# UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

## FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

## DEPARTAMENTO DE PROGRAMACIÓN DE COMPUTADORAS

## FUNDAMENTOS DE PROGRAMACIÓN

Año 2019

# ÍNDICE

# CONCEPTOS FUNDAMENTALES DE SISTEMAS

## Enfoque Sistémico

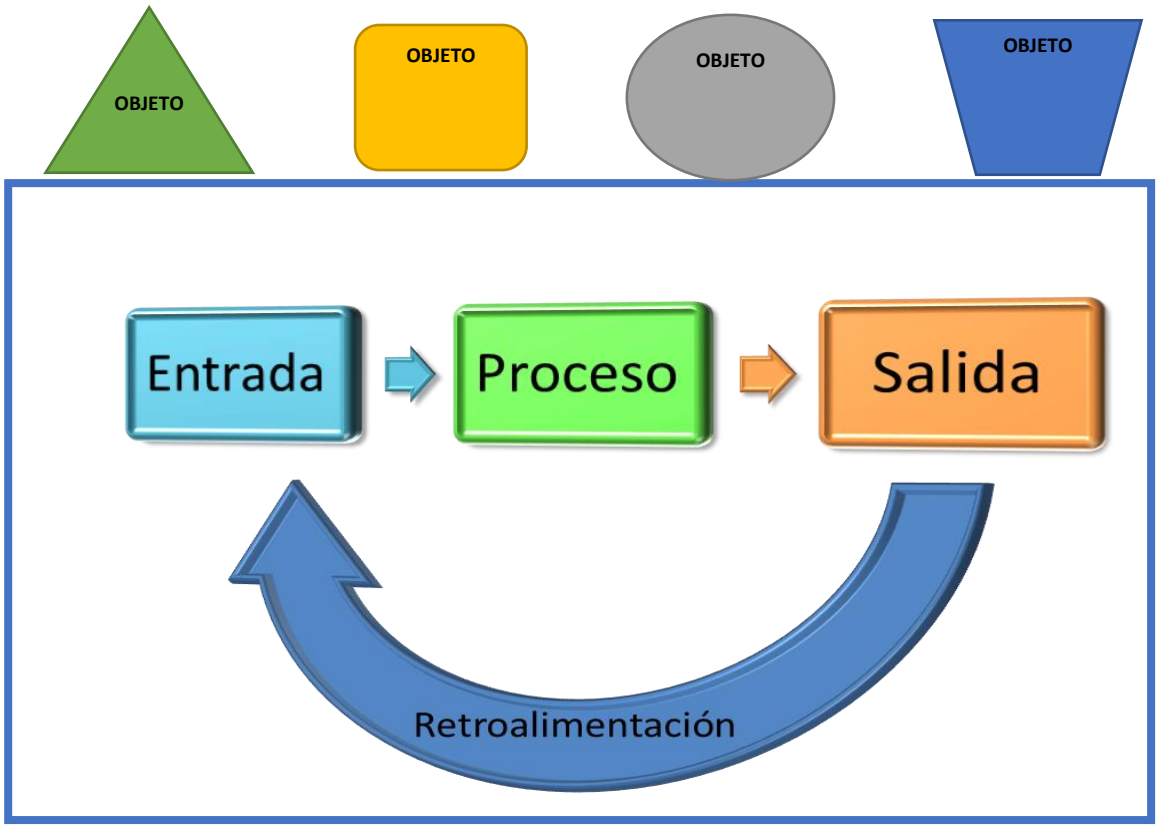
Cuando escuchamos la frase enfoque sistémico, debemos inmediatamente pensar en la forma de abordar o estudiar un fenómeno o un objeto como sistema. Dicho de otra forma, sería incorrecto realizar el estudio como un elemento aislado y no como un elemento que forma parte de un todo.

Al analizar el sistema se deben establecer sus límites, sus propiedades, sus componentes funcionando como un todo (las relaciones entre los componentes), sus flujos de energía, materia o información, identificar subsistemas o suprasistemas, etc.

## Sistemas

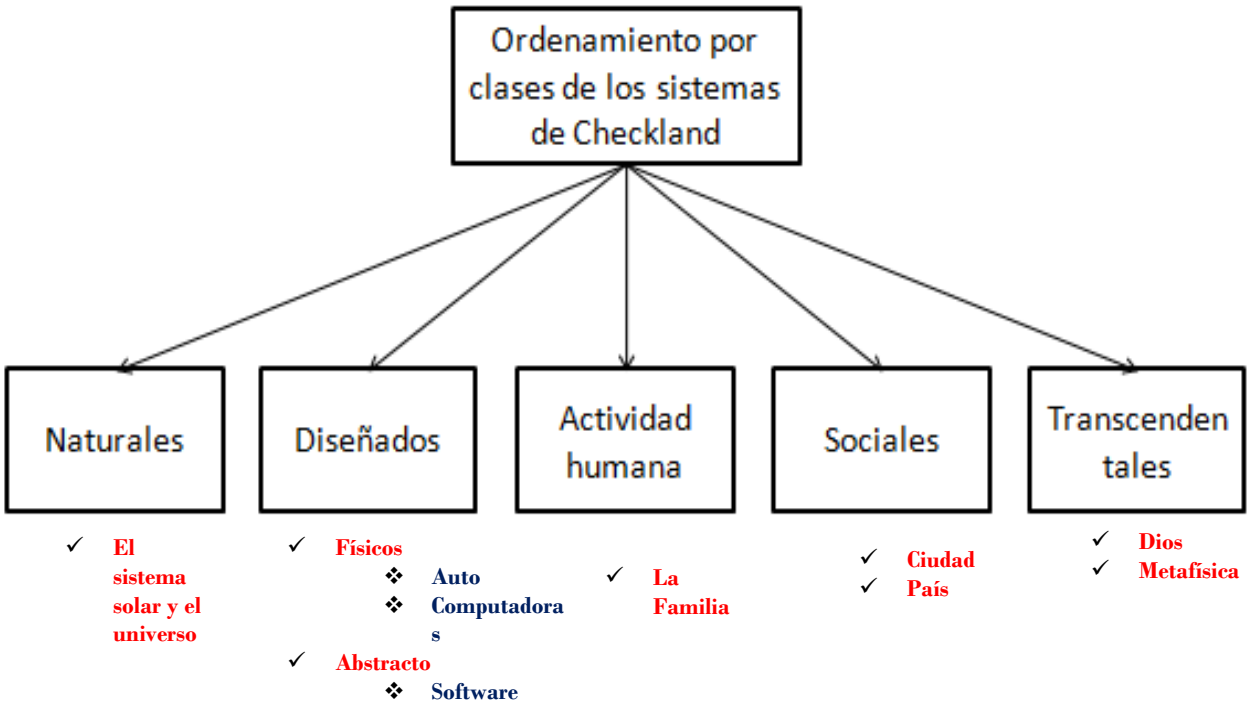
Definimos como un Sistema es un conjunto organizado de elementos que interactúan entre sí o son interdependientes, formando un todo complejo, identificable y distinto.

Un sistema puede estar formado por un grupo de **subsistemas** si mantienen una relación entre sí que los hace también un conjunto identificable y distinto.

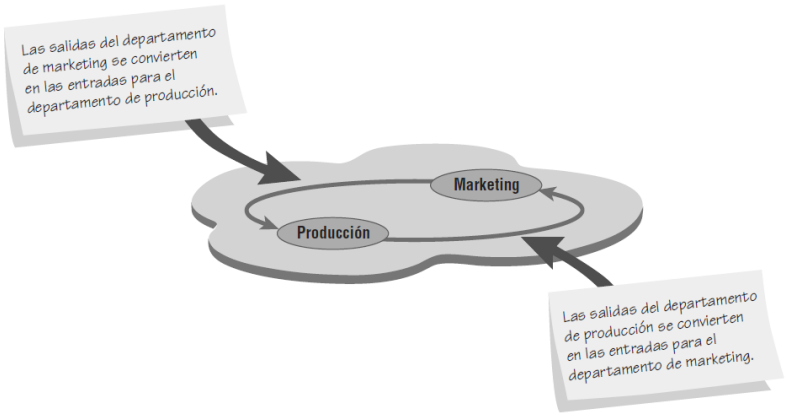


- Los sistemas reciben del exterior **entradas** (*inputs*) en forma, por ejemplo, de información, o de recursos físicos, o de energía.
- Las entradas son sometidas a **procesos** de transformación como consecuencia de los cuales se obtienen unos resultados o **salidas** (*outputs*).
- Se dice que hay **realimentación** o retroalimentación (*feed-back*): cuando parte de las salidas de un sistema vuelven a él en forma de entrada. La realimentación es necesaria para que cualquier sistema pueda ejercer control de sus propios procesos.
- Cuando de un subsistema se conocen solo las entradas y las salidas pero no los procesos internos se dice que es una **caja negra**.

