

Interfaces en java

- ▶ Una interface es similar a una clase que no tiene atributos.
- ▶ Una interface solo declara un conjunto de métodos sin implementarlos.
- ▶ Es decir, los métodos de una interface solo tienen el prototipo (nombre, tipo y lista de parámetros) y carecen de implementación.

Interfaces en java

- ▶ Los métodos de una interface, son similares a un método **abstracto**.
- ▶ Una Interface es similar a una **clase abstracta**.
- ▶ Tanto la declaración de la interface como la de cada uno de sus métodos, no requieren la palabra reservada **abstract**.

Interfaces en java

- ▶ Una interface provee un mecanismo para definir métodos sin implementarlos.
- ▶ Estos métodos posteriormente se implementan por clases que se relacionan con la interface.
- ▶ Cada clase asociada a la interface puede tener una implementación propia o distinta de los métodos de la interface, tratándolos de este modo como métodos polimórficos.

Interfaces en java

- ▶ Cualquier clase puede relacionarse con una interface; lo cual, no depende de las relaciones que la clase pueda tener con otras clases; en particular, no se requiere que la clase pertenezcan a una jerarquía de clases.
- ▶ Así, el uso de una interface no depende ni requiere de una relación de herencia entre clases.

Interfaces en java

- ▶ Sin embargo, para aquellas clases que no tienen un **ancestro común**, pero que **implementan una misma interface**, dicha interface se comporta como un **padre** para esas clases.
- ▶ Es decir, que una interface se convierte en un **ancestro común para todas las clases** que se relacionen con ella, **independiente de las demás relaciones que las clases puedan tener**.

Interfaces en java

- ▶ Por esta razón, una interface suele representar los **pocos métodos comunes** que existen entre clases que pueden ser **bastante distintas** entre si; es decir, clases que tienen mas diferencias que similitudes, especialmente en términos de métodos.
- ▶ Una interface puede relacionar a conjuntos de objetos muy **disimiles**, agrupando lo **poco común** que hay entre ellos y haciéndolos **compatible**.

Interfaces en java

- ▶ El uso de una interface es para relacionar clases que **comparten** pocos métodos **comunes** sin que necesariamente exista una relación de herencia entre ellas.
- ▶ Una interface es adecuada en aquellos casos en los que las clases son tan **distintas** que no convenga definir un **padre común** para ellas.
- ▶ Las interface se usan frecuentemente en la programación de **eventos**.

Interfaces en java

- ▶ Cualquier clase que se relacione con una interface debe implementar todos los métodos de la interface.
- ▶ En consecuencia, la relación entre una clase y una interface, es similar a la relación entre una clase hija y una clase padre abstracta.
- ▶ La excepción a esta regla es la misma que en la relación de herencia; es decir, que la clase asociada a la interface sea abstracta.

Interfaces en java

- En UML una interfaz se simboliza por un rectángulo dividido en dos secciones:
 - En la parte superior se indica el nombre de la interfaz.
 - En la parte inferior se indican los prototipos de los métodos.
- Opcionalmente el nombre incluye el estereotipo <<interface>> para mayor diferenciación con una clase.

Interfaces en java

- Símbolo UML para una interfaz:

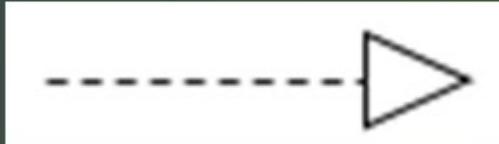


- Ejemplo:

```
<<interface>> ActionListener
```

```
+ void actionPerformed(ActionEvent Evt);
```

Interfaces en java

- ▶ En **UML** la relación entre una interfaz y una clase que la implementa se le llama **realización**.
- ▶ Una **realización** en UML se simboliza por una flecha triangular de línea discontinua:

- ▶ Esta flecha parte de las clases y apunta hacia la interface.

Interfaces en java

- ▶ En esta relación, la codificación de los métodos declarados por una interface la deben hacer las clases que se relacionan con ella.
- ▶ Así, las clases que implementan (realizan) a una interface “heredan” todos los métodos de esta, pero no en términos de relación de herencia, sino de relación contractual.

Interfaces en java

- ▶ Una clase que implementa (realiza) a una interface también puede **heredar** de otra clase.
- ▶ Una misma clase puede implementar **varias** interfaces simultáneamente; lo cual es una alternativa de solución a problemas de **herencia múltiple**, en lenguajes donde no hay un soporte para la herencia múltiple.

