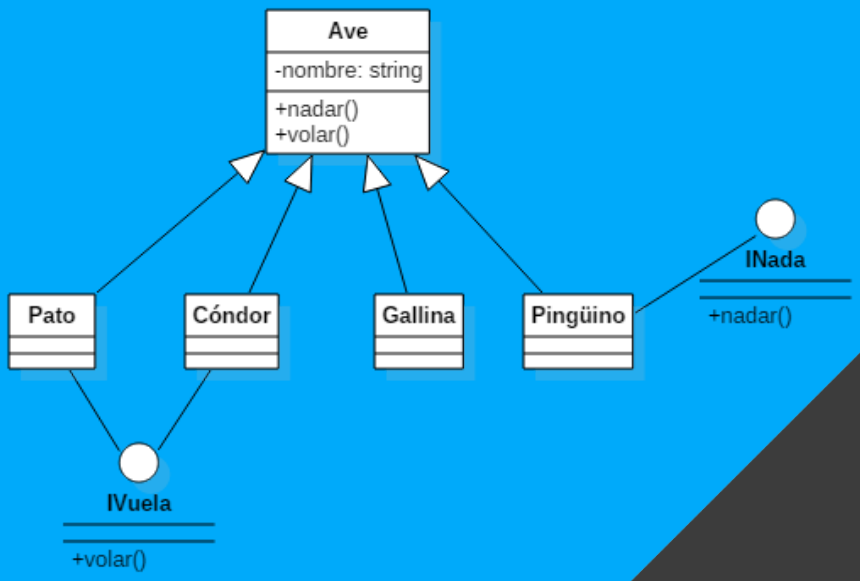
Java

//Interfaces



UTEC
Universidad Tecnológica

LICENCIATURA EN TECNOLOGÍAS
DE LA INFORMACIÓN

Interfaces

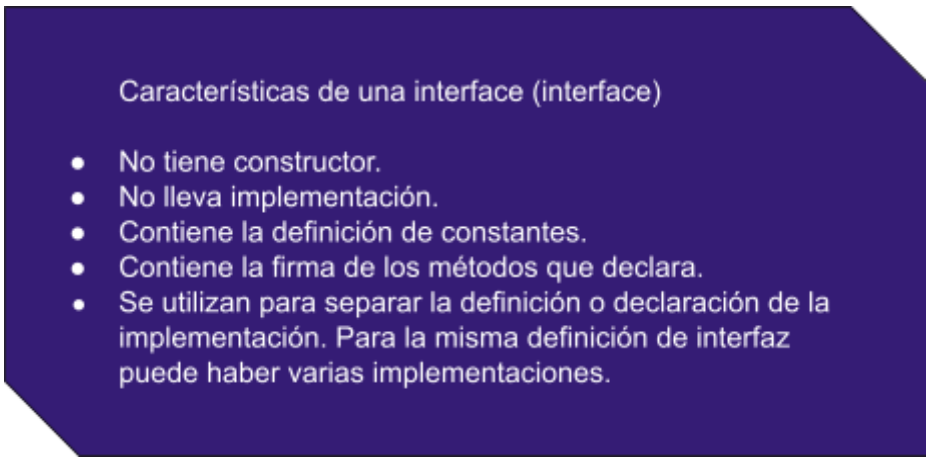
En este documento veremos a qué llamamos interfaz, como se define y cuándo debemos utilizar al programar. Una interfaz es un conjunto de métodos y de constantes cuya funcionalidad es determinar el funcionamiento de una clase. Es decir, funciona como un molde o como una plantilla a ser implementada luego por otras clases.

Definición de interfaces

Sintaxis de definición de interfaz:

```java
public interface Figura{
	static final double PI = 3.1416;
	public double area();
}
```

Como vemos en lugar de la palabra class debemos utilizar interface, para indicar que estamos definiendo una interfaz de nombre Figura. Vemos que esta no tiene constructor, tiene una constante PI y su único método es area, y se puede observar que la implementación de este no está dentro de la interfaz, es más, ni se abre un bloque de código ({}).



Implementación de interfaces

Las interfaces, a diferencia de las clases, no pueden ser instanciadas, ya que por sí mismas no tienen constructores ni implementación. Entonces, ¿cómo utilizamos una interfaz?

Para definir una clase que implementa una interfaz usamos "**implements**". **Toda clase que implemente una interfaz, debe implementar obligatoriamente los métodos definidos en esta.** Por lo tanto, una clase que implemente la interfaz Figura debe implementar la función area.

Cabe aclarar que es posible heredar e implementar al mismo tiempo, es decir que una clase que implementa una interfaz, sea hija de otra clase.

