Java Básico - Avanzado

Contenido 4 Uso de Objetos

Contenido

- La Programación Orientada a Objetos.
- Instanciación y uso de objetos.
- Estructura y componentes de una clase:
  - Variables de instancia
  - Constructores
  - Métodos
    - Sin retorno (void)
    - Con retorno
- Notación y referencias.
- Particularidades de los Parámetros.
- Arreglos de Objetos.
- Sobrecarga de constructores y métodos.
- Particularidades de la clase String.

Sección 1

Uso de Objetos

Datos y código

- Todo programa está formado por 2 elementos: datos y código.
- Los datos es lo que se desea almacenar y procesar. Corresponden a la memoria.
- El código corresponde a las instrucciones que definen lo que se quiere hacer con esos datos. De esta manera definen el comportamiento.

Datos y código

- Todo programa debe organizar estos dos elementos.
- Existen 2 enfoques (o paradigmas) para hacerlo:
  - Programación basada en procesos.
  - Programación basada en objetos.

Programación basada en procesos

- El programa se organiza en base a las operaciones que se quieren realizar.
- Estas operaciones se organizan en grupos llamados procedimientos.
- Entonces el programa es un conjunto de procedimientos.

Programación basada en procesos

Problemas:

- Los datos quedan dispersos en el programa.
- Cualquier procedimiento puede hacer cualquier cosa con cualquier dato.
- En problemas muy grandes, este enfoque se vuelve caótico.

Programación basada en objetos

- El programa se organiza en base a los datos que se quieren almacenar y procesar.
- Estos datos se organizan en grupos, y para cada grupo se definen las operaciones que se quieren realizar con esos datos.
- Ese conjunto de datos y sus operaciones conforman un objeto.

Programación basada en objetos

- Entonces el programa es un conjunto de objetos que interactúan.

Entonces:

- Los datos quedan organizados en el programa.
- Una operación no puede hacer cualquier cosa con cualquier dato. Sólo puede hacerlo con los datos de su objeto.
- En problemas muy grandes, este enfoque permite reducir complejidad.

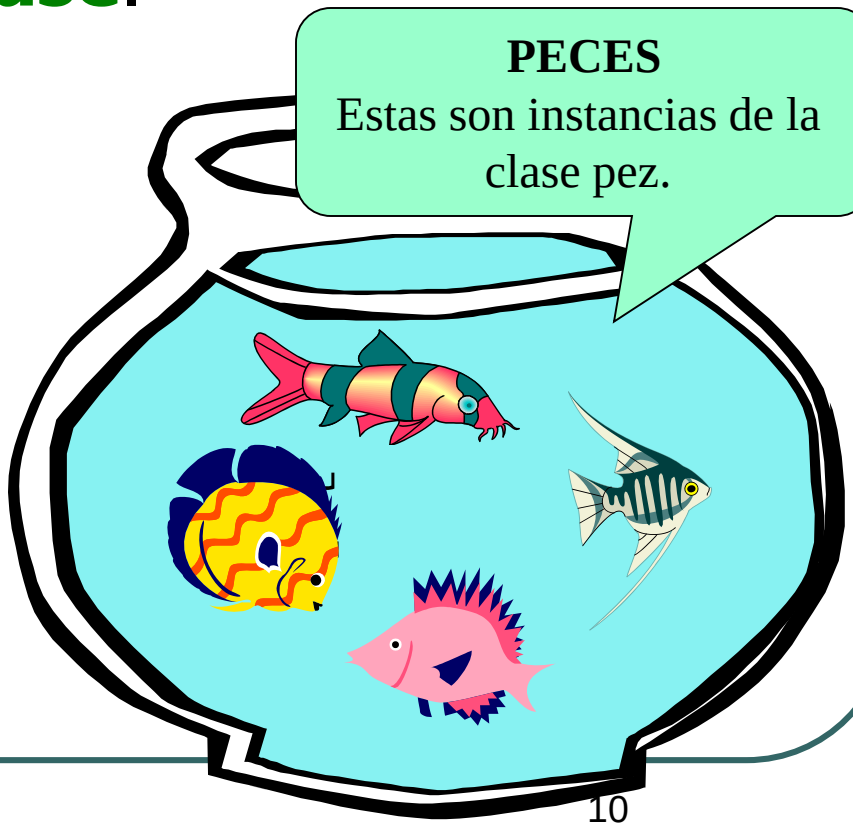
Clases y objetos

Una clase es un tipo al cual pertenecen **objetos** o **instancias de la clase**.

