

INTRODUCCIÓN A TYPESCRIPT

ANDREA MAGÁN & HACK A BOSS

AMR



ANDREA MAGÁN

FRONT DEVELOPER

Datail Technology / Improving Metrics

PROJECT&BRAND MANAGER

Tokio New Technology School

INTERIOR DESIGNER

SEAT Design Center



andrea.magan@outlook.com



/in/andreamaganrey



AMR



Are you ready for this jelly?

ukfabcier

ÍNDICE

CONTENIDOS

- ¿Qué es Typescript?
- ¿Por qué utilizarlo?
- ¿Que necesitamos para empezar?
- Hola Mundo

2

Declaraciones
Tipos básicos

3

Introducción P00
Tipos avanzados

4

P00
Decoradores

5

□

”

JAVASCRIPT THAT SCALES.

**TypeScript is a typed superset of JavaScript
that compiles to plain JavaScript.
Any browser. Any host. Any OS. Open source.**

typescriptlang.org

ANDERS HEJLSBERG

ARQUITECTO JEFE DE MICROSOFT

Diseñador de Delphy

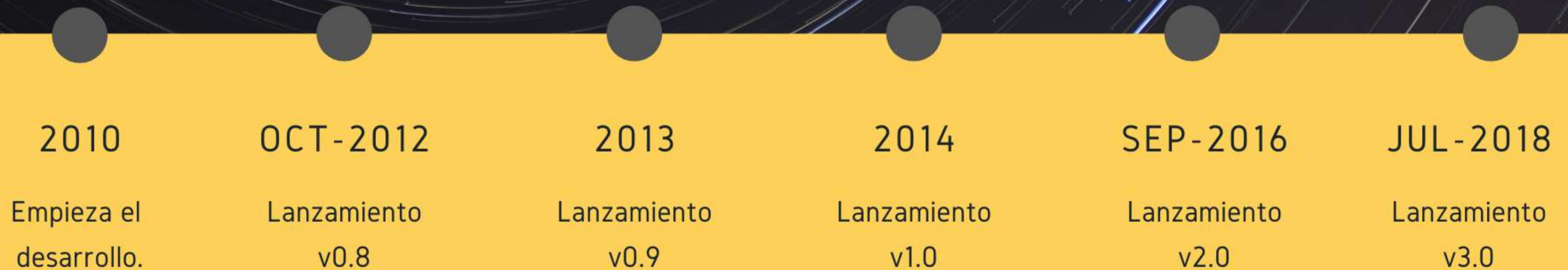
Diseñador de Turbo Pascal

Diseñador de C#

Arquitecto jefe Typescript

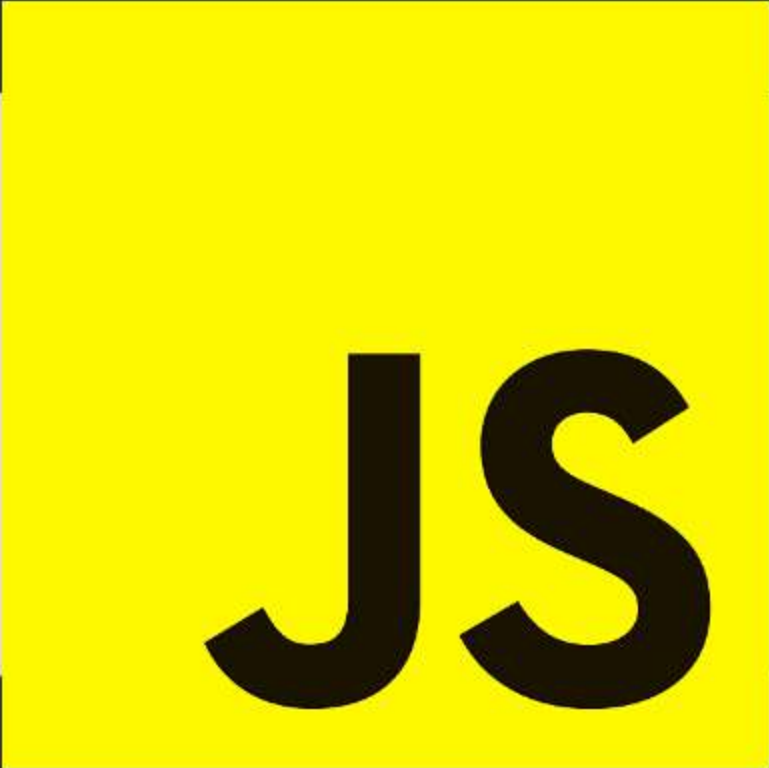


HISTORIA



¿QUÉ ES TYPESCRIPT?

TypeScript es un superset de Javascript, de código abierto, desarrollado por Microsoft, el cual cuenta con herramientas de programación orientada a objetos, pensado para escalar las aplicaciones.

A yellow square with the letters 'JS' in a large, bold, black sans-serif font.

JS

TIPADO DINÁMICO

El interprete define el tipo en tiempo de ejecución.

TIPADO ESTÁTICO

El tipo se define en desarrollo y no se modifica en tiempo de ejecución

A black square with the letters 'TS' in a large, bold, light gray sans-serif font.

TS

¿POR QUÉ UTILIZARLO?



El código se compila a JS puro

Fácil aprendizaje si ya se conoce JS

Ayudas para el desarrollo

Respaldo de grandes
comunidades y empresas

Utilizar funcionalidades modernas
antes de que tengan soporte en
todos los navegadores



Se necesita compilación

Se necesitan conocimientos de JS

Configuración de proyecto más
compleja

Es más verboso

Falsa sensación de seguridad

JS



TS

¿Qué necesitamos?



NODE JS

<https://nodejs.org/es>



TSC

`npm install -g typescript`



EDITOR

<https://code.visualstudio.com>

A young child with dark hair, wearing a red shirt, is standing in a playground. They are covering their eyes with their hands. The background is a blurred playground with a blue slide and a yellow heart-shaped balloon. The entire image has a warm, yellowish tint.

Playground

<https://www.typescriptlang.org/play/>

KAHOOT

KAHOOT.IT



ÍNDICE

CONTENIDOS

- Declaración de variables
- Declaración de funciones
- Tipos de datos básicos
- Tipos de datos avanzados

DÍA 1

Introducción
Primeros pasos

DÍA 3

POO

DÍA 4

Decoradores
Config. proyecto

DÍA 5

□

DECLARACIÓN DE VARIABLES

const

let

var

+

identificador

+

: tipo

=

valor ;

DECLARACIÓN DE FUNCIONES

function + identificador + (argumentos) + { sentencias }

TIPOS DE DATOS BÁSICOS I

TIPOS DE DATOS

> STRING

NUMBER

BOOLEAN

ARRAY

ANY

VOID

STRING

Para referirnos a tipos de datos textuales.

Al igual que JavaScript, se utilizan comillas dobles (") o comillas simples (') para rodear los datos de cadena. También puede usar plantillas de cadena de texto. Estas cadenas están rodeadas por el carácter backtick / backquote (`), y las expresiones incrustadas \${ expr }.

```
1 //Variable de tipo string con valor definido
2 let color: string = "azul";
3
4 //Cambiamos el valor con comillas simples
5 color = 'negro';
6
7 //Creamos una nueva variable con una cadena de texto
8 let description = `El ${color} es mi color favorito`;
9 // que sería equivalente a:
10 description = "El " + color + " es mi color favorito";
```

TIPOS DE DATOS

STRING

> NUMBER

BOOLEAN

ARRAY

ANY

VOID

NUMBER

Como en JavaScript, todos los números en TypeScript son valores de coma flotante. Además de los literales hexadecimales y decimales, TypeScript también admite literales binarios y octales introducidos en ECMAScript 2015.

```
1  let int: number = 6;
2  let decimal: number = 6.5;
3  let hex: number = 0xf00d;
4  let binary: number = 0b1010;
5  let octal: number = 0o744;
6
7  let total: number = int + decimal + hex + binary + octal
8
9
```


