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

Sergio Paraiso
sergio.paraiso@upm.es **Juan José Moreno**
juanjose.moreno@upm.es

Luis Miguel Danielsson
lm.danielsson@alumnos.upm.es

Normas.

- Fechas de entrega y nota máxima alcanzable:

Hasta el Martes 19 de octubre, 12:00 horas	10
Hasta el Miércoles 20 de octubre, 12:00 horas	8
Hasta el Jueves 7 21 octubre, 12:00 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

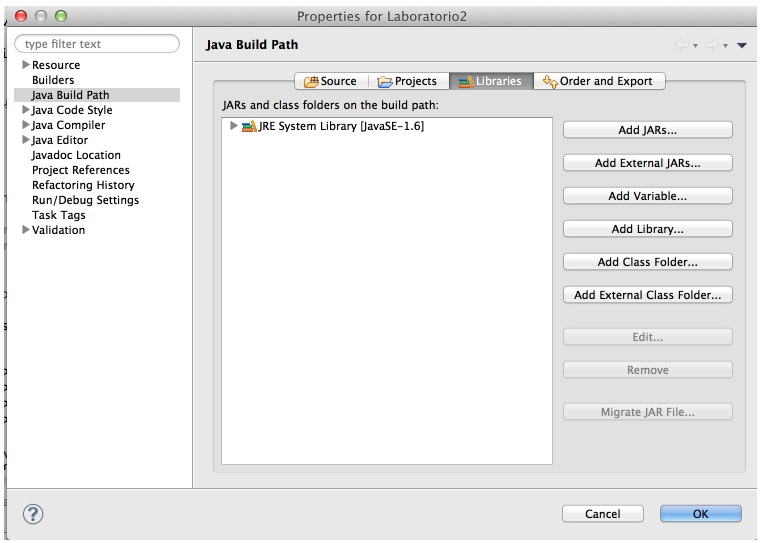
- Todos los ejercicios de laboratorio se deben entregar a través de `http://vps142.cesvima.upm.es`
- El fichero que hay que subir es `MultiSetListIterator.java`.

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.invididual4` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Laboratorios y Entregas Individuales → Individual 4 → Individual4.zip; descomprimidlo
- Contenido de Laboratorio2.zip:
  - `TesterInd4.java`, `MultiSetListIterator.java`

Configuración previa

- Importad al paquete `aed.invididual4` los fuentes que habéis descargado (`TesterInd4.java`, `MultiSetListIterator.java`)
- Añadid al proyecto `aed` la librería `aedlib.jar` que tenéis en Moodle (en Laboratorios y Entregas Individuales).

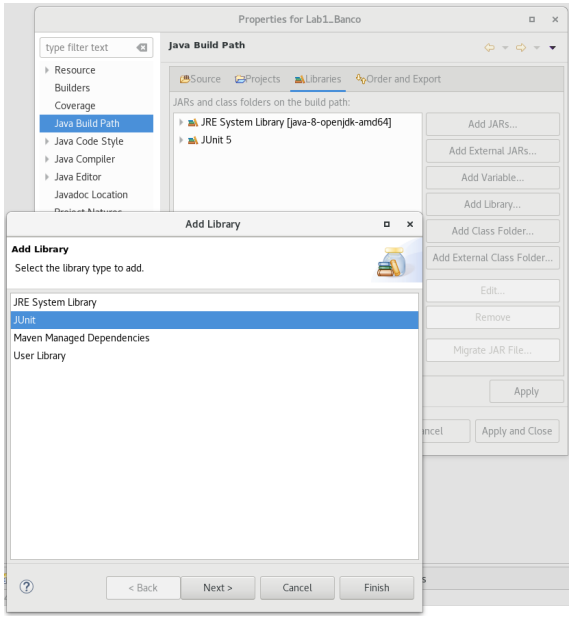


Para ello:

- 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`



Para ello:

- 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/entrega/aed/docs/aedlib/`
- Tambien 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/entrega/aed/docs/aedlib/`

    y presionar el buton "Apply and Close"

Tarea para hoy

- En este individual vamos a implementar un iterador *eficiente* para una representación de multiconjuntos
- La tarea consiste en terminar la implementación de la clase `MultiSetListIterator` completando la implementación de los métodos: `hasNext()` y `next()`
- De manera opcional, también se puede completar la implementación del método `remove` (*no es obligatorio para hacer la entrega*)
- Si la implementación de `hasNext()` y `next()` es correcta la puntuación máxima será 8. Si además la implementación de `remove()` es correcta la puntuación máxima será 10

Multiconjuntos

- Un "multiconjunto" se comporta como un conjunto, excepto que los multiconjuntos admiten elementos repetidos
- Ejemplo: el multiconjunto {'a','b','c','a'} contiene dos caracteres 'a', un 'b', y un 'c'
- En esta practica individual se representa un multiconjunto, con elementos de tipo `E`, como una lista de tipo `PositionList<Pair<E,Integer>>`. Es decir, una lista de pares compuesto por un elemento y un numero que indica cuantas veces occurre el elemento en el multiconjunto
- Por ejemplo, el multiconjunto {'a','b','c','a'} esta representado como una lista de tipo `PositionList<Pair<Character,Integer>>` que contiene los elementos (pares) `<'a',2>,<'b',1>,<'c',1>`.
- **Notad**: se puede asumir que para todas las pares $\langle e, n \rangle$ en la lista de posiciónes, el entero $n > 0$.

Implementación

- El constructor

```
MultiSetListIterator(PositionList<Pair<E,Integer>> list)
```

  recibe una lista sobre que el iterador debe iterar
  - Debéis guardar la referencia a la lista en el atributo `list`
  - Nótese que la lista `list` representa el multiconjunto como explicado anteriormente
- Probablemente necesitaréis definir dos atributos nuevos (esta permitido) del estilo de:

```
Position<Pair<E,Integer>> cursor; // posicion por donde va el iterador
int contador;                     // numero del elemento en la posicion
```

- Si completáis la implementación del método `remove()` está necesitaréis añadir otro atributo a la clase para guardar el último elemento devuelto por `next()`
- Los atributos deben ser inicializados en el constructor, y comprobados en `hasNext()` y modificados por `next()` y `remove()`

Tarea para hoy: Ejemplo

- Dada la lista `list = <a,2>,<b,4>,<c,5>` el iterador debe devolver los elementos, en el orden, `a,a,b,b,b,b,c,c,c,c,c`
- **Notad:** el orden de los elementos devueltos por el iterador importa. Si la lista de posiciones contiene un par `<a,2>` como primer elemento, entonces el iterador deberia devolver primero `a`, despues otro `a`, y después continuar con el segundo par de la lista de posiciónes.
- El método `remove()` debe borrar el último elemento devuelto por `next`, es decir:

```
// list = <a,2>,<b,4>,<c,5>
Iterator<E> it = new MultisetListIterator<E>(list)
it.next() -> a
it.next() -> a
it.remove() -> list = <a,1>,<b,4>,<c,5>
it.next() -> b
it.remove() -> list = <a,1>,<b,3>,<c,5>
```

Tarea para hoy: Notas

- Esta **prohibido** crear objetos nuevos dentro `MultiSetListIterador.java`, es decir no se puede llamar `new` **excepto** para crear una excepción y lanzarlo.
- Si no ha habido una llamada a `next()` antes de una llamada a `remove()`, la llamada a `remove()` debe lanzar la excepción `IllegalStateException`.
- No está permitido llamar dos veces consecutivas a `remove()`
- Para más detalles sobre el comportamiento de los métodos `hasNext()`, `next()` y `remove()` la documentación sobre Iteradores en la documentación estandar de Java. Por ejemplo en, `https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html`.

Comentarios generales

- 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)
- Todos los ejercicios se comprueban manualmente antes de dar la nota final