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, Viernes 5 de octubre de 2012, a las 13:00 horas
- El fichero que hay que subir es
  `ExtendedNodePositionList.java`
- La entrega se hace a través de la siguiente URL:
  `http://lml.ls.fi.upm.es/~entrega`
- El paquete extendedPositionList esta documentado con Javadoc en
  `http://babel.ls.fi.upm.es/~fred/courses/aed/extendedPositionList/`
- El proyecto debe compilar sin errores, cumplir la especificación y pasar el `Tester`.

Configuración

- Arrancad Eclipse.
- Descargar la "Librería de la Asignatura" desde

  `http://net3.datastructures.net/lib5/net-datastructures-5-0.jar`
- Pinchar en el proyecto *aed* con el botón derecho:
  - Seleccionar *Propiedades*, *Java Build Path*, *Libraries*.
  - Dependiendo de dónde lo hayais salvado hay que seleccionar *Add JARs* o *Add External JARs*
  - Seleccionar `net-datastructures-5-0.jar`.
  - Debe apareceros dentro del proyecto `aed` un nuevo elemento `Referenced Libraries` con el `.jar` que hemos seleccionado.
- Cread un paquete `extendedNodePositionList` en el proyecto `aed`, dentro de `src`.
- Aula Virtual $\rightarrow$ AED $\rightarrow$ Sesiones de laboratorio $\rightarrow$ Laboratorio 2 $\rightarrow$ codigo_lab2.zip (formato zip).
- Importad al paquete `extendedNodePositionList` las fuentes que habéis descargado
- Ejecutad `Tester`. Veréis que lanza una excepción.

Tareas para hoy

- Hoy trabajaremos con el API `PositionList` del paquete `net.datastructures`. (Recordamos que el todo el código fuente del paquete está disponible en el Aula Virtual.)
- Se pide completar los métodos `nth` y `fairMerge` de la clase `ExtendedNodePositionList` que implementa un `PositionList`. Esta permitido añadir nuevos métodos o variables.
- El método `Position<E> nth(int n)` debe devolver el nodo `n`-ésimo de la lista, siempre que `n` sea menor o igual que la longitud de la lista. En caso contrario debe lanzar la excepción `BoundaryViolationException`. Puede asumirse que `n` será siempre mayor que cero.
- Ejemplo: Si `l` es una lista con los elementos $10, 20, 15, 4, 5$:
  - `l.nth(1)` debe devolver el nodo con el elemento 10.
  - `l.nth(3)` debe devolver el nodo con el elemento 15.
  - `l.nth(6)` debe lanzar la excepción `BoundaryViolationException`.

Tareas para hoy (2)

- El método `fairMerge` combina los elementos de dos listas `la` y `lb` en una **nueva** lista `l`. El método **no puede modificar** las listas `la` o `lb`.
- Se construye `l` alternando entre `la` y `lb`.
- Concretamente
  - el primer elemento de `l` es el primer elemento de `la` (si existe),
  - y el segundo elemento es el primer elemento de `lb` (si existe),
  - y el tercer elemento es el segundo elemento de `la`,
  - y el cuarto elemento es el segundo elemento de `lb`, etc
- Ejemplo: si asumimos una lista la con los elementos $1, 2, 3$ y una lista lb con los elementos $-1, -3, -3, -4, -5$
  - la llamada `la.fairMerge(lb)` deberia devolver una lista nueva $1, -1, 2, -3, 3, -3, -4, -5$
  - la llamada `lb.fairMerge(la)` devuelve una lista nueva $-1, 1, -3, 2, -3, 3, -4, -5$