

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

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 5 de octubre, 12:00 horas	10
Hasta el Miércoles 6 de octubre, 12:00 horas	8
Hasta el Jueves 7 de 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`

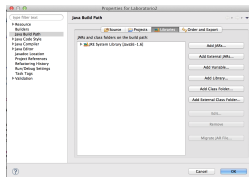
- ▶ El fichero que hay que subir es `Utils.java`.

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.filter` en el proyecto aed, dentro de `src`
- ▶ Aula Virtual → AED → Laboratorios y Entregas Individuales → Individual 3 → Individual3.zip; descomprimidlo
- ▶ Contenido de Individual3.zip:
 - ▶ `Utils.java`, `GreaterThan.java`, `TesterInd3.java`

Configuración previa

- ▶ Importad al paquete `aed.filter` los fuentes que habéis descargado (`Utils.java`, `GreaterThan.java`, `TesterInd3.java`)
- ▶ Añadid al proyecto `aed` la librería `aedlib.jar` que tenéis en Moodle (en Laboratorios y Entregas Individuales).

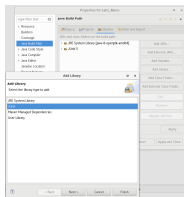


Para ello:

- ▶ Project → Properties → Java Build Path. Se abrirá una ventana como la de la izquierda
- ▶ Usad la opción “Add External JARs...”.
- ▶ Si vuestra instalación 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 TesterInd3 tenéis las pruebas, para ejecutarlas, abrid el fichero TesterInd3, 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/>
- ▶ 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/entrega/aed/docs/aedlib/>
y presionar el boton “Apply and Close”

Tarea: filtrar elementos en un iterable

- ▶ Se pide implementar el método

```
static <E> Iterable<E>  
    filter(Iterable<E> l, Predicate<E> pred)
```

dentro la clase `Utils`.

- ▶ Recibe un estructura de datos *Iterable* `l`, y devuelve una **nueva** estructura de datos *Iterable* `r` con los mismos elementos que el `l` y en el mismo orden, **excepto**:
 - ▶ `r` no podrá contener elementos `null`
 - ▶ `r` no contendrá los elementos filtrados por el predicate `pred`
- ▶ Si `l==null` el programa debe lanzar la excepción `IllegalArgumentException`
- ▶ No se debe modificar la estructura de entrada `l`

Predicados en Java

- ▶ Los predicados en Java nos permiten definir cuándo un objeto cumple una determinada condición
- ▶ Un predicado en Java es una instancia del interfaz `java.util.function.Predicate<E>`:

```
public interface Predicate<T> {  
    // Evaluates the predicate on its argument t  
    boolean test(T t);  
}
```

- ▶ Dado un `Predicate pred` y un valor `e`, para comprobar si se debe incluir `e` en el iterable resultante se puede llamar `pred.test(e)`.
- ▶ Por ejemplo, el `Tester` usa la clase `GreaterThan` que implementa el interfaz `Predicate`. La clase tiene un constructor `GreaterThan(E e)` y su método `test(E arg)` devuelve `true` si “`arg > e`” y `false` si no lo es

Ejemplos

Ejemplos. Usamos el sintaxis de listas $[e1, e2, e3]$ para iterables de tres elementos $e1, e2, e3$:

```
filter([], new GreaterThan(5))           --> []  
filter([1], new GreaterThan(5))          --> []  
filter([10], new GreaterThan(5))         --> [10]  
filter([1,2,4,5,6,7], new GreaterThan(5)) --> [6,7]  
filter([10,10,5,5,1,1], new GreaterThan(5)) --> [10,10]
```

Reglas y Consejos

- ▶ **Es obligatorio trabajar con iteradores** para acceder a los elementos del argumento `l`, concretamente llamar a los métodos `hasNext()` y `next()` de los iteradores
- ▶ **No esta permitido bucles “for-each”**. Es decir, no se puede codificar un bucle de la siguiente forma

```
for (E e : it) {  
    ...  
}
```

Notas importantes

- ▶ 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 cualquier entrada posible).
- ▶ Todos los ejercicios se comprueban manualmente antes de dar la nota final.