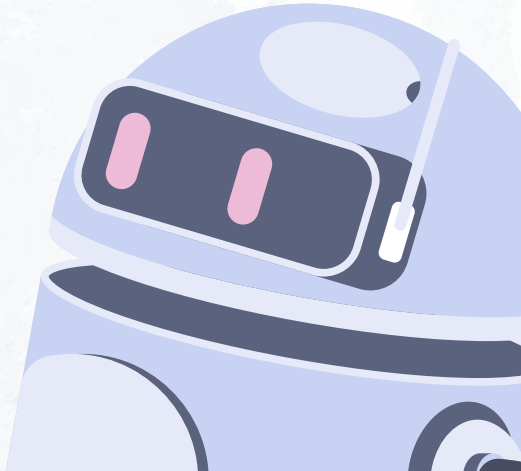
Interfaces en Java

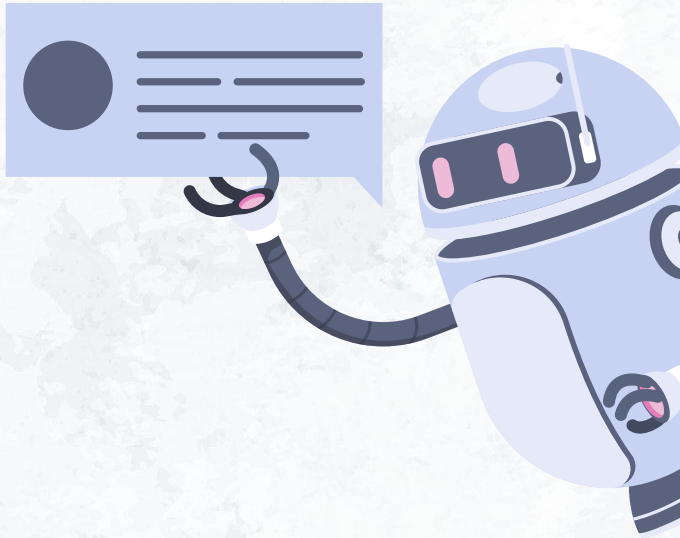


Índice

01 →

¿Qué es y cómo se declara una interfaz en Java?

(Java) = Introducción a las interfaces

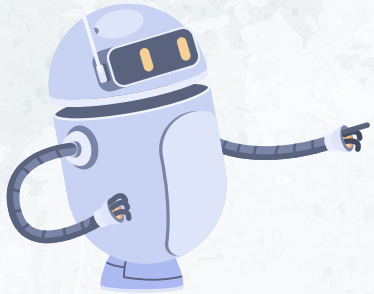


¿Qué es una interfaz?

Esta es una colección de métodos abstractos que pueden ser implementados por cualquier clase que la utilice. Es una estructura que define un conjunto de comportamientos que una clase puede proporcionar.

¿Cómo se declara una interfaz?

Una interfaz se declara usando la palabra clave “interface” seguida del nombre de la interfaz. Después se definen los métodos abstractos los cuales no tienen una implementación concreta, sólo definen la firma de los métodos.



Ejemplo Caso Vehiculo

A continuación se presenta el caso Vehículo, el cual es una interfaz que define métodos y comportamientos que se esperan de cualquier vehículo. Después se crea la clase Automóvil y Bicicleta las cuales heredan los métodos abstractos de la clase Vehículo.

Clases Vehículo, Automóvil y Bicicleta

```java
package Modelo;
// Implementación de la interfaz en una clase
public class Automovil implements Vehiculo {
    private int velocidad;

    public void acelerar(int velocidad) {
        this.velocidad += velocidad;
        System.out.println("El automóvil acelera a " + velocidad + " km/h");
    }

    public void frenar() {
        this.velocidad = 0;
        System.out.println(x:"El automóvil se detiene en seco.");
    }

    public int obtenerVelocidad() {
        return velocidad;
    }
}
```

```java
package Modelo;

// Implementación de la interfaz en otra clase
public class Bicicleta implements Vehiculo
{
    private int velocidad;

    public void acelerar(int velocidad) {
        this.velocidad += velocidad;
        System.out.println("La bicicleta acelera a " + velocidad + " km/h");
    }

    public void frenar() {
        this.velocidad = 0;
        System.out.println(x:"La bicicleta se detiene en seco.");
    }

    public int obtenerVelocidad() {
        return velocidad;
    }
}
```

Modelo > J Vehiculo.java > ...

```java
1   package Modelo;
2
3   // Definición de la interfaz Vehiculo
4   public interface Vehiculo
5   {
6       void acelerar(int velocidad);
7       void frenar();
8       int obtenerVelocidad();
9   }
10
```

