

## Práctica 4

**Objetivo:** Practicar con el ADT Listas

**Descripción:** Escribir un programa para simular un sorteo de premios

- disponemos de
  - una lista de  $n$  nombres de personas que participan en un sorteo
  - una lista de  $m$  premios ( $m \leq n$ ), cada uno identificado por un String
- Los premios se sortean por su orden en la lista; al sortearse un premio se elige aleatoriamente a una persona de la lista
  - esa persona se elimina de la lista de candidatos a recibir premio
  - el premio recibido se elimina de la lista de premios
  - y ambos se añaden al final de una lista de premiados
- el proceso se repite hasta sortear todos los premios

## Práctica 4 (cont.)

Realizar una clase que contenga

- la lista de candidatos a premio
- la lista de premios
- la lista de premiados (parejas nombre-premio)

Los métodos de la clase serán

- **constructor:** se le pasa una lista de  $m$  premios y una lista de  $n$  personas, que se copian en las lista de candidatos y premios
  - si  $m > n$  se lanza una excepción
- **sortea( ):** simula un paso del algoritmo, consistente en sortear el premio que toque en ese momento
  - retorna el nombre identificador del premio sorteado, o `null` si ya no quedan premios por sortear

## Práctica 4 (cont.)

- **sorteaTodos()**: llama sucesivamente a **sortea()** hasta que no queden más premios por sortear
  - retorna la lista de premiados (parejas nombre-premio)
- **muestra()**: muestra en pantalla el estado actual del objeto
  - lista de candidatos a premio, lista de premios sin sortear, y lista de premiados

Escribir también un programa de prueba que muestre las listas iniciales y permita sortear premios de uno en uno, o todos a la vez, mostrando a cada paso el estado de las listas.

## Práctica 4 (cont.)

### Entregar:

- diagrama de la clase
- diseño de los métodos
- una tabla con las eficiencias de operaciones de **ArrayList** y **LinkedList**
- evaluación de la eficiencia de los métodos (en función de **n** y **m**) suponiendo dos alternativas de implementación, según el tipo de lista:
  - **vector**: lista con acceso posicional  $O(1)$ , inserción y eliminación al final  $O(1)$ , resto de inserciones y eliminaciones  $O(n)$  y recorrido  $O(1)$
  - **lista enlazada**: lista con acceso posicional  $O(n)$ , e inserción y eliminación con el iterador  $O(1)$ , y recorrido con el iterador  $O(1)$
- código de la clase y del programa de prueba