#POO(Programación Orientada a Objetos)

Es un metodo de implementación donde los programas se organizan como colecciones colaborativas de objetos, las cuales son instancias de alguna clase, las mismas que se representan en un diagrama jerarquico

#Características

Abstracción: Permite identificar caracteristicas esenciales de un objeto, ejemplos:

mochila:

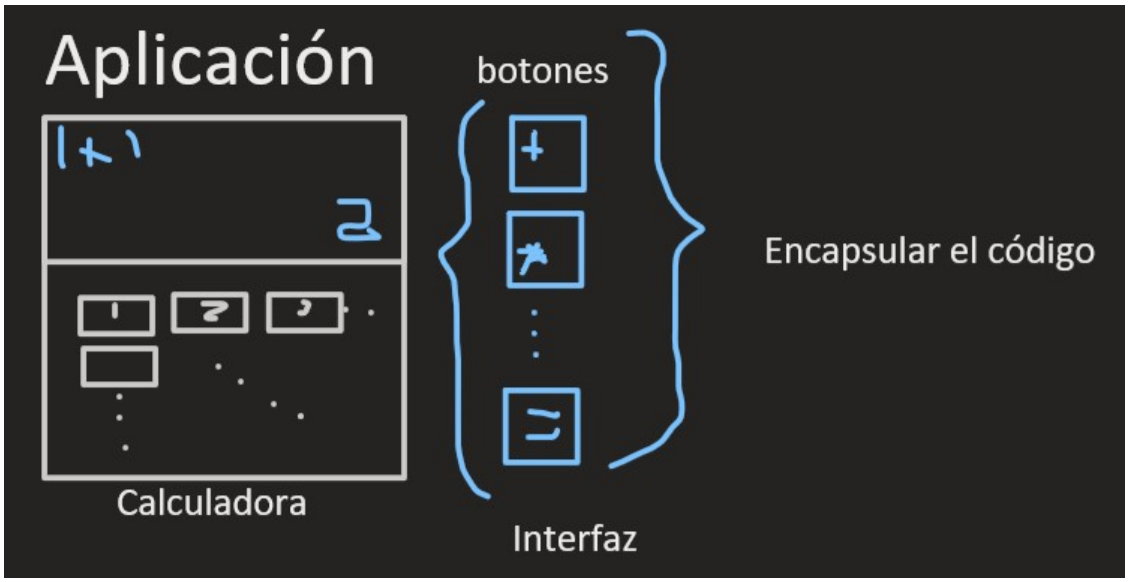
- Marca = "totto"
- Precio = 150
- Color = "Celeste"
- Capacidad = 5

Estudiante:

- Nombre = "Alan Brito"
- Edad = 18
- CI = 12345678
- RU = 189563
- Semestre = 3
- Record = 61.23
- AnioIngreso = 2021

Encapsulamiento

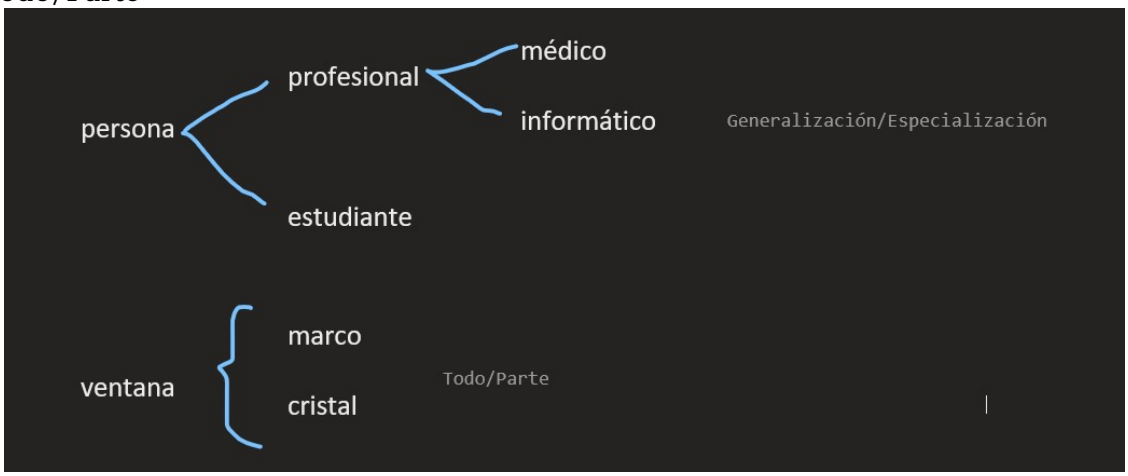
Es la manera de "ocultar" datos e información hacia el exterior



Jerarquia: Es la organización de las abstracciones, donde un objeto tiene prioridad al resto de los objetos.