## Clasificación de Sistemas según Checkland



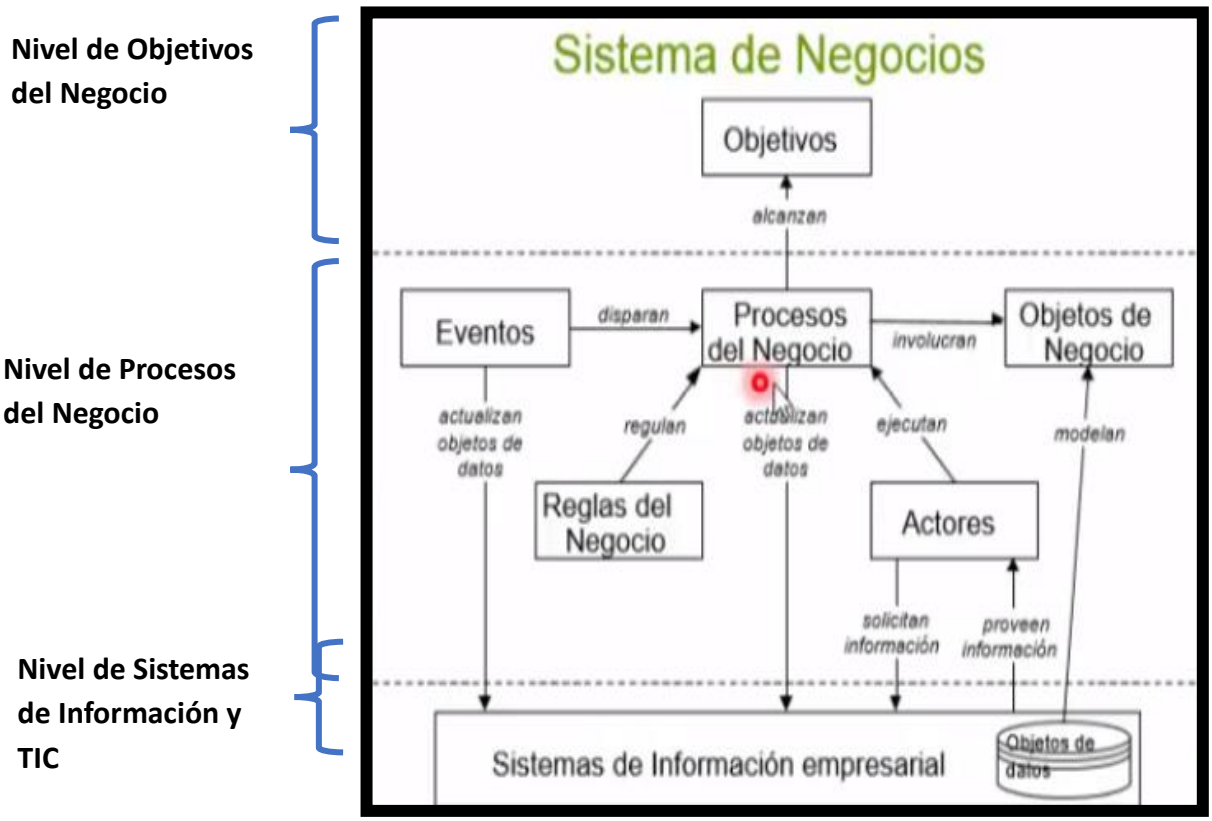
## Las Empresas Vistas como Sistemas




Las organizaciones son sistemas extensos compuestos por subsistemas interrelacionados.

Un sistema de negocios es un conjunto organizado de procesos de negocios, los cuales:

1. Interactúan para alcanzar objetivos predefinidos.
2. Están compuestos de actividades (tareas y acciones).
3. Sus actividades son ejecutadas por actores humanos o autómatas.
4. Siguen reglas del negocio previamente establecidas.
5. Responden a eventos determinados.
    a. Los procesos se ejecutan cuando ocurren estos eventos.
6. Utilizan tecnología para hacerlos más eficientes y efectivos.




## Sistema de Negocios

Para dividir un sistema de negocios en subsistemas se puede emplear diferentes criterios:

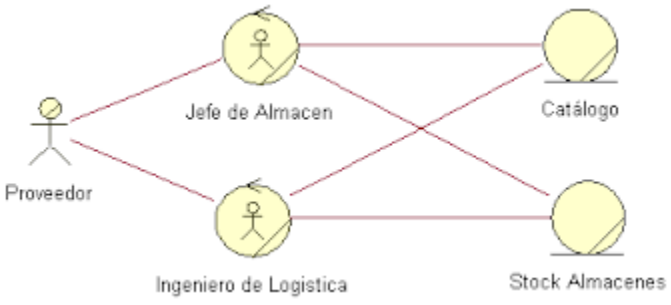
1. Descomposición **por procesos** del negocio
    a. Se identifican los procesos medulares y de apoyo de la empresa.
    b. Cada proceso medular o de apoyo es un subsistema de negocios.
2. Descomposición **por objetos** del negocio
    a. Se identifican los principales objetos de la empresa.
    b. Cada objeto principal tiene asociado un sistema de negocios.
3. Descomposición basada en la **estructura organizacional**
    a. Cada unidad del organigrama de la empresa puede ser considerado un subsistema de negocios.

## Descomposición por procesos del negocio

- Usando la cadena de valor de la empresa, cada proceso medular o de apoyo puede ser considerado como un subsistema de negocios.


## Descomposición por objetos del negocio

- Consiste en identificar las clases de objetos de negocios que son gestionados en la empresa.
- Cada objeto de negocios tiene asociado un sistema de negocios que se encarga de su gestión.



**Descomposición basada en la estructura organizacional**

- Usando el organigrama de la empresa, se definen como subsistemas aquellas unidades organizacionales que se deseen modelar como un sistema de negocios.

## Arquitectura de una Empresa

Los objetos o elementos que integran un sistema de negocios se agrupan en tres niveles diferentes:

- ✓ **Nivel de objetivos del negocio**
  - Agrupa aquellos objetos relacionados con el propósito de la empresa
- ✓ **Nivel de procesos del negocio**
  - Agrupa a todos aquellos elementos que están relacionados con los procesos que la empresa ejecuta para alcanzar sus objetivos
- ✓ **Nivel de sistemas de información y comunicación**
  - Agrupa a todos aquellos sistemas y tecnologías de información y comunicación que la empresa requiere para apoyar sus procesos

## Sistemas de Software

Un sistema de software es un sistema abstracto, hecho por el hombre, con la finalidad de ejecutar un conjunto de acciones en un computador

Un sistema de software consta de:

1. Un conjunto de programas interrelacionados
2. Un conjunto de cero, uno o más repositorios de datos
    a. Archivos, bases de datos, almacenes de datos, etc.
3. Un conjunto de documentos que describen:
    a. Cómo usar el sistema
        i. Ejemplo manuales o guías de uso
    b. Cómo el sistema está estructurado e implementado

i. Ejemplo modelos, manuales o guías de diseño y mantenimiento del software.

