

Taller: Programación orientada a objetos

Deiry Sofía Navas Muriel
Misión TIC ciclo 2





Objetivos del taller

Conceptos básicos de Programación Orientada a Objetos (POO)

Conocer a la perfección en qué consiste la estructura de un programa en Java

Comprender los diagramas UML

Tipos de Datos en Programación Orientada a Objetos

Tipo Estático

Los datos de tipo estático son expresados en tiempo de **compilación** con esto se dice que mediante un análisis de compilación puede determinarse el tipo de dato.

Lenguajes que utilizan un tipado estático: C++, Java, C#.

```
class example
{
    private string Name;

    public string Estatic()
    {
        //Asignamos el nombre a la propiedad privada
        Name = "Daniel";
        Name = 70;           //esta asignación nos brinda un error.
        return Name;
    }
}
```

Tipo Dinámico

En este tipo de dato las comprobaciones se realizan en tiempo real es decir en **tiempo de ejecución**.

Esto permite que una variable tome valores de tipos diferentes en diferentes momentos. Los lenguajes que podemos mencionar que utilizan este tipado estan Phyton y PHP.

```
class dinamic
{
    private $id;

    public function typeDinamic()
    {
        $this->id = "Daniel";
        $this->id = 2;
        return "Hola " . $this->id    // Resultado: "Hola 2"
    }
}
```

Tipado fuerte vs débil

Estamos hablando de los tipos de datos que se manejan en un determinado lenguaje de programación, se refiere a los tipos y la forma en que deben usarse

```
> let resultado = "x" + 5;  
< undefined  
  
> resultado  
< "x5"
```

```
>>> resultado = "x" + 5  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: must be str, not int  
>>>
```

```
numero = 5  
if numero > 3:  
    print("SI se cumple")  
else:  
    print("NO se cumple")
```

```
int numero = 5;  
if (numero > 5){  
    System.out.println("Si cumple");  
}else{  
    System.out.println("No cumple");  
}
```

Ventajas Tipo fuerte

1. Código expresivo: ahora sí sabremos de qué tipo espera un argumento una función
2. Menos errores: Nos olvidaremos de ver el tipo de variable antes de hacer operaciones con ésta.
3. Detección temprana de errores durante la compilación
4. Aumento de rendimiento en tiempo de ejecución

Desventajas Tipo débil

1. Escribir más código: tenemos que declarar el tipo de variable al declararla
2. Curva de aprendizaje

```
>>> resultado = "x" + 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>>
```

Ventajas Tipo débil

1. Nos olvidamos de declarar el tipo
2. Podemos cambiar el tipo de la variable sobre la marcha. Por ejemplo, asignarle un string a un int
3. Escribimos menos código.
4. Prototipos rápidos

