Entrega Individual 4

Guillermo Román
groman@fi.upm.es
Julio García
juliomanuel.garcia@upm.es
Lars-Åke Fredlund
lfredlund@fi.upm.es
Manuel Carro Liñares
mcarro@fi.upm.es
Marina Álvarez
marina.alvarez@upm.es
Raúl Correal
raul.correal@upm.es
Tonghong Li
tonghong@fi.upm.es

Normas

- **La entrega del ejercicio es individual**
- Fechas de entrega y nota máxima alcanzable:

Hasta el Viernes 21 de Octubre, 23:59 horas	10
Hasta el Domingo 23 de Octubre, 23:59 horas	8
Hasta el Lunes 24 de Octubre, 23:59 horas	6

- Después la máxima puntuación será 0
- Se comprobará plagio y se actuará sobre los detectados

Sistema de Entrega

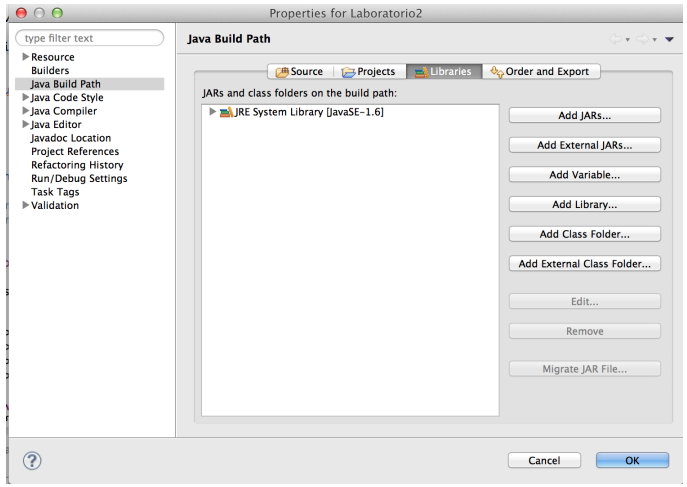
- Todas los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`
- Los ficheros a subir son `PositivePositionListIterator.java`

Configuración previa al desarrollo del ejercicio.

- Arrancad Eclipse. Es suficiente con que tengáis la *Eclipse IDE for Java Developers*
- Cambiad a "Java Perspective"
- Cread un proyecto Java llamado `aed`:
  - Seleccionad separación de directorios de fuentes y binarios
- Cread un *package* `aed.positiveiterator` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Sesiones de laboratorio → Entrega Individual 4 → `EntregaIndividual4.zip`; descomprimidlo
- Contenido de EntregaIndividual4.zip
  - `TesterInd4.java`, `PositivePositionListIterator.java`
- Descargad también el fichero `aedlibraries.jar`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `aed.positiveiterator` los fuentes que habéis descargado
- Añadid al proyecto `aed` la librería `aedlibraries.jar` que habéis descargado. Para ello:
- Project → Properties. Se abrirá una ventana como esta:



- Java Build Path → Libraries → Add external JARs → Seleccionad el fichero `aedlibraries.jar`
- El fichero `PositivePositionListIterator.java` no compila, es tarea vuestra completar los métodos que faltan

Tarea para hoy: Entrega Individual 4

- Se pide implementar la clase

  `public class PositivePositionListIterator<E extends Integer>`

  `implements Iterator<Integer>`
- Dicha clase debe implementar un iterador sobre una lista de tipo `PositionList<Integer>` que únicamente itere sobre los números positivos contenidos en la lista
- Por tanto, el método `next` devolverá únicamente los elementos positivos contenidos en la lista
- Los elementos contenidos en la lista podrán ser `null`, pero estos elementos `null` no deberán ser devueltos por `next()`

Tarea para hoy: Entrega Individual 4

- El método `hasNext` devolverá `true` si quedan en la lista elementos positivos después del último elemento devuelto por `next()` y `false` e.o.c.
- El método `remove()` del iterador lanzará siempre la excepción `UnsupportedOperationException`
- La clase tiene un constructor

  `PositivePositionListIterator(PositionList<Integer> list)`
- Si el parámetro `list` es `null` el constructor lanzará la excepción `IllegalArgumentException`

Ejemplos

- Dadas las siguientes listas, el siguiente bucle del iterador:

```
while (it.hasNext()){
  System.out.print(it.next()+ "_");
}
```

debe imprimir los siguientes elementos:

{2,2,2,2} ⇝ 2 2 2 2

{2,2,null,null,2,3,-1,-1} ⇝ 2 2 2 3

{2,-2,1,2} ⇝ 2 1 2

{} ⇝ Ninguno!

{1,2,3,null,null,0} ⇝ 1 2 3

{null} ⇝ Ninguno!

{-1,-2,null} ⇝ Ninguno!

{-1,-2,3} ⇝ 3

null ⇝ IllegalArgumentException

Comentarios generales

- Al usar

  ```
  PositivePositionListIterator<E extends Integer> implements
  Iterator<Integer>
  ```

  el iterador devolverá siempre elementos de tipo Integer, con lo que el tipo devuelto por `next()` debe ser un `Integer`, es decir:

  ```
  public Integer next()throws NoSuchElementException {...}
  ```
- En el fichero `Example.java` tenéis un main para poder hacer vuestras pruebas
- Podéis crear los atributos y métodos privados que consideréis oportunos
- 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 cada posible entrada)