**Clasificación de los Sistemas de Software**

| Software de Aplicación | Software de Desarrollo | Software de Sistema |
|---|---|---|
| • Aplicaciones empresariales<br>• Aplicaciones Web<br>• Videojuegos<br>• Herramientas de productividad<br>• Sistemas ERP, BPM, CRM, etc. | • Ambientes de Desarrollo de software (IDE)<br>• Herramientas CASE<br>• Herramientas de programación<br>• DBMS, etc | • Sistemas operativos<br>• Compiladores<br>• Interpretadores<br>• Utilitarios<br>• Librerías de programas |

## Arquitectura de un Sistema de Software

Todo sistema de software tiene una estructura u organización denominada arquitectura del software

Una arquitectura de software consta de:

1. **Componentes**: elementos arquitectónicos, tales como:
   - Programas, funciones, servicios de software, módulos, subprogramas.
   - Bases de datos, archivos.
2. **Conectores**: relaciones entre los elementos arquitectónicos.
3. **Principios**: lineamientos que gobiernan el diseño y la lógica de las conexiones.

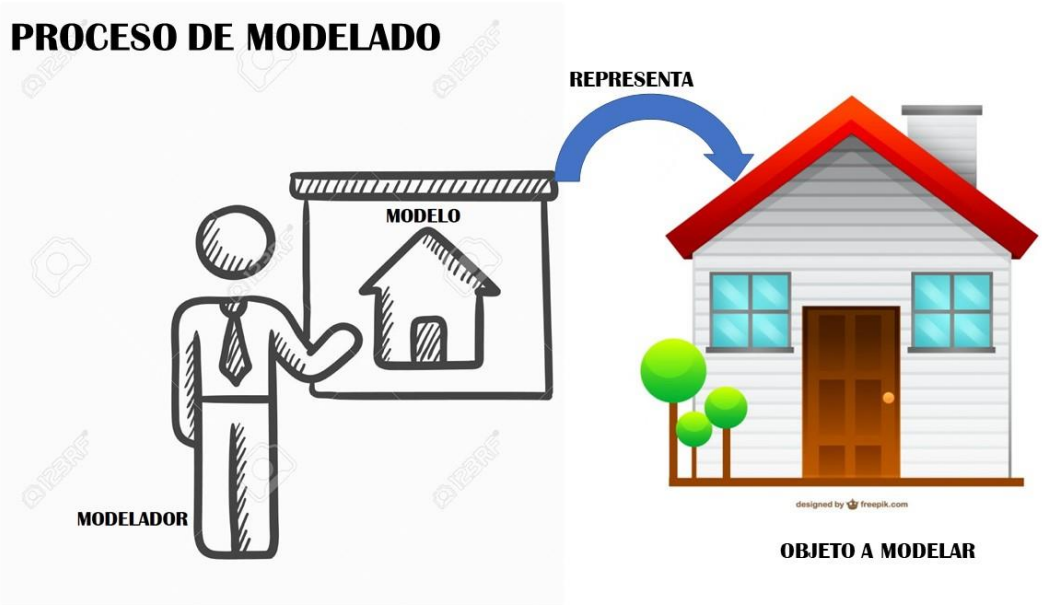
**Arquitectura = Componentes + Conectores + Principios**

# MODELADO

## Concepto de Modelado

**¿Qué es el modelado?** Acción y efecto de modelar. Modelar es el acto de "configurar o conformar" algo [RAE, 2001].

Es también, un proceso intelectual en el cual un sujeto (modelador) representa, a través de un modelo, ciertas características o cualidades de un objeto (cosa, fenómeno, hecho) o sistema




El modelado de sistemas de software es una técnica para tratar la complejidad inherente a los sistemas.

El uso de modelos ayuda al ingeniero de software a "visualizar" el sistema a construir. Los modelos permiten una mejor comunicación con el cliente.

Una condición fundamental del modelado:

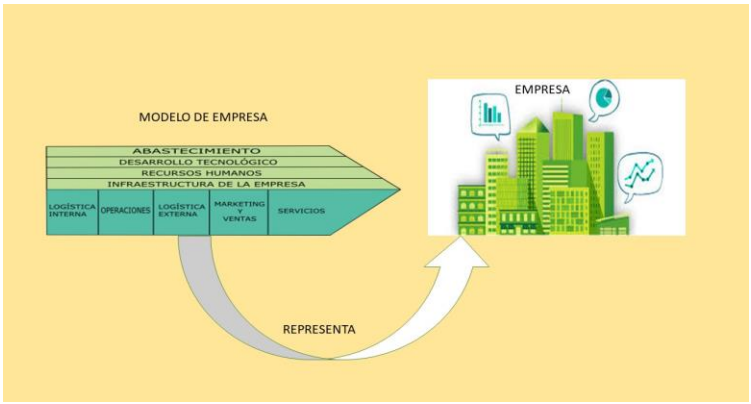
- Para modelar es necesario que el sujeto conozca el objeto o sistema que va a modelar
- No se puede modelar lo que no se conoce

Conocer es el acto mediante el cual un sujeto aprende (capta o concibe) las cualidades y relaciones (propiedades) que tiene un objeto o sistema


El resultado del proceso de modelado es un modelo

- Un modelo es una representación de un objeto (cosa, fenómeno, hecho o sistema).

El modelo es una simplificación del objeto o sistema modelado



# Modelado de Sistemas

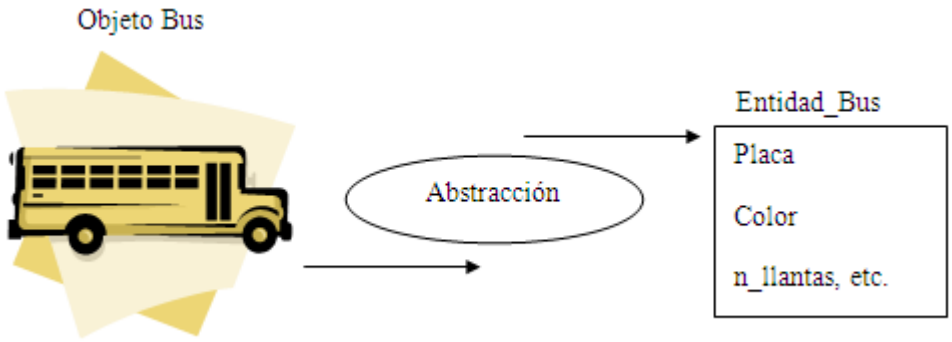
En el modelado de sistemas de negocios y/o software se emplean, fundamentalmente, tres procesos cognitivos:

- Abstracción
- Conceptualización
- Simbolización

## Abstracción

La abstracción es el proceso mental mediante el cual el modelador (sujeto):

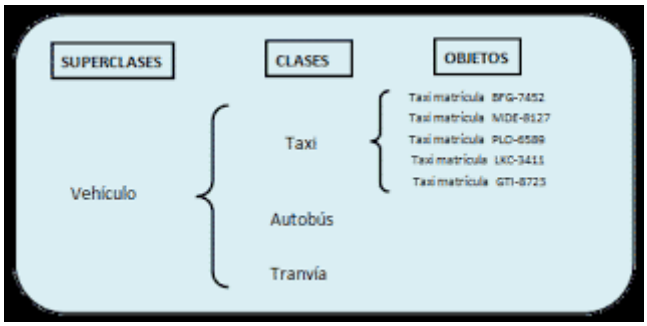
- Centra su atención en algunos aspectos o cualidades del objeto y
- Aisla o deja de lado otros que no le interesan.



## Conceptualización

La conceptualización es el proceso cognitivo de formación de conceptos.

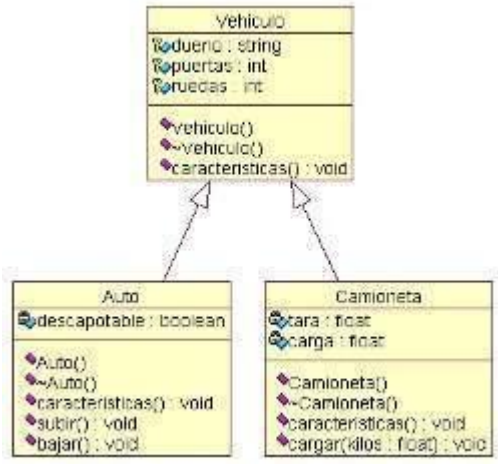
- Conocimiento que un sujeto tiene de un objeto.
- Un concepto es el conjunto de cualidades y relaciones (propiedades) que un sujeto le atribuye a un objeto.




