Practica: Ordenar arrays de objetos

Tal como hemos construido los distintos algoritmos de ordenación, sería necesaria una implementación diferente para cada tipo de datos de las casillas del array que queramos ordenar.

Si queremos por ejemplo ordenar por el método de la burbuja, tendríamos que hacer una implementación de este método para cada uno de los tipos distintos que podamos tener en un *array*:

```
void  ordenarEnteros (int[] array)

void  ordenarReales (double[] array)

void ordenarCaracteres (char[] array)

void  ordenarPuntos (Punto[] array)

void  ordenarCasillas (Casilla[] array)

void  ordenarPersonas (Persona[] array)
```

…

Para solucionar este problema podemos usar el polimorfismo, propiedad fundamental de la POO, para tener una sola implementación que nos sirva para ordenar un array de objetos de cualquier clase que incorpore un método de comparar adaptado a sus propias características.

¿Cómo se hace?

1. Declarando el array como elementos de la interfaz Comparable (recuerda que puedes declarar variables del tipo de una interfaz pero ¡no puedes instanciarla!). Así definiríamos por ejemplo:

```
Comparable[] personas= new Persona [TAMAÑO];
Comparable[] arraydobles = new Double[TAMAÑO]
Comparable[] arrayCasillas = new Casilla[TAMAÑO]
```

2. Implementando el método ordenar (que implementaría el método de la burbuja) cuyo argumentos en un elemento del tipo Comparable. La firma sería tal que así:

```
static void ordenar (Comparable[] personas);
static void ordenar (Comparable[] arraydobles);
static void ordenar (Comparable[] arrayCasillas);
```

3. La implementación del método de la burbuja para una array concreto de Personas sería algo así:

// Método específico

```
private static void ordenar (Persona[] personas){
int i, j;
Persona aux;
for (i=0 ;  i< Persona.length -1 ;  i++)
      for (j = Persona.length -1 ; j > i ; j--)

      if (Persona [j].compareTo(Persona [j-1])<0){
      //intercambio de elementos
            aux = Persona [j];
            Persona [j] = Persona [j-1];
            Persona [j-1] = aux;
      }
}
```

4. Sin embargo, la implementación del método de la burbuja para una array declarado como Comparable, sería algo así:

// Metodo polimórfico

```
private static void ordenar (Comparable[] comparable(*)){
```

//(*) comparable puede ser personas, arraydobles, arrayCasillas….

```
int i, j;
Comparable aux;
for (i=0 ;  i< comparable.length -1 ;  i++)
      for (j = comparable.length -1 ;  j > i ; j--)
      if (comparable [j].compareTo(comparable [j-1])<0){
      //intercambio de elementos
            aux = comparable
            [j]; comparable [j]
            = comparable [j-1];
            comparable [j-1] =
            aux;
      }
}
```

Para poder usar segunda versión la clase ha de implementar (*implelments*) la interfaz Comparable<T>, que le obliga a implementar el método compareTo:

```
public class Persona implements Comparable <Persona> {
```

[…..]

```
@Override
```

```
        public int compareTo(Carta c){
        [.....]
        }
}
```

Ver ejemplo en https://github.com/EsperanzaE/OrdenacionPolimorfica.git

Práctica

1. En el ejemplo de Git hemos usado el método de la burbuja. Repite los ejemplos con el método de ordenación directa
2. Prueba a cambia el criterio de comparación (modificando SÓLO el método compareTo) y verás que obtienes una secuencia ordenada diferente.