# Unidad 1:

## 1. Introducción a la Orientación a Objetos

### 1.1. Paradigma Orientado a Objetos

Como primer concepto abordar vamos a explicar el concepto de "*paradigma":*

*"Un paradigma define un conjunto de reglas, patrones y estilos de programación que son usados por un grupo de lenguajes de programación".*

En la programación tenemos diferentes paradigmas, dentro de los cuales podemos destacar los siguientes:

- ***Paradigma por Imperativo o Procedural:*** es el paradigma original de la programación y quizá el de uso más común en la actualidad. La programación estructurada es un actor principal en este paradigma. Ej. C o Pascal.
- ***Paradigma Lógico:*** se definen reglas lógicas que luego por medio de un motor de inferencias lógicas, resuelve los problemas presentados al sistema. Ej. PROLOG
- ***Paradigma Orientado a Objetos:*** este paradigma consiste en el uso de clases y objetos. Este es uno de los paradigmas de más auge en estos tiempos. En el desarrollo de esta unidad profundizaremos los conceptos que involucran todo su desarrollo. Ej: Java, Python…

Cabe destacar que mayormente los lenguajes de programación aplican uno o más paradigmas.

El paradigma orientado a objetos surge en la historia como un intento para dominar la complejidad que posee el software. Desde sus principios, la manera de encarar el desarrollo del software fue a través de la programación estructurada, su resolución a grandes rasgos consiste en dividir el problema principal en sub-problemas cada vez más pequeños. Así de esta manera, se hace más simple afrontar un problema complejo.

En el paradigma orientado a objetos es otra manera de afrontar los problemas, existe una descomposición pero de una manera diferente. Lo que se hace aquí es la fragmentación a través de objetos. Para llevar adelante esta división lo que vamos hacer es analizar el entorno del problema para tratar de abstraer el problema lo que más se pueda, para poder crear un modelo del escenario del mismo.

### 1.2. Historia y Lenguajes

La programación orientada a objetos surgió con la necesidad de describir y simular fenómenos, como las redes neuronales, el flujo de tráfico, etc.

En 1961 *Kristen Nyguard,* dio origen a las ideas de un lenguaje de doble propósito, primero que nada sería describir el sistema y el otro programar la simulación. Junto con *Ole-Johan Dal, Nyguard* perfecciono el desarrollo del lenguaje *Simula 1.* Algunos de los conceptos madurados a partir de este lenguaje fueron las construcciones de datos y acciones agrupados como: encapsulamiento y objetos. También acceso externo a atributos de objetos y ya se observaba también la herencia en sus inicios. La maduración de estos conceptos dio nacimiento al *Simula 67*. Después de esto tenemos varios lenguajes descendientes de estos conceptos y que lograron profundizarlos.

En la década de los 70's el modelo se amplia y surge *Smalltalk,* desarrollado por *Alan Kay.*

En los 80's surge *C++,* desarrollado Bjarne Stroustrup, que era un antiguo usuario de Simula, quien plasmo la características de este lenguaje en *C.*

En 1995 surge en el lenguaje *JAVA,* y sus creadores son *James Gosling y Bill Joy.* Este lenguaje desciende de otro llamado *OAK.*

## 2. Propiedades del Paradigma Orientado a Objetos.

### 2.1. Objetos

El termino objeto fue utilizado formalmente por primera vez en *Simula* y se usó en sus desarrollos para moldear algunas características de la realidad.

Los objetos representan entidades, que se pueden clasificar en:

- Cosas Tangibles
- Roles
- Incidentes
- Interacciones
- Especificaciones

Algunas definiciones que podemos encontrar de ***Objeto***, entre las que me quedo con la de algunos de los autores más reconocidos, son las siguientes:

"*Un objeto representa elementos identificables, unidades o entidades individuales, reales o abstractas, pero con un rol bien definido en el dominio del problema*" por *Smith y Tockey.*

*"Un objeto es un concepto, abstracción o cosa con fronteras bien definidas y significado para el manejo del problema"* por *Rumbaugh.*

*"Un objeto tiene estado, comportamiento e identidad; la estructura y comportamiento de objetos similares son expresados a traves de una clase; los términos objeto e instacia son intercambiables"* por *Booch.*

