# TDA

Definición, Especificación, Creación y Uso

## Temas: Abstracción de datos

Clases y objetos

Propiedades

Ciclo de vida

Atributos

Uso del This

Constructores

Categorías

# Tipo de dato

- Primitivo
- Establecido por el lenguaje
- TDA (tipo de dato abstracto)

# Abstracción de datos

- Proceso de definirlo, implementarlo y utilizarlo
- **Se** ocultan las características de una cosa para obviarlas
- Ejemplo: Se abstraen todas las características de todos los perros en un solo término abstracto. “Perro”

- TDA : generalización de los tipos de datos básicos y de las operaciones primitivas.

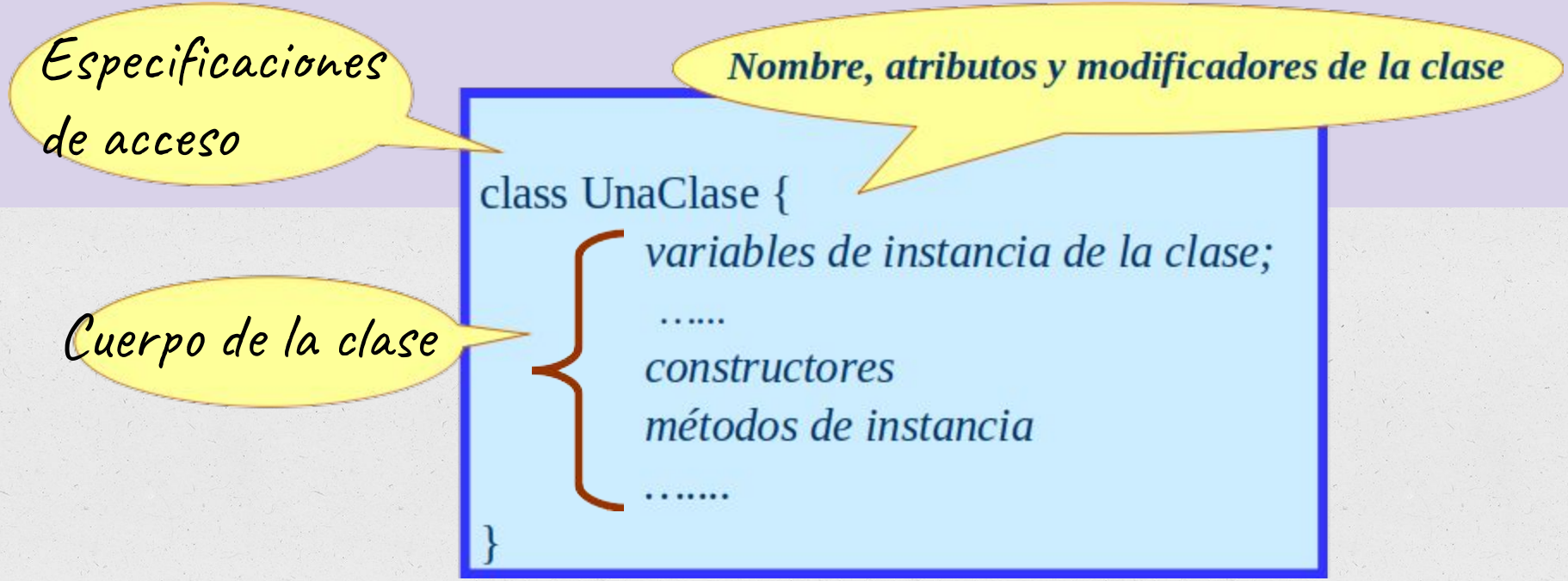
- define un concepto nuevo que puede manejarse con independencia de la estructura de datos para representarlo.

- TDA = **Representacion** (estructuras de datos) + **Operaciones** (metodos)

