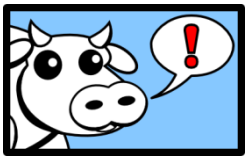
Programación Orientada a Objetos con Java

1. Programando Objetos

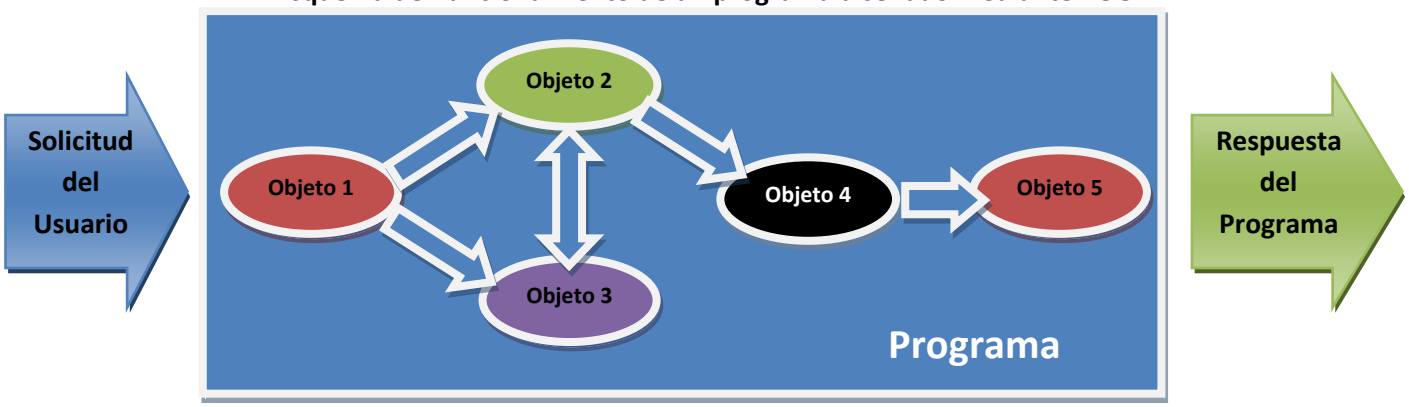
La Programación Orientada a Objetos (POO) es un **paradigma de programación** que surge como una respuesta a la necesidad de métodos y técnicas de programación que permitieran describir mejor la realidad, más ágiles y más potentes. El razonamiento detrás de la POO es que resulta mucho más natural programar si pensamos en los diferentes componentes de un programa como ***objetos que interactúan entre sí***.

De esta manera, obtenemos la definición de **Programa** en la POO, que es:



DEFINICIÓN : Programa
Conjunto de **OBJETOS** que **CUMPLEN DIVERSAS FUNCIONES** e **INTERACTÚAN ENTRE SÍ**, intercambiando **INFORMACIÓN.**

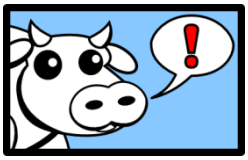
Esquema del funcionamiento de un programa diseñado mediante POO



De esta manera, para diseñar un programa lo que se hace es **diseñar y crear los objetos** que necesitamos para que cumplan las diversas tareas que componen el programa y relacionarlos entre sí.

¿Qué es un objeto?

En la POO, un objeto puede ser **cualquier cosa que yo le pueda describir a la computadora**. Puede basarse en un objeto de la realidad, o ser imaginario, o una abstracción.



DEFINICIÓN : Objeto
Conjunto de recursos (variables, algoritmos, funciones, etc.) que, juntos, **proporcionan una funcionalidad determinada dentro de un programa**. Un objeto puede representar una realidad física (los datos de una persona, por ejemplo) o un concepto completamente abstracto (como la conexión a una base de datos).

Pero todos los objetos tienen una misma estructura: ***Clase, Nombre, Atributos y Métodos.***

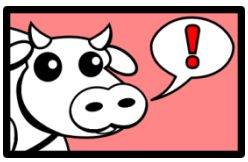
La ***CLASE*** se puede definir como la **"familia" a la que pertenece un objeto**. Igual que una familia determina ciertas cualidades de una persona, la clase determina las características generales del objeto. De esta manera, todos **los objetos de una misma clase tienen las mismas características generales y las mismas capacidades**.

El ***NOMBRE*** es el **identificador de un objeto dentro del programa**. Un mismo programa puede, durante su ejecución, generar **varios objetos de la misma clase** (mediante un proceso llamado ***Creación de Instancias***, que

veremos más adelante) pero para poder identificarlos y usar sus funciones, cada objeto debe tener **un nombre propio único** dentro del programa.