TIPOS DE DATOS

STRING

NUMBER

> BOOLEAN

ARRAY

ANY

VOID

BOOLEAN

El tipo de dato lógico o booleano es en computación aquel que puede representar valores de lógica binaria, esto es 2 valores, que normalmente representan falso o verdadero

```
1  let edad: number;
2
3  // variable de tipo boolean
4  let mayorDeEdad: boolean;
5
6  // función que retorna un valor de tipo boolean
7  function esMayorDeEdad(edad: number): boolean {
8      if (edad >= 18) {
9          return true;
10     } else {
11         return false;
12     }
13 }
14
15 // Pedro tiene 21 años
16 edad = 21;
17 mayorDeEdad = esMayorDeEdad(edad); // true
18 console.log(edad, mayorDeEdad);
19
20 // María tiene 8 años
21 edad = 8;
22 mayorDeEdad = esMayorDeEdad(edad); // false
23 console.log(edad, mayorDeEdad);
24
```

TIPOS DE DATOS

STRING

NUMBER

BOOLEAN

> ARRAY

ANY

VOID

ARRAY

Tipo de dato estructurado que permite almacenar una colección de elementos.

```
1 // Definiendo el tipo y añadiéndole la notación []
2 let listaDeNumeros: number[] = [1, 2, 3];
3
4 // Usando un tipo genérico
5 let otraListaDeNumeros: Array<number> = [1, 2, 3];
6
7
```


TIPOS DE DATOS

STRING

NUMBER

BOOLEAN

ARRAY

> ANY

VOID

ANY

Indica que la variable puede ser de cualquier tipo. No es aconsejable utilizarlo de forma continua, pero es muy útil cuando trabajamos con librerías de terceros o con proyectos JS antiguos que empezemos a refactorizar a TS.

```
1 // Variable de tipo any. Puede asignarse cualquier
2 // valor y se inferirá su tipo.
3 let notSure: any = 4;
4
5 // Ahora es un string
6 notSure = "maybe a string instead";
7
8 //Ahora un boolean
9 notSure = false;
10
11 // Array sin tipo definido
12 let list: any[] = [1, true, "free"];
13
14 list[1] = 100;
15
```

TIPOS DE DATOS

STRING

NUMBER

BOOLEAN

ARRAY

ANY

> VOID

VOID

Void es lo opuesto a any, la ausencia de cualquier tipo. Normalmente se usa para indicar que una función no devolverá ningún valor.

Declarar variables de tipo void es inútil porque no podríamos asignarle ningún valor.

```
1 // La función no devuelve nada
2 function alerta(): void {
3   console.log("Nininoninonino");
4 }
5
6
7 // Definir una variable como void no tiene sentido porque
8 // nos permite asignarle ningún valor
9 let vacio: void;
10
11 // Error: Type '"cualquier valor"' is not assignable
12 // to type 'void'
13 vacio = 'cualquier valor';
14 vacio = null;
15 vacio = 1;
16 vacio = [1,2,3]
17
```


TIPOS DE DATOS BÁSICOS II

TIPOS DE DATOS

> TUPLE

NULL &
UNDEFINED

NEVER

ENUM

OBJECT

TUPLE

Similar al array, pero con un número fijo de elementos escritos.

```
1
2 // Declarar una tupla
3 let x: [string, number];
4
5 // Asignar valores correctos
6 x = ["hello", 10]; // OK
7
8 // Asignar valores incorrectos
9 x = [10, "hello"]; // Error
10
```


TIPOS DE DATOS

TUPLE

> NULL &
UNDEFINED

NEVER

ENUM

OBJECT

NULL

Representa un valor nulo o que aun no está disponible.

```
1 // Declarar una variable de tipo null
2 let x: null;
3
4 // El unico valor válido es null
5 x = null; //Ok
6 x = 2; // Error
```

UNDEFINED

Se utiliza para definir que aun no se ha inicializado.

```
1 // Declarar una variable de tipo undefined
2 let x: undefined;
3
4 // El unico valor válido es undefined
5 x = undefined; // Ok
6 x = null; // Error
7
```

TIPOS DE DATOS

TUPLE

NULL &
UNDEFINED

> NEVER

ENUM

OBJECT

NEVER

Este tipo representa el tipo de valores que nunca se producen.

```
1  // Función que lanza una excepción
2  function error(message: string): never {
3      |    throw new Error(message);
4      |
5      |
6      // Función que retorna un error
7      function fail(): never {
8          |    return error("Something failed");
9          |
10         |
11         // Bucle infinito
12         function infiniteLoop(): never {
13             |    while (true) {
14                 |    }
15             |
16             }
```


TIPOS DE DATOS

TUPLE

NULL &
UNDEFINED

NEVER

> ENUM

OBJECT

ENUM

Un enum es una forma de organizar una colección de valores relacionados.

```
1  enum Estaciones {  
2      Primavera,  
3      Verano,  
4      Otono,  
5      Invierno  
6  }  
7  
8  // Definimos una variable del tipo del enumerado  
9  let estacion: Estaciones;  
10  
11 // Asignamos uno de los valores disponibles  
12 estacion = Estaciones.Primavera; // 0  
13
```

TIPOS DE DATOS

TUPLE

NULL &
UNDEFINED

NEVER

ENUM

> OBJECT

OBJECT

Un objeto es el tipo que representa el tipo no primitivo, es decir, cualquier cosa que no es number, string, boolean, symbol, null, o undefined.

```
1  let persona = {  
2      nombre: "Andrea",  
3      apellido: "Magán",  
4      aficiones: ["series", "bailar", "aprender"]  
5  };  
6  
7  //Acceso a los valores  
8  console.log(persona.nombre); // Andrea  
9  console.log(persona.apellido); // Magán  
10 console.log(persona.aficiones[1]); // bailar  
11  
12 console.log(typeof(persona)); // object  
13
```


TIPOS DE DATOS AVANZADOS

TIPOS DE DATOS

> GENÉRICOS

UNIONTYPES

TYPEGUARDS

INTERSECTION
TYPES

TYPE
ASERTIONS

TYPE ALIASES

GENÉRICOS

Podemos entender los genéricos como una especie de "plantilla" o "variable", mediante la cual podemos aprovechar código, sin tener que duplicarlo por causa de cambios de tipo y evitando la necesidad de usar el tipo "any".

```
1 // La función que recibe un tipo genérico
2 function displayGeneric<T>(valor: T): T {
3     console.log('Display generic -->' , valor);
4     return valor;
5 }
6
7 displayGeneric<number>(2);
8 displayGeneric<string>('string');
9
```


TIPOS DE DATOS

GENÉRICOS

> UNION TYPES

TYPEGUARDS

INTERSECTION
TYPES

TYPE
ASSERTIONS

TYPE ALIASES

UNION TYPES

Nos permiten definir varios tipos para una variable.

```
1  interface Guess {  
2      identifier: string;  
3      canRead: boolean;  
4  }  
5  
6  interface Admin {  
7      identifier: string;  
8      canWrite: boolean;  
9  }  
10  
11  type Role = Guess | Admin;  
12
```

TIPOS DE DATOS

GENÉRICOS

UNION TYPES

TYPE GUARDS

> INTERSECTION
TYPES

TYPE

ASSERTIONS

TYPE ALIASES

INTERSECTION TYPES

Un tipo de intersección combina múltiples tipos en uno.

```
1 interface ErrorHandler {
2   success: boolean;
3   error?: { message: string };
4 }
5
6 interface ArtworksData {
7   artworks: { title: string }[];
8 }
9
10 interface ArtistsData {
11   artists: { name: string }[];
12 }
13
14 type ArtworksResponse = ArtworksData & ErrorHandler;
15 type ArtistsResponse = ArtistsData & ErrorHandler;
16
```


TIPOS DE DATOS

GENÉRICOS

UNIONTYPES

TYPEGUARDS

INTERSECTION
TYPES

TYPE
ASERTIONS

>TYPE ALIASES

TYPE ALIASES

Los alias de tipo crean un nuevo nombre para un tipo. Son similares a las interfaces, pero pueden nombrar primitivas, uniones, tuplas y cualquier otro tipo.

```
1  type Name = string;
2  type NameResolver = () => string;
3  type NameOrResolver = Name | NameResolver;
4
```

