# Laboratorio A.E.D.

**Lars-Åke Fredlund**
lfredlund@fi.upm.es
**Tonghong Li**
tonghong@fi.upm.es
**Manuel Carro Liñares**
mcarro@fi.upm.es
**Germán Puebla Sánchez**
german@fi.upm.es
**Pablo Nogueira**
pnogueira@fi.upm.es

Viernes 11:00-13:00

# Entrega

- La fecha límite para optar a la máxima nota es
  Lunes 29 de octubre de 2012, a las 13:00 horas
- Los ficheros que hay que subir son `MyElementIterator.java` y `MySparseElementIterator.java` (son dos ficheros)
- La entrega se hace a través de la siguiente URL:
  `http://lml.ls.fi.upm.es/~entrega`
- El paquete `positionListIterators` esta documentado con Javadoc en
  `http://babel.ls.fi.upm.es/~fred/courses/aed/positionListIterators/`
- El proyecto debe compilar sin errores, cumplir la especificación y pasar el `Tester`.

# Configuración

- Arrancad Eclipse.
- Cread un paquete `positionListIterators` en el proyecto `aed`, dentro de `src`.
- Aula Virtual → AED → Sesiones de laboratorio → Laboratorio 5 → codigo_lab5.zip (formato zip).
- Importad al paquete `positionListIterators` las fuentes que habéis descargado
- Ejecutad `Tester`. Veréis que lanza una excepción:

```
Testing MyElementIterator...
Exception in thread "main" java.lang.IllegalStateException: remove
at positionListIterators.MyElementIterator.remove(MyElementIterator.java:4
at positionListIterators.Tester.doTest(Tester.java:78)
at positionListIterators.Tester.main(Tester.java:34)
```

# Tareas para hoy

- Hoy trabajaremos otra vez con los APIs `PositionList` del paquete `net.datastructures` e `Iterator` del paquete `java.util`. Esta vez vamos a **implementar** iteradores para recorrer las listas.
- Se pide completar las clases `MyElementIterator` y `MySparseElementIterator` que implementan iteradores, es decir, implementan la interfaz `Iterator<E>`. Ambas clases son variantes de la clase `ElementIterator`, cuya implementación habéis visto en clase.
- Esta permitido añadir nuevos métodos, atributos privados, o variables locales.
- Los iteradores están explicados en el libro de texto, durante las clases, y también en la web (`http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Iterator.html`).

# Tareas para hoy (2)

- Se pide implementar el método `remove()` de la clase `MyElementIterator`.
- El método `remove()` debe borrar (de la lista iterada) el nodo que corresponde al elemento devuelto por la última llamada a `next()`.
- Probablemente tendréis que realizar algún cambio pequeño en los otros métodos, por ejemplo `next()`.
- Como consejo, estudia atentamente la explicación sobre el método `remove` en el enlace web `http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Iterator.html`

# Ejemplo `MyElementIterator`

Ejemplos: Si `l` es una lista con los elementos $\{10, 5, 20\}$ y se crea un iterador usando `t = new MyElementIterator(l);` este fragmento de programa debería ejecutar así:

```
t.remove();              lanza excepción – next() no ha sido llamado
t.next();                devuelve 10
t.remove();              no lanza excepción y borra el nodo con 10
t.remove();              lanza excepción
t.hasNext();             devuelve true
t.next();                devuelve 5
// observad que la lista l ha cambiado
l.size();                devuelve 2 (no 3)
l.first().element();     devuelve 5 (no 10)
```

# Tareas para hoy (3)

- Se pide cambiar la clase `MySparseElementIterator`.
- El codigo de `MySparseElementIterator` implementa un iterador "normal" sobre listas.
- Se pide modificar `MySparseElementIterator`:
  - El método `next()` debe devolver sólo los elementos distintos a `null`
  - El método `hasNext()` sólo toma en cuenta los elementos que no son `null`.

# Ejemplo `MySparseElementIterator`

Ejemplos: Si `l` es una lista con los elementos $\{10, null, 20\}$ y se crea un iterador usando

```
t = new MySparseElementIterator(l);
```

el siguiente fragmento de programa debe ejecutar así:

| | |
|---|---|
| `t.hasNext();` | devuelve `true` |
| `t.next();` | devuelve 10 |
| `t.hasNext();` | devuelve `true` |
| `t.next();` | **devuelve 20** |
| `t.hasNext();` | **devuelve false** |

Es decir, los métodos `next` y `hasNext` "saltan" valores `null`.