## Simbolización

La simbolización es el proceso mediante el cual el sujeto designa (simboliza) el concepto que tiene de un objeto

El sujeto usa un lenguaje (sistema de signo) para designar (modelar) el concepto que él/ella tiene del objeto



## Modelado de Sistemas y el Lenguaje UML

El modelado de sistema requiere del uso de lenguajes artificiales y/o naturales que permitan representar y describir:

- El Sistema
- Sus partes
- Sus interacciones y
- Su ambiente

El lenguaje UML se ha convertido en la lengua adoptada del modelado de:

- Sistemas de software
- Sistemas de negocio

# MODELADO UML

El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Cada uno de ellos trabajaban en empresas diferentes en la década de los años 80 y principios de los años 90.

Cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los años noventa empezaron a intercambiar ideas entre sí y decidieron desarrollar su trabajo en conjunto.

1994 Rumbaugh ingresó a Rational Software Corporation, donde ya trabajaba Booch. 1995 ingresa Jacobson a Rational. Los anteproyectos del UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones.

Conforme diversos corporativos vieron que el UML era útil a sus propósitos, se conformó un consorcio del UML. Los primeros miembros fueron:

- DEC
- Hewlett-Packard
- Intellicorp
- Microsoft
- Oracle
- Texas Instruments
- Rational

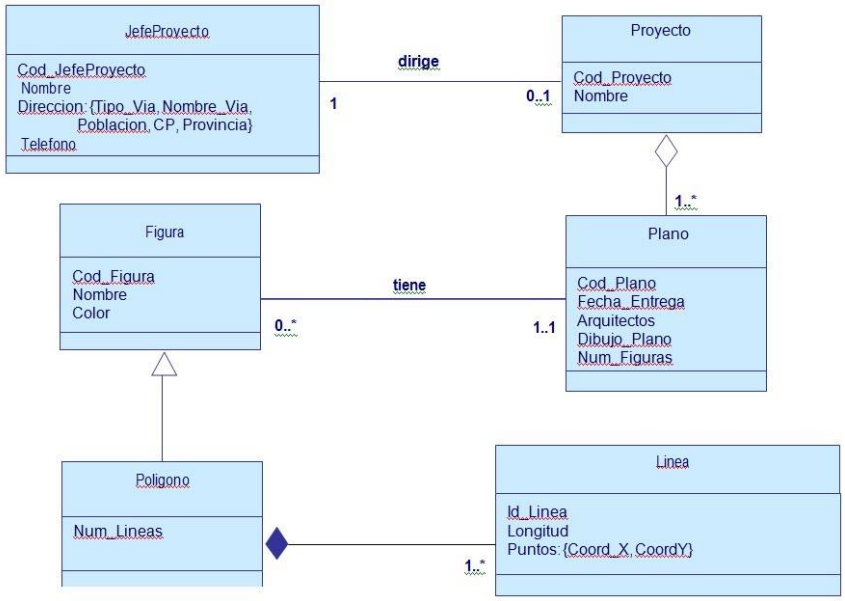
En 1997 el consorcio produjo la versión 1.0 del UML y lo puso a consideración del OMG (Grupo de Administración de Objetos) como respuesta a su propuesta para un lenguaje de modelado estándar.

En 1997 el consorcio produjo la versión 1.0 del UML y lo puso a consideración del OMG (Grupo de Administración de Objetos) como respuesta a su propuesta para un lenguaje de modelado estándar.

## Conceptos

### Qué es un modelo

Un modelo es la representación de los aspectos importantes de lo que se está modelando, desde cierto punto de vista, y simplifica u omite el resto.
La ingeniería, la arquitectura y muchos otros campos creativos usan modelos.
Un modelo de un sistema está construido en un lenguaje modelado, como UML.
El modelo tiene semántica y notación y puede adoptar varios formatos que incluyen textos y gráficos.



### Para qué Sirven los Modelos

1. Para captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento, de forma que todos los implicados puedan entenderlos y estar de acuerdo con ellos.
2. Para pensar del diseño de un sistema.
3. Para capturar decisiones del diseño de forma mutable a partir de los requisitos.
4. Para generar productos aprovechables para el trabajo.
5. Para organizar, encontrar, filtrar, recuperar, examinar y corregir la información en grandes sistemas.
6. Para explorar económicamente múltiples soluciones.
7. Para domesticar los sistemas complejos.

## Diagramas del UML

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas.

Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos

Modelo: permite presentar por medio de diagramas las diversas perspectivas de un sistema.

El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio.

Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

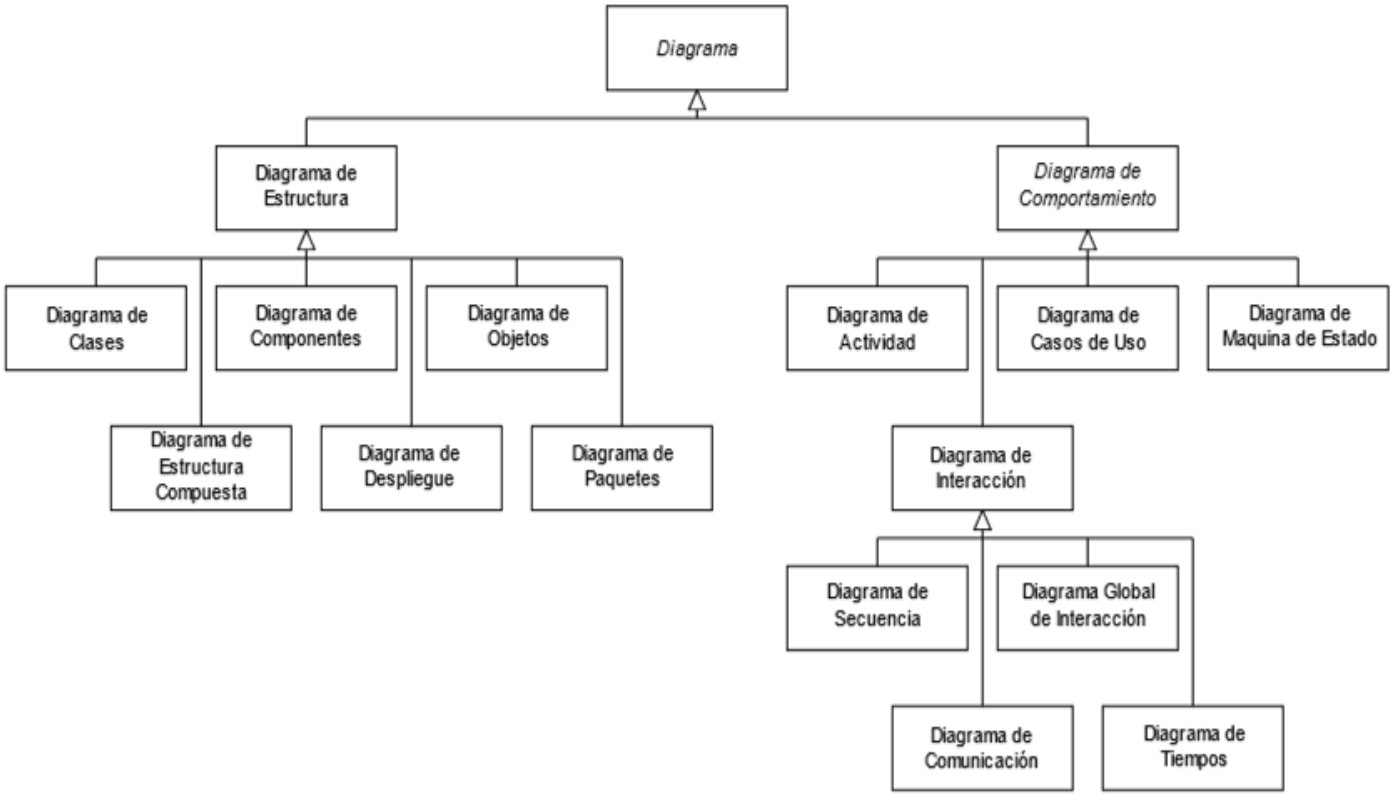
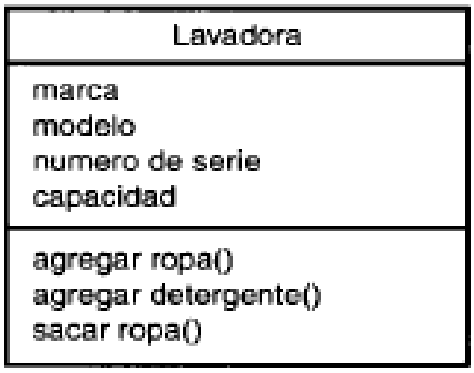