Clase Main y ejecución del programa

```java
package Ejecutable;

import Modelo.*;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Vehiculo auto = new Automovil();
        Vehiculo bicicleta = new Bicicleta();
        System.out.println(x:"Paco agarra una bicicleta y su primo le sigue con su carro");

        auto.acelerar(velocidad:60);
        bicicleta.acelerar(velocidad:20);

        System.out.println("Velocidad del automóvil: " + auto.obtenerVelocidad() + " km/h");
        System.out.println("Velocidad de la bicicleta: " + bicicleta.obtenerVelocidad() + " km/h");

        bicicleta.frenar();
        auto.frenar();
    }
}
```

```
Paco agarra una bicicleta y su primo le sigue con su carro
El automóvil acelera a 60 km/h
La bicicleta acelera a 20 km/h
Velocidad del automóvil: 60 km/h
Velocidad de la bicicleta: 20 km/h
La bicicleta se detiene en seco.
El automóvil se detiene en seco. (El primo de Paco acaba de atropellar a Paco)
```

02 →

Métodos e implementación de múltiples interfaces

Tipos de métodos en interfaces

(a) Métodos abstractos ⟶

Son aquellos que no tienen una implementación concreta y sólo proporcionan la firma del método (su nombre, parámetro y/o tipo de retorno).

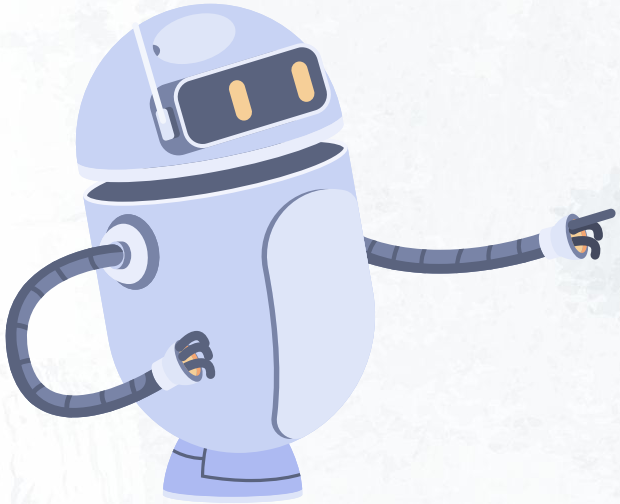
(b) Método default ⟶

Estos métodos tienen una implementación predeterminada en la interfaz y pueden ser utilizados directamente por las clases que implementan la interfaz. Estos se definen usando la palabra clave “default” antes de la declaración del método.

(c) Métodos estáticos ⟶

Estos métodos se pueden llamar directamente utilizando el nombre de la interfaz, sin necesidad de crear una instancia de la interfaz. Los métodos estáticos en una interfaz se definen utilizando la palabra clave “static” antes de la declaración del método.

Ejemplo Caso Teléfono



En este ejemplo, se crean las interfaces "Encendible", "Recargable" y "Conectable" que contienen un método abstracto, un método default y un método estático cada una. La clase "Teléfono" implementa las tres interfaces y proporciona las implementaciones correspondientes para los métodos abstractos.

Clase Teléfono, Conectable, Encendible y Recargable

```java
package Modelo;

public class Telefono implements Encendible, Recargable, Conectable
{
    public void encender()
    {
        System.out.println(x:"Encendiendo el teléfono");
    }

    public void recargar()
    {
        System.out.println(x:"Recargando el teléfono");
    }

    public void conectar()
    {
        System.out.println(x:"Conectando el teléfono a otros dispositivos");
    }
}
```

```java
package Modelo;

public interface Encendible
{
    void encender();

    default void apagar()
    {
        System.out.println(x:"Apagando el dispositivo");
    }

    static void reiniciar()
    {
        System.out.println(x:"Reiniciando el dispositivo");
    }
}
```

```java
package Modelo;

public interface Recargable
{
    void recargar();

    default void verificarNivelBateria()
    {
        System.out.println(x:"Verificando nivel de batería");
    }

    static void mostrarMensajeCargaCompleta()
    {
        System.out.println(x:"El dispositivo está completamente cargado");
    }
}
```

```java
package Modelo;

public interface Conectable
{
    void conectar();

    default void desconectar()
    {
        System.out.println(x:"Desconectando el dispositivo");
    }

    static void mostrarMensajeConexionExitosa()
    {
        System.out.println(x:"La conexión se ha establecido correctamente");
    }
}
```