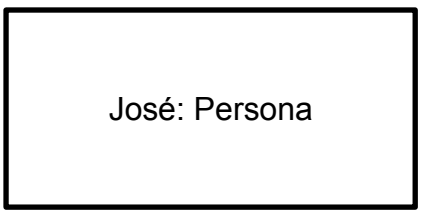


**Diagrama de Objetos**

- Instancia: Una instancia es un objeto creado de una clase. La clase describe la estructura (comportamiento e información) de la instancia. Al utilizar un lenguaje se generan objetos instanciando clases. La diferencia que existe entre instancia y objeto es puramente temporal ya que en el momento del análisis se habla de “objetos” y en tiempo de diseño e implementación se habla de “instancias”.

- Atributos: Son los datos que definen el interior de un objeto.

- Comportamiento: Es la forma en que un objeto actúa y reacciona en términos de sus cambios de estado y la transferencia de mensajes; la actividad exterior y visible de un objeto que puede ser captada y modificada por otros objetos a través de los protocolos de comunicación del objeto receptor.

- Operación: Es una actividad que pertenece a un objeto y que puede:

  a) Ser considerada parte de la interfaz del objeto con su medio en cuyo caso se pone en marcha como resultado de la recepción de un mensaje.
  b) Ser privada del objeto, es decir, de uso exclusivo del objeto, en cuyo caso se activa como resultado de una inicialización interna como por ejemplo: tareas de validación del formato de los datos que contendrá un objeto.

- Estado: Es el valor de los atributos de un objeto como resultado de las operaciones realizadas sobre ellos. Es el resultado acumulativo del comportamiento de un objeto; es una de las posibles condiciones en las que un objeto puede existir. En un momento determinado del tiempo el estado de un objeto se define por sus propiedades (usualmente estáticas) y los valores de las mismas (usualmente dinámicas).

- Estímulos: Eventos por los cuales un objeto se comunica con otro. Un estímulo incentiva algún comportamiento para que un objeto se ponga en acción y no necesariamente incluye algún mensaje de información.

- Método: Es una operación que pertenece a un objeto que puede considerarse como parte de la interfaz que existe entre el objeto y su medio. Los métodos definen el comportamiento de un objeto y son puestos en funcionamiento a través de mensajes. Un método es un tipo de operación (de interfaz) solo en aquellos lenguajes que permiten la definición de operaciones de tipo "privadas" mientras que en los lenguajes que no permiten dicha definición "método" y "operación" son términos equivalentes.
- Mensaje: Es una acción por la cual un objeto influye sobre otro, es decir, que un objeto envía un estímulo a otro a través de mensajes los cuales pueden ser comprendidos por el receptor debido a que dichos mensajes hacen referencia a operaciones de la interfaz del objeto (o sea, a su protocolo de comunicación). Un mensaje incluye la mención hacia la operación a realizar y los parámetros por ella requeridos mientras que un estímulo representa solo la forma en que un objeto emisor capta la atención del objeto receptor; en otras palabras, un mensaje transporta un estímulo.

- Identidad: Es la naturaleza de un objeto que lo distingue del resto de los objetos (es decir, es el objeto en sí mismo). Un objeto puede cambiar de estado pero no de identidad. Específicamente la identidad de un objeto está definida por la posición lógica de memoria ocupada por el objeto.

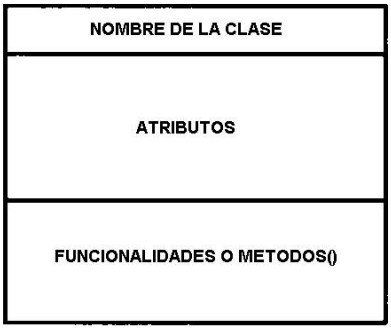
### 2.2. Clases

Una clase representa un modelo de varios objetos y describe cómo esos objetos están internamente estructurados. Los objetos de la misma clase tienen la misma definición para sus operaciones y para su información.

Las clases son implementaciones de tipos de datos que pueden:

a) Ser predefinidos por el lenguaje.
b) Ser definidos por los usuarios del lenguaje.

De la siguiente manera vamos a representar una clase gráficamente, para cuando dibujarlas.