Diagrama
Diagrama de Estructura
Diagrama de Comportamiento
Diagrama de Clases
Diagrama de Componentes
Diagrama de Objetos
Diagrama de Estructura Compuesta
Diagrama de Despliegue
Diagrama de Paquetes
Diagrama de Actividad
Diagrama de Casos de Uso
Diagrama de Maquina de Estado
Diagrama de Interacción
Diagrama de Secuencia
Diagrama Global de Interacción
Diagrama de Comunicación
Diagrama de Tiempos

## Diagramas Estructurales

El modelo estructural representa el marco de trabajo del sistema y este marco de trabajo es el lugar donde existen los demás componentes.

### Diagrama de Clases




Los diagramas de clase representan la estructura estática en términos de clases y relaciones.

Utilizan clases de interfaz para capturar detalles sobre las entidades que constituyen su sistema y las relaciones estáticas entre ellas.

Los diagramas de clases son uno de los diagramas UML más utilizados en modelado y en la generación de código fuente en un lenguaje de programación.

Los diagramas de clase son, sin duda, el tipo de diagrama UML más utilizado.

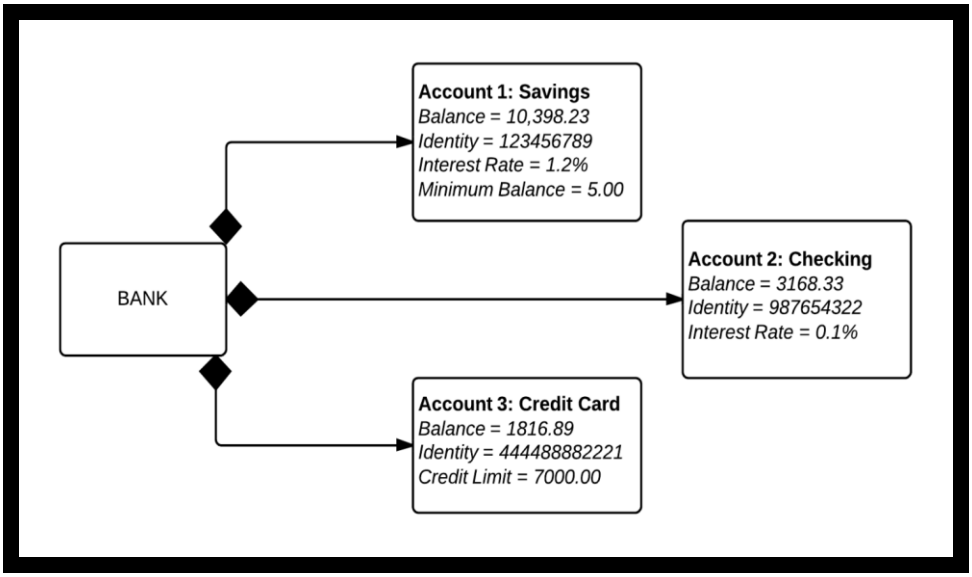
Es el bloque de construcción principal de cualquier solución orientada a objetos.

Muestra las clases en un sistema, atributos y operaciones de cada clase y la relación entre cada clase.

En la mayoría de las herramientas de modelado, una clase tiene tres partes, nombre en la parte superior, atributos en el centro y operaciones o métodos en la parte inferior.

En sistemas grandes con muchas clases relacionadas, las clases se agrupan para crear diagramas de clases.
Las Diferentes relaciones entre las clases se muestran por diferentes tipos de flechas.

## Diagrama de Objetos



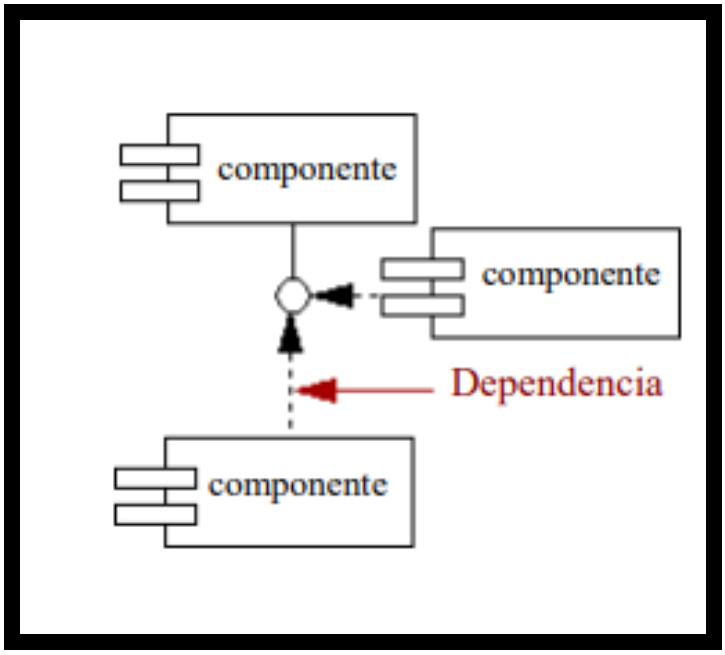

Un objeto es una instancia de clase.
Un diagrama de objetos es un gráfico de instancias, incluyendo objetos y datos.
Un diagrama de objetos es una instancia de un diagrama de clases; muestra una foto del estado de un sistema en un punto determinado.

Los diagramas de objetos están ligados a los diagramas de clase y comparten virtualmente los mismos símbolos para la notación.
Los diagramas de objetos, a veces denominados diagramas de instancia, son muy similares a los diagramas de clases.
Al igual que los diagramas de clases, también muestran la relación entre los objetos, pero usan ejemplos del mundo real.

Se utilizan para mostrar cómo se verá un sistema en un momento dado.

Debido a que hay datos disponibles en los objetos, a menudo se utilizan para explicar relaciones complejas entre objetos.

**Diagrama de Componentes**




El moderno desarrollo de software se realiza mediante componentes, lo que es particularmente importante en los procesos de desarrollo en equipo.

Un diagrama de componentes muestra la relación estructural de los componentes de un sistema de software.

Estos se utilizan principalmente cuando se trabaja con sistemas complejos que tienen muchos componentes.

Los componentes se comunican entre sí mediante interfaces.

Las interfaces se enlazan mediante conectores.