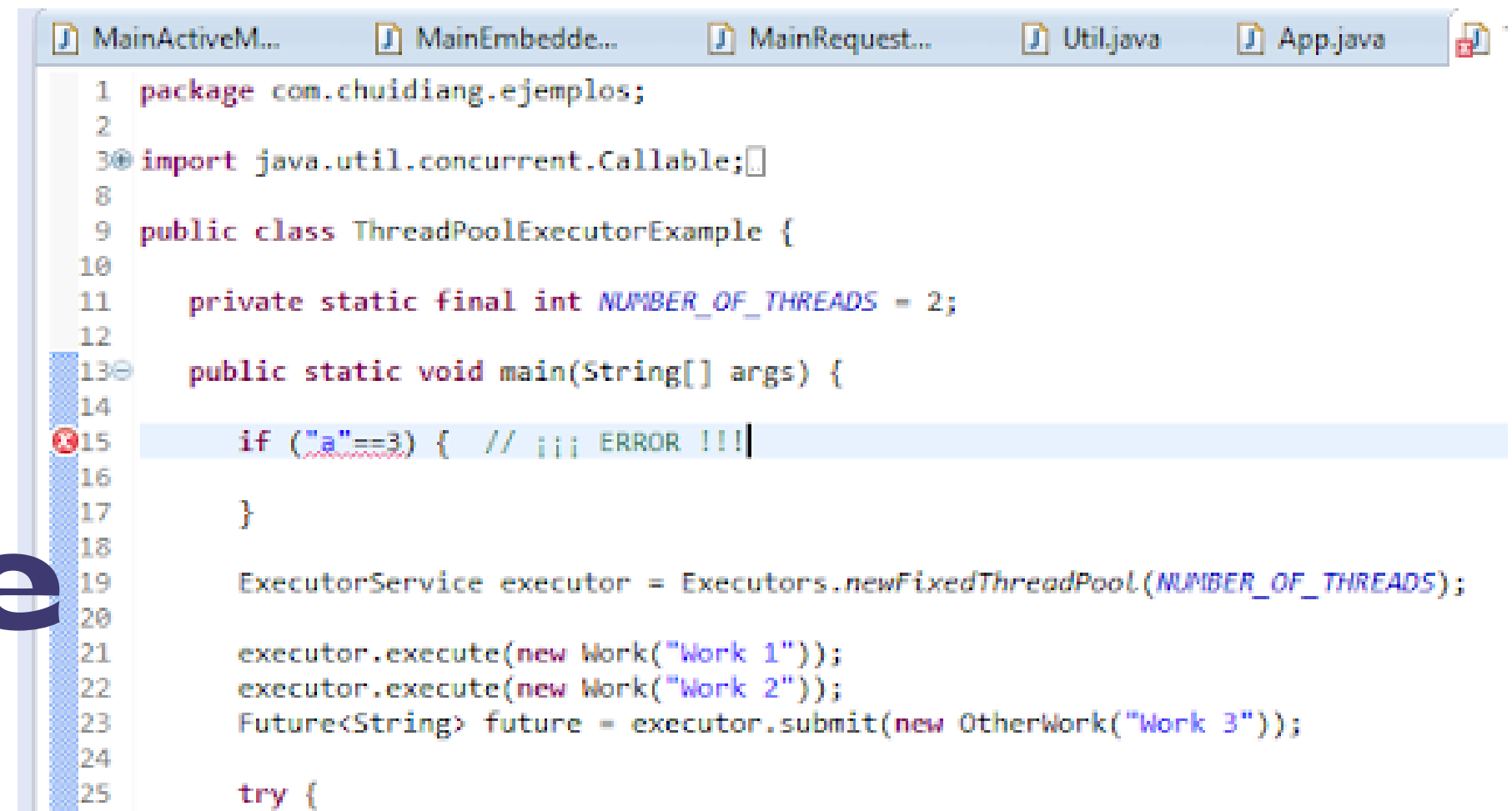
Desventajas Tipo débil

1. Al hacer operaciones, a veces éstas salen mal.
2. Comparar números que creemos que son números, pero no lo son.
3. Hay que castear muchas veces.
4. Código menos expresivo. Al declarar los argumentos de una función no sabemos si ésta espera un flotante, un entero, un string.

```
> let resultado = "x" + 5;
< undefined
> resultado
< "x5"
```

Java es un lenguaje compilado y estáticamente tipado

La programación orientada a objetos es un paradigma de programación en el cual los programas se organizan como colecciones de objetos que colaboran entre sí.



The screenshot shows a Java IDE window with several tabs: MainActiveM..., MainEmbedde..., MainRequest..., Util.java, and App.java. The active file is Util.java, which contains the following code:

```
1 package com.chuidiang.ejemplos;
2
3 import java.util.concurrent.Callable;
4
5 public class ThreadPoolExecutorExample {
6
7     private static final int NUMBER_OF_THREADS = 2;
8
9     public static void main(String[] args) {
10
11         if ("a"==3) { // !!! ERROR !!!
12
13         }
14
15         ExecutorService executor = Executors.newFixedThreadPool(NUMBER_OF_THREADS);
16
17         executor.execute(new Work("Work 1"));
18         executor.execute(new Work("Work 2"));
19         Future<String> future = executor.submit(new OtherWork("Work 3"));
20
21         try {
```

Line 11 is highlighted in blue, and a red error icon is visible in the left margin next to it, indicating a syntax error in the if statement.

Estructura de un programa java

Al momento de escribir un programa, a menudo es necesario modelar una parte del mundo que nos rodea (por modelar entendemos el representar de alguna manera esa parte del mundo, teniendo en cuenta solo los elementos que nos interesan en ese momento, y dejando por fuera todo lo demás).

