9.11 Implementar la clase **Lista** para almacenar elementos de tipo **String**.

9.12 Definir las interfaces **Cola** y **Pila** para objetos **String** e implementarlos en la clase Lista definida en la actividad anterior 9.11

9.13 Diseñar la clase **Futbolista** con los siguientes atributos: dni, nombre, edad y número de goles. Implementar:

- Un constructor y los métodos **toString( )** y **equals( )** (este último basado en el DNI).
- La interfaz **Comparable** con un criterio de ordenación basado también en el DNI.
- Un comparador para hacer ordenaciones basadas en el nombre y otro basado en la edad.

Crear una tabla con 5 futbolistas y mostrarlos ordenados por DNI, por nombre y por edad.

9.14 Añadir a la Actividad anterior 9.13 un comparador que ordena los futbolistas por edades y, para aquellos que tienen la misma, por nombres.

9.15 Implementar la clase **Supercola**, que tiene como atributos dos colas para enteros, en las que se encola y desencola por separado. Sin embargo, si una de las colas queda vacía, al llamar a su método desencolar, se desencola de la otra mientras tenga elementos. Solo cuando las dos colas estén vacías, cualquier llamada a desencolar devolverá **null**. Escribir un programa con el menú:

1. Encolar en **cola1**.
2. Encolar en **cola2**.
3. Desencolar de **cola1**.
4. Desencolar de **cola2**.
5. Salir.

Después de cada operación se mostrará el estado de las dos colas para seguir su evolución.

9.16 Definir las interfaces **Cola** y **Pila** para **Object** e implementarlos en la clase **Lista** definida en la Actividad Resuelta 9.11.

9.17 Escribir un programa donde se use una **Lista** para los elementos **Object** para encolar y desencolar objetos de distintos tipos, mostrándolos por pantalla.

9.18 Repetir la actividad anterior con la interfaz **Pila** apilando y desapilando.

9.19 Implementar la interfaz **Comparable** en la clase **Socio** para que el criterio de ordenación sea de menor a mayor edad.

9.20 Repetir la actividad anterior para que se ordene por edades y, si dos socios tienen la misma edad, vaya antes el que tenga un número de socio menor.

9.21 Repetir la actividad anterior (9.20) con un criterio que ordene por fechas de nacimiento.

9.22 Definir una clase comparadora que ordene los socios por orden alfabético de nombre.

9.23 Repetir la actividad anterior (9.22) con un orden alfabético de nombre invertido (que empieza por la letra 'z').

9.24 Implementar en la clase **Lista** para elementos **Object** las funciones sobrecargadas:

- **void ordenar( ),** que ordena la lista con el orden natural de sus elementos.
- **void ordenar(Comparator c)**, que la ordena con el criterio que establezca **c**. Aquí tendremos que ser muy cuidadosos con que todos los elementos insertados sean del mismo tipo.

9.25 Usar la **Lista** de la actividad anterior (9.24) para insertar cadenas de caracteres y ordenarlos por orden alfabético.

9.26 Repetir la actividad anterior (9.25) con elementos de tipo **Socio** cuyo orden natural es el de la edad.

9.27 Manteniendo el mismo orden natural de la clase **Socio** (por edades), ordenar la lista de socios por orden alfabético de nombres.