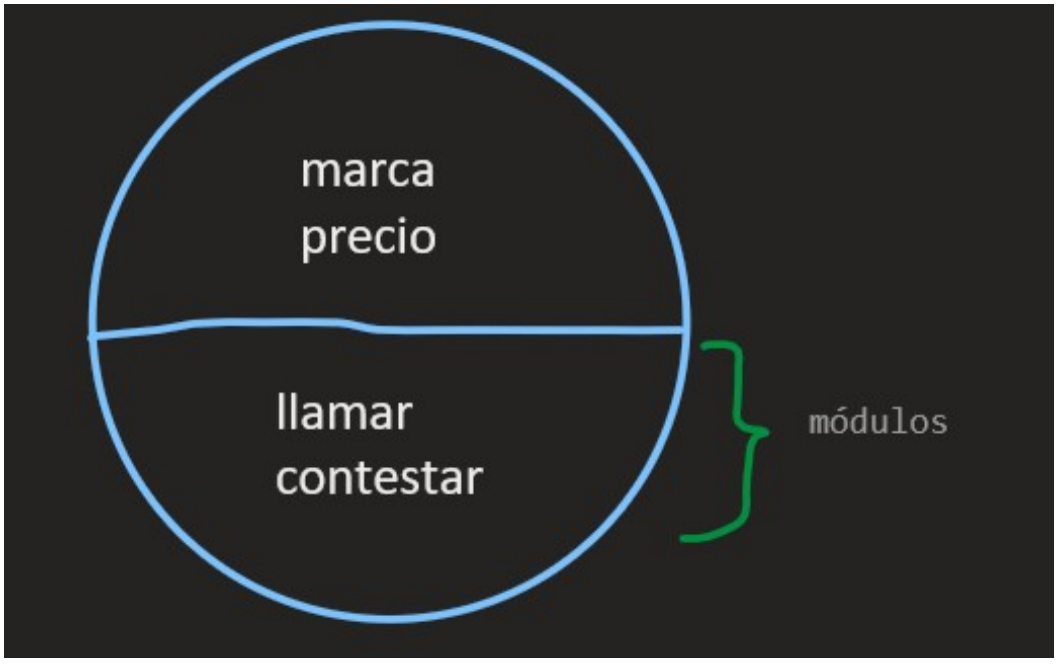
La organización se puede representar en un diagrama:

- Generalización/Especialización
- Todo/Parte

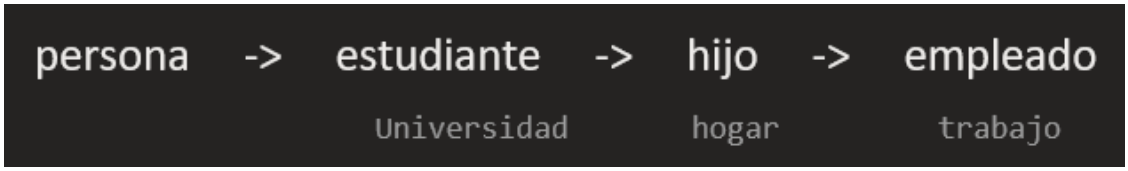


Modularidad: Consiste en subdividir el programa en partes mas pequeñas denominadas funciones o módulos, cada módulo debe ser independiente del resto de los módulos y del programa inicial.

Ejemplo: Objeto celular



Polimorfismo: Cuando un objeto u ente tiene varias maneras de comportamiento de acuerdo a su contexto.