Definición de clase

Definición de atributo o características

Definición de métodos



Definición clase

Descripción abstracta de un conjunto de objetos con propiedades similares, con un comportamiento común y relaciones similares con otras clases. Una clase también se puede entender como una especie de molde, modelo o plantilla a partir de la cual se crean objetos individuales.

Definición atributos

Normalmente, los atributos son privados, y por lo tanto, no son directamente accesibles. La consulta o modificación de los atributos se debe realizar a través de métodos públicos; de esta manera, el objeto controla las operaciones válidas que se pueden realizar.

Un atributo es una información que se almacena y es parte representativa del estado. Los atributos se pueden sustentar mediante tipos primitivos o clases.

Definición métodos

La razón de establecer los atributos en vista privada, es poder controlar la modificación del estado del objeto y no permitir que este evolucione hacía estados incoherentes. Pero debemos establecer los mecanismos para poder modificar el objeto, y son los métodos, el sistema adecuado. Podemos organizar los métodos en tres tipos, teniendo en cuenta aspectos sintácticos y semánticos

Constructores. Métodos que inicializan la instancia

Métodos genéricos. Realizan acciones utilizando los atributos

Métodos para accesos directo a los atributos (getters & setters)

Estos métodos son opcionales y suelen utilizarse más en clases de tipo "entidad", es decir, que tienen persistencia en bases de datos, o en los "beans", donde tienen una correspondencia con formularios. Aunque estos métodos son realmente generales, los distinguimos por separado por tener unas normas concretas de estilo.

Objetos

Recordemos que la clase representa el modelo para crear objetos, pero son los objetos los elementos con los que podemos trabajar.

Los objetos **ocupan memoria y tiene un estado**, que representa el conjunto de valores de sus atributos. Podemos crear todos los objetos que queramos, todos ellos tiene existencia propia, pudiendo evolucionar por caminos diferentes e independientes.

Cuando se crea un objeto se debe asociar con una referencia, por lo tanto, el proceso debe tener dos partes:

1. Declarar una variable que pueda referenciar al objeto en cuestión
2. Crear el objeto y asociarlo a la variable. Para crear el objeto, se realiza con la sentencia new, recordar que lleva implícito llamar al método constructor para que inicialice el objeto

Objetos

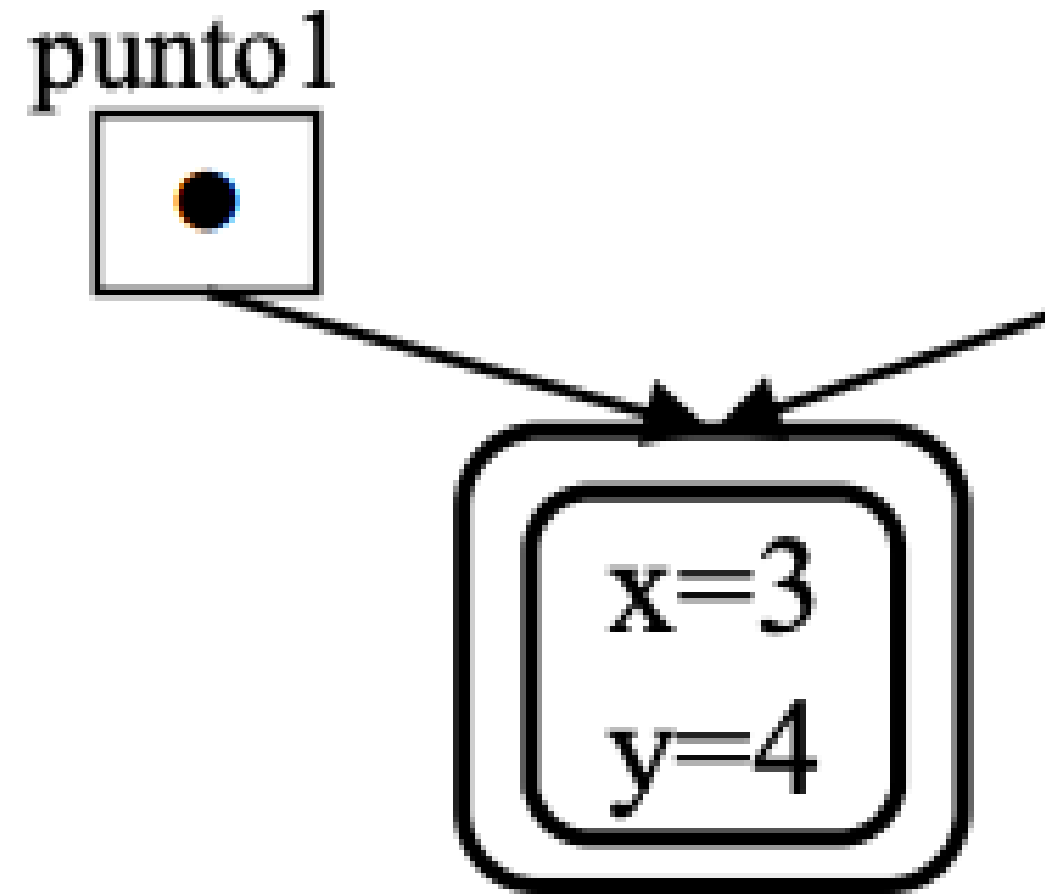
Recordemos que la clase representa el modelo para crear objetos, pero son los objetos los elementos con los que podemos trabajar.

