Laboratorio A.E.D. Ejercicio Individual 3

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 nota máxima alcanzable:

Hasta el Lunes 2 de octubre, 23:59 horas	10
Hasta el Martes 3 de octubre, 23:59 horas	8
Hasta el Miércoles 4 de octubre, 23:59 horas	6
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

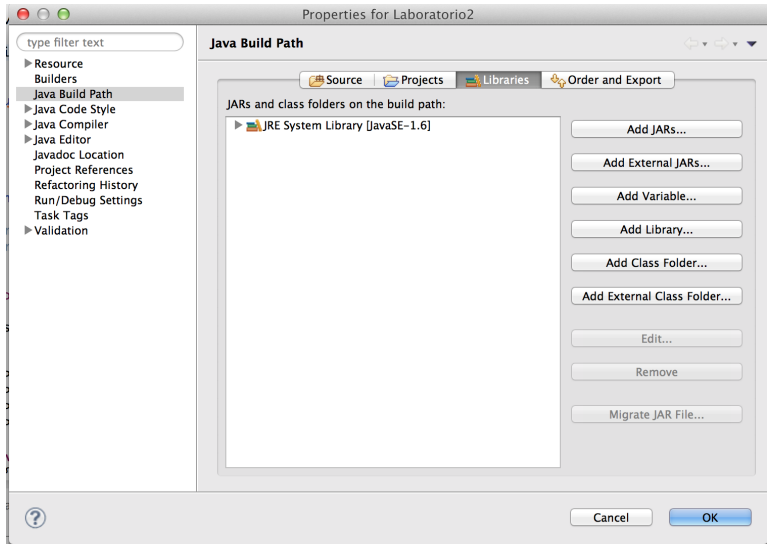
- Todos los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`.
- El fichero que hay que subir es `MergeLists.java`.

Configuración previa

- Arrancad Eclipse
- Si trabajáis en portátil, podéis utilizar cualquier versión relativamente reciente de Eclipse. Debería valer cualquier versión a partir de la versión 3.7. Es suficiente con que instaléis 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.mergelists` en el proyecto `aed`, dentro de `src`
- Aula Virtual $\rightarrow$ AED $\rightarrow$ Laboratorios y Entregas Individuales $\rightarrow$ Individual 3 $\rightarrow$ `Individual3.zip`; descomprimidlo
- Contenido de `Individual3.zip`:
  - `MergeLists.java`,`TesterInd3.java`, `AscendingComparator.java`, `DescendingComparator.java`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `aed.mergelists` los fuentes que habéis descargado (`MergeLists.java`,`TesterInd3.java`, `AscendingComparator.java`, `DescendingComparator.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 esta:



- Usad la opción “Add External JARs...”.
- Intentad ejecutar `TesterInd3`

Documentación de la librería aedlib.jar

- La documentación de la API de la librería aedlib.jar esta disponible en `http://lml.ls.fi.upm.es/~entrega/aed/docs/aedlib/`.
- También se puede añadir la documentación de la librería a Eclipse (*no es obligatorio*): en el "Package Explorer": "Referenced Libraries" $\rightarrow$ 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://lml.ls.fi.upm.es/ entrega/aed/docs/aedlib/
  y presionar el botón "Apply and Close"

Tarea: Combinar Dos Listas Ordenadas

- Se pide implementar dos métodos de la clase `MergeLists`:

```
static <E> PositionList<E> merge(PositionList<E> l1,
                                 PositionList<E> l2,
                                 Comparator<E> cmp);
static <E> IndexedList<E> merge(IndexedList<E> l1,
                                IndexedList<E> l2,
                                Comparator<E> cmp)
```

  Ambos métodos reciben dos listas, `l1` y `l2`, *ordenadas* según el comparador `cmp`, y devuelven una lista *ordenada* **nueva** que contiene todos los elementos de `l1` y `l2`

- La correcta implementación del método `merge` con argumentos de tipo `PositionList` es **obligatoria**
- La implementación del método `mergeList` con argumentos de tipo `IndexedList` **no es obligatoria**:
  - La nota máxima sin implementar correctamente el método es **8**
  - La nota máxima con una implementación correcta es **11**

Reglas de la Implementación

- Para crear una lista de posiciones se puede usar

```
PositionList<E> list = new NodePositionList<E>();
```

- No se debe modificar las listas de entrada `l1` ni `l2`
- Para la implementación con `PositionList` el uso de `first()` o `last()`, y `next(...)` o `prev(...)`, para recorrer la lista es obligatorio
- Para `IndexedLists` el uso del método `get(int)`, para travesar la lista, es obligatorio
- No esta permitido el uso de iteradores para implementar los dos métodos `merge`: no se puede llamar al método `iterator()` ni usar bucles "*for-each*" . Por ejemplo, no esta permitido el fragmento de código abajo:

```
for (E elem : l1) ... // cuando l1 es una lista
```

  Sin embargo bucles *for* "normales" están permitidos
- No esta permitido llamar los métodos `toArray` de las listas

Ejemplos

Ejemplos (asumimos que `cmp` es una comparador que ordena los integers en orden ascendente):

```
merge([],[],cmp)                  -->  []
merge([],[2,3,4],cmp)             -->  [2,3,4]
merge([1],[2],cmp)                -->  [1,2]
merge([2],[1],cmp)                -->  [1,2]
merge([1,3],[4],cmp)              -->  [1,3,4]
merge([1,4],[3,4,4],cmp)          -->  [1,3,4,4,4]
merge([1,4,4],[2,3,3,4],cmp)      -->  [1,2,3,3,4,4,4]
```

Notas

- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar
- Debe ejecutar `TesterInd3` 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)
- Todos los ejercicios se comprueban manualmente antes de dar la nota final