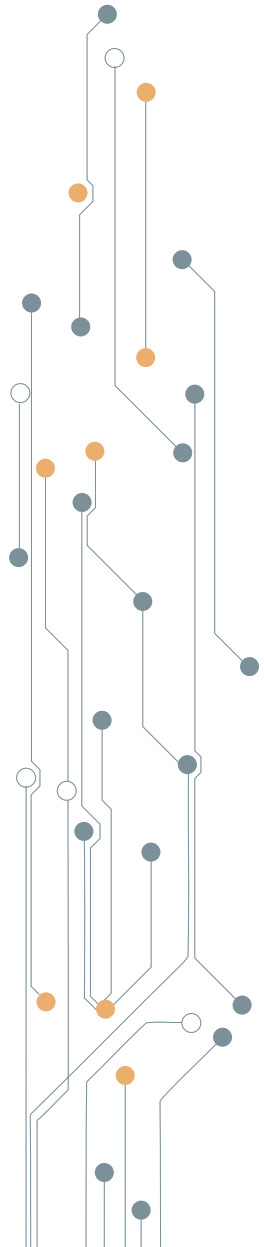
Interfaces

Telefónica
EDUCACIÓN DIGITAL

Índice



Interfaces

1. Definición de interfaces

Una interfaz es un conjunto de declaraciones de funciones. Sólo se declara la "firma" de las funciones, el tipo que retorna, el identificador con el que debe ser invocada y la lista de parámetros de la llamada. El código de las funciones se implementara en clases.

La palabra reservada para definir una interfaz es **interface**.

Esto se parece a las clases abstractas y las funciones abstractas. Pero las diferencias son:

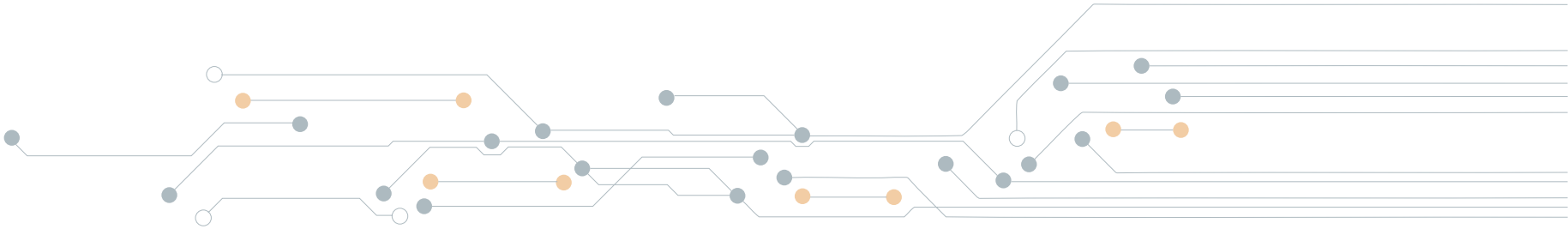
- Las funciones abstractas de las clases abstractas deben ser implementadas en las subclases. Una subclase sólo puede tener una superclase.
- Todas las funciones de una interface tienen que ser implementadas en una clase. Una clase puede implementar varias interfaces.
- En una clase abstracta puede haber funciones que no lo sean y por tanto este implementado su código.
- Una clase sólo puede tener una subclase pero puede implementar varias interfaces.

Ejemplo de definición de **interfaces**:

```
package com.juan.parejas.interfaces;

public interface Visualizar {
	public void verUnaPareja(int pos);
	public void verUnaParejaPorClave(Object
clave);
	public void listarParejas();
}
```

A las interfaces se les pueden aplicar los modificadores de acceso **public** y de **ámbito de paquete**.