Los objetos **ocupan memoria y tiene un estado**, que representa el conjunto de valores de sus atributos. Podemos crear todos los objetos que queramos, todos ellos tiene existencia propia, pudiendo evolucionar por caminos diferentes e independientes.

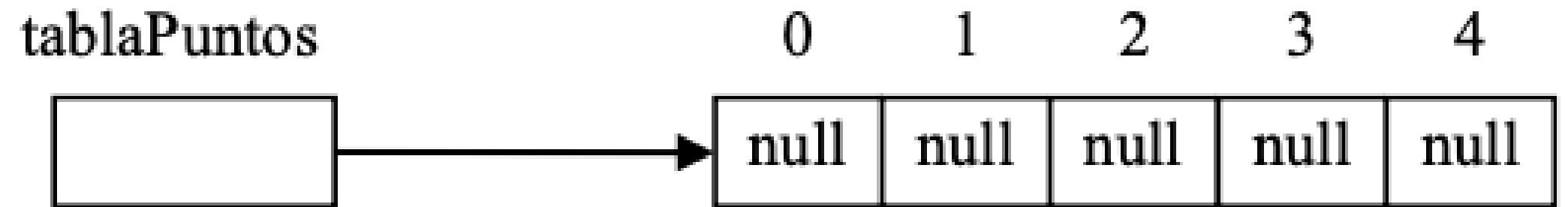
Cuando se crea un objeto se debe asociar con una referencia, por lo tanto, el proceso debe tener dos partes:

1. Declarar una variable que pueda referenciar al objeto en cuestión
2. Crear el objeto y asociarlo a la variable. Para crear el objeto, se realiza con la sentencia `new`, recordar que lleva implícito llamar al método constructor para que inicialice el objeto