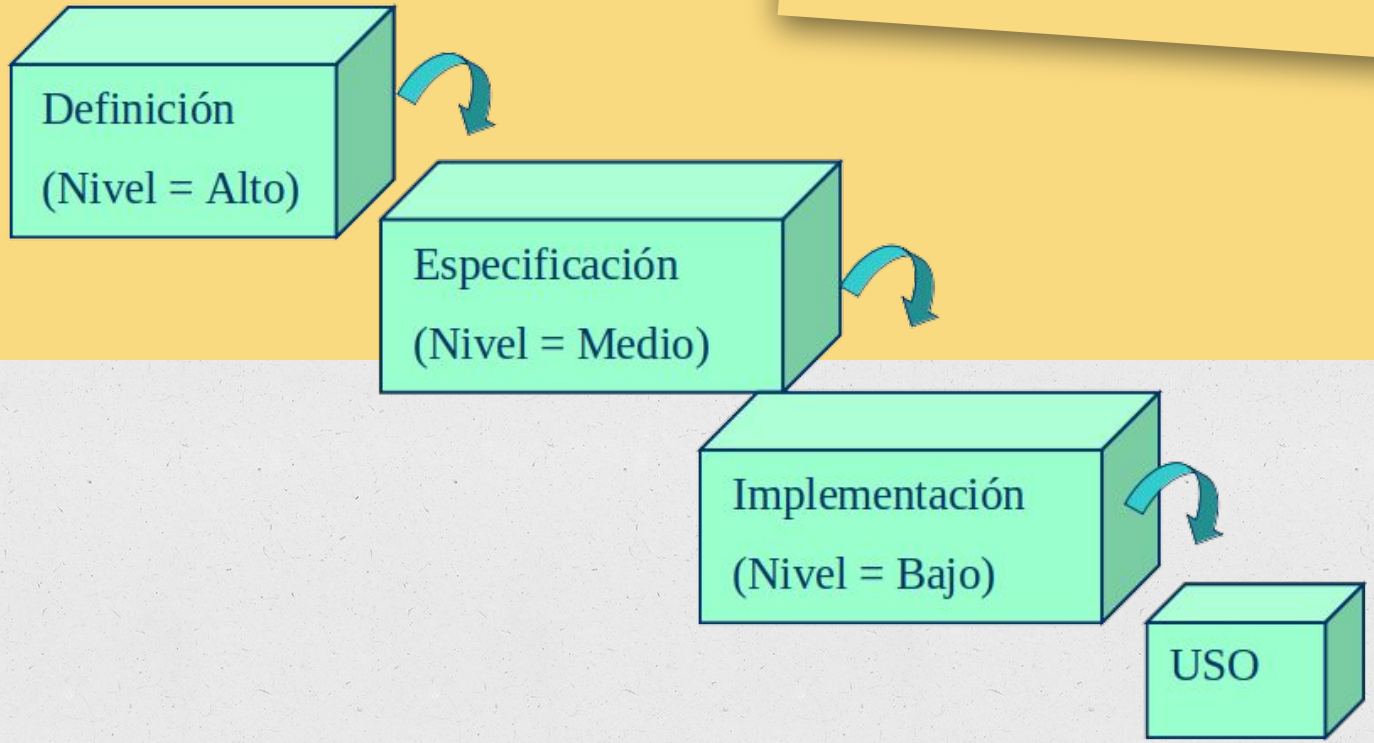
# TDA Estructura de datos

- Sólo pueden ser accedida mediante operaciones definidas, **interfaz**
  - Exporta un conjunto de operaciones (interfaz)
  - Las operaciones de la interfaz son el único y exclusivo mecanismo de acceso a la estructura de datos del TDA.

Un archivo PUEDE tener mas de una clase pero solo **una** puede ser **PUBLICA**



# Ciclo de vida del TDA



## DEFINICION

tipos de la instancia

- Mutable: puede ser modicado después de creado
- Inmutable: no puede ser modicado

Identificar el dominio

Identificar operaciones

## ESPECIFICACION

- Que datos o valores conforman el tipo de dato
- Especificar operaciones
  - sintácticamente (reglas de una operación)
  - semánticamente (significado de cada operación)

# Características

- Reusabilidad del código
- Facilidad de refinamiento
- Testeo simple
- Facilidad de Mantenimiento


## Lenguaje de modelado unificado

Estándar diseñado para visualizar, especificar, construir y documentar software orientado a objetos -

nosotros los usamos para especificar TDA

<NombreTDA>

<+> (público)/<->(privado) <tipoDeDato><identificadorDeDato>
…

UML

**Constructoras**
<NombreTDA>( <tipoDeDato> arg1, …)
…
**Observadoras**
<+> (público)/<->(privado) get<identificadorOperacion>
( <tipoDeDato> arg1, …): <tipoDeSalida> // pre-condiciones
…
**Modificadoras**
<+> (público)/<->(privado) set<identificadorOperacion>
( <tipoDeDato> arg1, …): <tipoDeSalida> // pre-condiciones
…
**Propias del tipo**
<+> (público)/<->(privado) <identificadorOperacion>
( <tipoDeDato> arg1, …): <tipoDeSalida> // pre-condiciones
…

```
package vehiculos;

class Auto{
   private String patente;
   private int km;
   …
Auto (String pat, ..,..) {
        ..
        }

public void andar (..) {
        …
        }
}
```

Clase a probar

```
package vehiculos;
   class TestAuto {
        ...
        main ( ..){
        …
   Auto miAuto;
   miAuto= new Auto (“AK…”, ,);

    miAuto.andar(..);
        ….
        }
   }
```

Definición de Clase

**Nombre de la Clase:** Automovil

**Datos:**

patente _____

marca _____

modelo _____

**Métodos (acciones):**

Incrementar Velocidad:

**Cómo:** Presionar el acelerador.

Parar:

**Cómo:** Presionar pedal de freno.

Objetos: **instancias** de la clase Automovil

**nombre del Objeto:** cocheDePedro
**patente:** “RRS 739”
**marca:** “Renault”
**modelo:** “Clio”

**nombre del Objeto:** cocheDeJuan
**patente:** “AGF 696”
**marca:** “Peugeot”
**modelo:** “206 XR”

**nombre del Objeto:** miCoche
**patente:** “HHG 246”
**marca:** “Renault”
**modelo:** “Megane”

```
/*
   *La clase TestAuto crea *una objeto auto, muestra sus datos, recorre algunos
    kilómetros y muestra la cantidad de kilómetros recorridos.
    */
    class TestAuto{
          public static void main(String[]   args) {
               Auto miAauto= new Auto('AKQ146', 'Sedan', 'rojo', 0);
               miAuto.andar(280);
               System.out.println("Patente nro " + miAuto.getPatente());
               System.out.println("kilómetros recorridos: " + miAuto.getKm());
          }
    }
```

declaración de la variable *miAuto*, como referencia a un objeto que será instancia de la clase *Auto*

Métodos que deben estar implementados en la clase Auto

## Tips de buena programación

- Se usa el mismo nombre para la clase y para el archivo (más la extensión ".java")
- Buena práctica de programación:
  - El nombre de la clase con letra mayúscula Ej. CocheDeCarrera.java
- Por ahora, colocaremos todas las clases que necesitamos para ejecutar un programa en la misma carpeta (paquete)

- Categorías: Constructores, visualizadores, modificadores, comparativos, propios del tipo
- La referencia **this**
- Para qué sirven los métodos **estáticos**
- Existen los métodos **privados**?