Pez

Clase PEZ
Los objetos de esta clase tienen **color, tamaño** y tienen la capacidad de **respirar bajo el agua**, **nadar** y **alimentarse**.

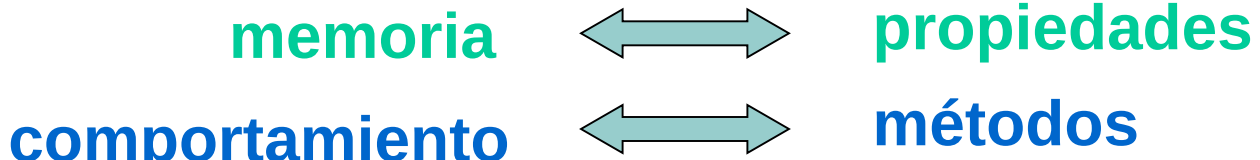
PECES
Estas son instancias de la clase pez.



Clases y objetos de software

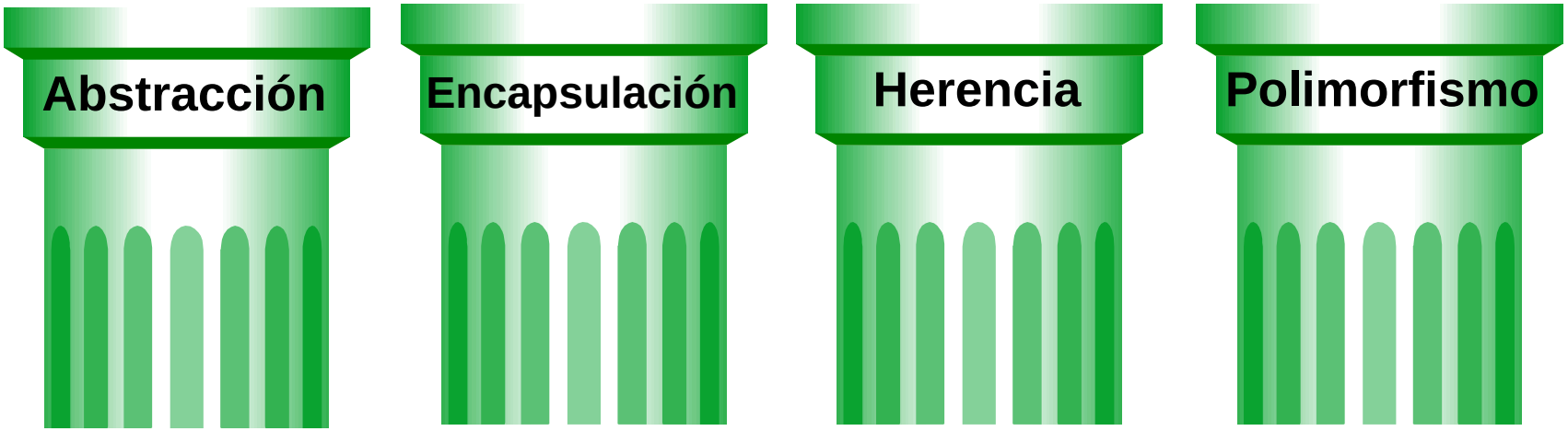
Una clase es una unidad de software que posee **memoria** y **comportamiento**.

Una clase es el "*plano*" que permite "*construir*" un objeto: define sus **propiedades** (datos que almacena el objeto) y el código de sus **métodos** (comportamiento).



Fundamentos de la P.O.O.