Una clase tiene:

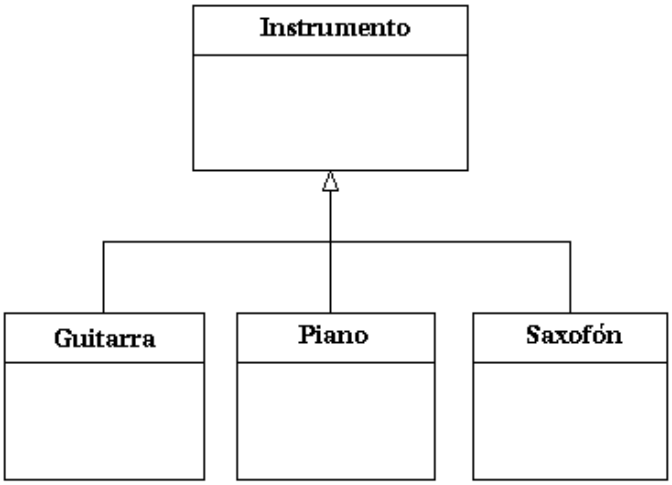
- Atributos: datos o variables que se caracterizan el estado de un objeto.
- Métodos: Procedimientos o acciones que cambian el estado de un objeto.

Estos dos componentes principales de una clase, constan de distintos niveles de visibilidad. Para modificar la visibilidad de los atributos y métodos, dependiendo del lenguaje utilizado, tenemos diferentes modificadores de acceso. Los más comunes que podemos encontrar son los siguientes:

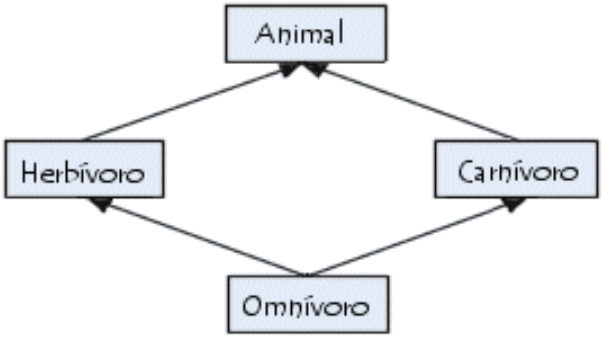
- **Public**: Todo el mundo puede acceder al elemento. Si es un dato miembro, todo el mundo puede ver el elemento, es decir, usarlo y asignarlo. Si es un método todo el mundo puede invocarlo.
- **Private:** Sólo se puede acceder al elemento desde métodos de la clase, o sólo puede invocarse el método desde otro método de la clase.
- **Protected:** proporciona acceso público para todas las clases derivadas.

Relaciones estáticas: Son relaciones que existen por un largo período y que implican que dos objetos conocen sobre la existencia del otro y que existe una cierta pertenencia mutua entre sus respectivas clases. Las relaciones estáticas pueden definirse antes de la puesta en funcionamiento del sistema.

Herencia: Es una relación entre clases, donde una o más clases comparten la estructura o comportamiento definidos en una (herencia simple) o más (herencia múltiple) clases.




**Herencia Simple**




**Herencia Múltiple**

Jerarquía: Es un ordenamiento de abstracciones.

Especializar: Si es necesario crear una nueva clase puede identificarse otra que contenga algunas de sus características, es posible entonces heredarlas y adicionar las características particulares de la nueva clase (esta técnica recibe el nombre de especialización).

Generalizar: Extraer y compartir características comunes colocando a una clase en el más alto nivel de la jerarquía.

Subclase: Es una clase que hereda de una o más clases.

Superclase: Es la clase de la cual otras heredan.

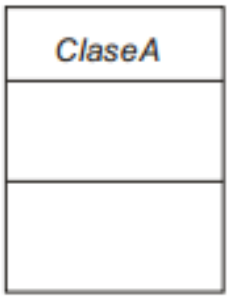
Encapsulamiento: Se define desde el interior del objeto hacia su medio (exterior del objeto). Es el proceso por el cual se dividen los elementos de una abstracción que constituyen su estructura y comportamiento; el encapsulamiento sirve para separar la interfaz de una abstracción y su implementación, lo cual significa que todo lo que se conoce de un objeto es su interfaz y que para usarlo solo es necesario conocer las operaciones que el objeto puede realizar.

