INTERFACES EN JAVA

En el tema anterior hemos visto el concepto de clase abstracta. Una clase que contenga un método abstracto se debe declarar como abstracta. También sabemos que una clase abstracta puede contener atributos, constructores y métodos no abstractos. De una clase abstracta no se pueden crear objetos y su finalidad es ser la base para construir la jerarquía de herencia entre clases y aplicar el polimorfismo.

Llevando estos conceptos un poco más allá aparece en concepto de Interface. Podemos considerar una **interface** como una clase que sólo puede contener:

- Métodos abstractos
- Atributos constantes

A partir de **Java 8** el concepto de Interface se ha ampliado y a pertir de esa versión de Java las interfaces tambíen pueden contener:

- Métodos por defecto
- Métodos estáticos
- Tipos anidados

Y a partir de **Java 9** se le ha añadido una nueva funcionalidad a las interfaces y también pueden contener:

- Métodos privados

Una Interface tiene en común con una clase lo siguiente:

- Puede contener métodos.
- Se escribe en un archivo con extensión .java que debe llamarse exactamente igual que la interface.

Pero una inteface se diferencia de una clase en lo siguiente:

- No se pueden instanciar. No podemos crear objetos a partir de una Interface.
- No contiene constructores.
- Si contiene atributos todos deben ser *public static final*
- Una interface puede heredar de varias interfaces. Con las interfaces se permite la herencia múltiple.

Conceptos importantes a tener en cuenta cuando trabajamos con interfaces:

- Las clases no heredan las interfaces. Las clases **implementan** las interfaces.
- Una clase puede implementar varias interfaces.

Explicándolo de una manera simple, implementar una interface sería equivalente a copiar y pegar el código de la interface dentro de la clase. Cuando una clase implementa una interface está obligada a implementar todos los métodos abstractos que contiene ya que de otra forma debería declararse como clase abstracta.

Usando Interfaces, si varias clases distintas implementan la misma interface nos aseguramos que todas tendrán implementados una serie de métodos en comunes.

Las interfaces juegan un papel fundamental en la creación de aplicaciones Java.

Las interfaces definen un protocolo de comportamiento y proporcionan un formato común para implementarlo en las clases.

Utilizando interfaces es posible que clases no relacionadas, situadas en distintas jerarquías de clases sin relaciones de herencia, tengan comportamientos comunes.
Una interface se crea utilizando la palabra clave **interface** en lugar de class.

```
[public] interface NombreInterface [extends Interface1, Interface2, ...]{
        [métodos abstractos]
        [métodos default]
        [métodos static]
        [métodos privados]
        [atributos constantes]
}
```

Algunas características de las interfaces:

La interface puede definirse *public* o sin modificador de acceso, y tiene el mismo significado que para las clases. Si tiene el modificador *public* el archivo .java que la contiene debe tener el mismo nombre que la interfaz.

En los métodos abstractos no es necesario escribir *abstract.*

Los métodos por defecto se especifican mediante el modificador *default*.

Los métodos estáticos se especifican mediante la palabra reservada *static*.

Los métodos privados se especifican mediante el modificador de acceso *private*.

Todos los atributos son constates públicos y estáticos. Por lo tanto, se pueden omitir los modificadores *public static final* cuando se declara el atributo. Se deben inicializar en la misma instrucción de declaración.

Los nombres de las interface suelen acabar en *able* aunque no es necesario: Configurable, Arrancable, Dibujable, Comparable, Clonable, etc.
Ejemplo: Creamos una interface llamada *Relacionable* con tres métodos abstractos.

```
//Interfaz que define relaciones de orden entre objetos.
public interface Relacionable {
   boolean esMayorQue(Relacionable a);
   boolean esMenorQue(Relacionable a);
   boolean esIgualQue(Relacionable a);
}
```

IMPLEMENTACIÓN

Para indicar que una clase implementa una interface se utiliza la palabra clave **implements**.

```
public class UnaClase implements Interface1{
```

.....
}

EJERCICIO:

Escribe un programa para una biblioteca que contenga libros y revistas.

Las características comunes que se almacenan tanto para las revistas como para los libros son el código, el título, la temática y el año de publicación. Estas cuatro características se pasan por parámetro en el momento de crear los objetos.

La temática sólo pueden tomar los valores: Aventura, Historia y Acción.

Los libros tienen además un atributo prestado. Los libros, cuando se crean, no están prestados.

Las revistas tienen un número. En el momento de crear las revistas se pasa el número por parámetro.

Tanto las revistas como los libros deben tener (aparte de los constructores) un método toString() que devuelve el valor de todos los atributos en una cadena de caracteres. También tienen un método que devuelve el año de publicación, y otro el código.

Para prevenir posibles cambios en el programa se tiene que implementar una interfaz Prestable con los métodos prestar(), devolver() y prestado. La clase Libro implementa esta interfaz

Realiza un programa principal que cree un array de cómo máximo 100 libros y revistas y muestre el siguiente menú:

1.- Dar de alta libro

2.- Dar de alta revista.

3.- Comprobar si libro está prestado.

4.- Prestar libro

5.- Prestar revistas

6.- Mostrar información

7.- Salir