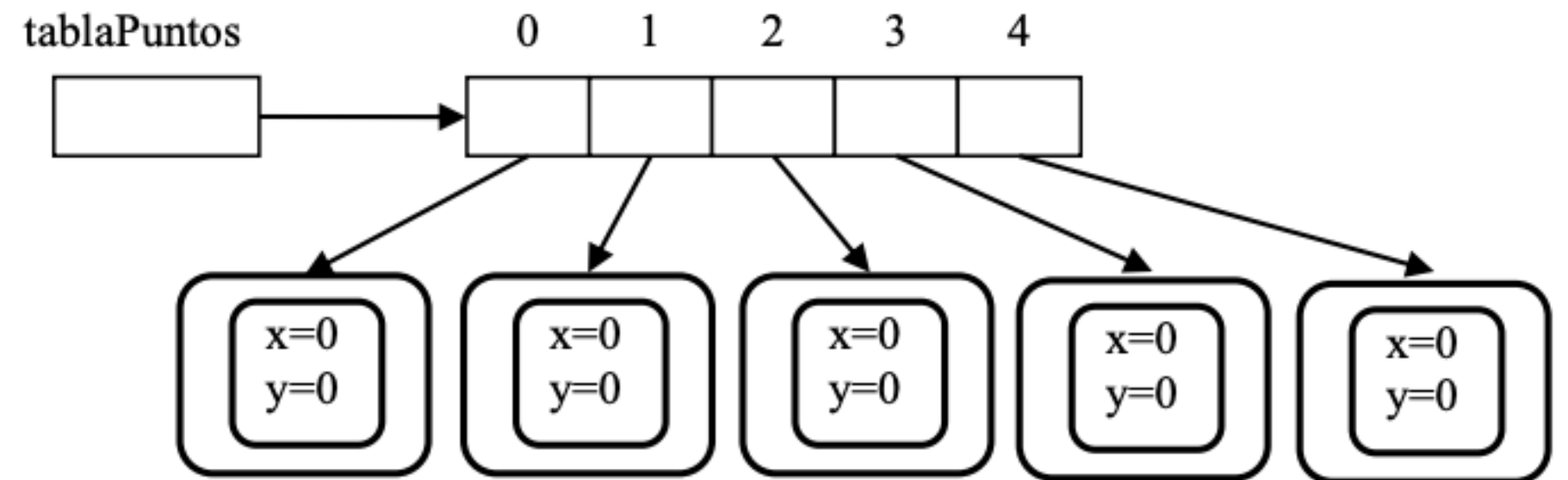
```
Punto punto1 = new Punto(3,4)
```



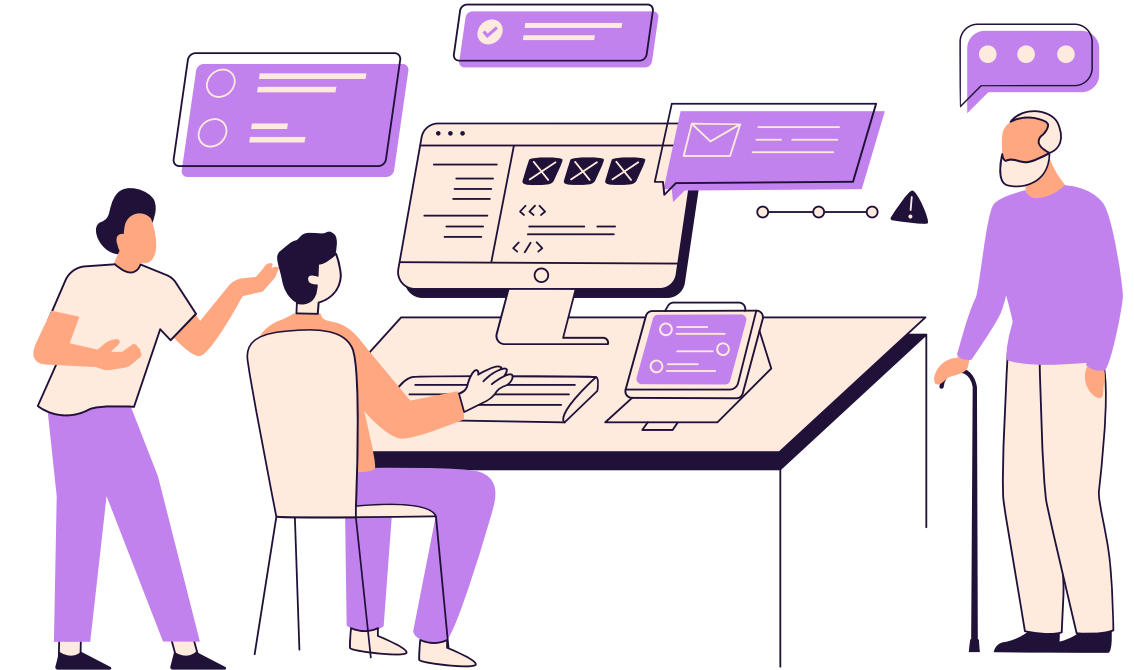
```
Punto[] tablaPuntos;  
tablaPuntos = new Punto[5];
```



```
for (int i = 0; i < 5; i++)  
tablaPuntos[i] = new Punto();  
tablaPuntos[i]= new Punto();
```



