UTN F r a
Avellaneda

Programación II

Clase 03

Programación Orientada a Objetos

Temario

Resumen POO

- Características
- Objetos
- Clases
- Abstracción

Herencia

- Definición
- Tipos
- Encapsulamiento

Encapsulamiento

- ¿Qué es?
- Objetivo
- ¿Cómo se implementa?
- Propiedades

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.

02.

Encapsulamiento

Encapsulamiento

¿Qué implica?

- **Objeto**

  Posee estados e información [datos].

- **Ocultar información**

  Como mecanismo de protección ante uso incorrecto.

CONTRARIA A LA LÓGICA DEL NEGOCIO

- **Ofrecer servicios**

  Exponer información a través de métodos.

  No revelar los detalles de la implementación ni permite modificarla.

Encapsulamiento

DATOS

CONSUMO_KM: 0.1

NIVEL_NAFTA: 24

IMPLEMENTACIÓN

RECORRER_KM(KM)

NIVEL_NAFTA

= NIVEL_NAFTA – KM * CONSUMO_KM

NIVEL_NAFTA()

NIVEL_NAFTA

ENCAPSULAMIENTO

RECORRER_KM(10)

NIVEL_NAFTA()

23

Encapsulamiento

Responsabilidad

- **El objeto es responsable de sí mismo**

  Administra sus estados.

  Define la interfaz.

Acoplamiento

- **Dependencia entre objetos**

  Mayor acople, más compartida la responsabilidad.

Encapsulamiento

Buenas prácticas

- **No exponer estados**

  Sólo a través de propiedades.

GETTER
SETTER

- **Exponer interfaz**

  Métodos para realizar acciones.

- **Minimizar acople**

Encapsulamiento

¿Cómo encapsular?

- **public**

  Accesible para todas las clases.

- **private**

  Accesible dentro de la misma clase.

- **protected**

  Accesible dentro de la misma clase y subclases.

- **internal**

  Accesible dentro del mismo proyecto

Encapsulamiento

UTNFra Avellaneda

Propiedades

- **GETTER**

  Método para obtener un estado.

  Lectura.

- **SETTER**

  Método para configurar un estado.

  Escritura.

03.

Herencia

Herencia

¿Qué es?

ATRIBUTOS MÉTODOS

- **Heredar**

  Obtener estructura y comportamiento de otra **clase**.

