TEMA 2 - ORIENTACIÓN A OBJETOS

Paradigma de programación que considera las aplicaciones como un conjunto de objetos que interaccionan.

Intento de mejorar el proceso de construcción y mantenimiento de aplicaciones.

Origen en los años 60 : Smalltalk

Conceptos fundamentales:

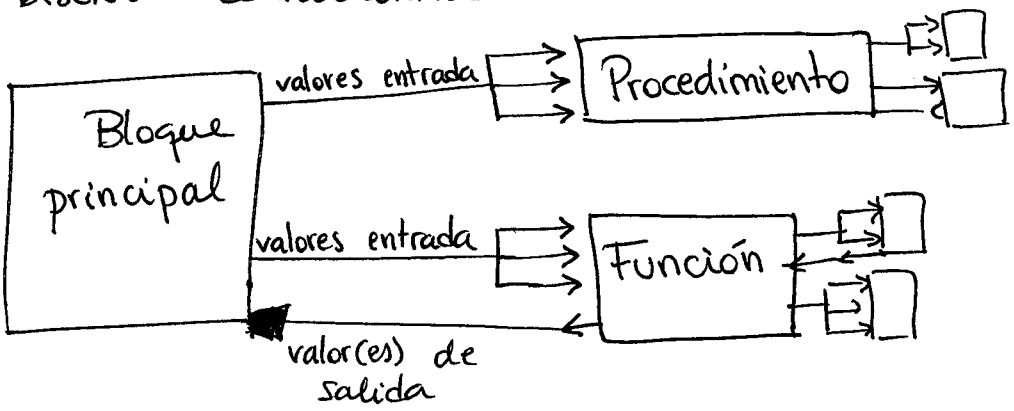
- clase: "plantilla" que describe los datos y comportamiento de un conjunto de objetos.
- objeto: instancia en tiempo de ejecución de una clase.
- encapsulación: ocultación de información. Mostrar la interfaz del objeto
- polimorfismo de tipos: refinamiento/generalización, herencia de tipos. Poder usar de manera segura un objeto especialización en lugar del objeto más general.

Ventajas:
- modela conceptos del mundo real de manera natural.
- extensibilidad de los diseños:
  - mediante herencia: añadir nuevas clases, extender el comportamiento de métodos.
  - mediante encapsulamiento: el usuario de una clase no ve detalles innecesarios.
- potencia la reutilización.

PROGRAMACIÓN ESTRUCTURADA

- Esquema de programación procedimental, propia de lenguajes como Pascal o C.
- Separación de algoritmos y estructuras de datos.
- Programas: llamadas entre procedimientos. Diseño "top-down".

DISEÑO ESTRUCTURADO



ABSTRACCIÓN DE OPERACIONES

- Estructura de un módulo:
  - datos de entrada
  - datos de salida
  - descripción funcional
- Sintaxis del lenguaje
  - organización del código en bloques de instrucciones. Definición de funciones y procedimientos.
  - Extensión del lenguaje con nuevas operaciones llamadas a nuevas funciones y procedimientos.

VENTAJAS

- Facilita el desarrollo
  - Se evita la repetición del trabajo
  - trabajo de programación dividido en módulos.
  - diseño top-down: descomposición en subproblemas
- Facilita el mantenimiento.
  - claridad del código.
  - independencia de los módulos.
- Favorece la reutilización.

TIPOS ABSTRACTOS DE DATOS

Consisten en → estructura de datos: almacena información para representar un determinado concepto.

→ funcionalidad: conjunto de operaciones que se pueden realizar sobre el tipo de datos.

Sintaxis del lenguaje → módulos asociados a tipos de datos.

→ no introduce necesariamente variaciones respecto a la programación modular.

PROGRAMACIÓN ORIENTADA A OBJETOS

POO = soporte sintáctico para los tipos abstractos de datos
+
prestaciones asociadas a las jerarquías de clases
+
cambio de perspectiva

ELEMENTOS POO

- ▷ Objetos: atributos + métodos
- ▷ Métodos: operaciones sobre los objetos
- ▷ Clases: categorías de objetos con propiedades y operaciones comunes.
- ▷ Herencia: Jerarquía de clases.
- ▷ Relaciones: entre objetos. Objetos compuestos.

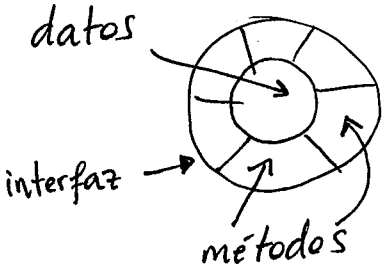
ENCAPSULAMIENTO

Podemos controlar el acceso a los atributos y métodos de una clase desde el exterior:

- Elementos privados (-): no accesibles ni visibles desde el exterior. Un método privado no se puede invocar desde un objeto de tipo distinto.
- Elementos públicos (+): accesibles desde el exterior. Un método público se puede invocar desde un objeto distinto.
- Elementos protegidos (#): accesibles solo desde la clase y subclas

ENCAPSULAMIENTO: sólo exponemos la interfaz relevante al resto del sistema.

OCULTACIÓN DE INFORMACIÓN: facilita el diseño, lo hace más simple y extensible.



→ Los datos están protegidos (no visibles)
→ El acceso al estado del objeto es a través de los métodos de la interfaz.