TIPOS DE DATOS

GENÉRICOS

UNIONTYPES

> TYPEGUARDS

INTERSECTION
TYPES

TYPE

ASERTIONS

TYPE ALIASES

TYPE GUARDS

Las guardias de tipos nos permiten conocer el tipo cuando usamos union types. Para ello podemos usar `typeof`, `in` o `instanceof`.

```
1  function foo(x: number | string) {  
2      // Dentro del if TypeScript sabe que `x`  
3      // debe ser un string  
4      if (typeof x === 'string') {  
5          // 'substr' no existe en `string`  
6          console.log(x.substr(1)); // Error  
7          console.log(x.substr(1)); // OK  
8      }  
9      //No hay garantía de que `x` sea un `string`  
10     x.substr(1); // Error  
11 }  
12  
13
```


TIPOS DE DATOS

GENÉRICOS

UNIONTYPES

TYPEGUARDS

INTERSECTION
TYPES

> TYPE
ASERTIONS

TYPE ALIASES

TYPE ASERTIONS

Las aserciones de tipo son una forma de decirle al compilador que use el tipo que le damos en lugar del definido anteriormente.

```
1 let someValue: any = "this is a string";
2 let strLength: number = (<string>someValue).length;
3
4
5 let someValue: any = "this is a string";
6 let strLength: number = (someValue as string).length;
7
8
```

KAHOOT

KAHOOT.IT



ÍNDICE

CONTENIDOS

- Introducción a P00
- P00 en Typescript

DÍA 1

Introducción

Primeros pasos

DÍA 2

Declaraciones

Tipos de datos

DÍA 4

Configuración de proyecto

Modulos

DÍA 5

□

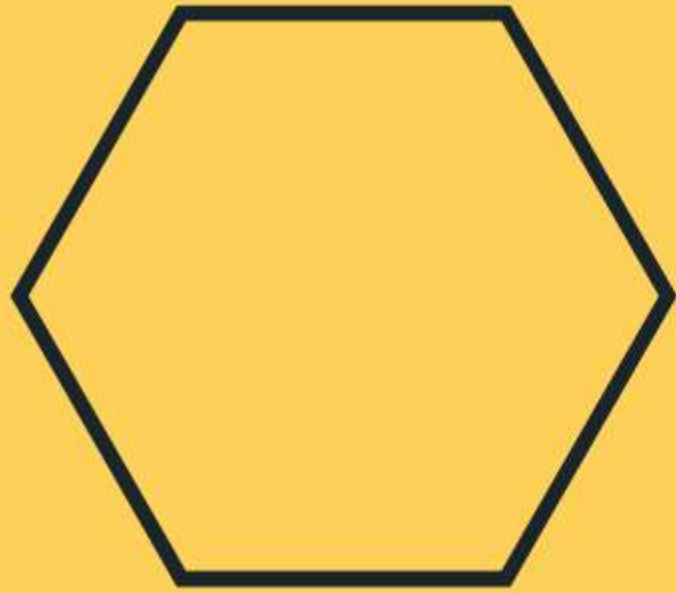
¿QUÉ ES POO?

La Programación Orientada a Objetos es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial

[Wikipedia.org](https://es.wikipedia.org)

CLASE

Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de determinado tipo.



OBJETO

Un objeto es una entidad concreta que se crea a partir de la plantilla que es la clase.



DECLARACIÓN DE CLASES

class + identificador + extends
implements + identificador { atributos
constructor
metodos }

BENEFICIOS DE LA POO

BENEFICIOS

ABSTRACCIÓN

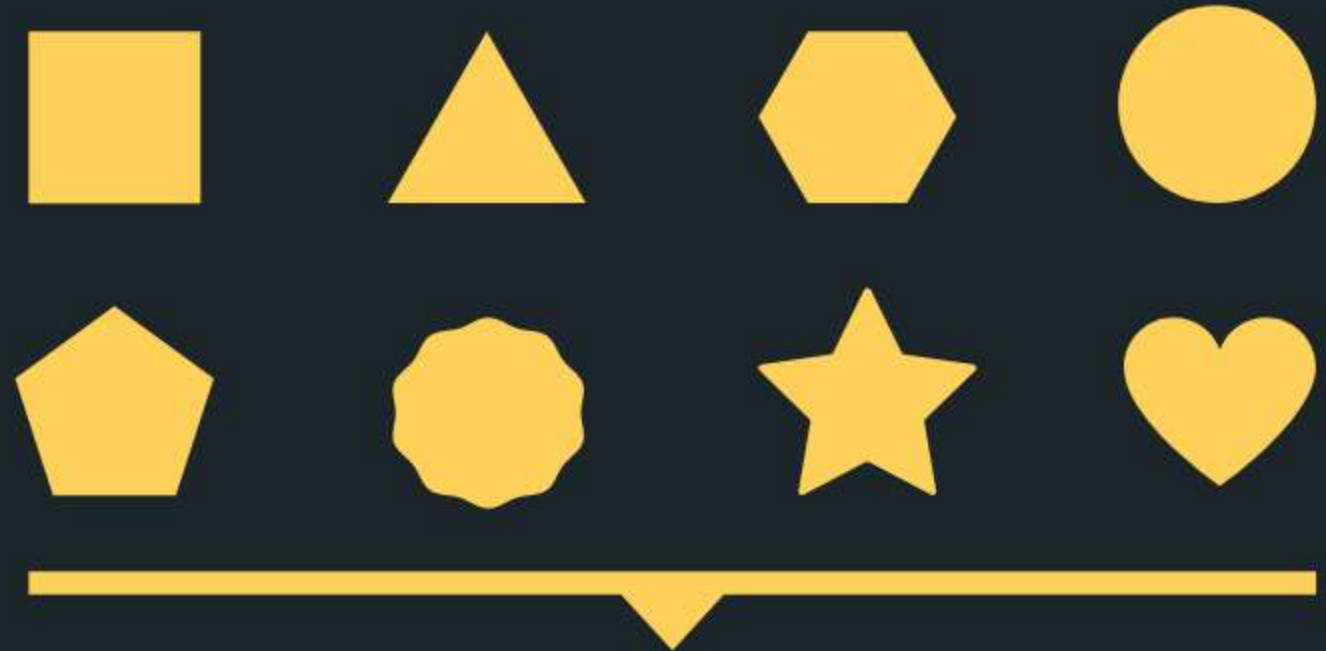
ENCAPSULACIÓN

MODULARIZACIÓN

JERARQUIZACIÓN

ABSTRACCIÓN

La abstracción consiste en captar las características esenciales de un objeto, así como su comportamiento.



Formas

BENEFICIOS

ABSTRACCIÓN

ENCAPSULACIÓN

MODULARIZACIÓN

JERARQUIZACIÓN

ENCAPSULACIÓN

Es el proceso por el cual se ocultan los detalles del soporte donde se almacenan las características de una abstracción

