UTN F r a
Avellaneda

Programación II

Clase 04

Programación Orientada a Objetos

Temario

Resumen POO

- Características
- Objetos
- Clases
- Abstracción
- Encapsulación
- Herencia

UML

- Diseño
- Sintaxis

Polimorfismo

- Definiciones
- Tipos
- Interfaces y Clases Abstractas

01.

Resumen POO

Resumen POO

Características

- **Abstracción**

  Características principales.

- **Encapsulación**

  Visibilizar datos no implementaciones.

- **Herencia**

  Jerarquía.

- **Polimorfismo**

UTNFra Avellaneda

Abstracción

- **Objetos**

  Modelo reducido de la realidad.

- **Clases**

  Plantilla. Definición de comportamiento y estructura.

  Abstractas.
  Estáticas.
  Concretas.

Abstracción

- **Objetos**

  Modelo reducido de la realidad.

- **Clases**

  Plantilla. Definición de comportamiento y estructura.

  Abstractas.
  Estáticas.
  Concretas.

Encapsulación

- **Seguridad**

  Imponer la utilización de un mecanismo que permita manipular estados.

  MÉTODOS PÚBLICOS

- **Ocultar**

  No exponer estados directamente.
  No revelar detalles de la implementación.

UTN F r a Avellaneda

Herencia

- **Clases**

  Implementar jerarquías.
  Obtener estructura y comportamiento de otra clase.
  Reutilización de código.

02.
Polimorfismo

Polimorfismo

Definiciones

- **Estático**

  El compilador elige cómo responder el método adecuado con el que un objeto responde un mensaje.

- **Dinámico**

  Capacidad que tienen dos o más objetos de responder el mismo mensaje cada uno a su manera.

- **Código flexible**

  Es compatible con distintos objetos.
  Existentes y futuros (especializados).

BENEFICIA A LOS

OBJETOS
USUARIOS

Polimorfismo

Estático

- **Sobrecarga de métodos y constructores**

  Métodos con el mismo nombre.

  Se diferencian por:

  Cantidad de parámetros.
  Tipo de los parámetros.
  Orden de los parámetros.

Polimorfismo

Dinámico

- **Herencia**

  El polimorfismo es inherente en la presencia de **herencia**.

- **Interfaces**

  Se subscribe a un comportamiento común.

Polimorfismo

Interfaces

- **Agrupa métodos**

  Define comportamiento bajo un nombre **[tipo]**.
  Los métodos no tienen implementación **[firma]**.

- **Una clase puede implementar múltiples interfaces**

  Las clases que implementen una interfaz tiene que definir las implementaciones de los métodos.
  Es como un **contrato**.

- **No deben crecer**

Polimorfismo

Interfaces vs Clases Abstractas

SIMILITUDES

No pueden instanciarse objetos.

No poseen constructor.

DIFERENCIAS

Los métodos son por defecto abstractos y públicos.

No poseen atributos.

Todos los métodos de la clase son **override.**

Polimorfismo

Interfaces vs Clases Abstractas

- **¿Cuándo usar Interfaces?**

  Clase abstracta **[herencia]** implica jerarquía.

  Polimorfismo en cierto contexto.

  Se requiere polimorfismo pero que tiene sentido
  que esté incluido en la jerarquía ya que no comparte otra cosa que la necesidad de ser polimórfico en un escenario particular.

02.

UML

Unified Modeling Language

- **¿Qué es?**

  Lenguaje gráfico que permite definir diseños de software, particularmente orientado a objetos.

  Sintaxis propia.