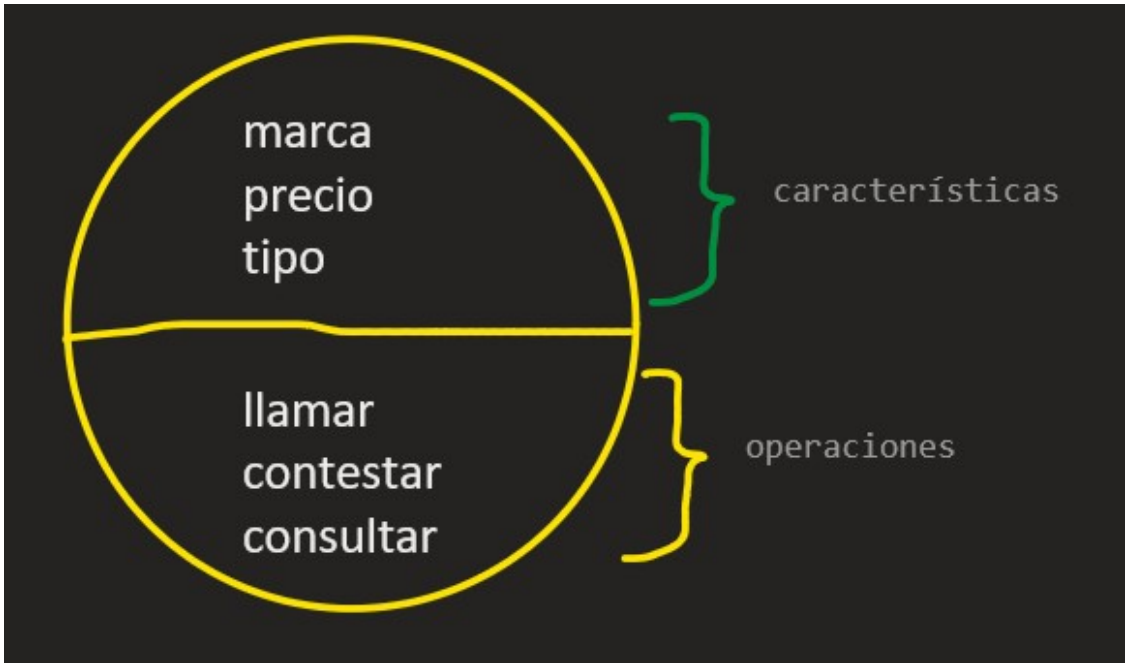


Conceptos

Objeto: Un objeto es la representación u abstracción de la realidad, el cual presenta ciertas caracteristicas y operaciones.

Un objeto es una instancia de una clase.

Ejemplo:



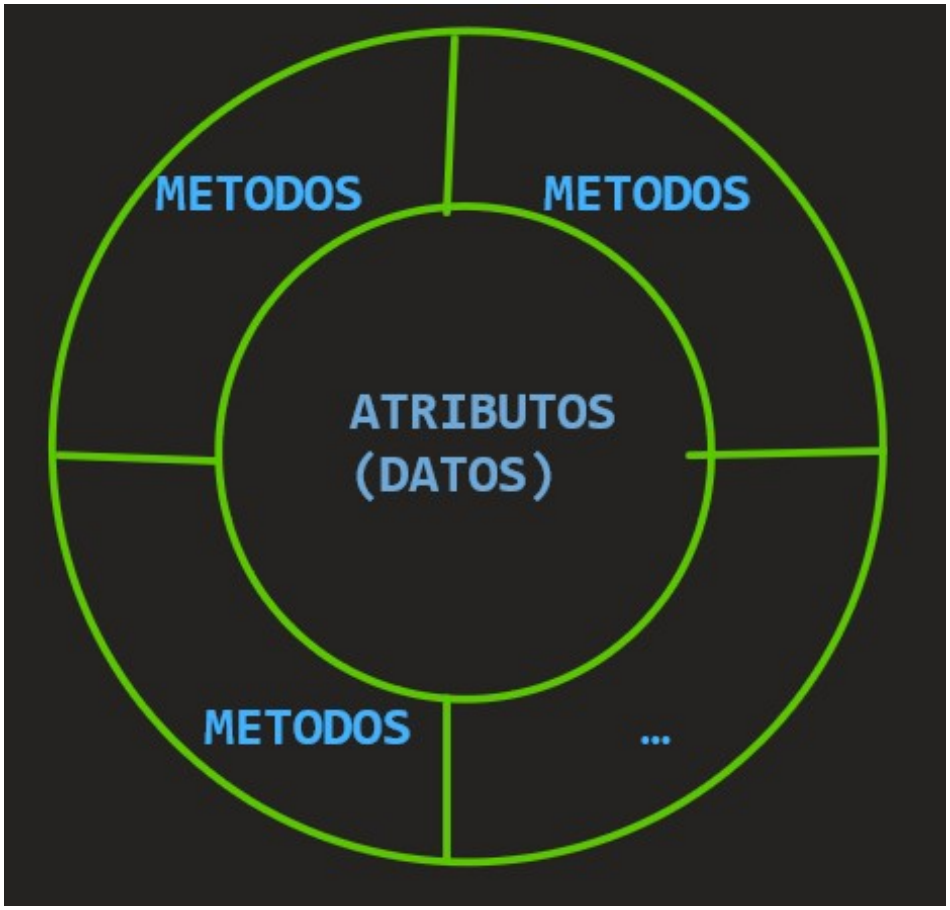
Tipos de objetos:

- Físicos o tangibles: Persona, vehiculo, marcador, etc.
- Intangibles: Aire, humo, etc.
- Incidentes: Accidente, etc.
- Roles: Estudiante, conductor, etc.
- Organizaciones: Universidad, posta médica, etc.
- Etc.

Clase: Una clase es un conjunto de objetos que tienen características en comun, en una clase se definen características(atributos) y operaciones(métodos).

A partir de una clase se van creando(instanciar) objetos.

Método: Un método o función es la operación que realiza un objeto, este método permite interactuar en un contexto, como ser objetos de otras clases, tambien es considerado como la interfaz de interacción.



Mensaje: Es una orden que se le da a un objeto para que se realice una determinada operación.

SeudoCodigo para representar una clase

```
class nomClase{
    tipoAccesoAtributos:
        tipoDato nomAtrib1;
        tipoDato nomAtrib2;
        ...
        ...
    tipoAccesoMetodos:
        nomMetodo1(){
            ...
            ...
        }
        nomMetodo2(tipoDato nomVariable, tipoDato nomVariable,...){
            ...
            ...
        }
        tipoDatoADevolver nomMetodo(tipoDato nomVariable,...){
            ...
            ...
            return valor;
        }
        ...
        ...
}
```

#Tipos de acceso

- **Público**: (+¿ , $public$) Se tiene acceso desde cualquier lugar del programa.
- **Privado**: (− , $private$) Se tiene acceso solamente desde sus propios métodos.
- **Protegido**: (¿ , $protected$) Se tiene acceso solamente desde sus propios métodos y desde sus clases hijas.

Contructores y destructores

Constructor: Es un metodo cuyo objetivo y funcion es la de crear un objeto con ciertos valores/datos

- **Contructor por defecto**: Crear un objeto con datos iniciales
- **Constructor con parametros**: Crear un objeto con datos proporcionados como parametros
- **Sintaxis**:

```
class nomClase{
    tipoAccesoAtributos:
        tipoDato nomAtrib1;
        tipoDato nomAtrib2;
        ...
        ...
    tipoAccesoMetodos:
        //contructor por defecto
        nomClase(){
           atrib1 <- valor1;
           atrib2 <- valor2;
           ...
           ...
        }
        //constructor con parametros
        nomClase(tipoDato p1, tipoDato p2, ...){
           atrib1 <- p1;
           atrib2 <- p2r;
           ...
           ...
        }
        ...
        ...
}
```

- **Características**:

  El nombre del constructor es el mismo que de la clase Una clase puede tener cero o mas constructores.

Destructor: Es un método que tiene por objetivo destruir al objeto creado(libera la memoria asignada al objeto).

Un destructor de caracteriza por:

- Tener el mismo nombre de la clase
- Solo existe un solo destructor en la clase
- No retorna ningún valor
- Se ejecuta(automáticamente) de manera independiente finalizado el programa.
- No lleva parametros

```
class NomClase(){
    ...
    ...
    public:
        ...
```

```
        ~nomClase(){
            print("objeto destruido");
        }
}
```