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
  Hoy, "Miércoles" 10 de octubre de 2012, a las 13:00 horas
- Después la puntuación máxima se reduce progresivamente:

Hasta (hoy) 10 de Octubre, 13:00 horas	10
(Jueves) 11 de Octubre, 13:00 horas	8
(Lunes) 15 de Octubre, 13:00 horas	6
(Martes) 16 de Octubre, 13:00 horas	4
...	

- El fichero que hay que subir es
  `MoreExtendedNodePositionList.java`
- La entrega se hace a través de la siguiente URL:
  `http://lml.ls.fi.upm.es/~entrega`
- El paquete `moreExtendedNodePositionList` esta documentado con Javadoc en
  `http://babel.ls.fi.upm.es/~fred/courses/aed/moreExtendedNodePositionList/`
- El proyecto debe compilar sin errores, cumplir la especificación y pasar el `Tester`.

Configuración

- Arrancad Eclipse.
- Cread un paquete `moreExtendedNodePositionList` en el proyecto `aed`, dentro de `src`.
- Aula Virtual → AED → Sesiones de laboratorio → Laboratorio 3 → codigo_lab3.zip (formato zip).
- Importad al paquete `moreExtendedNodePositionList` las fuentes que habéis descargado
- Ejecutad `Tester`. Veréis que lanza una excepción:

```
Testing reverse...
*** Error: the reverse of list {0,1,2,3,4} should be {4,3,2,1,0}
           but is null
Exception in thread "main" java.lang.Error
at moreExtendedNodePositionList.Tester.doTest(Tester.java:54)
at moreExtendedNodePositionList.Tester.main(Tester.java:30)
```

Consejos

- Haced un primer diseño del algoritmo en papel, abstrayendo los detalles menores
- Es muy útil dibujar las estructuras de datos que uséis
- Simulad el algoritmo en papel:
  - ¿qué pasa con los variables durante los bucles?
  - ¿qué pasa con las estructuras de datos?
- Si falla algo:
  - Para depurar vuestro código os será **muy** útil imprimir los valores de las variables del programa:

    ```
    System.out.println("The current element is "+currPos.element());
    ```

  - o usar el "debugger" de Eclipse
  - y, obviamente, revisar el codigo

Tareas para hoy

- Hoy trabajaremos otra vez con el API `PositionList` del paquete `net.datastructures`.
  - Recordad: todo el código fuente del paquete está disponible en el Aula Virtual.
- Se pide completar los métodos `reverse()` y `unique()` de la clase `MoreExtendedNodePositionList` que implementa `PositionList`. Esta permitido añadir nuevos métodos o variables locales.
- El método `reverse()` debe devolver una nueva lista que contiene los elementos en orden inverso.
- Ejemplo: Si `l` es una lista con los elementos $10, 4, 15, 4, 5$, `l.reverse()` debe devolver una nueva lista con los elementos $5, 4, 15, 4, 10$. Si `l` es una lista vacía, `l.reverse()` debe devolver otra lista vacía.

El método no debe cambiar la lista sobre la que se invoca (accesible usando `this`).

Tareas para hoy (2)

- `d = l.unique()` devuelve en `d` una lista sin los elementos duplicados que pudiesen aparecer en `l`. Para los elementos duplicados, `unique()` debe preservar la primera ocurrencia del elemento (la más cercana al principio de la lista).
- Ejemplos:
  - Si `l` es 0,3,1,2,3 entonces la llamada `l.unique()` debe devolver una nueva lista con los elementos 0,3,1,2.
  - Si `l` es una lista vacia, `l.unique()` debe devolver una nueva lista vacia.
  - Si `l` es 0,3,9,0,2,9,0 la llamada `l.unique()` debe devolver una nueva lista 0,3,9,2.
- Para comparar dos elementos `e1` y `e2` se puede usar la comparación `e1.equals(e2)` que devuelve `true` si son iguales y `false` si no son iguales.

El método no debe cambiar la lista sobre la que se invoca (accesible usando `this`).