- La POO (Programación Orientada a Objetos) se basa en cuatro conceptos:



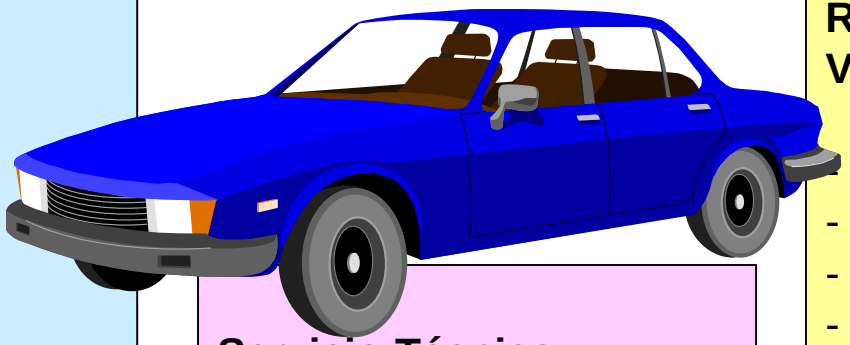
Abstracción

- Es el proceso de reconocer y seleccionar las características esenciales de los entes que forman parte de un sistema bajo estudio.
- Reduce la complejidad del problema real, permitiendo su manejo.
- Aplicada a la POO, la abstracción permite reconocer las propiedades de las clases de un software en desarrollo.

Abstracción (cont.)

Sistema de Ventas Casa Comercial

- Marca
- Modelo
- Nº Motor
- Nº chasis
- Color
- Equipamiento
- Valor de venta
- Fecha de venta
- Comprador
- Vendedor



Servicio Técnico:

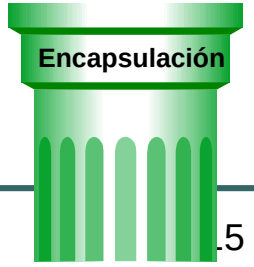
- Propietario
- Marca
- Modelo
- Patente
- Fecha ingreso al taller
- Fecha salida de taller
- Kilometraje vehículo
- Tipo de falla

Registro Nacional de Vehiculos Motorizados:

- Propietario
- Marca
- Modelo
- Patente
- Nº Motor
- Nº chasis
- Color

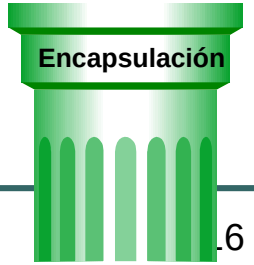
Encapsulación

- Mecanismo que permite juntar el código y los datos que maneja en una sola estructura: la clase.
- Se forma como un envoltorio protector que mantiene al código y datos alejados de posibles interferencias o usos indebidos.



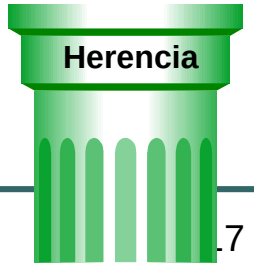
Encapsulación

- El acceso al código y a los datos se realiza de forma controlada a través de una interfaz bien definida.



Herencia

- Proceso en el cual un objeto adquiere las propiedades de otro.
- La herencia permite crear cada vez clases más especializadas (y complejas) a partir de las clases anteriores.
- Entonces, se forma una clasificación jerárquica de clases.

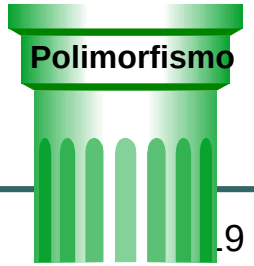


Polimorfismo

- “Muchas formas”, en griego.
- Permite que la misma interfaz se utilice para una clase general de acción. Entonces, el compilador selecciona la acción específica que se debe aplicar a cada situación.

Polimorfismo

- Con esto se reduce la complejidad.
- Se resume con la frase “Una interfaz, varios métodos”.



Posibilidades de la P.O.O.

- Usar clases previamente implementadas.

  Ejemplos:

  BufferedReader

  String

- Definir e implementar nuevas clases.

  Ejemplos:

  Persona

  Lista

  Auto