Los ***ATRIBUTOS*** son las **características del objeto**, lo que indica "cómo es" un objeto y en general, podemos decir que son ***variables*** tal y como las vimos antes. Algo importante a tener en cuenta es que, mientras que dos objetos de la misma clase tienen **el mismo conjunto de atributos**, **los valores** de esos atributos **pueden ser diferentes** (de la misma manera que dos hermanos pertenecen a la misma familia y comparten los mismos rasgos, pero de forma distinta).

Los ***MÉTODOS*** son las **capacidades del objeto**, indican "qué puede hacer" el objeto. Se trata de conjuntos de algoritmos o conjuntos de algoritmos que permiten que el objeto realice una tarea concreta. Es mediante la ejecución de estos métodos que el programa lleva a cabo sus funciones.

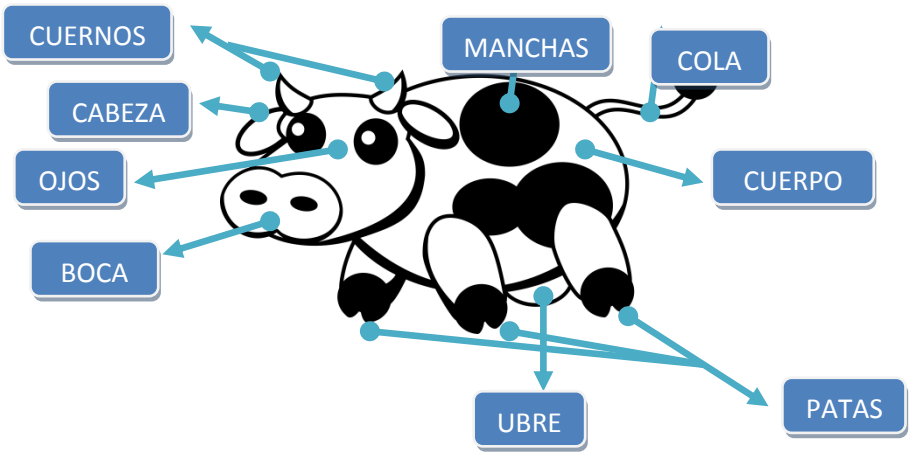


CONCLUSIÓN:

- **Todos los componentes** del programa **son objetos** que **se generan usando clases** .
- Todo objeto del programa tiene **atributos y métodos** que se definen **en su clase**.
- A los atributos **se les asigna valores**, los métodos **se ejecutan.**
- El programa está formado por **objetos creados usando ese conjunto de clases**.

Ejemplo 1:

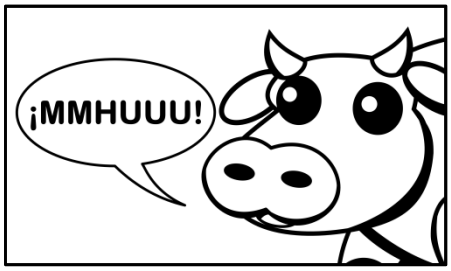
Como se explicó más arriba, la POO se basa en el concepto de objeto que manejamos en el mundo real, y para este ejemplo usaremos **una vaca**. Primero **¿cómo está formada una vaca?**, podríamos entrar en muchos detalles, pero generalicemos lo más posible:



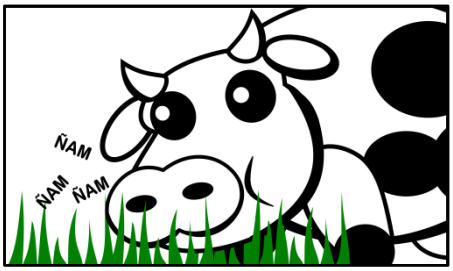
Lo que forma a nuestra vaca, sus características, serán sus **atributos**. Y podemos agruparlos de esta manera:

VACA
• **Cabeza** • **Cuernos** • **Ojos** • **Boca** • **Cuerpo** • **Manchas** • **Ubre** • **Cola** • **Patas**

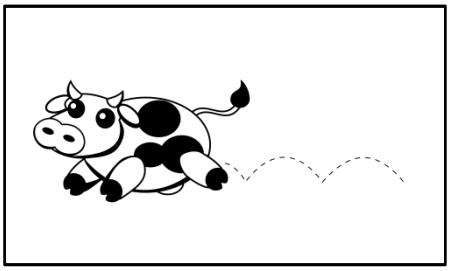
Ahora bien, una vaca también es capaz de **hacer cosas**. Por ejemplo:



Mugir



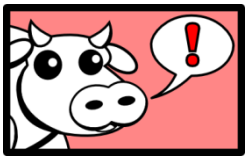
Comer



Correr

Las cosas que nuestra vaca **puede hacer** son sus habilidades o capacidades, y las llamaremos **métodos.** Al igual que hicimos con los atributos, podemos agrupar los métodos en una tabla, de esta manera:

VACA
• **Mugir()** • **Comer()** • **Correr()**



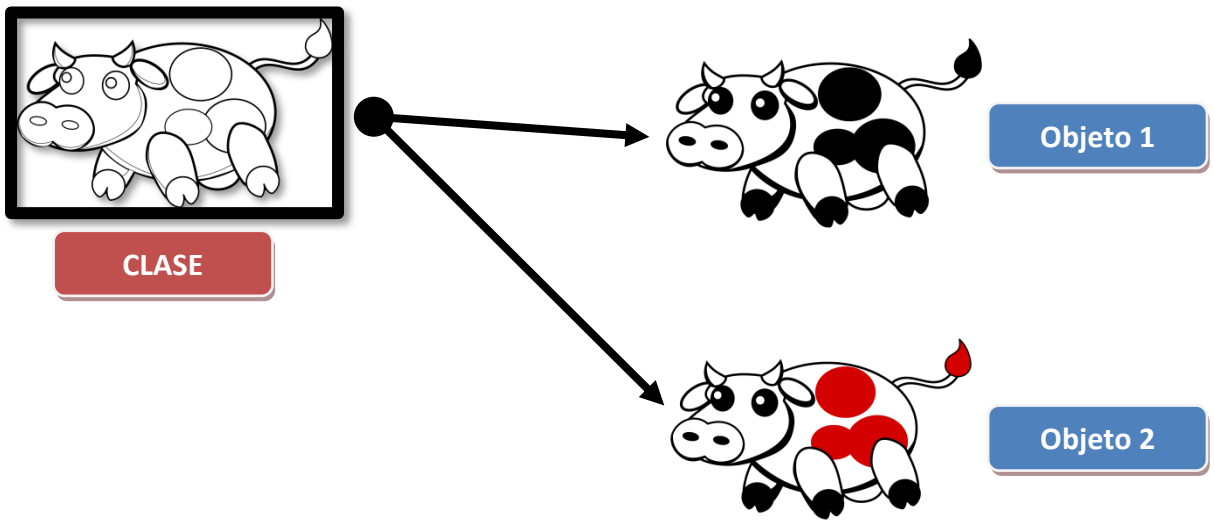
IMPORTANTE: Te habrás fijado que los métodos se escribieron con un paréntesis al final. Esto es algo importante a tener en cuenta, ya que los métodos **siempre** se deben anotar de esa manera, ya que es la forma en que los diferenciamos de los atributos.
Además, el paréntesis **tiene un propósito práctico** que explicaremos en el próximo párrafo.

¿Cómo se crea un objeto?

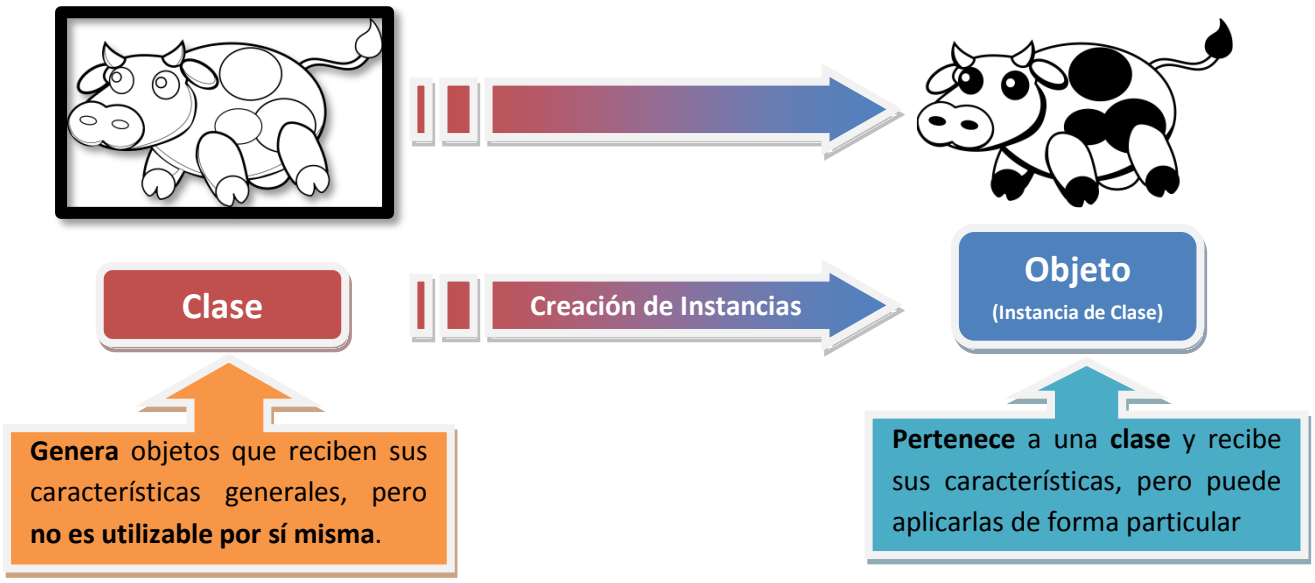
Para crear un objeto **primero debe existir su clase**. No podremos crear un objeto si su clase no ha sido definida aún.

