Práctica 17: Crear la interfaz: Ordenable e incluir como métodos: boolean esMayorQue() , esMenorQue(), esIgualQue() Implementar el interfaz en la clase Persona (tomaremos la clase abstracta que ya hemos usado en esta unidad) ¿ qué ocurre en el IDE cuándo escribimos: Persona implements Ordenable ? (tomar captura de pantalla) mediante alt+enter hacer el override

```java
/**
 *
 * @author Kevin Hernández García <kevinhg94@gmail.com>
 */
public interface Ordenable<T> {
    boolean esMayorQue(T elemento);
    boolean esMenorQue(T elemento);
    boolean esIgualQue(T elemento);
}
```

Creamos la interface Ordenable con un genérico, de esta forma nos aseguramos que siempre se use el objeto que deseamos según cada situación.

```java
/**
 *
 * @author Kevin Hernández García <kevinhg94@gmail.com>
 */
public abstract class Persona implements Ordenable<Persona>{
    String nombre;
    String apellido1;
    String apellido2;
    int edad;
    int altura;
    double peso;

    public Persona(String nombre, String apellido1, String apelli
```

Al implementar la interfaz Ordenable en la clase Persona no identificamos ningún error, esto se debe a que persona es una clase abstracta, de esa manera no lo hace falta implementar los métodos de ordenable, sin embargo, si miramos las clases Hombre y Mujer, que son herederas de Persona, encontramos un error.

```java
/**
 *
 * @author Kevin
 */
public class Mujer extends Persona {

    public Mujer(String nombre, String apellido1, String apellido2, int edad, int altura, double peso) {
        super(nombre, apellido1, apellido2, edad, altura, peso);
    }
```

Mujer is not abstract and does not override abstract method esIgualQue(Persona) in Ordenable
---
(Alt-Enter shows hints)

```java
/**
 *
 * @author Kevin
 */
public class Hombre extends Persona{

    public Hombre(String nombre, String apellido1, String apellido2, int edad, int altura, double peso) {
        super(nombre, apellido1, apellido2, edad, altura, peso);
    }

    @Override
```

Hombre is not abstract and does not override abstract method esIgualQue(Persona) in Ordenable
---
(Alt-Enter shows hints)

El error nos informa que no se ha sobrescrito el método abstracto que se ha creado en Ordenable. Para solucionar este error podemos implementar todas las clases abstractas en cada una de las clases, sin embargo, un método interesante de ordenar las personas podría ser por edad o por orden alfabético de su nombre.

Siendo estos dos atributos propios de la clase Persona, podemos añadir la implementación de los métodos directamente dentro de Persona.

```
15  -  */
16  public abstract class Persona implements Ordenable<Persona>{
17      String nombre;
18      String apellido1;
19      String apellido2;
20      int edad;
21      int altura;
22      double peso;
23
24      public Persona(String nombre, String apellido1, String apellido2, int edad, int altura, double peso) {
25          this.nombre = nombre;
26          this.apellido1 = apellido1;
27          this.apellido2 = apellido2;
28          this.edad = edad;
29          this.altura = altura;
30          this.peso = peso;
31      }
32
33      protected double calcularIMC(){
34          return this.peso/(Math.pow(altura/(double)100, 2));
35      }
36
37      protected abstract double pesoIdeal();
38
39      @Override
40      public String toString() {
41          String imc = String.format("%.3f", calcularIMC());
42          return nombre + " " + apellido1 + " " + apellido2 + ", Edad:" + edad + " años, Altura:" + altura + " cm, Peso:" + peso + " IMC: "+ imc;
43      }
44
45          @Override
46      public boolean esMayorQue(P17InterfaceOrdenable.Persona elemento) {
47          return this.edad > elemento.edad;
48      }
49
50      @Override
51      public boolean esMenorQue(P17InterfaceOrdenable.Persona elemento) {
52          return this.edad < elemento.edad;
53      }
54
55      @Override
56      public boolean esIgualQue(P17InterfaceOrdenable.Persona elemento) {
57          return this.edad < elemento.edad;
58      }
59
60      // Buscar informacion de formatter.
61
```

Con esto ya tendríamos a las clases Hombre y Mujer funcionando correctamente, y teniendo implementada la interfaz Ordenable.