Interfaces en java

Ejemplo: Consideremos el conjunto de clases para representar la facultad de nadar de algunos animales.

- ▶ Sabemos que existen diferentes especies de animales que son capaces de nadar.
- ▶ Aunque la acción de nadar es la misma para todos ellos, las especies de animales que nadan pueden ser muy distintas entre si.

Interfaces en java

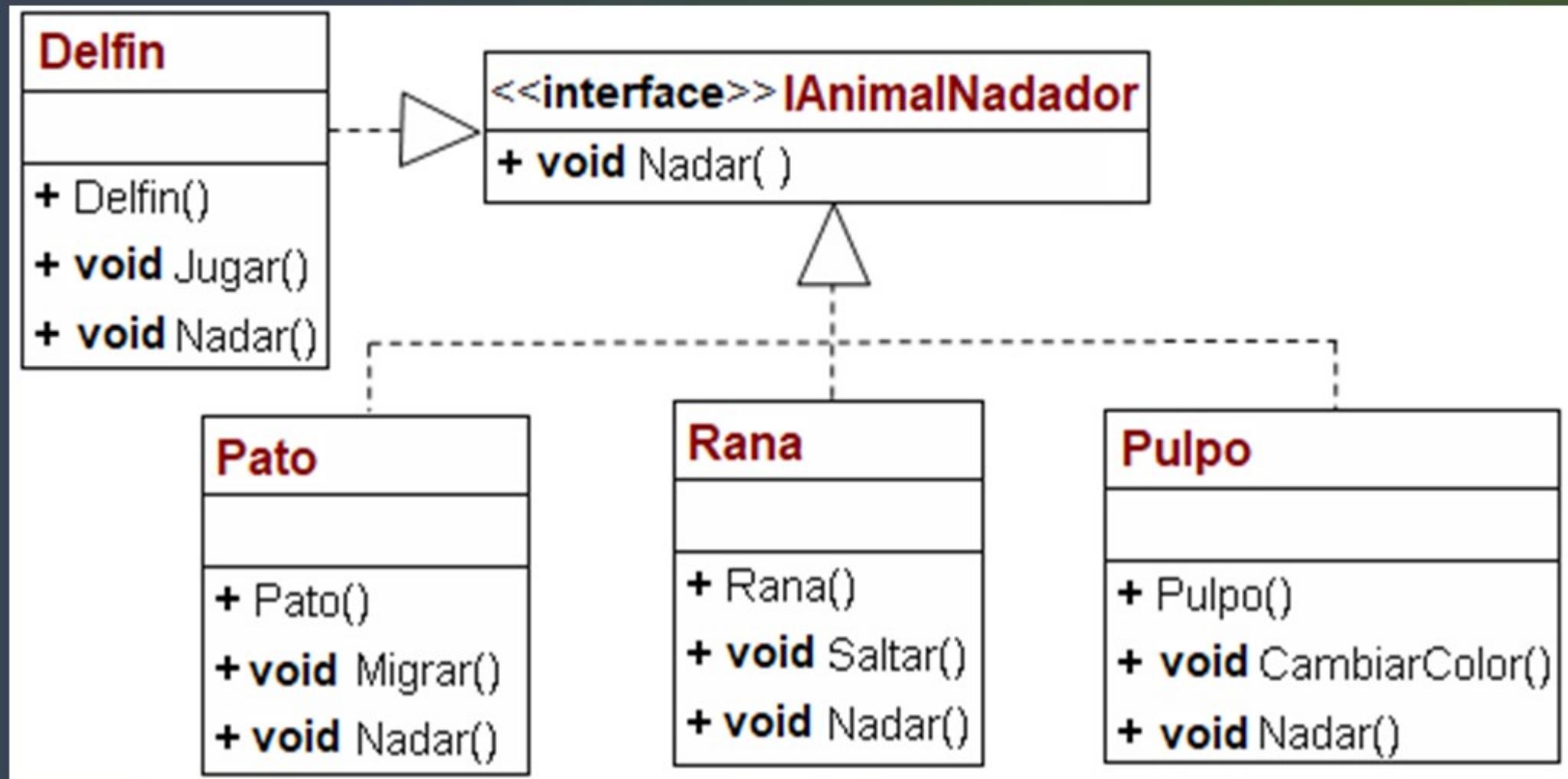
- ▶ Así un pato, una rana, un pulpo y un delfín son casos de animales que nadan.
- ▶ Sin embargo cada uno de estos animales pertenecen a especies diferentes (aves, anfibios, moluscos y peces respectivamente).
- ▶ Por esta razón, entre ellos existen muchas diferencias mas allá de la similitud de saber nadar.

