Sesión 17: Listas (2/2)

Hoja de problemas

Programación 2

Ángel Herranz
aherranz@fi.upm.es
Universidad Politécnica de Madrid

2024-2025

Ejercicio 1. Repasa las transparencias de la sesión 16 sobre el **tipo abstracto de datos** de las listas. En esta sesión tendrás que implementar todos los métodos de la interfaz `IListStr` usando la clase `LinkedListStr`.

Ejercicio 2. El objetivo de estos ejercicios es trasladar todo el conocimiento de cadenas simplemente enlazadas para implementar una clase que respresenta listas de strings: `LinkedListStr`.

Puedes seguir usando la herramienta de visualización **JavaVisualizer**:

`https://cscircles.cemc.uwaterloo.ca/java_visualize/`

`IListStr` **Ejercicio 3.** Empieza escribiendo la interfaz de Java que *especifica*, tal y como puedes ver en las transparencias, el tipo abstracto de datos de las listas de strings: `IListStr`.

`NodoStr` **Ejercicio 4.** Vamos a realizar una implementación para dicho tipo abstracto usando cadenas enlazadas. Por lo tanto empezaremos creando la clase que representa los nodos:

```
// Clase para representar los nodos de una cadena enlazada
class NodoStr {
  String dato;
  NodoStr siguiente;
}
```

Ejercicio 5. Recuerda que estamos usando la clase `NodoStr` para representar *cadenas enlazadas de strings* y que conceptualmente dicha estructura de datos vienen a representar listas de *strings*. En este documento, normalmente usaremos la palabra *cadena* para referirnos a las variables e instancias del tipo `NodoStr`.

`implements` **Ejercicio 6.** Vamos a empezar con una clase `LinkedListStr` que simplemente **implemente** `IListStr`:

```
public class LinkedListStr implements IListStr {
}
```

¿Qué ocurre al compilar? ¿Por qué?

toString **Ejercicio 7.** Ya puedes crear los atributos y el constructor de la clase `LinkedListStr` y tras ello, quizás, el primer método que puedes intentar implemntar es `toString`. Eso te ayudará a depurar el resto de tus métodos.

Pruebas **Ejercicio 8.** Antes de empezar con el resto de la implemnetación, escribe unas cuantas pruebas. Te dejo a continuación un par de pruebas a modo de inspiración. Explóralas y complétalas con las tuyas[1]:

```
public class IListTest {
  private static final int N = 5;

  public static void main(String[] args) {
    System.out.println("Probando IList");
    test0(new LinkedListStr());
    test1(new LinkedListStr());
    System.out.println("Todos los tests han pasado");
  }

  // PRE: la lista l es una IList vacía
  private static void test0(IList<String> l) {
    Assert.assertError(() -> l.get(1));
    Assert.assertError(() -> l.set(1, "Hola"));
    Assert.assertError(() -> l.remove(1));
  }

  // PRE: la lista l es una IList vacía
  private static void test1(IList<String> l) {
    Assert.assertEquals(0, l.size());
    Assert.assertEquals(-1, l.indexOf("Hola"));

    for (int i = 0; i < N; i++) {
      l.add(l.size(), "D" + i);
    }

    Assert.assertEquals("[D0, D1, D2, D3, D4]", l.toString());
    Assert.assertEquals("D3", l.get(3));
    Assert.assertEquals(2, l.indexOf("D2"));
    Assert.assertEquals(N, l.size());
    l.remove(2);
    l.remove("D3");
    l.add(l.size(), "D4");
    Assert.assertEquals(2, l.indexOf("D4"))
    l.remove("D4");
    Assert.assertEquals(3, l.size());
  }
}
```

Ejercicio 9. Ya “solo” queda implementar todas los métodos de `LinkedListStr`. Todas las

[1] Se está haciendo uso de nuestra minibiblioteca de aserciones `Assert.java`.

operaciones de manejo de cadenas enlazadas ya las has hecho en la hoja de ejercicios de la sesión 15. Mencionar que “simplemente” **tendremos que adaptar el API *funcional* a un API más orinetado a objetos**:

```
static String obtener(NodoStr cadena, int i)
```

vs.

```
public String get(int i)
```

Observa que ahora la cadena enlazada es un atributo de la clase `LinkedListStr` y que tienes que modificarla más que devolver una nueva.