2. Implementación de interfaces

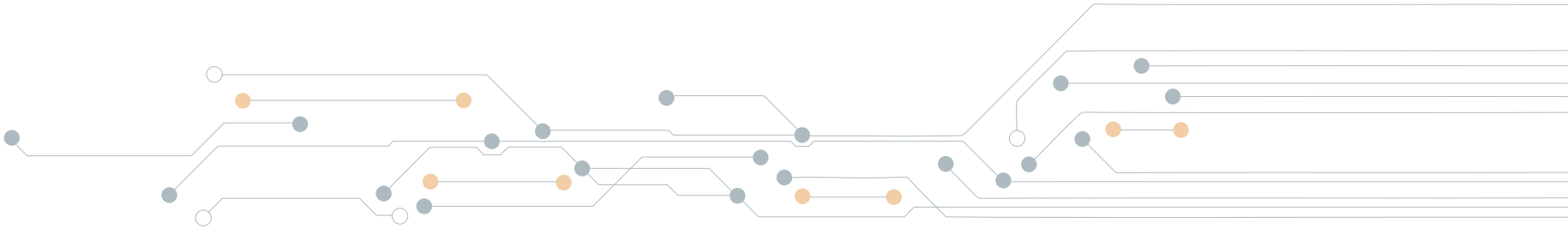
Cuando una clase tiene que implementar una **interface** debe utilizar la palabra reservada **implements**.
Una clase que implementa una **interface** esta obligada a definir todas las funciones de dicha **interface**.

El formato general de una clase que implementa una interfaz es el siguiente:

```
class <identificador> [extends <superclase>]
implements <lista_interfaces>{
      //datos
      //constructores y funciones
      //obligatoriamente definición de todas las
funciones de las interfaces
}
```

En el código se visualiza como una subclase declara que implementa dos interfaces:

```
public class Diccionario extends Parejas
                  implements Visualizar, Gestionar {
            //definición de variables, constructores
y
            //funciones de la clase
            //implementaciones de las dos interfaces
como public
}
```



En la imagen un ejemplo que muestra un diagrama de UML con una clase abstracta, dos subclases de esta y dos interfaces implementadas en las subclases.