Ocultamiento de información: Es una forma de llevar a la práctica el encapsulamiento. Los objetos manejan este concepto, por lo tanto, esconden su estructura interna a su medio. Toda la información en un sistema orientado a objetos es almacenado en sus objetos y puede ser solamente manipulada cuando los objetos deben realizar sus operaciones (el comportamiento y la información son encapsuladas en el objeto). En otras palabras, es el proceso de ocultar todos los secretos de un objeto que no contribuyen a sus características esenciales. Típicamente son ocultados la estructura de un objeto y la implementación de sus métodos.

Polimorfismo: Significa que el emisor de un estímulo no necesita conocer la clase a la cual pertenece la instancia receptora (la instancia receptora podría pertenecer a cualquier clase arbitraria). Es la posibilidad de que cualquier objeto de diferentes clases pueda responder de una forma particular a una misma operación. Es la propiedad según la cual un mismo nombre de función puede tener versiones diferentes, según cuál sea la clase de objetos a la que se aplica.

### 2.2.1 Clases Abstractas

Son clases que no tienen implementación para todos los métodos definidos. Esto indica que la clase no puede ser instanciada pues posee métodos abstractos (que aún no han sido definidos, es decir, sin implementación). La representación de este tipo de clases es igual que para las clases comunes, salvo que el nombre de la clase se escribe en cursiva.




Características de una Clase Abstracta:

- Una de las características principales es que se encarga de modelar el comportamiento común de sus clases derivadas.
- También establece métodos que han de ser implementados por sus subclases.
- Una clase abstracta puede contener métodos no abstractos, pero al menos uno de ello debe ser abstracto.
- La clase no puede ser instanciada.

## 2.3. Relaciones Entre Clases

Las relaciones que existen entre los distintos objetos de cada una de las distintas clases que podemos tener, nos avisan como se comunican estos entre sí.

Podemos decir entonces que los mensajes navegan entre las distintas relaciones que podemos tener entre las clases.

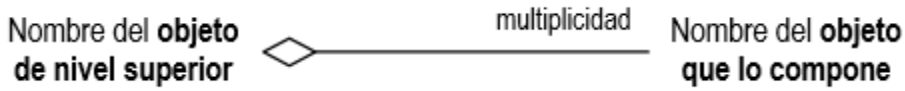
Según *G. Booch,* existen tres tipos básicos de relaciones entre los objetos:

- Agregación / Composición.
- Asociación.
- Generalización / Especialización.

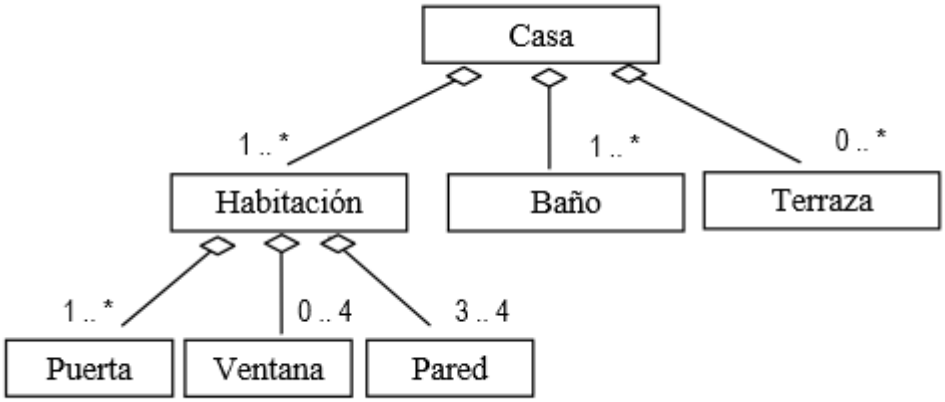
### 2.3.1 Agregación

Es una relación que representa a los objetos compuestos por otros objetos. A su vez hace referencia a objetos que a su vez están formados por otros. El objeto de nivel superior de la jerarquía es el todo y los que están en los niveles inferiores son las partes o componentes.

