Laboratorio A.E.D. Ejercicio Individual 4

Guillermo Román
guillermo.roman@upm.es
Lars-Åke Fredlund
lfredlund@fi.upm.es
Manuel Carro
mcarro@fi.upm.es
Marina Álvarez
marina.alvarez@upm.es
Julio García
juliomanuel.garcia@upm.es
Tonghong Li
tonghong@fi.upm.es

Normas.

- Fechas de entrega y penalización:

Hasta el Martes 11 de octubre, 12:00 horas	0
Hasta el Jueves 13 de octubre, 12:00 horas	20 %
Hasta el Viernes 14 de octubre, 12:00 horas	40 %

  Después la puntuación máxima será 0
- Se comprobará plagio y se actuará sobre los detectados
- Usad las horas de tutoría para preguntar sobre programación – son oportunidades excelentes para aprender

Entrega

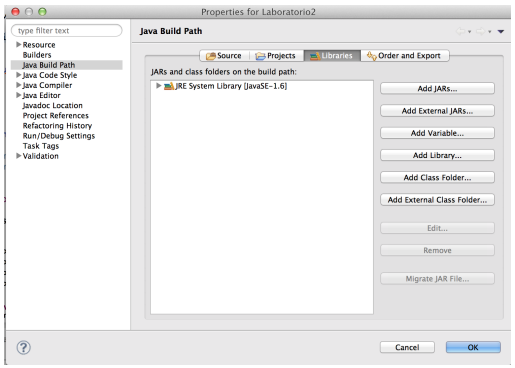
- Todos los ejercicios de laboratorio se deben entregar a través de `https://deliverit.fi.upm.es`
- El fichero que hay que subir es `NIterator.java`.

Configuración previa

- Arrancad Eclipse
- Si trabajáis en portátil, podéis utilizar cualquier versión reciente de Eclipse. Es suficiente con que instaléis la *Eclipse IDE for Java Developers*.
- Cambiad a "Java Perspective".
- Debéis tener instalado al menos Java JDK 8.
- Cread un proyecto Java llamado `aed`:
  - Seleccionad separación de directorios de fuentes y binarios.
  - **No debéis elegir la opción de crear el fichero** `module-info.java`
- Cread un *package* `aed.positionlistiterator` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Laboratorios y Entregas Individuales → Individual 4 → `Individual4.zip`; descomprimidlo
- Contenido de `Individual4.zip`:
  - `NIterator.java`, `TesterInd4.java`

Configuración previa

- Importad al paquete `aed.positionlistiterator` los fuentes que habéis descargado (`NIterator.java`, `TesterInd4.java`)
- Añadid al proyecto aed la librería aedlib . jar que tenéis en Moodle (en Laboratorios y Entregas Individuales).

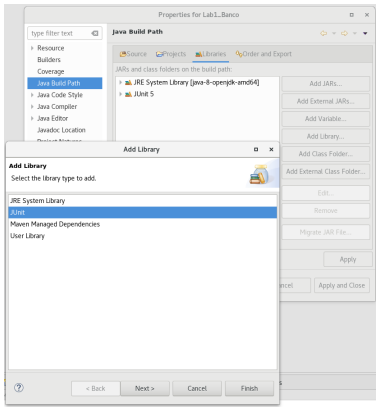


Para ello:

- Project → Properties → Java Build Path. Se abrirá una ventana como la de la izquierda
- Usad la opción “Add External JARs...”.
- Si vuestra instalacion distingue `ModulePath` y `ClassPath`, instalad en `ClassPath`

Configuración previa

- Añadid al proyecto aed la librería JUnit 5



  - Project → Properties → Java Build Path. Se abrirá una ventana como la de la izquierda;
  - Usad la opción “Add Library...” → Seleccionad “Junit” → Seleccionad “JUnit 5”
  - Si vuestra instalacion distingue `ModulePath` y `ClassPath`, instalad en `ClassPath`
- En la clase `TesterInd4` tenéis las pruebas, para ejecutarlas, abrid el fichero `TesterInd4`, pulsando el botón derecho sobre el editor, seleccionar “Run as...” → “JUnit Test”
- NOTA: Si al ejecutar, no aparece la vista “JUnit”, podéis incluirla en “Window” → “Show View” → “Java” → “JUnit”

Documentación de la librería aedlib.jar

- La documentación de la API de aedlib.jar está disponible en

`http://costa.ls.fi.upm.es/teaching/aed/docs/aedlib/`

- También se puede añadir la documentación de la librería a Eclipse (*no es obligatorio*):
  - En el "Package Explorer": "Referenced Libraries" → aedlib.jar y elige la opción "Properties". Se abre una ventana donde se puede elegir "Javadoc Location" y ahí se pone como "javadoc location path:"

`http://costa.ls.fi.upm.es/teaching/aed/docs/aedlib/`

y presionar el buton "Apply and Close"

Tarea: implementar un iterador sobre una lista de posiciones

- Se pide terminar la implementación de la clase

```
public class NIterator<E> implements Iterator<E> {

  // Constructor
  public NIterator(PositionList<E> list, int n) {
    ...
  }

}
```

- La clase `NIterator<E>` implementa un iterador sobre elementos de tipo (generico) `E`
- Tiene un constructor que recibe como argumento una lista de posiciones `list`, y un entero `n` > 0

Tarea: implementar un iterador sobre una lista de posiciones

- El iterador debe devolver una sequencia de elementos (a través de los métodos `hasNext` y `next`) que cumple con las siguientes condiciones:
  - El primer elemento devuelto es el primer elemento de la lista, *excepto* si es `null`. En este caso el iterador busca el primer elemento $\neq$ *null* en la continuación de la lista, y devuelve este elemento.
  - Después de haber devuelto un elemento, el siguiente elemento devuelto por el iterador debe ser el elemento n posiciones adelante en la lista. Sin embargo, si este elemento es `null`, se busca y devuelve el primer elemento $\neq$ *null* en la continuación de la lista.
- *No está permitido modificar la lista parámetro al constructor*
- No es necesario implementar el metodo `remove` del iterador
- Está permitido y es necesario añadir atributos dentro la clase. El uso de métodos axiliares está permitido y recomendado

Ejemplos

- En los ejemplos podéis ver el resultado a imprimir:

```
Iterator<E> it = new NIterator<Integer>(l,n);
while (it.hasNext()) {
  System.out.println(it.next()+",");
}
```

- Ejemplos:

```
l = [];                                    =>
l = [1], n=3                               => 1
l = [1,2,null,3,null], n=1                 => 1,2,3

l = [1,2,3,4], n=2                         => 1,3
l = [null,null,1,2,3,4], n=2               => 1,3
l = [1,2,null,null,3,4,null,5,6], n=2      => 1,3,5
```

Un Iterador Eficiente

- **Es obligatorio que el iterador implementado por la clase `NIterator` sea eficiente**:
  - No está permitido guardar los elementos que el iterador debe devolver en otra estructura de datos, y devolver el iterador sobre esta estructura de datos
  - No está permitido hacer una llamada a `new` dentro la clase `NInterface`, excepto para lanzar una excepcion

Notas importantes

- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar.
- Debe ejecutar `TesterInd4` correctamente sin mensajes de error.
- Nota: una ejecución sin mensajes de error **no** significa que el método sea correcto (es decir, que funcione bien para cualquier entrada posible).
- Todos los ejercicios se comprueban manualmente antes de dar la nota final.