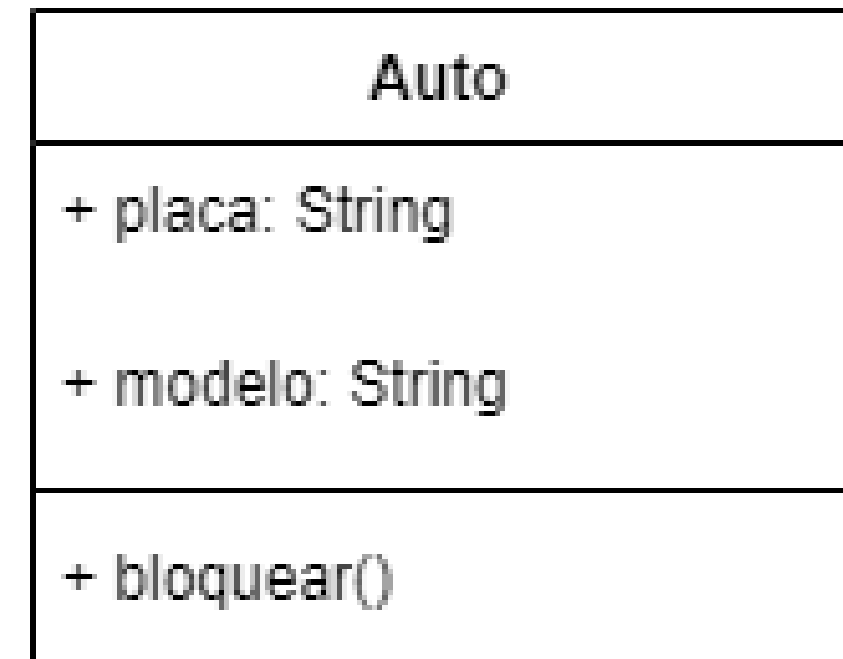
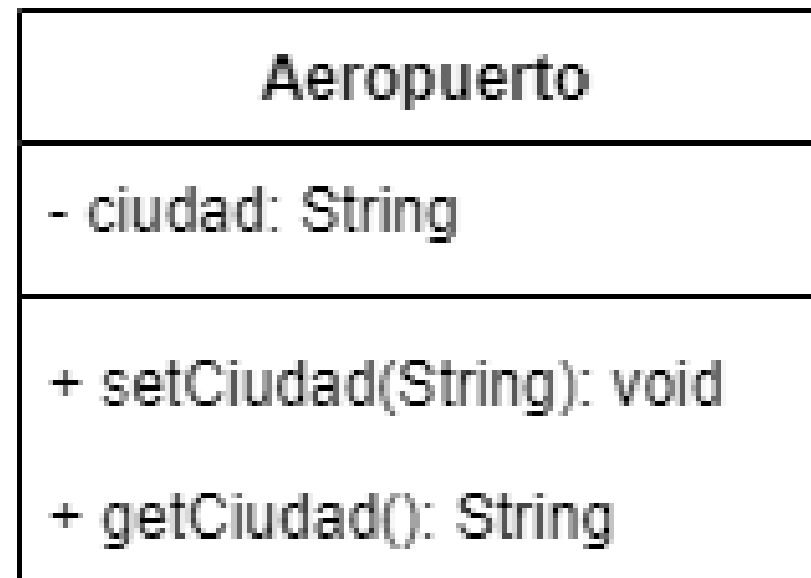
abstract	final	public
assert	finally	return
boolean	float	short
break	for	static
byte	if	strictfp
case	implements	super
catch	import	switch
char	instanceof	synchronized
class	int	this
continue	interface	throw
default	long	throws
do	native	transient
double	new	true
else	null	try
enum	package	void
extends	private	volatile
false	protected	while



Palabras reservadas

Diagrama UML

Para el modelado de sistemas de información se usa principalmente el estándar UML, que desde 2017 se encuentra en su versión 2.5.1. Dicho estándar propone y dicta la especificación de varios diagramas usados para el modelado de sistemas de información. Los diagramas UML pueden ser estáticos o dinámicos.



En primer lugar, se pone un símbolo que depende de si el atributo es de acceso público (+), protegido (#) o privado (–).

- Luego, se pone el nombre del atributo, seguido de dos puntos (:).
- Y, finalmente, se pone el tipo de dato del atributo, que para todos los casos de arriba serán atributos de tipo String.

Taller: Programación orientada a objetos

Realizado por: Deiry Sofía Navas Muriel

MISSION TIC

CICLO 2