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.

```
void  ordenarEnteros (int[] array)
```

Así necesitaríamos distintos métodos

```
void  ordenarReales (double[] array)

void  ordenarCaracteres (char[] array)

void  ordenarPuntos (Punto[] array)

void  ordenarCasillas (Casilla[] 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.

Implementación específica para la clase Carta (puede ser cualquier clase)

```
// Metodo especifico para la clase Carta
private static void ordenaCartas  (Carta[] mazo){
int i, j;
Carta aux;
for (i=0 ;  i< mazo.length -1 ;  i++)
      for (j = mazo.length -1 ;  j > i ; j--)
      if (mazo[j].compareTo(mazo[j-1])<0){
      //intercambio de elementos
            aux = mazo[j];
            mazo[j] = mazo[j-1];
            mazo[j-1] = aux;
      }
}
// Metodo polimórfico
private static void ordenaCartas  (Comparable[] mazo){
int i, j;
Comparable aux;
for (i=0 ;  i< mazo.length -1 ;  i++)
      for (j = mazo.length -1 ;  j > i ; j--)
```

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

Esta segunda versión podemos usarla para ordenar un array de objetos de cualquier clase que implemente la interfaz Comparable<T>, que le obliga a implementar el método compareTo

```
public class Carta implements Comparable <Carta> {
      [.....]
      @Override
      public int compareTo(Carta c){
      [.....]
      }
}
```

Práctica

1. Para este ejemplo hemos usado el método de intercambio directo. Escribe los otros métodos directos (selección e inserción) de forma polmórfica.
2. Prueba los métodos con arrays de cualquier clase que hayas implementado en las unidades anteriores y a la que hayas dotado de un método compareTo. Usa al menos dos clase distintas.
3. Comprueba cómo, al cambiar el criterio de comparación (modificando el método compareTo) se obtiene una secuencia ordenada diferente.