Forma



Nombre

BENEFICIOS

ABSTRACCIÓN

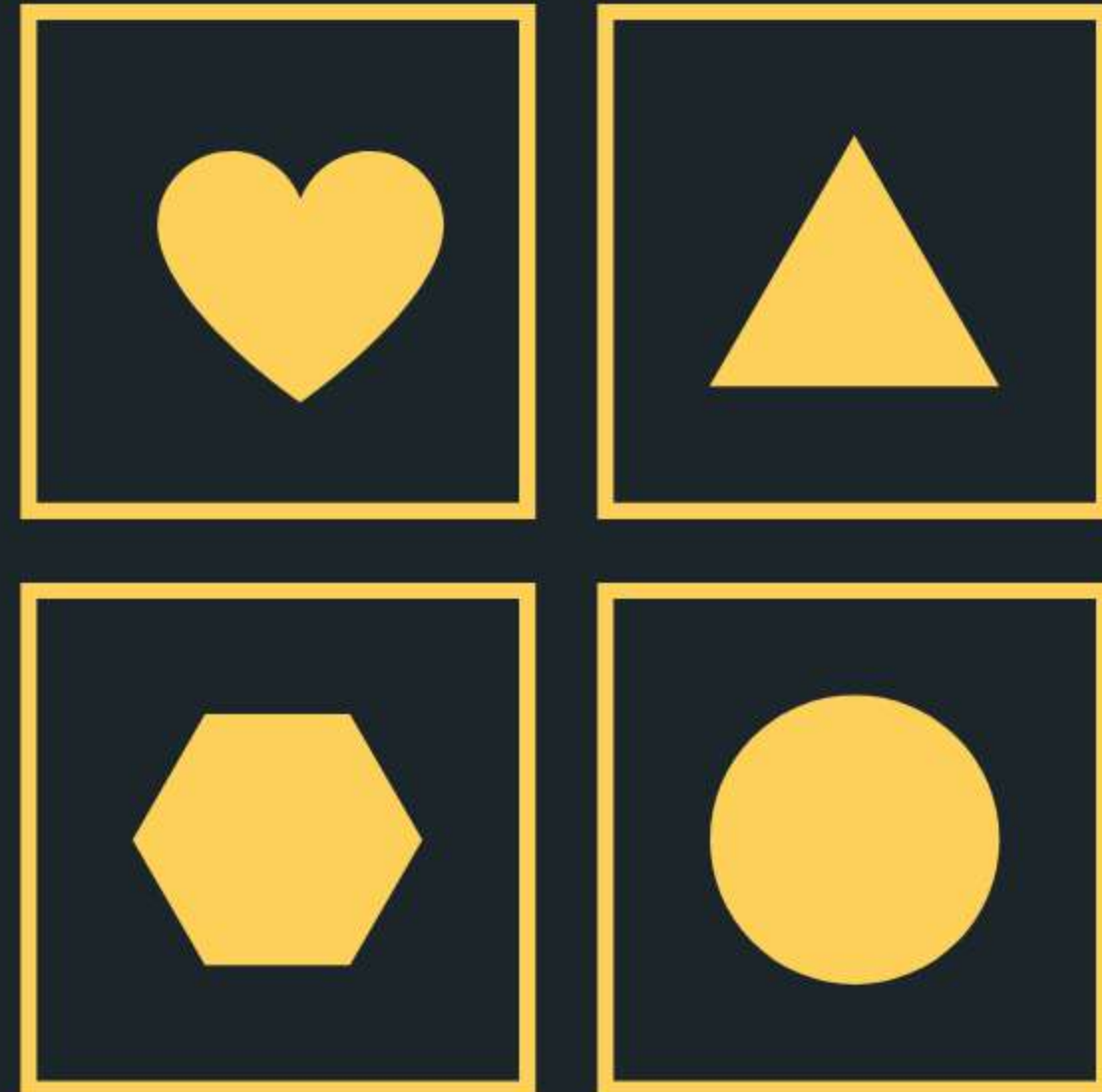
ENCAPSULACIÓN

MODULARIZACIÓN

JERARQUIZACIÓN

MODULARIZACIÓN

Es la descomposición de un sistema, creando una serie de piezas que colaboran entre si, poco acoplados y cohesivos



BENEFICIOS

ABSTRACCIÓN

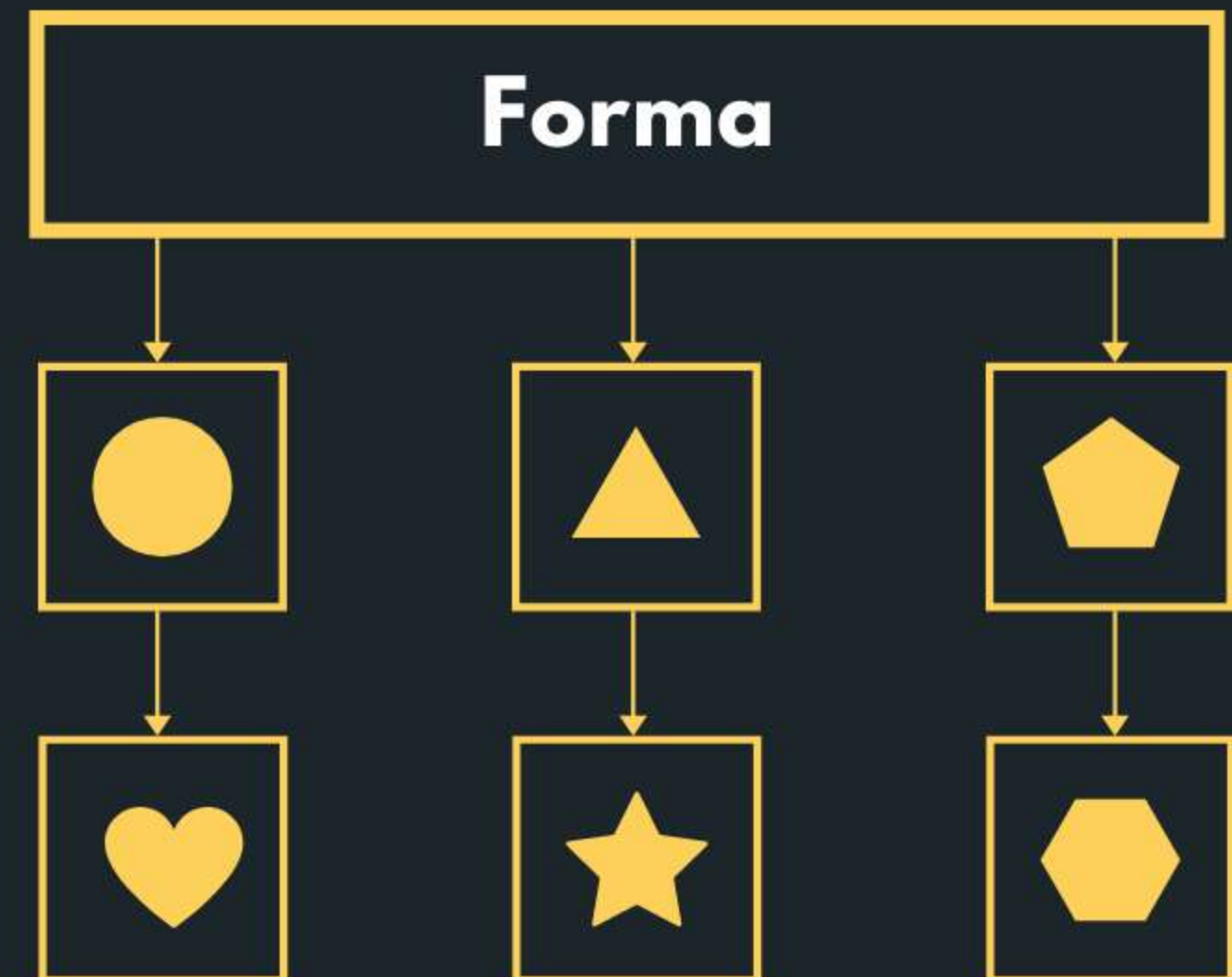
ENCAPSULACIÓN

MODULARIZACIÓN

JERARQUIZACIÓN

JERARQUIZACIÓN

Es la estructuración por niveles de los módulos o elementos que forman parte de un sistema