Clase Main y Ejecución del programa

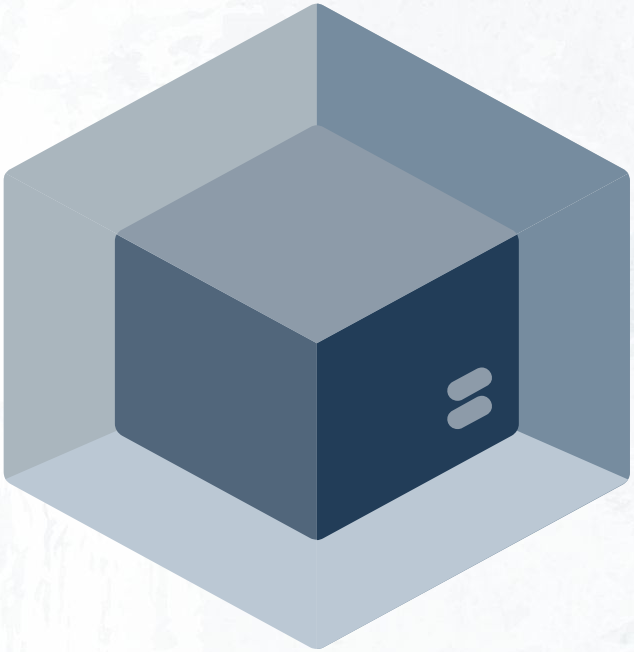
```java
package Ejecutable;
import Modelo.*;
public class Main
{
    Run | Debug
    public static void main(String[] args) {
        Telefono teléfono = new Telefono();

        teléfono.encender();
        Encendible.reiniciar();

        teléfono.recargar();
        teléfono.verificarNivelBateria();
        Recargable.mostrarMensajeCargaCompleta();

        teléfono.conectar();
        Conectable.mostrarMensajeConexionExitosa();
        teléfono.desconectar();

        teléfono.apagar();
    }
}
```

```
Encendiendo el teléfono
Reiniciando el dispositivo
Recargando el teléfono
Verificando nivel de batería
El dispositivo está completamente cargado
Conectando el teléfono a otros dispositivos
La conexión se ha establecido correctamente
Desconectando el dispositivo
Apagando el dispositivo
```

03 →

Interfaces funcionales y expresiones lambdas

¿Qué es una interfaz funcional?



Es un tipo especial de interfaz que contiene un solo método abstracto. Estas interfaces se utilizan principalmente en el contexto de la programación funcional y son la base para el uso de expresiones lambda en Java.

¿Qué es una expresión lambda?

Esta es una forma concisa y funcional de representar una función anónima. La expresión lambda se compone de los siguientes elementos:

(a) Parámetros

Son los valores que se pasan a la función lambda. Se especifican entre paréntesis y pueden ser opcionales si la función no toma ningún parámetro.

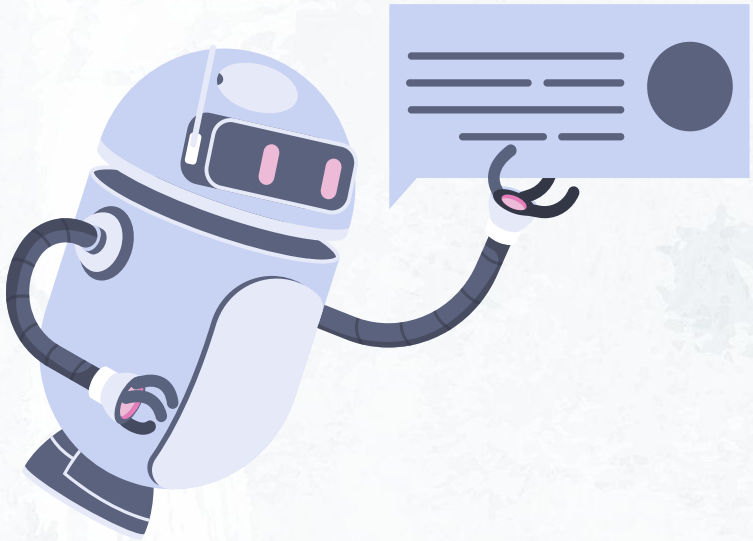
(b) Flecha

Se utiliza el operador '→' para separar los parámetros del cuerpo de la expresión lambda.

(c) Cuerpo

Es la implementación de la función lambda, que puede ser una sola expresión o un bloque de código. Si el cuerpo es una sola expresión, no es necesario utilizar llaves '{}' y el valor de la expresión se devuelve automáticamente.

Ejemplo Caso Operación



En este ejemplo, hemos definido una interfaz funcional llamada "Operación" que contiene un único método abstracto llamado "realizarOperacion()" que toma dos enteros como parámetros y devuelve un entero. Luego, utilizamos una expresión lambda (a, b) → a + b para implementar la función de suma. La expresión lambda toma dos enteros, los suma y devuelve el resultado.

