Sesión 22: Arrays redimensionables

Programación 2

Ángel Herranz

2022-2023

Universidad Politécnica de Madrid

En capítulos anteriores

- 👍 Tema 1: Intro a POO
- 👍 Tema 2: Clases y Objetos y TADs y módulos
- 👍 Tema 3: Colecciones con *arrays*
- 👍 Tema 4: Cadenas simplemente enlazadas
- 🕒 Tema 5: TAD Listas
  - `LinkedList<E>` **`implements`** `IList<E>`
- 👍 Tema 7: Polimorfismo
- 👍 Tema 8: Excepciones
  - **Precondiciones y postcondiciones**
  - **Ocultación**
  - **Excepciones en Java**

En el capítulo de hoy

👍 Tema 5: TAD Listas

- `ArrayList<E>` **`implements`** `IList<E>`
- Redimensionando arrays

IList

```
public interface IList<T> {
  void add(int index, T elem);
  T get(int index);
  int size();
  void set(int index, T elem);
  int indexOf(T elem);
  void remove(int index);
  void remove(T elem);
  IList<T> subList(int start, int end);
}
```

Recuperar y adaptar los tests

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

  public static void main(String[] args) {
    IList<String> l = new ArrayList<String>();

    System.err.println(l);

    assert l.size() == 0;

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

    System.err.println(l);

    ...
  }
}
```

Redimensionando de uno en uno

- `class ArrayList<T> implements IList<T>`
- Un único atributo: `private T[] data`
- El array se crea con longitud 0
- Se redimensiona en 1 al añadir
- Se redimensiona en 1 al borrar
- No se dejan huecos (ie. sin *nulles*)

30 minutos

Redimensionando de uno en uno

- `class ArrayList<T> implements IList<T>`
- Un único atributo: `private T[] data` 
- El array se crea con longitud 0
- Se redimensiona en 1 al añadir
- Se redimensiona en 1 al borrar
- No se dejan huecos (ie. sin *nulles*)

 30 minutos

Array genérico: el problema

```
private T[] data;

public ArrayList() {
  data = new T[0]; // error: generic array creation
}
```

- Es un problema técnico de análisis no trivial que tiene que ver con la *covarianza* de la herencia en arrays nativos:

```
Object[] os = new String[10];
os[0] = new Object();
```

Array genérico: una solución

```
private Object[] data;

public ArrayList() {
  data = new Object[0];
}

public T get(int index) {
  return (T)data[index];
}
```

- Sencillo y seguro en este caso de uso (sin errores en tiempo de ejecución)

De uno en uno: el problema

```
for (int i = 0; i < 100_000; i++) l.add(l.size(), "D"+ i);
```

```
angel@T440p: /22 $ javac IListTest.java
angel@T440p: /22 $ time -p java -ea IListTest
real 12.79
user 12.40
sys 0.43
angel@T440p: /22 $ cowsay -p "Okay, Houston, we've had a problem here"
 ________________________________________
< Okay, Houston, we've had a problem here >
 ----------------------------------------
        \   ^__^
         \  (@@)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
angel@T440p: /22 $
```

Redimensionando de N en N

- Otro atributo: `int size`
- El array se crea con longitud N
- Se redimensiona en N al añadir
- Se redimensiona en N al borrar cuando se desaprovecha espacio
- Huecos sólo al final

15 minutos

Redimensionando de $1, 2, 4, 8, 16, \ldots$

- Problemas de frontera en el redimensionamiento (ej. insertar y borrar en la frontera del redimensionamiento)
  - En vez de redimensionar linealmente ...
  - se puede redimensionar *geométricamente*
- ¿Para qué sirve esta aproximación?
- ¿Qué problemas tiene?