HERENCIA

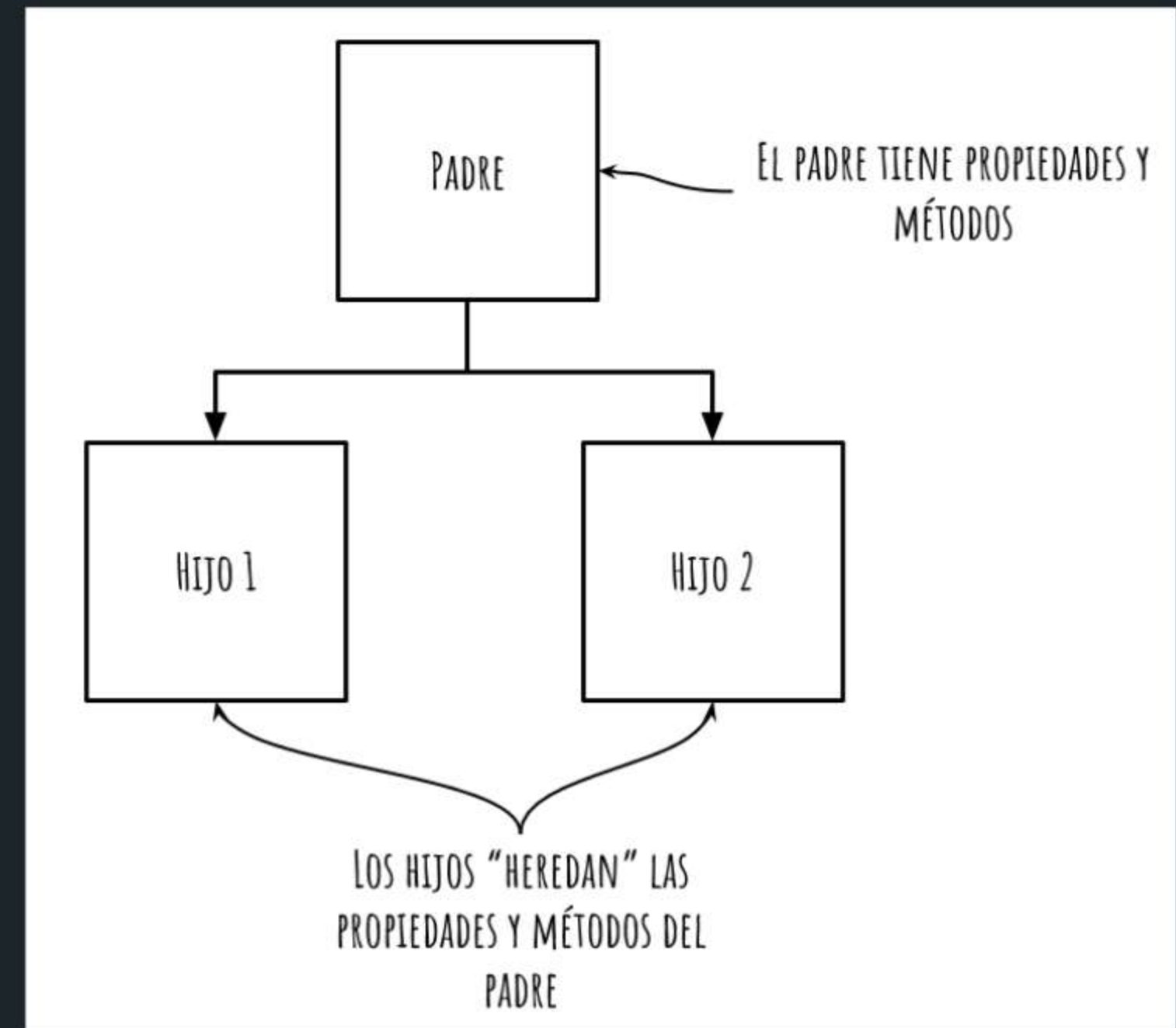
CONSTRUCTOR

CLASE PADRE

VISIBILIDAD

SOBREESCRIBIR
MÉTODOS

Las clases pueden herdar propiedades y métodos de otras si extendemos su comportamiento. Las clases derivadas a menudo se denominan subclases, y las clases base a superclases .



HERENCIA

CONSTRUCTOR

CLASE PADRE

VISIBILIDAD

SOBREESCRIBIR
MÉTODOS

Los atributos y métodos de una clase pueden ser:

PUBLIC

Podemos acceder sin ninguna restricción desde fuera de la clase.



¡HOLA, SOY **PUBLIC**! NO
TENGO SECRETOS,
¡CUALQUIERA PUEDE
VERME!

HERENCIA

CONSTRUCTOR

CLASE PADRE

VISIBILIDAD

SOBREESCRIBIR
MÉTODOS

Los atributos y métodos de una clase pueden ser:

PRIVATE

Solo podremos acceder desde dentro de la propia clase



¡HOLA, SOY **PRIVATE**!
TENGO MUCHOS
SECRETOS, **SÓLO** LA CLASE
QUE ME CREA PUEDE
VERME Y... ¡NADIE MÁS!

HERENCIA

CONSTRUCTOR

CLASE PADRE

VISIBILIDAD

SOBREESCRIBIR
MÉTODOS

Los atributos y métodos de una clase pueden ser:

PROTECTED

Podremos acceder desde la clase o desde sus subclases



HERENCIA

CONSTRUCTOR

CLASE PADRE

VISIBILIDAD

SOBREESCRIBIR
MÉTODOS

Los atributos y métodos de una clase pueden ser:

READ ONLY

Solo lectura. Podemos acceder a ellas pero no modificarlas. Las propiedades de solo lectura deben inicializarse en su declaración o en el constructor.

READONLY

¡HOLA, SOY READONLY!
TODOS PUEDEN VERME,
PERO NADIE MÁS QUE LA
CLASE QUE ME CREA PUEDE
MODIFICARME

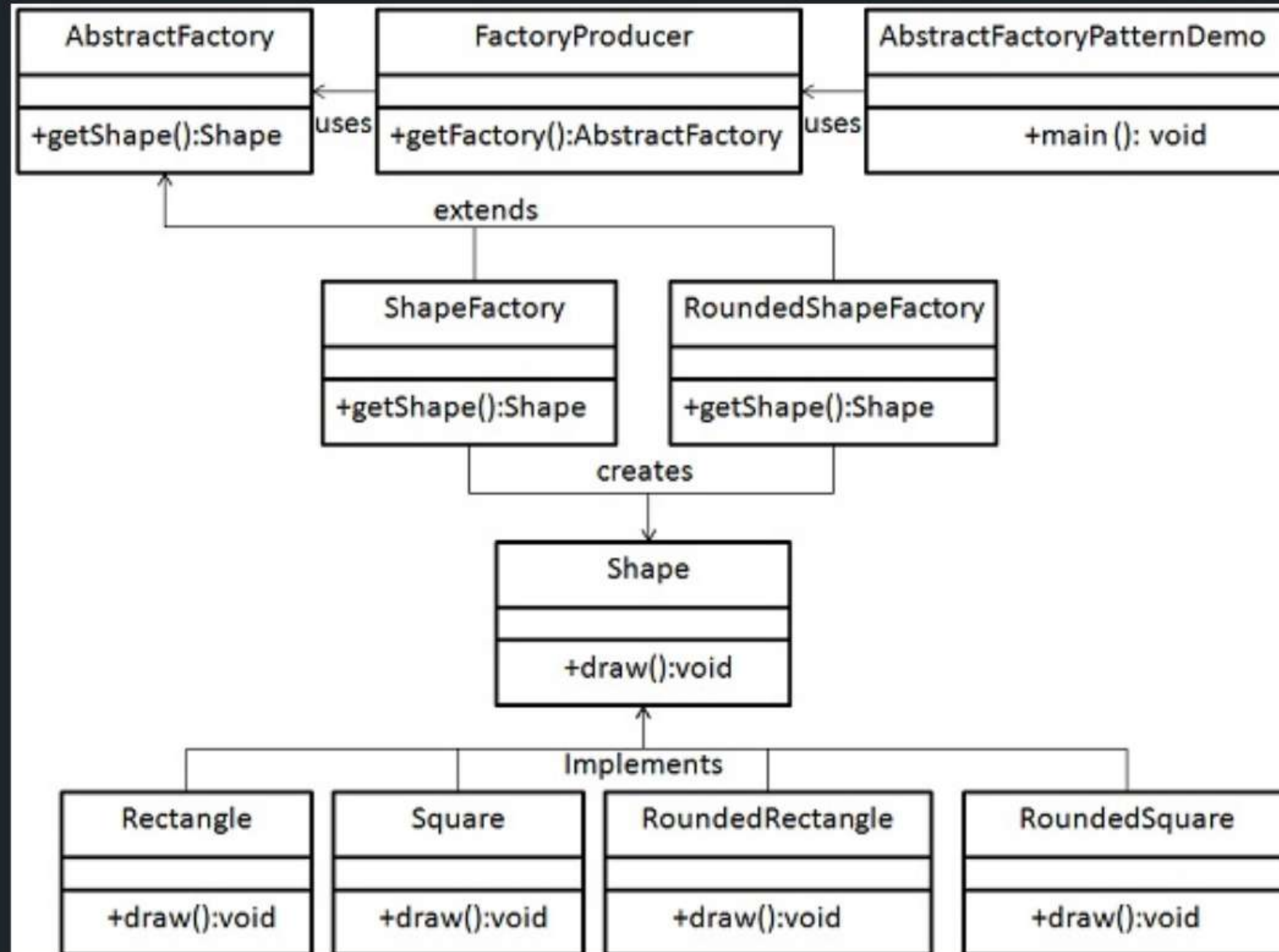
CLASES ABSTRACTAS

Son clases base a partir de las cuales se pueden extender otras clases.
No se pueden crear instancias de ellas mismas.

