El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 2

Semana 3

Herencia, Abstracción y Polimorfismo

Lectura 4 - Interfaces

Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.

Interfaces

Como se mencionó anteriormente las clases abstractas generan funciones que no necesitan ser implementadas por la clase padre. Las interfaces por su lado, aunque tienen un comportamiento similar, difieren con las clases abstractas en que sus métodos no se implementan, por tanto, una interface define el comportamiento de las clases que la implementen.

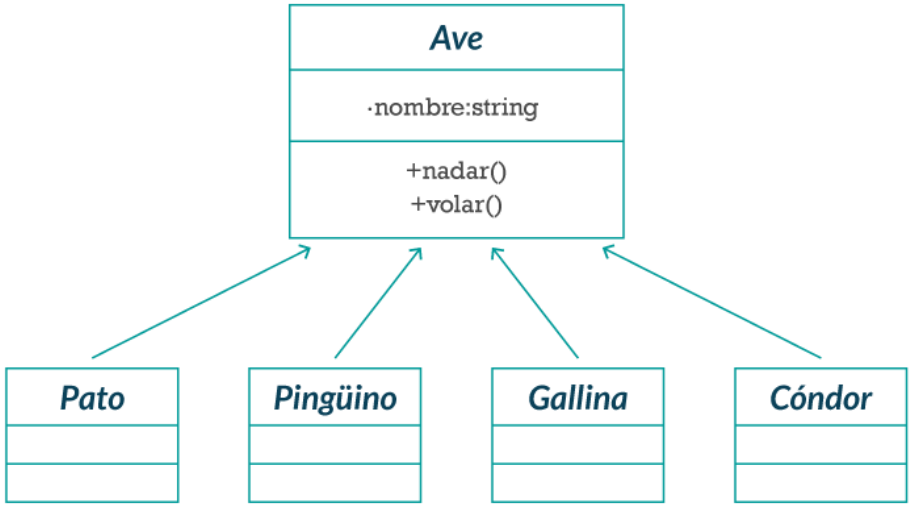
La interfaz realmente es una clase abstracta pura, donde todos los métodos son abstractos y generalmente no tiene atributos ni implementación pues no relaciona un espacio de almacenamiento, permitiendo combinar múltiples interfaces, algo que no permiten las clases abstractas.

En este sentido las Interfaces serian clases completamente abstractas con un conjunto de métodos abstractos y propiedades constantes que al igual que la clase define qué se debe hacer y muestran el código para dichos métodos, pero no su implementación y así son las clases que implementen estas interfaces las que finalmente describan las acciones que puede ejecutar un determinado objeto.

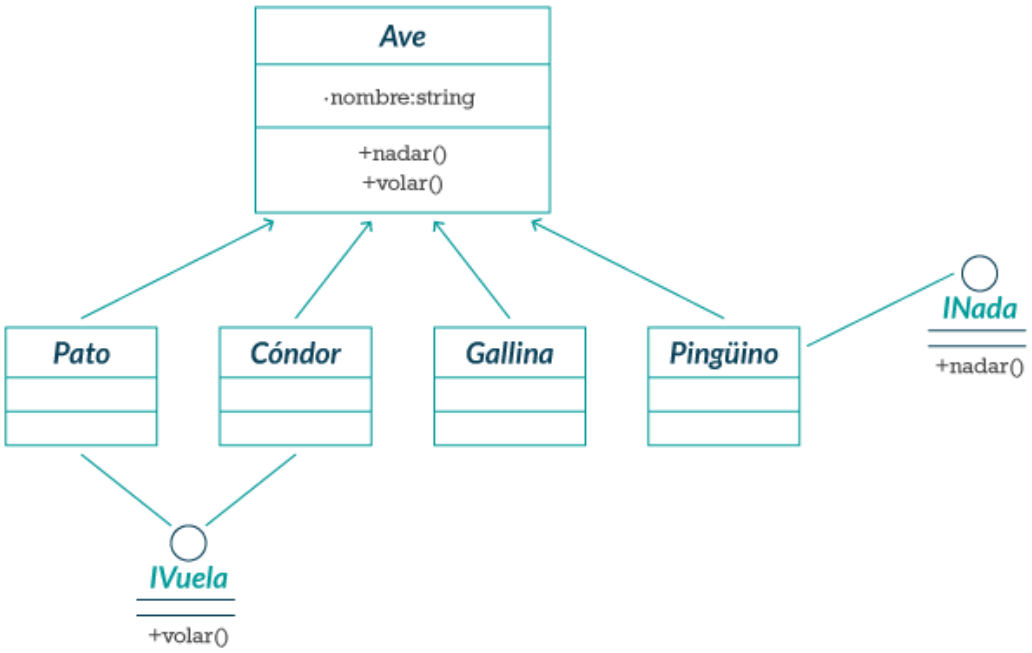
Un ejemplo que nos puede aclarar algo el tema sería aquel en el que tenemos unas clases definidas llamadas Pato, Pingüino, Cóndor y Gallina, a las cuales se les ha definido atributos y métodos similares las cuales se remiten a una clase llamada Ave, pero que al observarlas también tienen comportamientos diferentes en cómo estas se desplazan, como, por ejemplo:

- Pato: vuela y nada
- Pingüino: nada, pero no vuela
- Cóndor: vuela, pero no nada
- Gallina: no vuela ni nada.

Así podemos encontrar:



Podríamos tratar de buscar algunas soluciones para esto como definir la clase Ave como no abstracta, pero esto implica que se tendrían que crear e implementar dos métodos: nadar() y volar(), pero se debe tener en cuenta que esta superclase AVE es muy general, y de ahí se derivarán todos los tipos de aves que existen. Aquí la clase Cóndor heredaría las dos funciones, pero nadar no hace parte de su naturaleza, así que podría sobrescribir el método nadar y retornar algún valor no valido, lo que lo hace una construcción errónea. Si se define Ave como una clase abstracta no se implementa las funciones, pero obligara a lo eso se haga las clases derivadas y volvemos al mismo problema. Se pueden separar los comportamientos teniendo en cuenta que nadar y volar son independientes, separando estas en estructuras independientes, por tanto, el Pingüino heredará de la estructura que contenga nadar(), Cóndor de la que tenga volar() y Pato de las dos estructuras, un concepto de herencia múltiple o usando el tema de Interface, así con el uso de interface podrían plantear el comportamiento no solo para aves, sino para otras clases que deseen implementarlos.



En notación UML, las interfaces se representan con una circunferencia.

Las interfaces pueden tener una organización jerárquica, donde los métodos sean heredados y a diferencia de las clases que sólo heredar de una clase base por herencia simple, las interfaces pueden heredarse tanto como se precise.

En la siguiente tabla se pueden encontrar algunas diferencias entre las clases abstractas y las interfaces:

Clase Abstracta	*Interfaz*
La palabra clave abstract se usa para crear una clase abstracta y se puede usar con métodos.	La palabra clave de interface se usa para crear una interfaz, pero no se puede usar con métodos.
Una clase puede extender solo un clase abstracta.	Una clase puede implementar más de una interfaz.
Las variables no son definitivas por defecto. Puede contener variables no finales.	Las variables son finales por defecto en una interfaz.
Una clase abstracta puede proporcionar la implementación de una interfaz.	Una interfaz no puede proporcionar la implementación de una clase abstracta.
Puede tener métodos con implementaciones.	Proporciona una abstracción absoluta y no puede tener implementaciones de métodos.
Puede tener modificadores de acceso públicos, privados, estáticos y protegidos.	Los métodos son implícitamente públicos y abstractos en la interfaz de Java.
No admite herencias múltiples.	Es compatible con herencias múltiples.
Es ideal para la reutilización del código y la perspectiva de la evolución.	Es ideal para la declaración de tipo.

En resumen, se debe de tener en cuenta:

- Las interfaces solo tienen variables estáticas.
- No contiene constructores porque estas no pueden ser instanciadas.
- Las interfaces pueden extender otras interfaces.
- Una interfaz implícitamente es abstracta por tanto no se usa la palabra clave abstracta al declararla.
- Los métodos de la interfaz implícitamente son abstractos y públicos pues ésta es la naturaleza de la interfaz.
- Una clase hereda de una sola superclase, pero puede implementar múltiples interfaces.
- La clase que implementa la interfaz especifica la implementación o código de cada método, de lo contrario la clase se convierte en abstracta obligando así a que cada método de la interfaz se implemente.