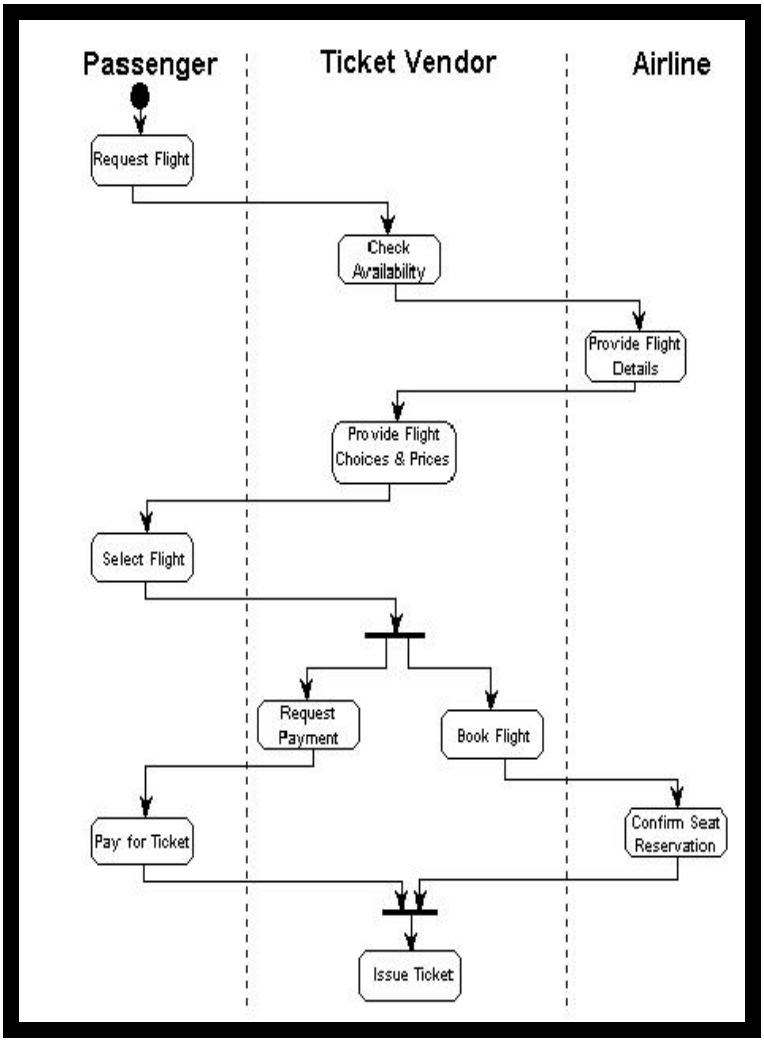
## Diagrama de Paquetes

Como su nombre indica, un diagrama de paquetes muestra las dependencias entre diferentes paquetes de un sistema.
Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones.
Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes.
Con estas líneas maestras sobre la mesa, los paquetes son buenos elementos de gestión.
Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

## Diagramas de Comportamiento

Los diagramas de comportamiento se centran en el comportamiento de los elementos de un sistema y describen la interacción en el sistema.

### Diagrama de Actividades

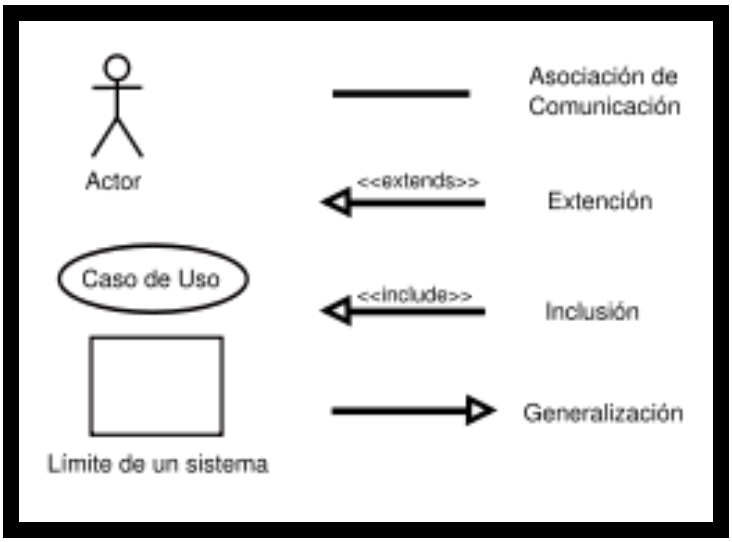



El Diagrama de Actividad es un diagrama de flujo del proceso multi-propósito que se usa para modelar el comportamiento del sistema.
Los diagramas de actividad se pueden usar para modelar un Caso de Uso, o una clase, o un método complicado.

Un diagrama de actividad es parecido a un diagrama de flujo; la diferencia clave es que los diagramas de actividad pueden mostrar procesado paralelo (parallel processing).

Esto es importante cuando se usan diagramas de actividad para modelar procesos 'bussiness' algunos de los cuales pueden actuar en paralelo, y para modelar varios hilos en los programas concurrentes.

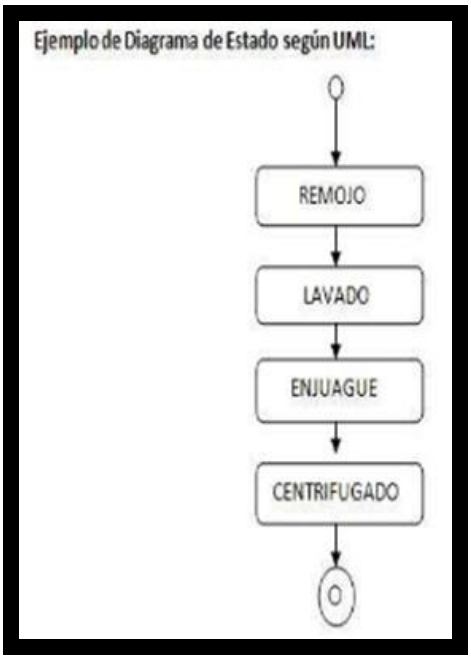
## Diagrama de Casos de Uso




Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario.

Los diagramas de casos de uso se utilizan para ilustrar los requisitos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

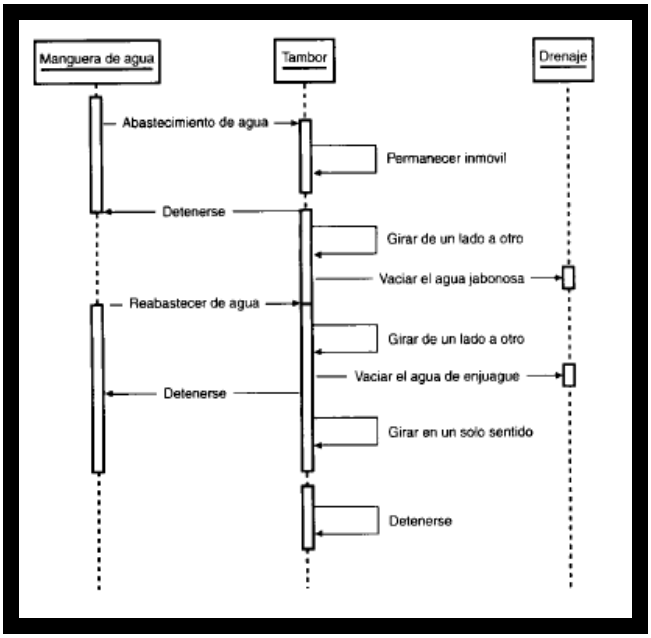
## Diagrama de Estados




En cualquier momento, un objeto se encuentra en un estado en particular.

Una lavadora podrá estar en la fase de remojo, lavado, enjuague, centrifugado o apagada.

## Diagrama de Secuencias




El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos.

Si vemos en el ejemplo de la lavadora la secuencia sería más o menos así:

1. El agua empezará a llenar el tambor mediante una manguera.
2. El tambor permanecerá inactivo durante cinco minutos.
3. La manguera dejará de abastecer agua.
4. El tambor girará de un lado a otro durante quince minutos.
5. El agua con jabón saldrá por el drenaje.
6. Comenzará nuevamente el abastecimiento de agua.
7. El tambor continuará girando.
8. El abastecimiento de agua se detendrá.
9. El agua del enjuague saldrá por el drenaje.
10. El tambor girará en una sola dirección y se incrementará su velocidad por cinco minutos.
11. El tambor dejará de girar y el proceso de lavado habrá finalizado.