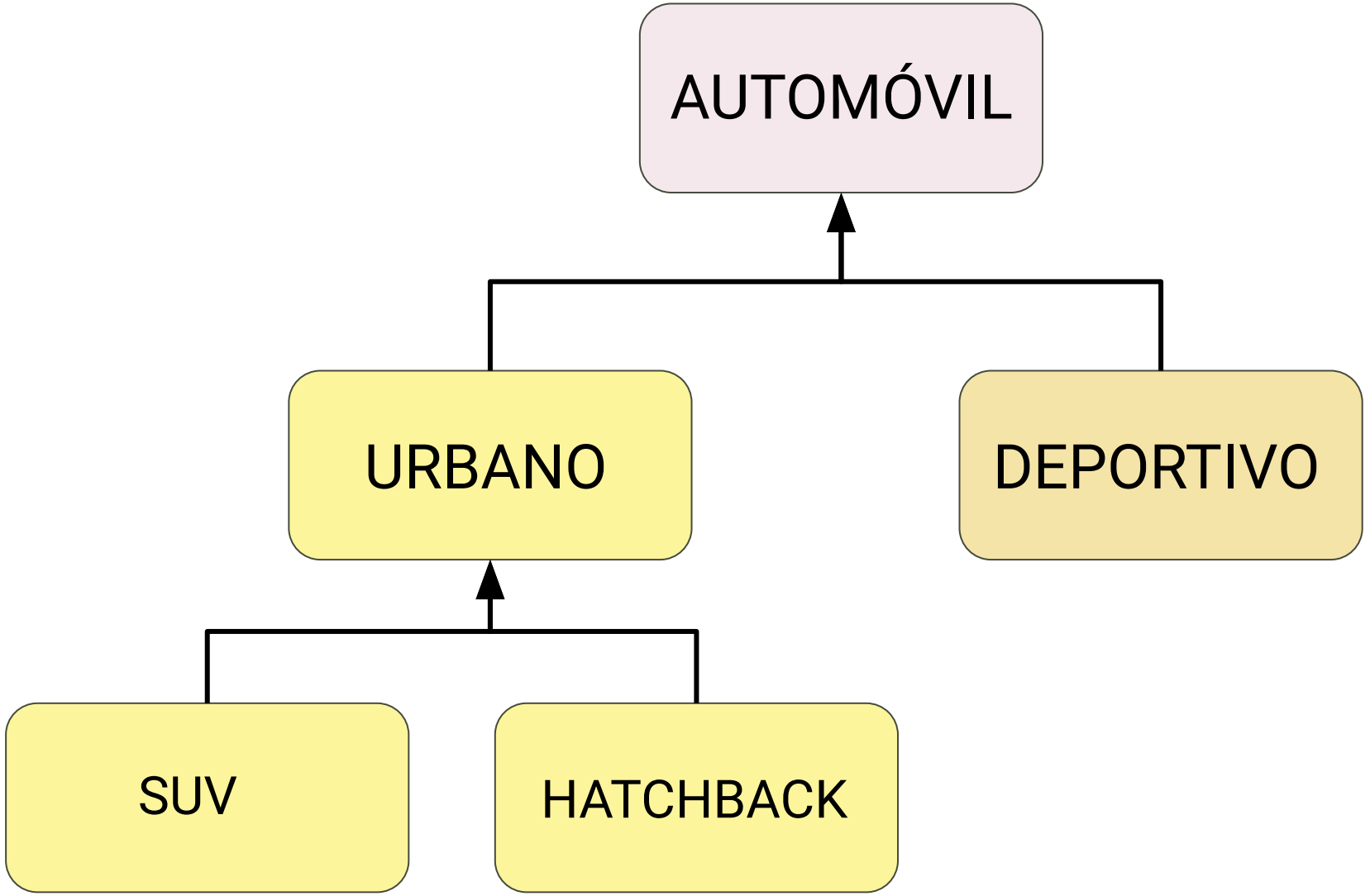
- **Objetivo**

  Reutilización de código.

- **Nuevos niveles de abstracción**

  Implementación de jerarquías.

