuevos tipos de objetos, que pudieran responder a mensajes existentes, pueden ser agregados en dicho sistema sin modificar el sistema base.

Persistencia: - Designa la capacidad de un objeto de trascender el tiempo o el espacio.

- Un objeto persistente conserva su estado en un sistema de almacenamiento permanente (pasivación del objeto).

- El objeto puede ser reconstruido (activación del objeto) por otro proceso y se comportará exactamente como en el proceso inicial.