Interfaces en java

- ▶ En consecuencia, no es fácil pensar en una clase padre con atributos y métodos comunes para cuatro tipos de animales tan disimiles.
- ▶ En estos casos es mas apropiada una interface, que tenga solo los pocos métodos comunes a un conjunto de objetos; sin importar las muchas diferencias entre ellos.

Interfaces en java

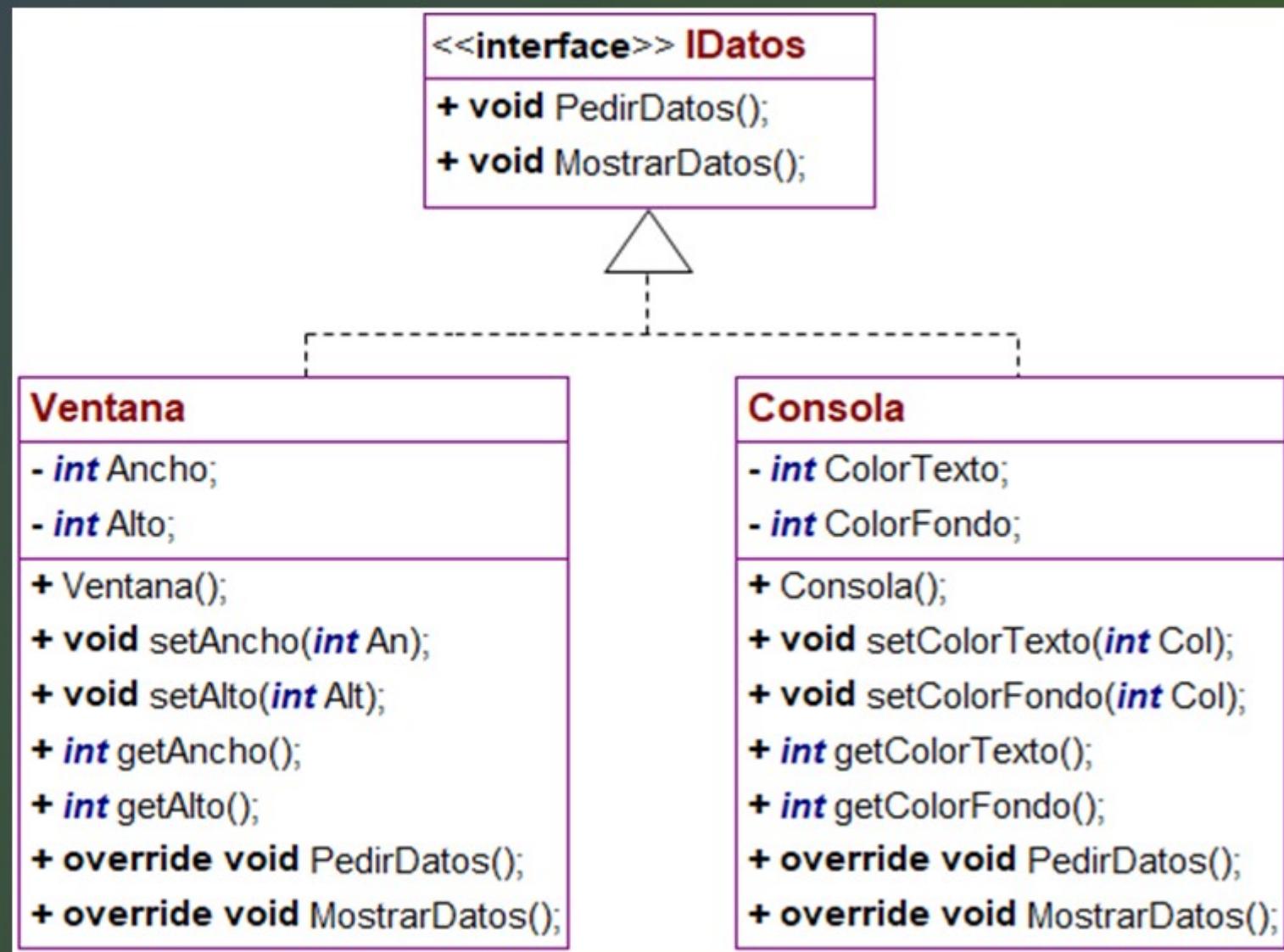
Diseño UML del ejemplo:



Interfaces en java

Ejemplo:

En este diagrama vemos una interface con métodos comunes para ingresar datos y mostrar datos para una aplicación de consola y de ventana.