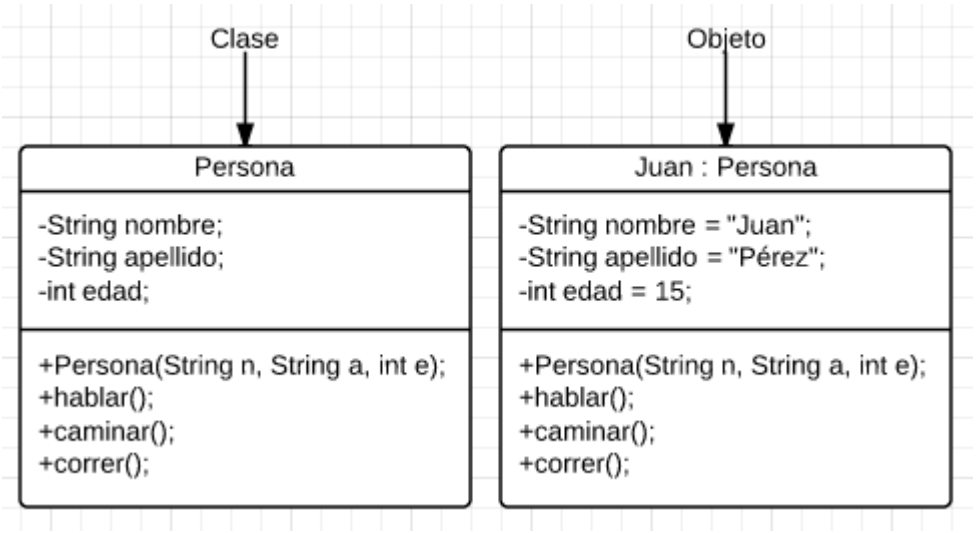
# PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos es un enfoque conceptual específico para diseñar programas, utilizando un lenguaje de programación orientada a objetos.
Las propiedades más importantes de la POO son:

1. Abstracción
2. Encapsulamiento y ocultación de datos
3. Polimorfismo
4. Herencia
5. Reusabilidad o reutilización de códigos

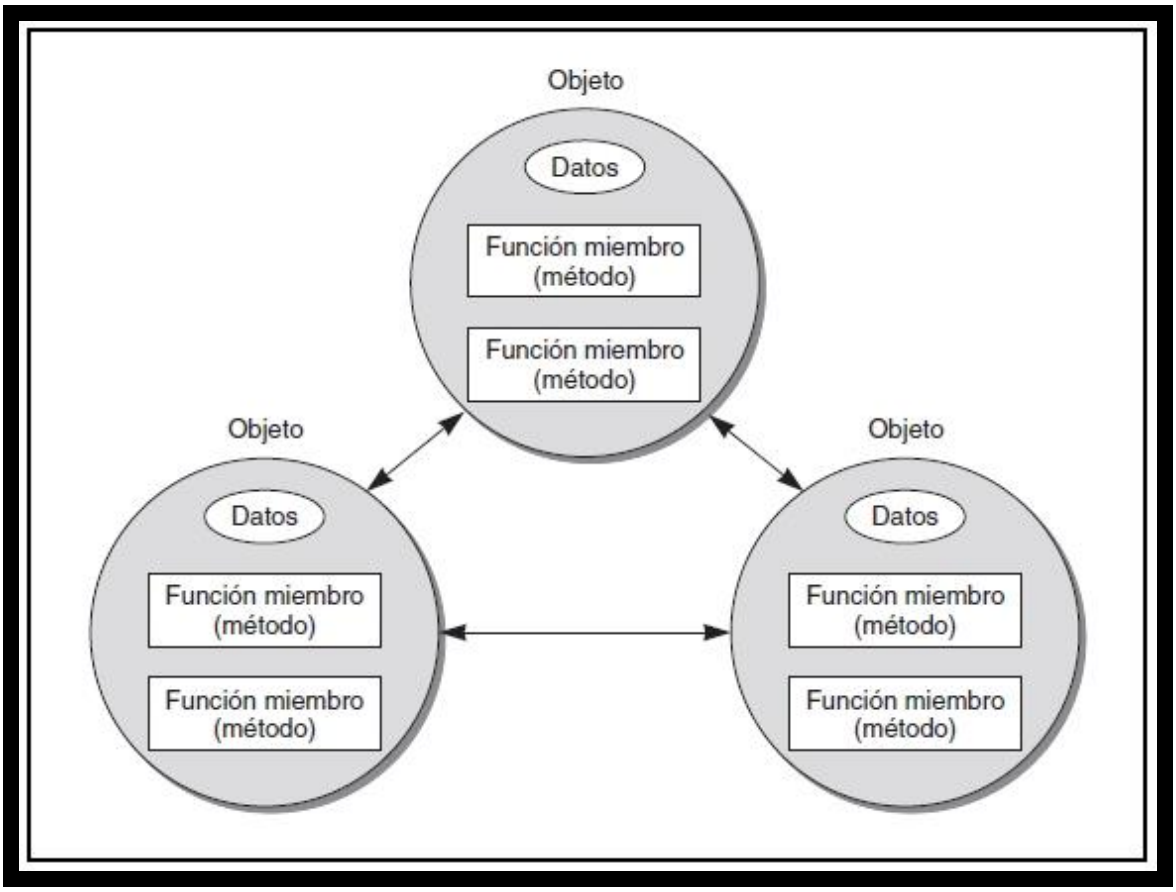
Elementos fundamentales:

1. Clases
2. Objetos



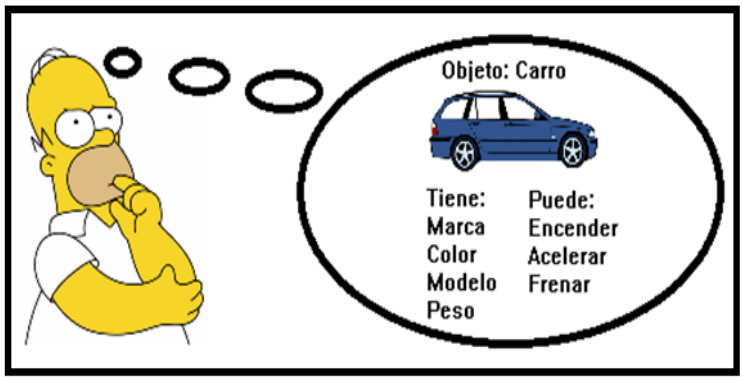

La POO se basa en el hecho de que se debe dividir el programa, no en tareas, sino en modelos de objetos físicos o simulados.
En la Programación Orientada a Objetos se combina (encapsula) en una sola unidad tanto los datos como las funciones que operan (manipulan) sobre los datos.

## Organización Típica de un Programa Orientada a Objetos




Etapas necesarias para modelar un sistema empleando orientación a objetos:

1. Identificación de los objetos del problema.
2. Agrupamiento en clases de los objetos con características y comportamiento comunes.
3. Identificación de los datos y operaciones de cada una de las clases.
4. Identificación de las relaciones existentes entre las diferentes clases del modelo.





Los objetos encapsulan datos y funciones miembro o métodos que manipulan los datos.

Las funciones miembros también se conocen como métodos.

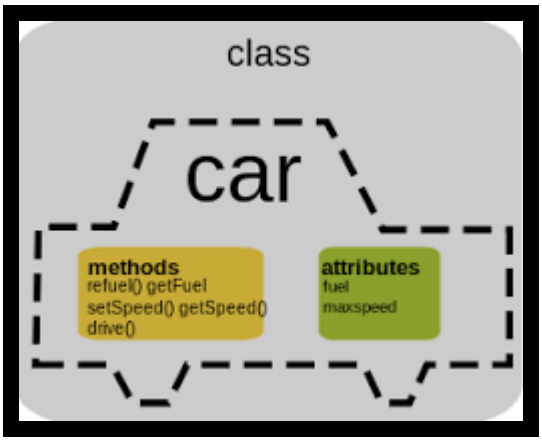
Los elementos datos de un objeto se conocen también como atributos o variables de instancia.

La llamada a una función miembro de un objeto se conoce también como envío de un mensaje al objeto.

## Objetos


Una forma de reducir la complejidad, es la abstracción. El **objeto** es el centro de la programación orientada a objetos. Un objeto es algo que se visualiza, se utiliza y que juega un papel o un rol. Un objeto no tiene que ser necesariamente algo concreto o tangible.

