INTERFACES Y CLASES ABSTRACTAS

1.

INTERFACES

INTERFACES EN JAVA

- Contrato de compromiso.
- Conjunto de operaciones que una clase se compromete a implementar.
- La interfaz marca qué métodos, con su firma
- Desde Java 8, pueden incluir también métodos con cuerpo (abstractos, estáticos y por defecto).
- También puede incluir constantes.

DEFINICIÓN DE INTERFACES

- **interface**
- Mismas normas de acceso que una clase.
- Mismas reglas de nombres que una clase.
- También existe herencia de interfaces (**extends**). En este caso, sí que puede ser múltiple.

```
public interface GroupedInterface extends Interface1, Interface2  {
     // constant declarations
     // base of natural logarithms
     double E = 2.718282;
     // method signatures
     void doSomething (int i, double x);
     int doSomethingElse(String s);
}
```

IMPLEMENTACIÓN DE INTERFACES

- **implements**
- Una clase puede implementar más de una interfaz

```
public class RectanglePlus implements Relatable {
//…
    public int isLargerThan(Relatable other) {
        RectanglePlus otherRect = (RectanglePlus)other;
        if (this.getArea() < otherRect.getArea())
            return -1;
        else if (this.getArea() > otherRect.getArea())
            return 1;
        else
            return 0;
    }
}
```

INTERFACES COMO TIPOS

- Una interfaz puede ser el tipo de dato a usar para crear una instancia de un objeto.
- La clase del objeto a crear debe implementar dicha interfaz.
- Muy útil, sobre todo, para recibir argumentos en métodos.

```
RectanglePlus rectangleOne = new RectanglePlus(10, 20);
Relatable rectangleTwo = new RectanglePlus(20, 10);
```

MÉTODOS POR DEFECTO

- Novedad en Java 8
- **default**.
- Un método puede tener una implementación por defecto, descrita en la interfaz.

```
public interface Interfaz {

     default public void metodoPorDefecto() {
         System.out.println("Este es uno de los nuevos
                                        métodos por defecto");
     }

}
```

MÉTODOS ESTÁTICOS

- Novedad en Java 8
- **static**.
- Misma sintaxis que los métodos estáticos en clases

```
public interface Interfaz {

     public static void metodoEstatico() {
         System.out.println("Método estático en un interfaz");
     }
}
```

2.

CLASES ABSTRACTAS

CLASES ABSTRACT

- Clase definida como **abstract**.
- No se pueden crear instancias de la misma.
- Puede tener métodos con implementación y atributos.

```
public abstract class AbstractaSencilla {

	public void saluda() {
		System.out.println("Hola mundo!!!");
	}

}
```

MÉTODOS ABSTRACT

- Deben estar en una clase definida como **abstract**.
- Definen la firma del método, pero sin implementación.
- Sus subclases se comprometen a implementarlo.
- Si no lo hacen, también deben ser abstractas.
- Pueden convivir con métodos normales.

```
public abstract class AbstractaConMetodos {

	public abstract void saludo(String s);

	public void saludar() {
		System.out.println("Hola mundo!!!");
	}
}
```

3.

CLASES ABSTRACTAS vs INTERFACES

INTERFACES	CLASES ABSTRACTAS
No se pueden instanciar	No se pueden instanciar
Métodos sin implementación	Métodos sin implementación
Métodos con implementación por defecto	Métodos con implementación por defecto
Atributos estáticos o constantes	Cualquier tipo de atributos
Métodos públicos o por defecto	Métodos públicos, privados, protegidos o por defecto.
Una clase puede implementar varios interfaces	Una clase solo puede heredar de otra

¿QUÉ USAR?

INTERFACES	CLASES ABSTRACTAS
Clases no relacionadas podrán implementar los métodos.	Compartir código con clases muy relacionadas.
Si se quiere indicar que existe un tipo de comportamiento, pero no sabemos quien lo implementa.	Las clases derivadas usarán métodos *protected* o *private*.
Si necesitamos tener *herencia múltiple*.	Queremos definir atributos que no sean estáticos o constantes.