Interfaces Comparable y Comparator

Interfaz Comparable

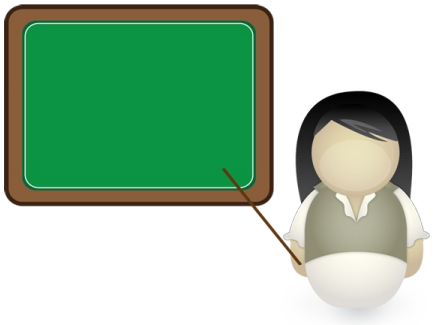
- Es una interfaz que permite que se puedan comparar los objetos de una clase, lo cual permite que se puedan ordenar.

- Para llevar a cabo lo mencionado en el ítem precedente la clase debe implementar la interfaz y definir el método compareTo.

- CompareTo devuelve un número negativo, 0 o un número positivo dependiendo de si el objeto que se compara es menor, igual o mayor que el objeto con el que se compara.

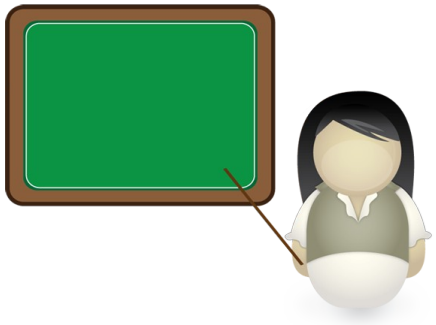
Interfaz Comparable

```
public class Persona implements Comparable<Persona>{
      Private int dni, edad;
       public Persona( int d, int e){
         this.dni = d;
         this.edad = e;
       }
 ......
      public int compareTo(Persona o) {
        int resultado=0;
         if (this.edad<o.getEdad()) {   resultado = -1;       }
          else if (this.edad>o.getEdad()) {    resultado = 1;       }
           else {
            if (this.dni<o.getDni()) {  resultado = -1;    }
             else if (this.dni>o.getDni()) {   resultado = 1;   }
             else {   resultado = 0;   }
           }
           return resultado;
      }
}
```



Interfaz Comparable

```
public class Programa {
  public static void main(String arg[]) {
    Persona p1 = new Persona(74999999,35);
    Persona p2 = new Persona(72759474,30);
       if (p1.compareTo(p2) < 0 ){
       System.out.println("p1: es menor."); }
       else if (p1.compareTo(p2) > 0 ) {
        System.out.println("p1: es mayor."); }
        else {
         System.out.println ("p1 es igual a p2"); }
      }
}
```



Interfaz Comparator

- Permite comprar dos elementos de una colección.
- Comparable obliga a implementar el método compareTo(Object o) (orden natural).
- Comparator obliga a implementar el método compare(Object o1, Object o2) (orden total).

Interfaz Comparator

```
import java.util.Comparator;
import java.util.ArrayList;
import java.util.Collections;

public class OrdenarPersonaPorAltura implements Comparator<Persona> {
    @Override
    public int compare(Persona o1, Persona o2) {
      return o1.getAltura() - o2.getAltura();
    }
}

public class Programa {
    public static void main(String arg[]) {
        ArrayList<Persona> listaPersonas = new ArrayList<>();
        listaPersonas.add(new Persona(1,"Maria",185));
        listaPersonas.add(new Persona(2,"Carla",190));
        listaPersonas.add(new Persona(3,"Yovana",170));
        Collections.sort(listaPersonas, new OrdenarPersonaPorAltura());
        System.out.println("Personas Ordenadas por orden total: "+listaPersonas);
      }
}
```

Comentarios Útiles: Imprimir un Objeto

```
Imprimir un Objeto
public class Persona implements Comparable<Persona> {
    private int idPersona;
    private String nombre;
    private int altura;

    public Persona (int idPersona, String nombre, int altura) {
        this.idPersona = idPersona;
        this.nombre = nombre;
        this.altura = altura;}

    @Override
    public String toString() {
        return "Persona-> ID: "+idPersona+" Nombre: "+nombre+" Altura: "+altura;}

    @Override
    public int compareTo(Persona o) {
        return this.nombre.compareTo(o.nombre);}

    public int getIdPersona() {return idPersona;}
    public String getNombre() {return nombre;}
    public int getAltura() {return altura;}
}
```

Conclusiones

- *Comparable y Comparator parecen iguales pero no lo son. La primera define un orden natural y la segunda define un orden total.*
- *En muchas situaciones se utiliza el orden natural pero existen situaciones donde esto no es así.*
- *Con Comparator se pueden ordenar colecciones utilizando clases que implementen el método compare por cada tipo de orden que se desee.*