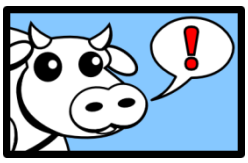
Las clases son los componentes básicos de todo proyecto de programación orientada a objetos. **Todo lo que existe y sucede dentro del programa, se define dentro de las clases, y se lleva a cabo mediante objetos que se generan a partir de esas clases.**

Podemos pensar en una clase como **en un molde**, con el cual **creamos nuevos objetos** que tendrán la "misma forma", o sea las mismas características generales, pero que serán **objetos diferentes** durante la ejecución del programa.



Este proceso de creación de objetos se llama ***Creación de Instancias*** o ***Instanciación*** , y cada objeto nuevo es una ***instancia*** de la clase a la que pertenece.





DEFINICIÓN : Instancia

Es la creación de un **objeto** a partir de las características definidas en una **clase.** Los objetos sólo pueden crearse mediante instancias, y las instancias solo son posibles mediante clases.
Es mediante la creación de un objeto que la clase puede usarse, ya que en muy pocos casos es posible (o recomendable) el utilizar los métodos y/o atributos de una clase de forma directa.
En general: Una **clase es inútil sin objetos que la instancien**, y **es imposible crear un objeto mediante una instancia si este no pertenece a una clase.**

¿Cómo se crea una clase?

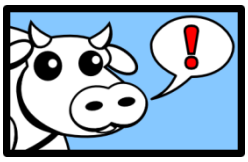
La creación de una clase se llama ***declaración***. Al declarar una clase, se establece la descripción de los objetos que se crearán a partir de esa clase tal como lo hemos definido más arriba: mediante sus atributos y sus métodos.

Para la declaración de una clase, es importante que primero tengamos claro los atributos y los métodos que necesitaremos definir. Los diagramas que utilizamos en el ejemplo de más arriba nos servirán para eso, primero consideremos los datos que tenemos hasta el momento y vamos a plantearlos en un solo diagrama

VACA	
Atributos • Cabeza • Cuernos • Ojos • Boca • Cuerpo • Manchas • Ubre • Cola • Patas	**Métodos** • Mugir() • Comer() • Correr()

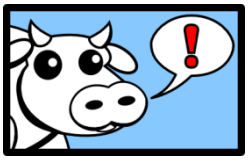
Para el desarrollo de clases en POO se utilizan diagramas muy similares a este mediante un lenguaje gráfico llamado **UML.** En este capítulo veremos el primer paso del diseño de clases en este lenguaje: la creación de entidades, pero para llegar a ese punto primero necesitamos formalizar un poco más nuestro diagrama de atributos y métodos. En primer lugar formalicemos un poco nuestros atributos y nuestros métodos.

En este punto nos tenemos que preguntar acerca de la **visibilidad** de los atributos y métodos de la vaca.



DEFINICIÓN : Visibilidad
Es una característica de los atributos y los métodos de un objeto que indica si el atributo o método puede ser modificado o utilizado por otro objeto.

En principio, consideraremos dos valores de visibilidad: **público** y **privado**. La accesibilidad significa cosas diferentes para un atributo y para un método:

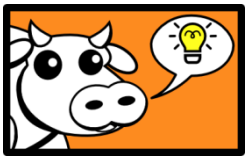


Visibilidad
En un Atributo:
- **Atributo Público:** *Cualquier objeto* puede acceder y modificar los valores del atributo.
- **Atributo Privado:** *Solo el objeto propietario del atributo* puede acceder y modificar los valores del mismo.