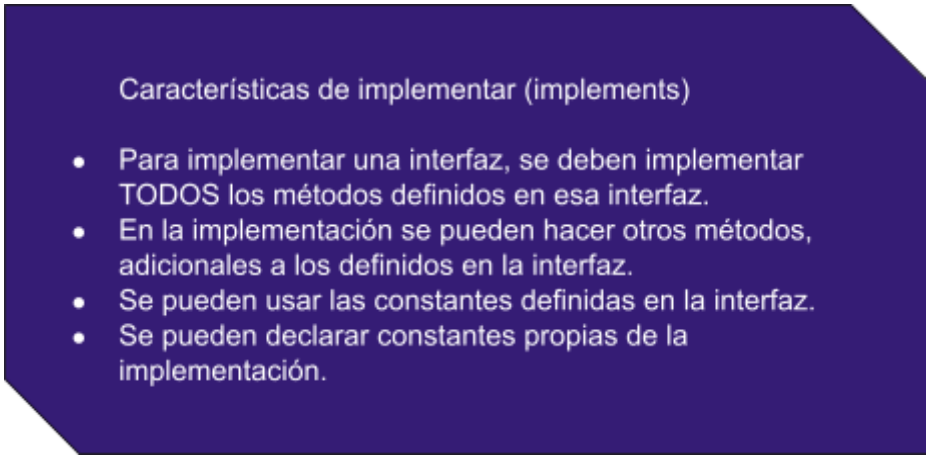
Java

```java
public class Cuadrado implements Figura {
  private double lado;

  public Cuadrado(double lado) {
    this.lado = lado;
  }

  @Override
  public double area() {
    return lado * lado;
  }
}

public class Rectangulo implements Figura {
  private double base;
  private double altura;

  public Rectangulo(double base, double altura) {
    this.base = base;
    this.altura = altura;
  }

  @Override
  public double area() {
    return base * altura;
  }
}
```

Creamos las clases Cuadrado, Rectangulo y Circulo que implementan la interfaz Figura, estas tienen "implements Figura" por lo que deben implementar todos los métodos definidos en la

interfaz Figura. En este caso, es únicamente area. Donde cada implementación de este método es distinta y se encuentra en cada una de las clases Cuadrado y Rectangulo.

Las clases que implementan una interfaz, como vimos, deben implementar sus métodos, pero también **pueden tener métodos y atributos propios**, estas no están restringidas a implementar solo lo que dicta la interfaz.



Implementación de múltiples interfaces

Una clase puede implementar más de una interfaz. Por ejemplo, nos definimos la interfaz “Dibujable” con el método “dibujar”:

```java
public interface Dibujable {
  void dibujar();
}
```

Y modificamos la clase “Rectangulo” para que también implemente “Dibujable”:

```java
public class Rectangulo implements Figura, Dibujable {
  private double base;
  private double altura;

  public Rectangulo(double base, double altura) {
    this.base = base;
    this.altura = altura;
  }

  @Override
  public double area() {
    return base * altura;
  }

  @Override
  public void dibujar() {
    System.out.println(" _________________");
    System.out.println("|                 |");
    System.out.println("|_________________|");
  }
}
```

Vemos como “Rectangulo” ahora debe implementar el método “dibujar”.

Instancia de clase que implementa una interfaz

Cuando instanciamos una clase en java lo hacemos de la siguiente forma:

```java
Cuadrado cuadrado = new Cuadrado(5);
```

Pero como dijimos anteriormente una interfaz no puede ser instanciada por sí misma, ya que no tiene constructor. Lo que sí podemos instanciar y tiene constructor son las clases que implementan dicha interfaz.

Por lo tanto, a estas clases las podremos instanciar y al ser la implementación de una interfaz las podremos tratar de la siguiente manera:

```java
Figura cuadrado = new Cuadrado(5);
```

Como vemos ponemos como tipo de dato a utilizar la interfaz, y como constructor, el constructor de una clase que la implementa.

¿Cuándo es oportuno utilizar ese tipo de instanciación?

Cuando queremos tratar de forma genérica las clases que implementan a una interfaz. Es decir, en este caso, cuando no nos interesa si estamos tratando con un cuadrado, círculo o rectángulo, ya que vamos a utilizar únicamente las funciones declaradas en la interfaz.

```java
Figura cuadrado = new Cuadrado(5);
Figura rectangulo = new Rectangulo(4, 3);

double areaCuadrado = cuadrado.area();
double areaRectangulo = rectangulo.area();
```

También nos es útil esta forma de instanciación cuando por ejemplo queremos tener una colección de Figuras (que puedan ser círculos, cuadrados y rectángulos). Ya que, si por ejemplo definimos una colección de círculos, no podremos agregar allí cuadrados y viceversa.

Por lo que al hacer una de figuras, nos abstraemos de qué figura es y podemos mantener todas en una misma colección.

```java
public static void main (String[] args) {
        //Cuadrado cuadrado = new Cuadrado(5);
        Figura cuadrado = new Cuadrado(5);
        Figura rectangulo = new Rectangulo(4, 3);

        Figura[] listaFiguras = new Figura[3];
        listaFiguras[0] = cuadrado;
        listaFiguras[1] = rectangulo;
}
```

Importante

Si modificamos una interfaz, por ejemplo: le agregamos un nuevo método, debemos modificar todas nuestras clases que la implementan de forma que provean la implementación de este método nuevo.