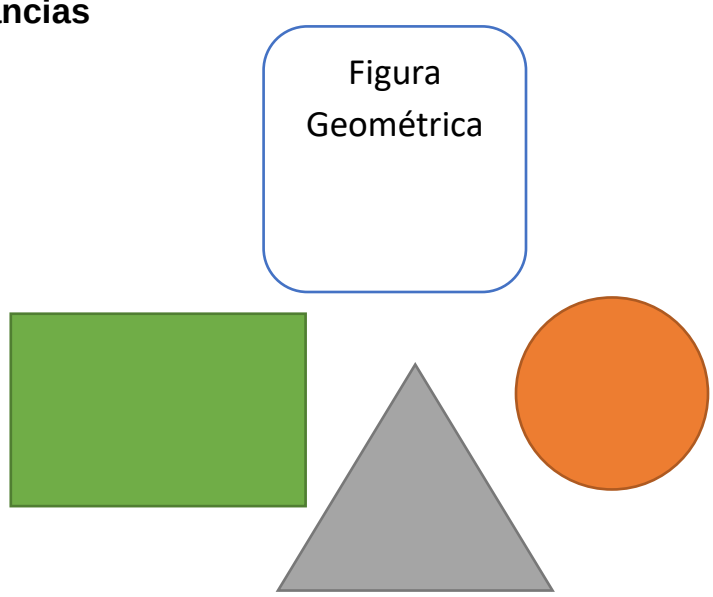
## Tipos abstractos de datos: Clases




Las clases con atributos y funciones miembro permiten gestionar los objetos dentro de los programas.

Una clase es la implementación de un tipo abstracto de dato y describe no solo los atributos (datos) de un objeto sino también sus operaciones (comportamiento)

**Instancias**




Una clase describe un objeto, en la práctica múltiples objetos.

Las instancias son la implementación de los objetos descritos en una clase. Estas instancias constan de los datos o atributos descritos en la clase y se pueden manipular con las operaciones definidas en la propia clase.

**Métodos**

Las operaciones definidas para los objetos se denominan métodos.

## Modelado e Identificación de Objetos

Un objeto en software es una entidad individual de un sistema que guarda una relación directa con los objetos del mundo real.

La correspondencia entre objetos de programación y objetos del mundo real es el resultado práctico de combiner atributos y operaciones, o datos y funciones.

Un objeto tiene:

1. Un estado,
2. Un comportamiento y
3. Una identidad.

## Estado

Conjunto de valores de todos los atributos de un objeto en un instante de tiempo específico. El estado de un objeto viene determinado por los valores que toman sus datos o atributos.

Estos valores han de cumplir siempre las restricciones que se hayan impuesto.

El estado de un objeto tiene un carácter dinámico que evoluciona con el tiempo, con independencia de que ciertos elementos del objeto puedan permanecer constantes.

## Comportamiento

Conjunto de operaciones que se pueden realizar sobre un objeto. Las operaciones pueden ser de observación, del estado interno del objeto, o bien de modificación de dicho estado.

El estado de un objeto puede evolucionar en función de la aplicación de sus operaciones.

Estas operaciones se realizan tras la recepción de un mensaje o estímulo externo enviado por otro objeto.

Las interacciones entre los objetos se representan mediante diagramas de objetos.

En UML se representarán por enlaces en ambas direcciones.

### Identidad

Permite diferenciar los objetos de modo no ambiguo independientemente de su estado.

Es posible distinguir dos objetos en los cuales todos sus atributos sean iguales.

Cada objeto posee su propia identidad de manera implícita.

Cada objeto ocupa su propia posición en la memoria de la computadora.

En un sistema orientado a objetos los programas se organizan en conjuntos finitos que contienen atributos (datos) y operaciones (funciones) y que se comunican entre sí mediante mensajes.

Los pasos típicos en el modelado de un sistema orientado a objetos son:

1. Identificar los objetos que forman parte del modelo.
2. Agrupar en clases todos aquellos objetos que tengan características y comportamientos comunes.
3. Identificar los atributos y las operaciones de cada clase.
4. Identificar las relaciones existentes entre las clases.

## Propiedades fundamentales de la orientación a objetos

Estos conceptos fundamentales son: abstracción, encapsulamiento y ocultación de datos, herencia, reutilización o reusabilidad y polimorfismo.

### Abstracción

La abstracción es la propiedad que considera los aspectos más significativos o notables de un problema y expresa una solución en esos términos.

### Encapsulamiento y ocultación de datos

La encapsulación o encapsulamiento es la reunión en una cierta estructura de todos los elementos que a un cierto nivel de abstracción se pueden

considerar pertenecientes a una misma entidad y es el proceso de agrupamiento de datos y operaciones relacionadas bajo una misma unidad de programación, lo que permite aumentar la cohesión de los componentes del sistema.

Los objetos que poseen las mismas características y comportamiento se agrupan en clases que no son más que unidades de programación que encapsulan datos y operaciones. La encapsulación oculta lo que hace un objeto de lo que hacen otros objetos del mundo exterior, por lo que se denomina también **ocultación de datos**.

## Herencia

La herencia modela el hecho de que los objetos tienden a organizarse en jerarquías. Esta jerarquía desde el punto de vista del modelado se denomina relación de generalización o es un (is-a). En programación orientada a objetos, la relación de generalización se denomina herencia.

# METODOLOGÍA PARA PROGRAMAR ORIENTADO A OBJETOS

Nos encontramos frente a la situación en donde debemos resolver el área de un rectángulo en base a la siguiente fórmula:

**Área = Base * Altura**

Cuando estamos frente a una situación como la descrita en el párrafo anterior debemos seguir una serie de pasos para construir un programa orientado a objetos:

1. Identificar la(s) clase(s) del problema.
2. Identificar los atributos de la(s) clase(s) identificadas en el punto I.
3. Identificar el(los) método(s) de la(s) clase(s).
4. Realizar el seudocódigo
5. Realizar la codificación utilizando Java.

Ahora pasaremos a realizar cada uno de los pasos para la situación que debemos resolver.

## I. Identificar la clase

| **AREA** |
|---|
| |

## II. Identificar los atributos

| **AREA** |
|---|
| base |
| altura |

## III. Identificar los métodos

| **AREA** |
| --- |
| - real base<br>- real altura |
| **+** Fijar_valor (real vbase, real valtura)<br><br>**+** real Calcular_area () |

## IV. Realizar el seudocódigo

```
/*Calcular el Área de un Rectángulo  */
clase Area {
/*Declaramos los datos de la clase */
privado real base, altura
/* Métodos de la clase  */
publico FijarValor(real vbase, real valtura){
        base = vbase
        altura = valtura}
publico real CalculaArea(){
real area
area = base * altura
retornar area}
}

INICIO
/* Se declaran las variables */
real vbase, valtura, area
/* Se crea el objeto de la clase */
Area ar

/*Solicitar los valores de entrada */
Escribir “Ingrese el valor de la base:”
Leer vbase
Escribir “Ingrese el valor de la altura:”
Leer valtura

/* Envio de mensaje */
ar.FijarValor(vbase, valtura)
area = ar.CalcularArea()

/*Escribir la salida*/
Escribir “El área del rectángulo es de:”, area
FIN
```

## V. Realizar la codificación utilizando Java

```
import java.util.Scanner;

public class Area {

    private double base;
    private double altura;

    public void FijarValor(double vbase,double valtura)
    {
            base =vbase;
            altura=valtura;
    }

    public double CalcularArea()
    {
            double area=0.0;
            area=base*altura;
            return area;
    }

    public static void main(String[] args)
    {
            double vbase=0, valtura=0, area=0;
            Scanner sc=new Scanner(System.in);
            Area ar=new Area();

            //Solicitar los valores

            System.out.println("Ingrese el valor de la base:");
            vbase=sc.nextDouble();
            System.out.println("Ingrese el valor de la altura:");
            valtura=sc.nextDouble();

            ar.FijarValor(vbase, valtura);
            area=ar.CalcularArea();

       System.out.printf("El area del rectangulo es de:%.2f",area);
    }
}
```