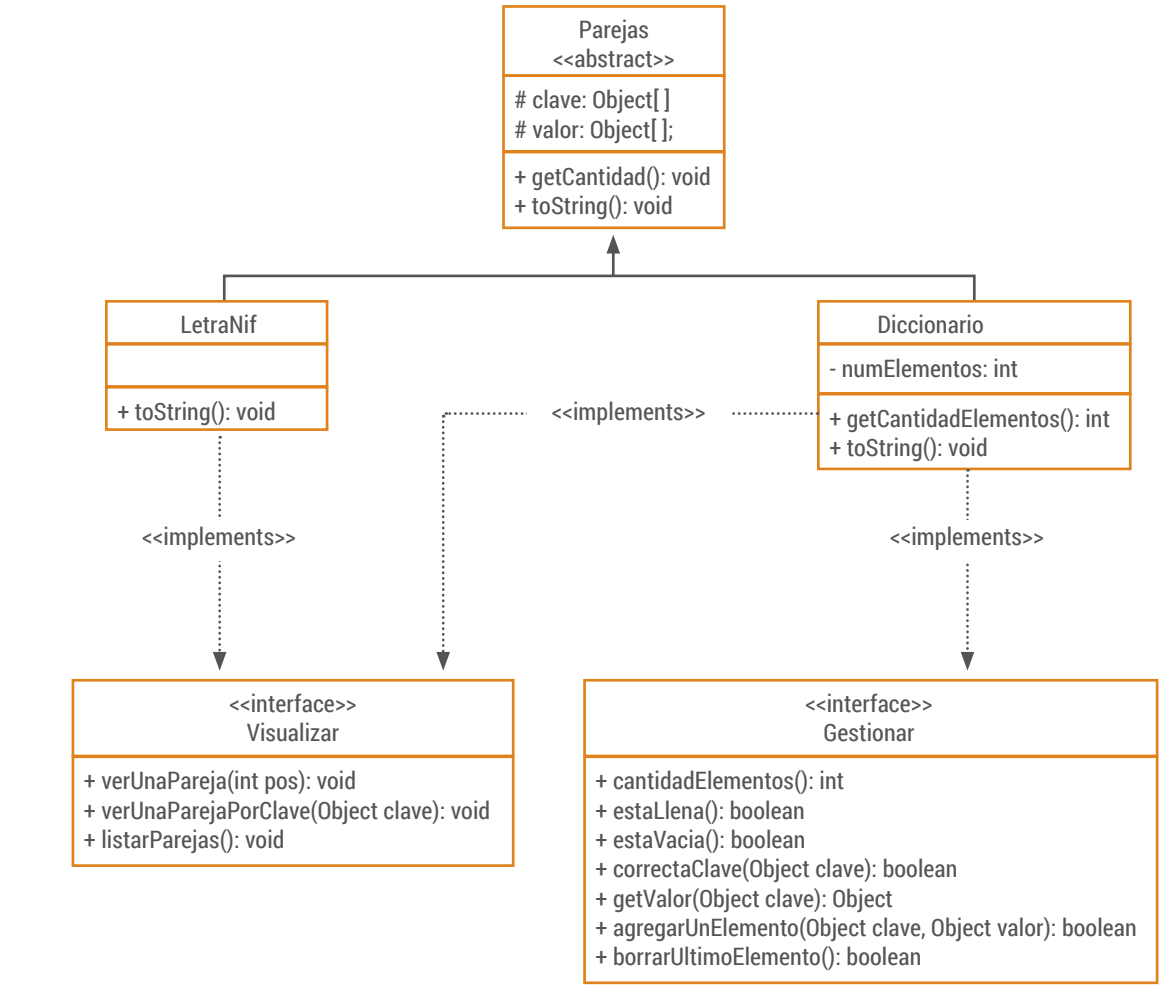


IMAGEN 8.1: DIAGRAMA UML CON CLASES E INTERFACES.

3. Métodos default

A partir de la versión 8 de JDK, se pueden definir en una interface funciones que implementen su código. Dichas funciones tiene que estar calificadas como default.

Los propósitos de la utilización de estos métodos predeterminados son:

- **Ampliación de una interfaz:** Si a una interface se le añade una función, hay que escribir su código en todas las funciones que implementen dicha interfaz, aunque el cambio no las afecte. Si la función que se añade es default, no hace falta definirla en ninguna clase excepto en las que se vean afectadas por el cambio.
- **Métodos opcionales:** Como las funciones default tienen código, su definición en las clases que implementan su interface es opcional y dependiente del uso de la misma.

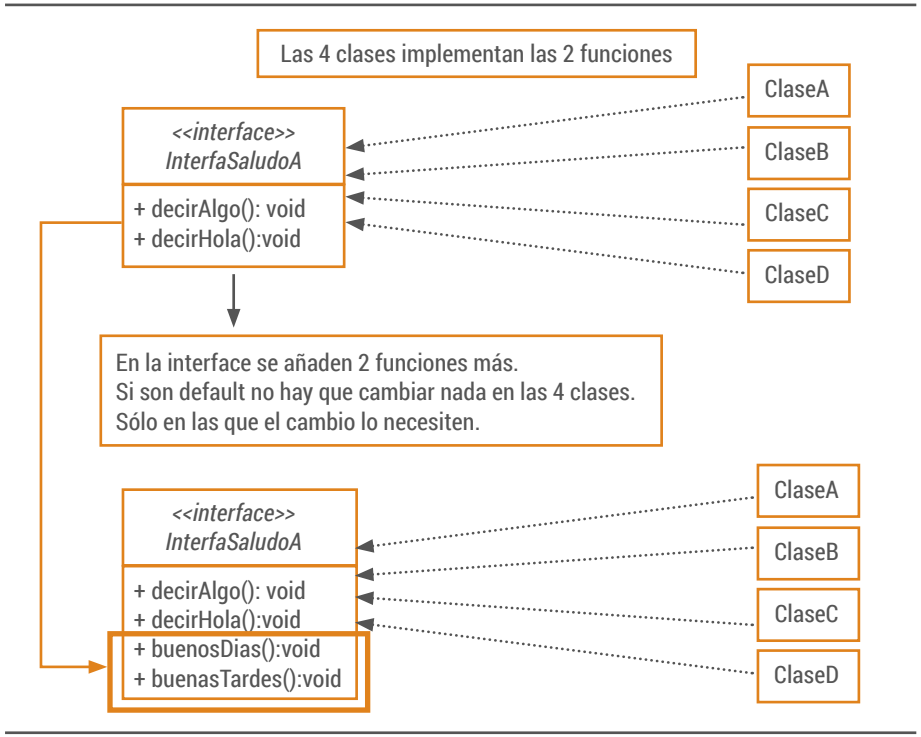


IMAGEN 8.2: FUNCIONES DEFAULT.

En cualquier caso, siempre habrá que seguir teniendo en cuenta que con las interfaces se debe de especificar el qué y no el cómo, por tanto las funciones default siempre serán funciones con un propósito especial y muy especifico.

4. Variables en interfaces

En las interfaces se pueden declarar variables de cualquier tipo, pero son implícitamente **public**, **static** y **final**. Son constantes públicas a las que se puede acceder desde las clases que las implementan.