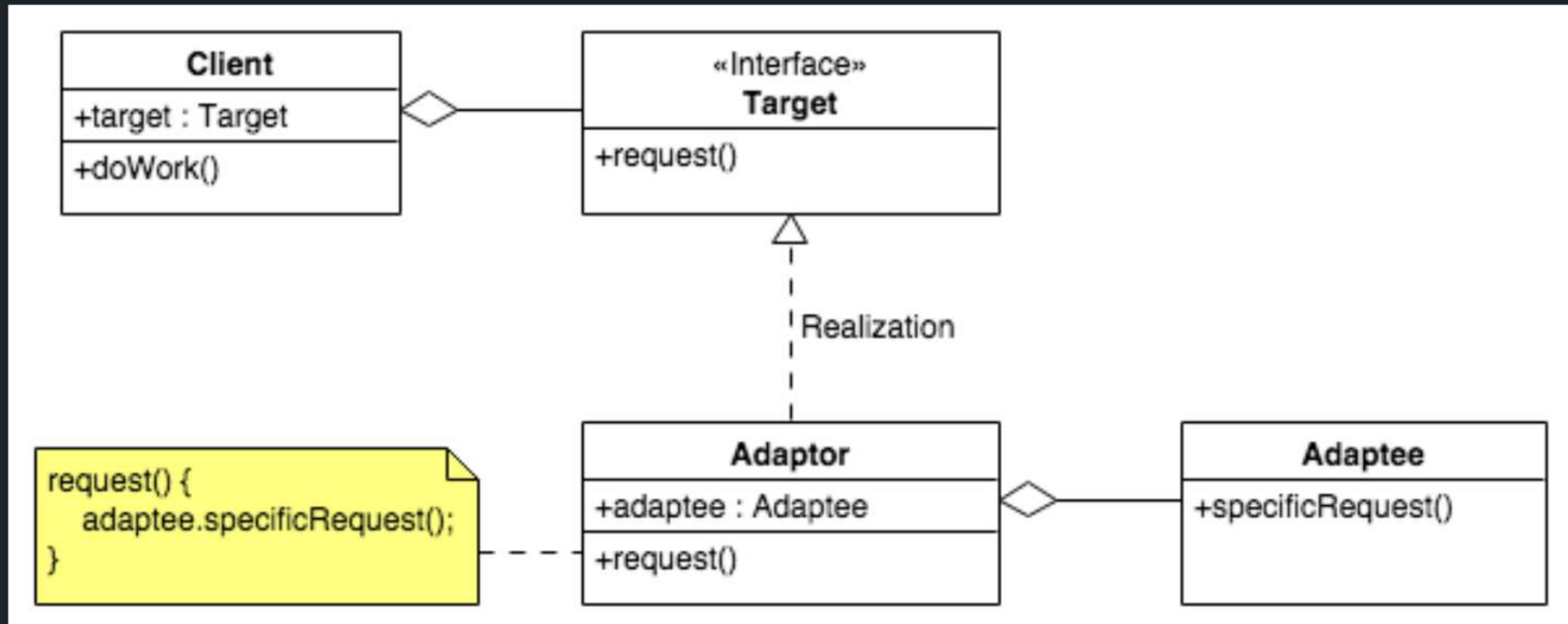
```
1  abstract class Department {  
2      constructor(public name: string) {  
3      }  
4  
5      printName(): void {  
6          console.log("Department name: " + this.name);  
7      }  
8  
9      // Debe implementarse en la clase  
10     abstract printMeeting(): void;  
11 }  
12
```