En un Método:
- **Método Público:** El método puede ser ejecutado por *cualquier objeto*
- **Método Privado:** El método solo puede ser ejecutado *por el objeto propietario del mismo*

Para definir cuál característica del objeto será pública y cuál será privada, tenemos que considerar **la forma en que queremos que nuestro objeto interactúe** con los demás.

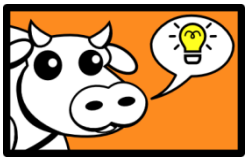
En general, para **los atributos**:



Un **atributo se definirá como público si:**
- Contiene o va a contener datos que deben ser **modificados por otros objetos**.
- Sus datos pueden o deben **modificarse durante la ejecución** del programa.

De lo contrario, **el atributo será definido como privado**.

Y para **los métodos**:



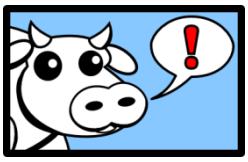
Un **método se definirá como público si:**
- El objeto lo va a necesitar para intercambiar información con otros objetos.
- El resultado de su ejecución es necesaria para el funcionamiento de otros objetos.

De lo contrario, **el método será definido como privado**.

Ahora solo debemos pensar cómo serán los atributos y los métodos de nuestra vaca. Como los atributos son sus características físicas, y estas no son fácilmente modificables, diremos que son atributos privados. Si fueran públicos, es como decir que podemos cambiar a voluntad las manchas de nuestra vaca, o cuántos ojos tiene.

En cuanto a los métodos, el razonamiento es parecido. El método **comer()** será privado (solo la vaca puede decidir cuándo comer). Pero los métodos **mugir()** y **correr()** pueden depender de estímulo externo (o sea, algo puede hacer que la vaca se ponga a correr o a mugir), así que los haremos públicos.

Para indicarlo en nuestro diagrama usaremos los siguientes símbolos delante de cada característica:

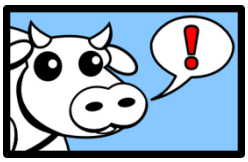


+ Característica	→ Indica que la característica es **pública**
- Característica	→ Indica que la característica es **privada**

De esta manera nuestro diagrama, por ahora, se verá de esta manera:

VACA	
Atributos	**Métodos**
- Cabeza	**+ Mugir()**
- Cuernos	**- Comer()**
- Ojos	**+ Correr()**
- Boca	
- Cuerpo	
- Manchas	
- Ubre	
- Cola	
- Patas	

Un último elemento de UML que veremos ahora es la forma correcta en que una clase se describe en este lenguaje. En UML, una clase se describe indicando primero el nombre, luego los atributos y finalmente los métodos, con lo que el diagrama queda de esta manera:



Representación de una Clase en UML:

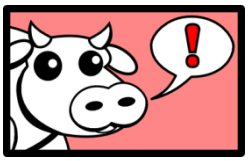
NombreDeLaClase
Atributos
Métodos()

Aplicándolo en la definición de nuestra vaca:

VACA
- Cabeza **- Cuernos** **- Ojos** **- Boca** **- Cuerpo** **- Manchas** **- Ubre** **- Cola** **- Patas**
+ Mugir() **- Comer()** **+ Correr()**

Así que definimos nuestra primera clase y estamos en condiciones de empezar a crear el código de programación para poder utilizar esta clase en nuestro programa, pero para ello necesitamos conocer el lenguaje que usaremos: **Java**.

En el próximo capítulo comenzaremos con los conceptos básicos de programación en Java y empezaremos a ver cómo transformar las clases que declaremos en código en un programa.



CONCLUSIÓN:

- Un objeto necesita una clase para poder crearse. En esa clase se definen los **atributos** y los **métodos** que tendrá el objeto cuando sea creado.
- La creación o definición de una clase se llama **declaración**, y la creación de un objeto a partir de una clase se llama **creación de instancias** o **instanciación**.
- Todo objeto es **instancia** de alguna clase.
- Los atributos y métodos de una clase tienen **visibilidad,** y pueden ser **públicos** o **privados.**

Para mostrar el proceso de declaración de una clase usaremos un ejemplo: la creación de una agenda electrónica. Para llevar a cabo esta tarea se puede usar una variedad muy grande de editores: NetBeans, Eclipse, Notepad++, incluso el mismísimo Bloc de Notas, pero en cualquier caso para empezar a programar en Java es recomendable instalar algunos componentes, el detalle de los mismos y lpuede encontrarse en este documento público de Google Drive

- **El Kit de Desarrollo de Java (JDK):**
  Necesario para poder compilar los programas que diseñaremos. Se puede descargar
-