Interfaces en java

La sintaxis para declarar una interface en java es:

```
public interface Nombre_Interfaz{  
    Prototipo_Metodo1();  
    Prototipo_Metodo2();  
    Prototipo_Metodo3();  
    ...  
    Prototipo_MetodoN();  
}
```

Interfaces en java

Ejemplos:

```
public interface IDatos {  
    void PedirDatos();  
    void MostrarDatos();  
}
```

```
public interface IAnimalNadador {  
    public void Nadar();  
}
```

Interfaces en java

- ▶ En una interface solo puede estar el área **public**; lo cual es opcional, pues todos los **métodos** de una interface son **públicos** por defecto
- ▶ En consecuencia Java generará un error de compilación, si en una interface usamos el área **private** o el área **protected**.
- ▶ Una clase que implemente una interface debe implementar todos los métodos de esta indicándolos como **public** explícitamente.

Interfaces en java

- En Java la sintaxis para que una clase implementa a una interface es usando la palabra reservada ***implements***:

```
public class Nombre_Clase implements NombreInterface {...}
```

- Si la clase va a implementar más de una interface los nombres de estas se separan entre sí por comas:

```
public class Nombre_Clase implements Interfaz1,Interfaz2, InterfazN {...}
```

Interfaces en java

- ▶ Si la clase **hereda** de otra e **implementa** a una o más interfaces, siempre se indica **primero** el nombre de la **clase padre** y después las interfaces separándolas con comas si son varias:

```
public class Nombre_Clase extends ClasePadre implements Interfaz1, Interfaz2, InterfazN {...}
```

Interfaces en java

Ejemplo:

```
public class Ventana implements IDatos {  
    private int Alto;  
    private int Ancho;  
  
    public Ventana() {  
        Alto=0;  
        Ancho=0;  
    }  
    • • •  
    @Override  
    public void PedirDatos(){  
    • • •  
    }  
  
    @Override  
    public void MostrarDatos(){  
    • • •  
    }  
    • • •  
}
```

Interfaces en java

- ▶ Para instanciar una interface, primero se indica el nombre de la interface y luego el nombre de la instancia.
- ▶ Una instancia de una interface DEBE inicializarse con el constructor o con una instancia de CUALQUIER clase que la implemente.
- ▶ Así por ejemplo, una instancia de la interface *IAnimalNadador* puede inicializarse con una instancia o constructor de la clase Pato, Rana, Pulpo o Delfín.

Interfaces en java

- ▶ En este sentido, una **instancia** de una **interface** puede verse de la misma manera que una **instancia** de una **clase padre abstracta**.
- ▶ Una instancia de interface asume el rol de cualquier instancia de clase que la implemente.
- ▶ Una instancia de una interface **contiene** a una instancia de cualquier clase que la implemente; aun cuando entre las clases que implementen **no exista relación alguna**, ni siquiera la de herencia.

Interfaces en java

Ejemplo:

```
IAnimalNadador IAN=null; //Inicializar en nulo  
IAN=new Pulpo(); //Inicializamos como pulpo  
IAN.Nadar(); //Ejecutamos el metodo nadar del pulpo  
if(IAN instanceof Pulpo){  
    //Convertimos interfaz a objeto pulpo y ejecutamos un metodo exclusivo de este  
    ((Pulpo)IAN).CambiarColor();  
}
```

Interfaces en java

Ejemplo:

```
IAN=new Delfin(); //Ahora inicializamos como delfin
IAN.Nadar(); //Ejecutamos el metodo nadar del delfin
if(IAN instanceof Delfin){
    //Convertimos interfaz a objeto delfin y llamamos un metodo propio del delfin
    ((Delfin)IAN).Saltar();
}
```

Interfaces en java

- Como excepción java permite incluir atributos en una interface, siempre y cuando estos:
 - Estén **inicializados** en su declaración.
 - Sean definidos como **static** (estáticos).
 - Sean definidos como **final**.
 - Los dos últimos puntos son **opcionales**; en caso de no indicarlos, java asume por defecto que el atributo es **static** y **final**.
 - En consecuencia un atributo en una interface se define una **constante** en la interface.

Interfaces en java

- ▶ Otra excepción en **java** a las interfaces es la inclusión de las **interfaces funcionales**:
- ▶ Una interface funcional es una interface especial caracterizada, entre otras cosas, por tener métodos de tipo **default**.
- ▶ Un método **default** en una interface funcional si puede tener una implementación.