En el ejemplo se utiliza una interface con variables:

```
interface LimitesSistema{
	int XMAX=300;
	int XMIN=0;
	int YMAX=500;
	int YMIN=0;
}
class Punto implements LimitesSistema{
	private int x;
	private int y;
	private normalizar(int x, int y){
	//con esta función se asegura que ningún punto estará
	//fuera de los límites del sistema
		if (x > XMAX) this.x = XMAX;
		else   if (x < XMIN) this.x = XMIN;
				else this.x = x;
		if (y > YMAX) this.y = YMAX;
		else          if (y < YMIN) this.y = YMIN;
				else this.y = y;
	}

	public Punto (int x, int y){
		normalizar(x, y);
	}
	//otros constructores y otras funciones
}

public class JugarConPuntos{
	public static void main(String [] args){
		Punto pt1 = new Punto(2000,3000);
		//Más sentencias
	}
```

5. Referencias a interfaces

Cuando una clase implementa una interface, los objetos instanciados de dicha clase pueden ser utilizados por referencias a la interface implementada.

Con dicha referencia a interfaz sólo se podrán invocar al objeto las funciones declaradas a la interfaz. Es un mecanismo parecido a utilizar una referencia a superclase para acceder a objetos instanciados de subclase.

Por ejemplo:

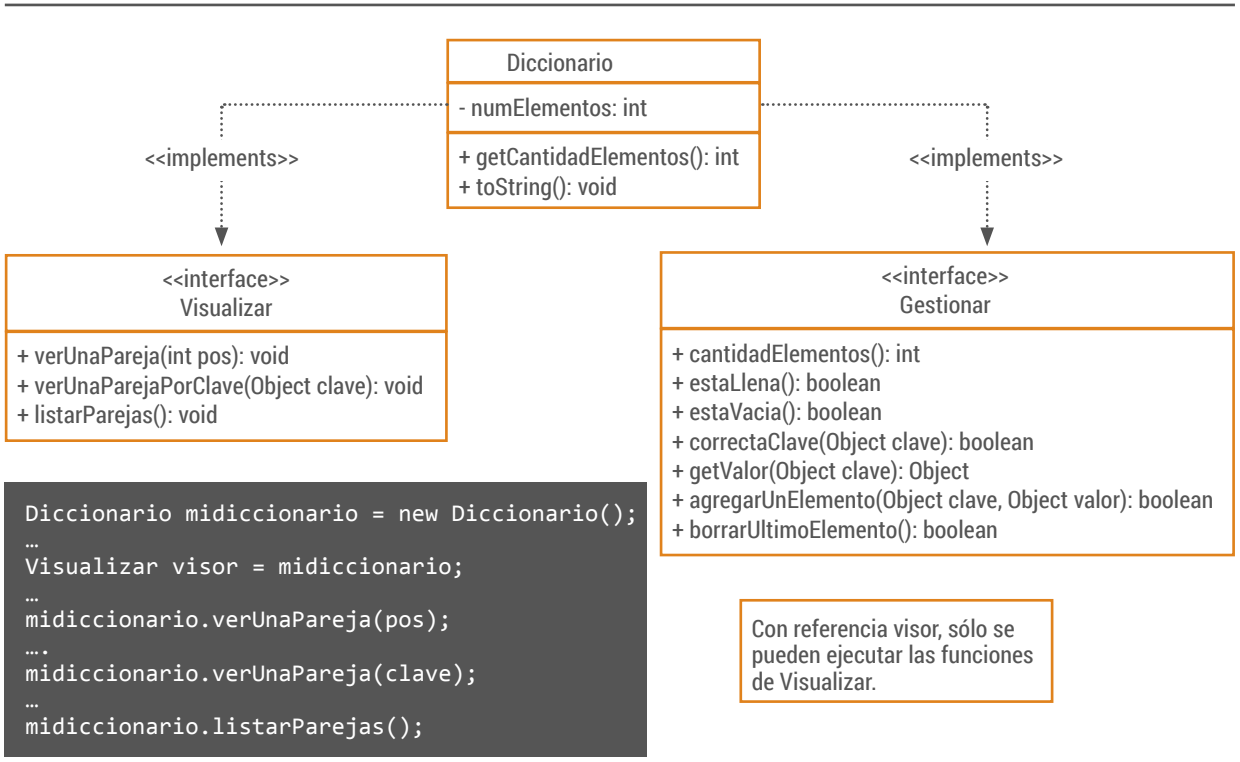


IMAGEN 8.3: REFERENCIAS A INTERFACES.

Telefónica

EDUCACIÓN DIGITAL