Clase Operación

```java
package Modelo;

// Declaración de una interfaz funcional
public interface Operacion
{
    int realizarOperacion(int a, int b);
}
```

Clase Main y Ejecución del programa

```java
package Ejecutable;
import Modelo.*;
public class Main {
    Run | Debug
    public static void main(String[] args) {
        // Utilización de una expresión lambda
        Operacion suma = (a, b) -> a + b;

        int a = 2;
        int b = 3;

        int resultado = suma.realizarOperacion(a, b);
        System.out.println("Resultado de la suma entre" + a + " y " + b + " es:");
        System.out.println(resultado);
    }
}
```

```
Resultado de la suma entre 2 y 3 es:
5
```

04 →

Recapitulación y conclusiones

Recapitulación sobre conceptos clave

(a) Definición de interfaces ⟶

Las interfaces son estructuras que permiten definir un contrato o conjunto de métodos que deben ser implementados por las clases que las utilizan. Una interfaz en Java se define mediante la palabra clave "interface".

(b) Características y beneficios ⟶

Las interfaces proporcionan abstracción y modularidad en el código, permitiendo la separación de preocupaciones y el diseño basado en contratos. Algunos beneficios de utilizar interfaces incluyen la reutilización de código, la flexibilidad en la implementación y la capacidad de trabajar con múltiples implementaciones.

(c) Declaración y uso de interfaces ⟶

La sintaxis de declaración de una interfaz en Java incluye la lista de métodos abstractos que deben ser implementados. Las clases pueden implementar una interfaz utilizando la palabra clave "implements" y proporcionando la implementación de los métodos requeridos.

(d) Relación entre interfaces y herencia ⟶

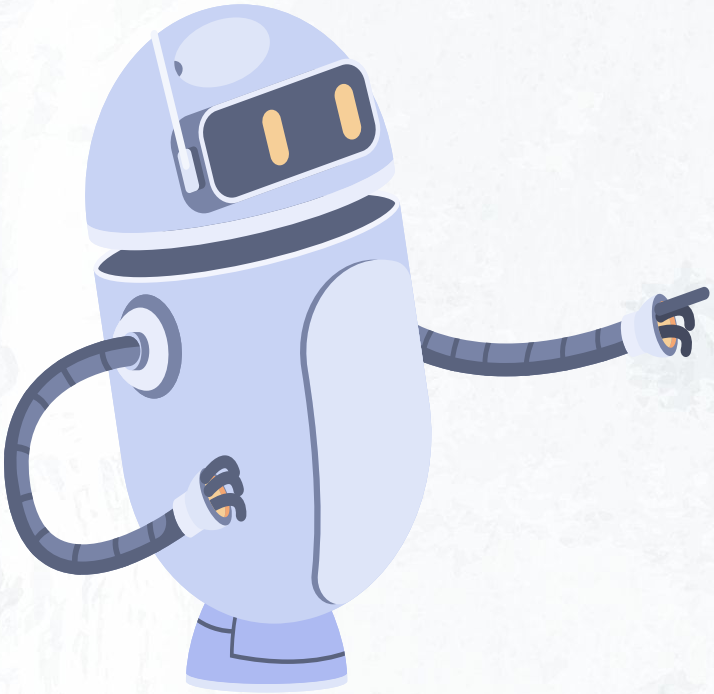
En Java, una clase puede implementar múltiples interfaces, lo que permite heredar y utilizar funcionalidades de varias fuentes diferentes. Esta capacidad de múltiple implementación de interfaces es una ventaja sobre la herencia de clases, que solo permite la herencia de una única clase.

(e) Métodos en interfaces ⟶

Los métodos en una interfaz pueden ser abstractos, lo que significa que solo se define su firma y deben ser implementados en las clases que las utilizan. Además, las interfaces también pueden contener métodos default y métodos estáticos que proporcionan implementaciones predeterminadas y utilidades adicionales.

(f) Interfaces funcionales y lambdas ⟶

Las interfaces funcionales son un tipo especial de interfaz que contiene un único método abstracto. Se utilizan en el contexto de la programación funcional y se pueden implementar utilizando expresiones lambda en Java, lo que proporciona una forma concisa de trabajar con interfaces funcionales.



En conclusión, las interfaces en Java son una herramienta fundamental para la definición de contratos y la implementación de la abstracción. Proporcionan flexibilidad, modularidad y reutilización de código en el desarrollo de software, permitiendo una programación más limpia, modular y extensible.

Gracias

Hecho por: Juan Sebastian Puerto

Universidad Industrial de Santander