PATRONES DE DISEÑO MÁS USADOS EN TS

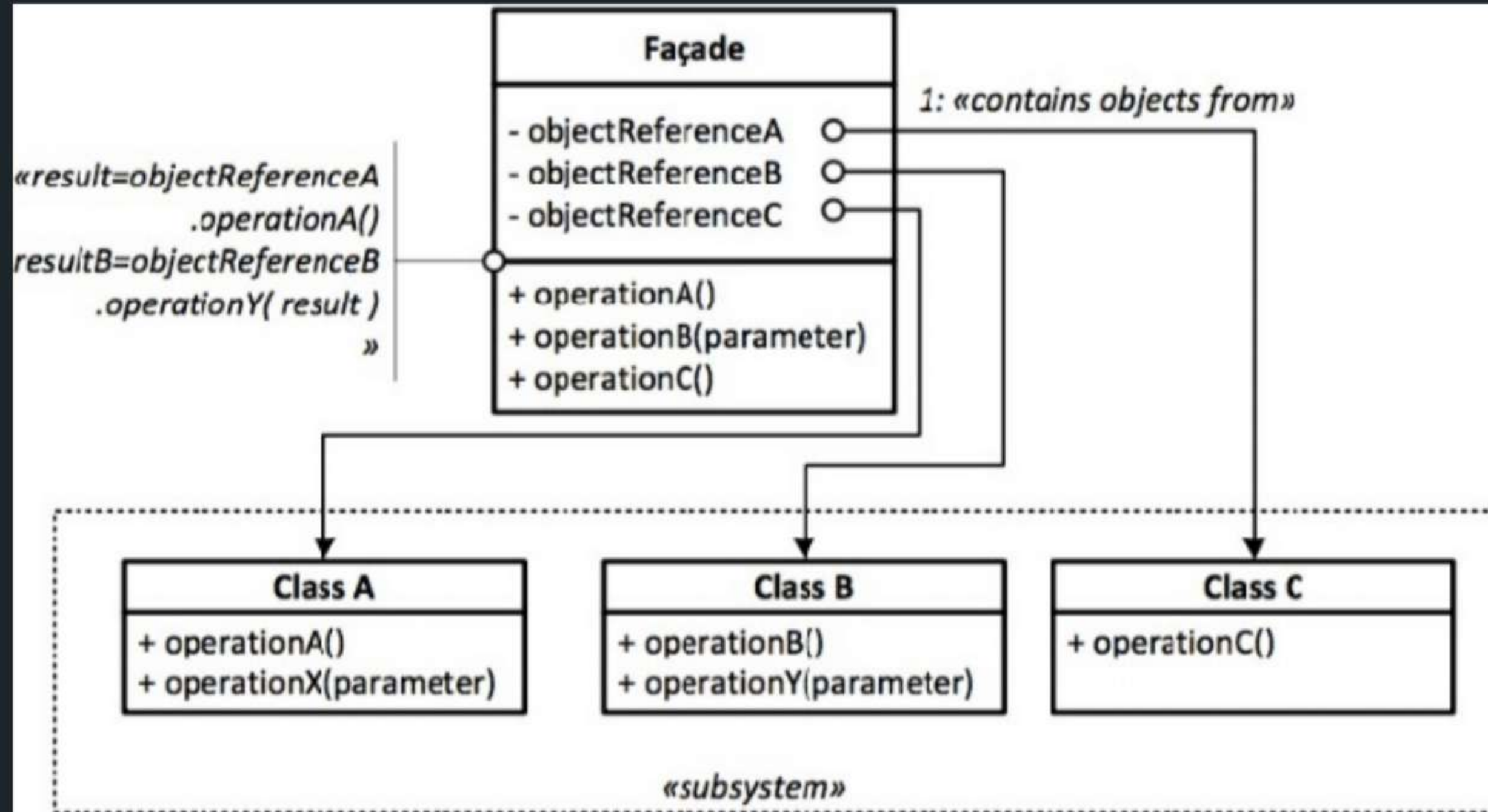
ABSTRACT FACTORY



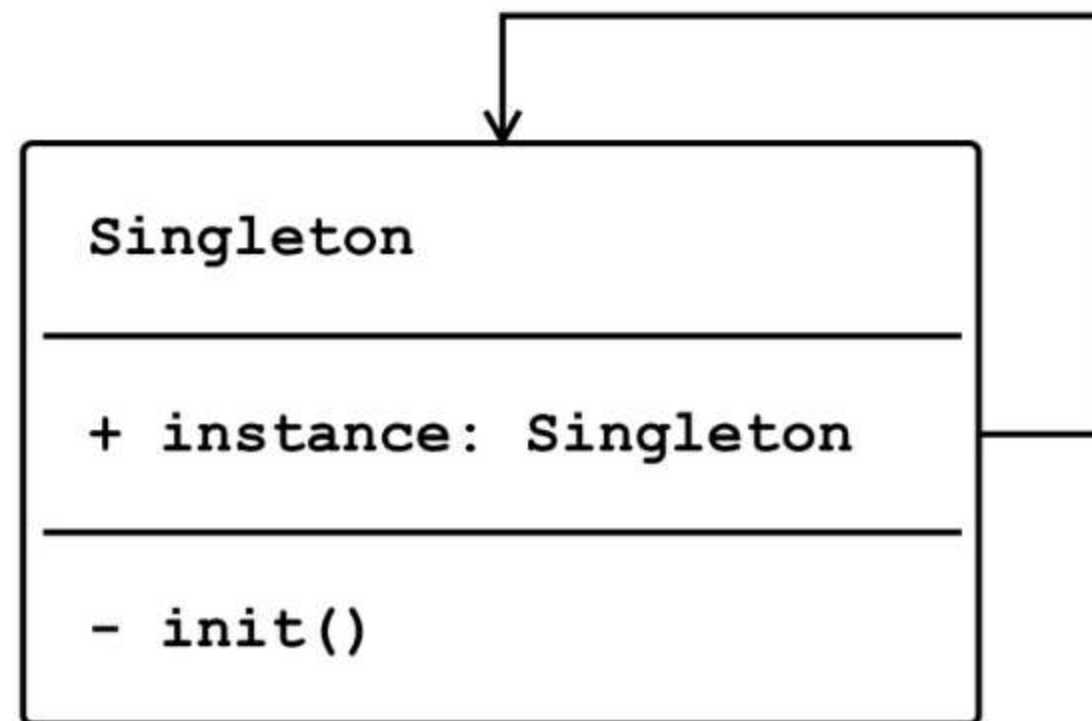
ADAPTER



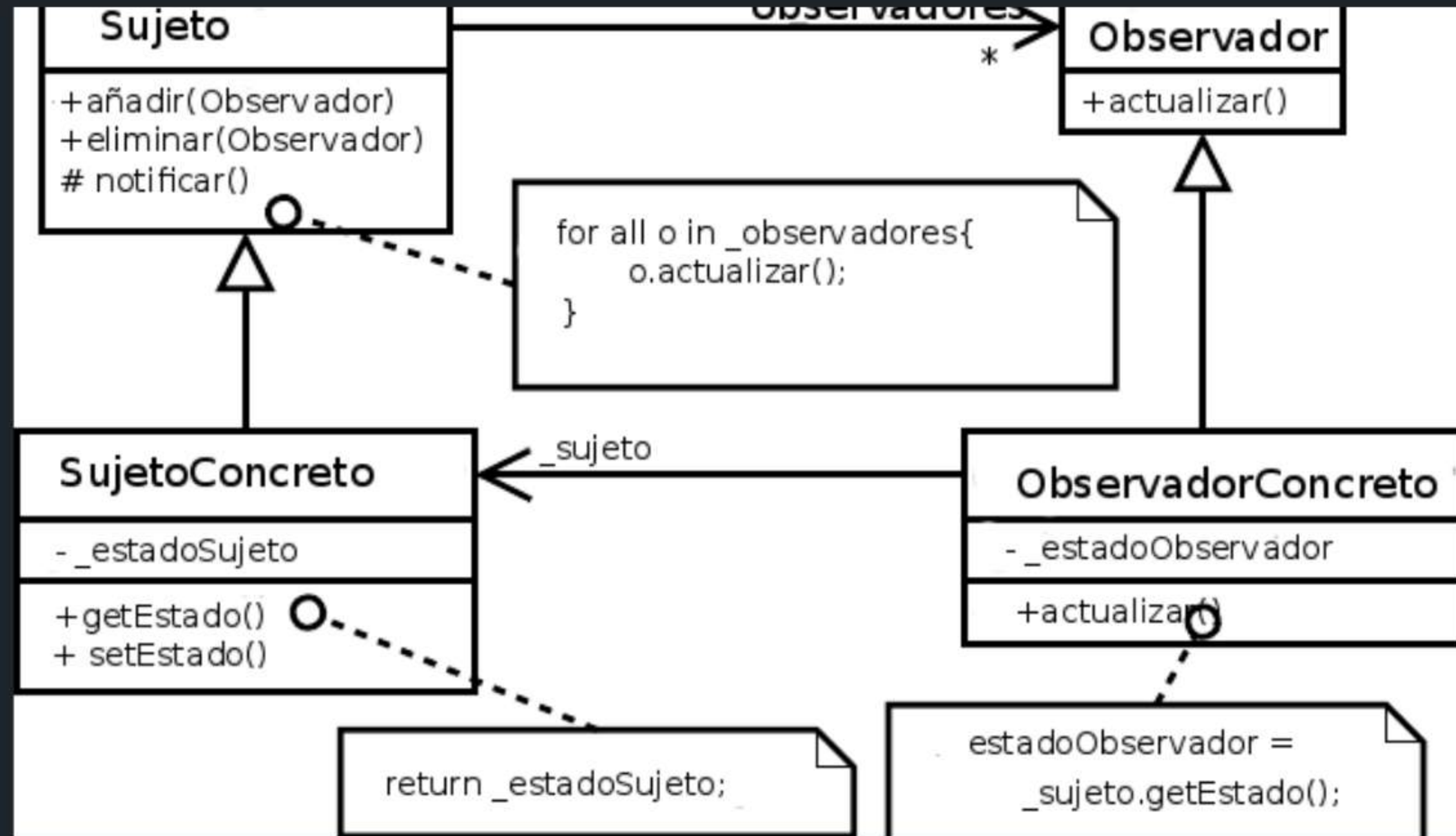
FACADE



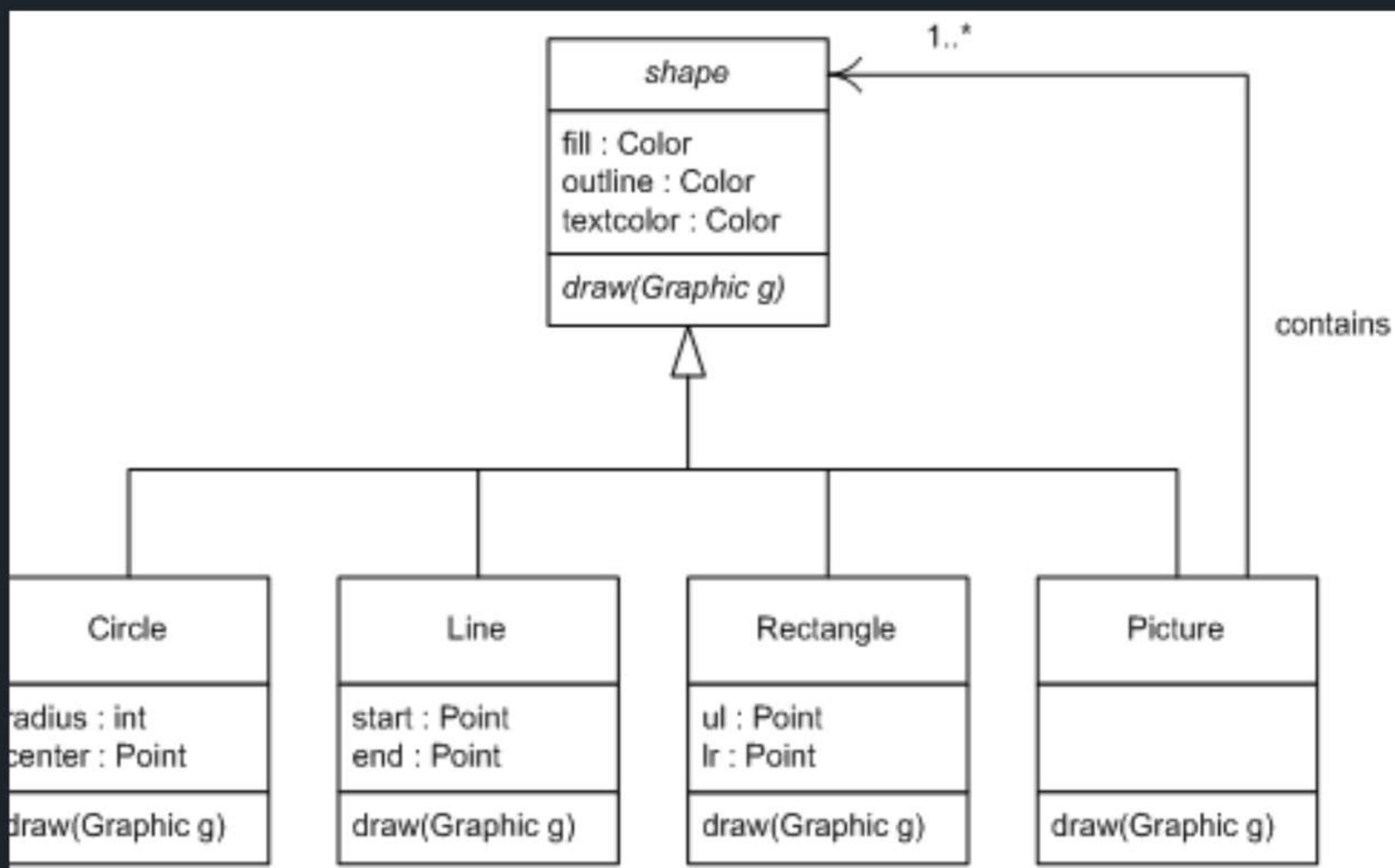
SINGLETON



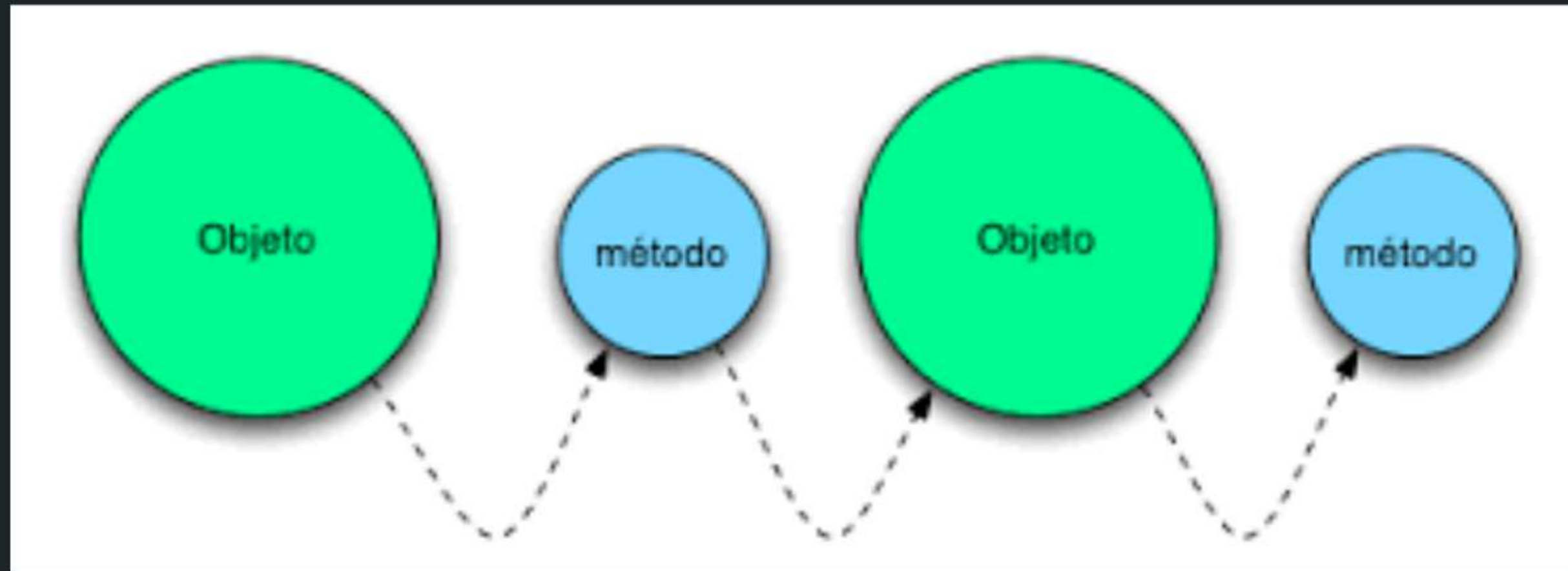
OBSERVABLE



COMPOSITE



FLUENT-INTERFACE



KAHOOT

KAHOOT.IT



ÍNDICE

CONTENIDOS

- tsconfig.json
- Namespaces
- Módulos
- Importación - exportación

DÍA 1

Introducción

Primeros pasos

DÍA 2

Declaraciones

Tipos de datos

DÍA 4

Configuración de proyecto

Modulos

DÍA 5

□

CONTEXTO DE COMPILACIÓN

El contexto de compilación es, básicamente, un término guay para definir el grupo de archivos que TypeScript va a analizar para determinar qué es válido y qué no. Junto con la información sobre qué archivos leer, el contexto de compilación contiene información acerca de qué opciones de compilación que serán utilizadas.

Para más info de todas las opciones de configuración:

typescriptlang.org/v2/en/tsconfig
json.schemastore.org/tsconfig

```
{ } tsconfig.json > ...  
1  {  
2      "compilerOptions": {  
3          "module": "commonjs",  
4          "target": "es5",  
5          "noImplicitAny": false,  
6          "sourceMap": false  
7      },  
8      "files": [  
9          "./some/file.ts"  
10     ],  
11     "include": [  
12         "./folder"  
13     ],  
14     "exclude": [  
15         "./folder/**/*.spec.ts",  
16         "./folder/someSubFolder"  
17     ]  
18 }  
19
```

NAMESPACES