  GRÁFICOS
  SÍMBOLOS

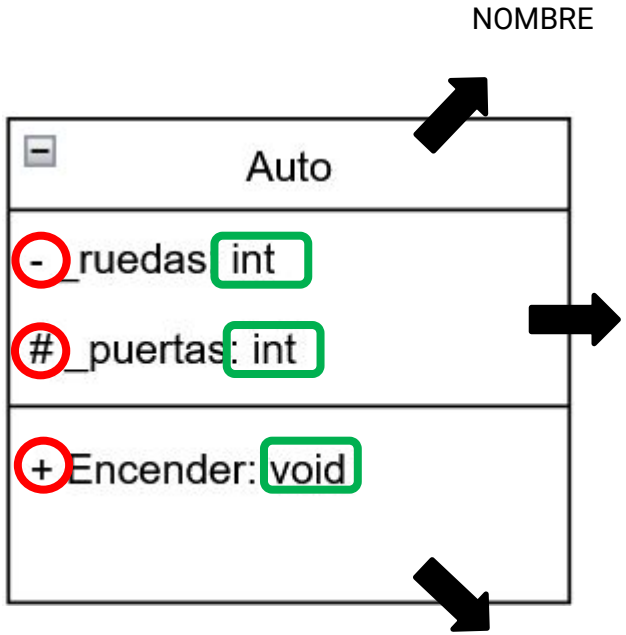
- **Usos**

  Diseño de soluciones.
  Diagramas de clases.

  - ESTRUCTURA
  - COMPORTAMIENTO
  - RELACIONES
  - COLABORACIONES
  - DEPENDENCIAS

UML

Clases Concretas



Definición de acceso

Tipo

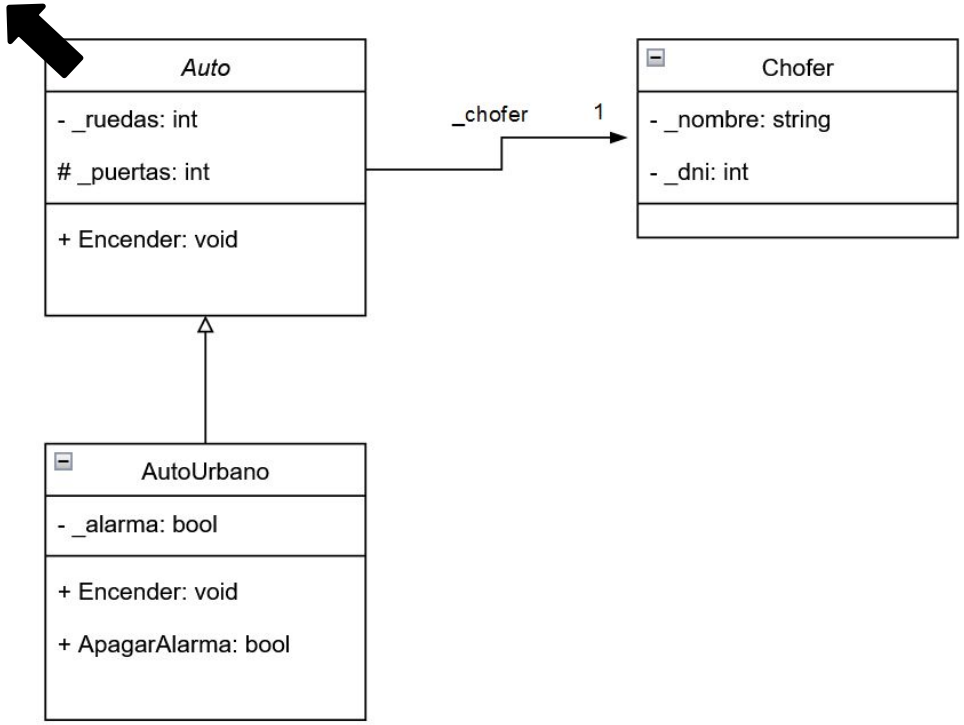
- PRIVADO
+ PUBLIC
# PROTECTED

Relaciones entre Clases

CLASE ABSTRACTA

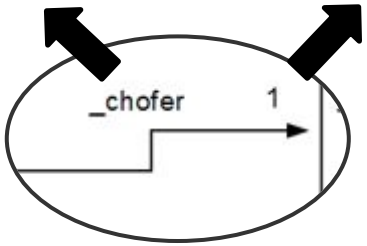
HERENCIA

COLABORACIÓN



NOMBRE ATRIBUTO

CARDINALIDAD



UML

Interfaces