A continuación mostramos la representación gráfica de este tipo de relación:




Lo siguiente es un ejemplo donde empleamos este tipo de relación:



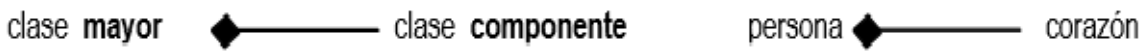

Acá también podemos ver, como se indica la multiplicidad en el correspondiente diagrama.

| Multiplicidad | Significado |
|---|---|
| 1 | Uno y sólo uno |
| 0..1 | Cero o uno |
| N..M | Desde N hasta M |
| * | Cero o varios |
| 0..* | Cero o varios |
| 1..* | Uno o varios (al menos uno) |

### 2.3.2 Composición

La relación es más fuerte que el caso de agregación, al punto que si el componente es eliminado o desaparece, la clase mayor deja de existir.

Este tipo de relación se gráfica de la siguiente manera:

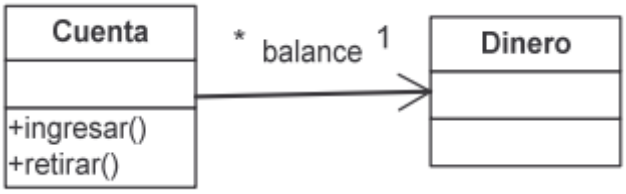



### 2.3.3 Asociación

La asociación es una relación estructural, que describe una relación de uso.

Gráficamente se representa así:


Las asociaciones por lo general son bidireccionales, pero puede haber alguna oportunidad donde se desee hacerla unidireccional.

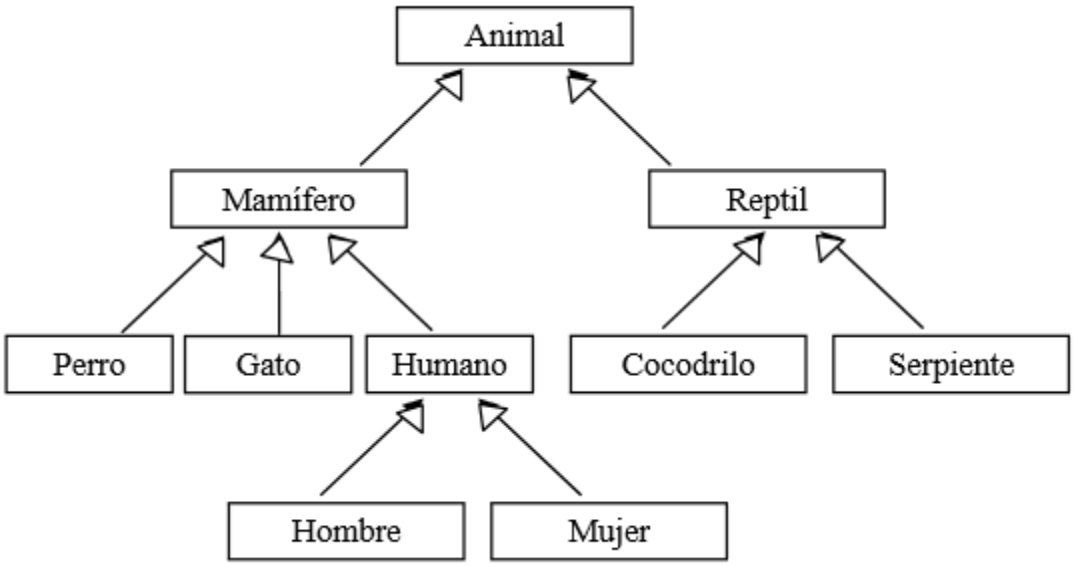



### 2.3.4 Generalización

Es un tipo de jerarquía de clases, en la que cada subclase contiene los atributos y métodos de una (herencia simple) o más superclases (herencia múltiple).

Cada subclase o clase hija en la jerarquía es siempre una extensión (esto es, conjunto estrictamente más grande) de la(s) superclase(s) o clase(s) padre(s) y además incorporar atributos y métodos propios, que a su vez serán heredados por sus hijas.

Gráficamente se representa de la siguiente manera:




### 2.3.5 Interfaces

Las interfaces son clases abstractas, lo que significa que no es posible crear instancias directamente a partir de ellas. Pueden contener operaciones, pero no atributos. Las clases pueden heredar de las interfaces (a través de una asociación de realización) y de estos diagramas sí es posible crear instancias.

Se representa con rectángulo y el nombre esta precedido con el estereotipo **<< *interfaz*>>**, o con una línea con un circulo en el extremo, etiquetado con el nombre de la interfaz. Nosotros utilizaremos por defecto la representación gráfica con el rectángulo.