Herencia



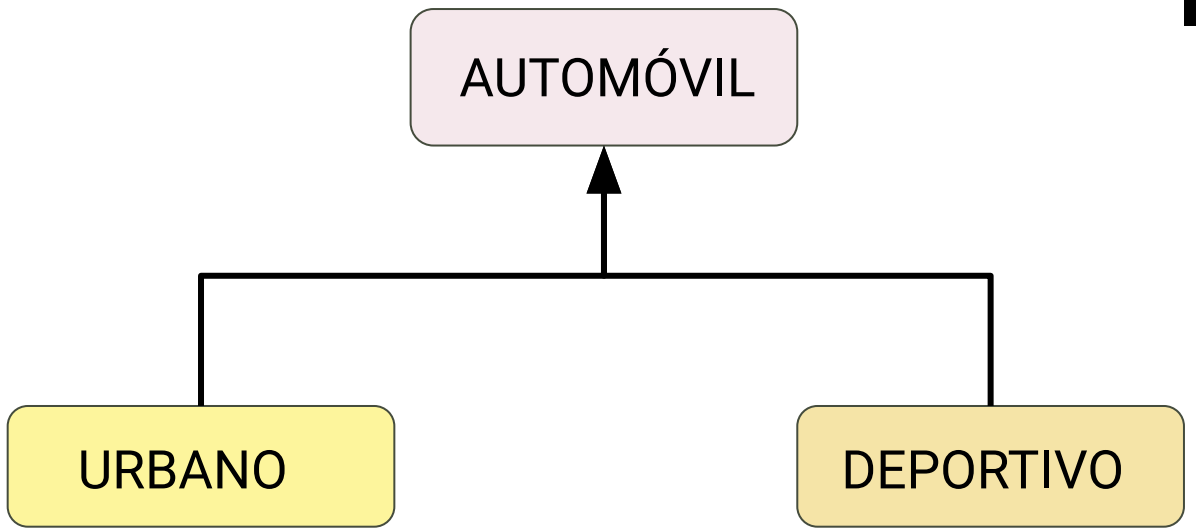
ABSTRACCIÓN

ABSTRACCIÓN

ABSTRACCIÓN

TRANSITIVIDAD

Herencia



DATOS

PUERTAS
RUEDAS

IMPLEMENTACIÓN

EENCENDER()

PUERTAS: 4
ALARMA: FALSE

ENCENDER()
DESACTIVAR_ALARMA()

OVERRIDE

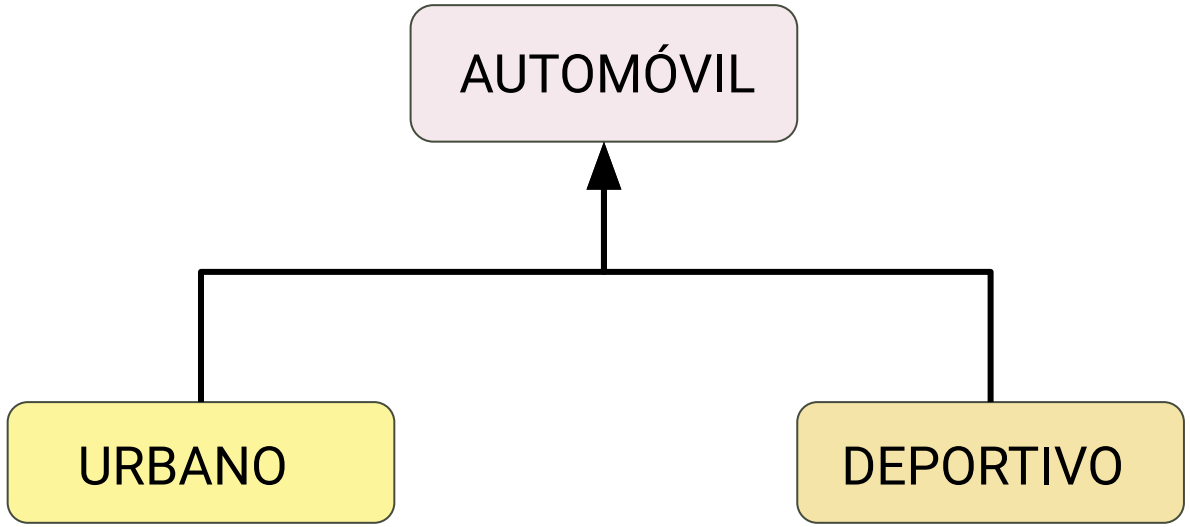
PUERTAS: 2

ENCENDER ()

OVERRIDE

Herencia

Relaciones



AUTOMOVIL es la **superclase o base** de URBANO y DEPORTIVO

URBANO es una **subclase o derivada** AUTOMOVIL

DEPORTIVO es una **subclase o derivada** AUTOMOVIL

URBANO Y DEPORTIVO **son del tipo** AUTOMOVIL

Herencia

Tipos de Herencia

- **Por defecto**

  Todas las clases heredan de **System.Object.**

- **Simple**

  Una única super clase.

- **Múltiple**

  Ejemplo tipado fuerte: C++, Scala
  Más de una super clase.
  Problema del “rombo”.

Encapsulamiento en Herencia

- **Acceso**

  Heredan el acceso de los atributos y métodos.

  La definición del acceso es desde el punto de vista de la clase donde está definida.

- **Sealed**

  Una clase **sellada** no puede ser superclase de otra.

Herencia

Clases y Herencia

Class Type		Can inherit from others	Can be inherited	Can be instantiated
normal	:	YES	YES	YES
abstract	:	YES	YES	NO
sealed	:	YES	NO	YES
static	:	NO	NO	NO