Los espacios de nombre son útiles para agrupar objetos y tipos relacionados lógicamente.

```
1  namespace Utility {
2      export function log(msg: string) {
3          console.log(msg);
4      }
5      export function error(msg: string) {
6          console.error(msg);
7      }
8  }
9
10 namespace Tools {
11     export function log(msg: string) {
12         console.log(msg);
13     }
14     export function error(msg: string) {
15         console.error(msg);
16     }
17 }
18
```


MÓDULOS

Los módulos se ejecutan dentro de su propio alcance, no en el alcance global.

Cualquier declaración (como una variable, función, clase, alias de tipo o interfaz) se puede exportar añadiendo la palabra clave export.

La importación de una declaración exportada se realiza mediante uno de los import

```
1  //log.ts
2  ✓ export function log(msg: string) {
3      |     console.log(msg);
4      | }
5
6  // error.ts
7  import { log } from '../log';
8  ✓ export function error(msg: string) {
9      |     console.error(msg);
10     | }
11
12
13
```

KAHOOT

KAHOOT.IT



ÍNDICE

CONTENIDOS

- Proyecto
- Despedida

DÍA 1

Introducción TS

Primeros pasos

DÍA 2

Declaraciones

Tipos básicos

DÍA 3

Introducción P00

P00 en TS

DÍA 4

Configuración de proyecto

Módulos

Proyecto

Los usuarios se pueden hacer login social con Facebook, LinkedIn y Google.

Una aplicación para que los usuarios puedan crear playlist compartidas, donde pueden añadir y borrar canciones. Opción de rebovinar canción.

El login será social, pudiendo usar Facebook, LinkedIn y Google.



KAHOOT

KAHOOT.IT



It's a Wonderful World