# Laboratorio A.E.D. Viernes 13:00 - 15:00 y 15:00 - 17:00

**Guillermo Vigueras**
guillermo.vigueras@imdea.org
**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
**Tonghong Li**
tonghong@fi.upm.es

# Normas.

- **¡Solo debe entregar una persona por grupo!**.
- Fechas de entrega y nota máxima alcanzable:

| | |
|---|---|
| Hasta el miércoles 18 de noviembre, 13:00 horas | 10 |
| Hasta el jueves 19 de noviembre, 13:00 horas | 8 |
| Hasta el viernes 20 de noviembre 13:00 horas | 6 |
| Hasta el lunes 21 de noviembre, 13:00 horas | 4 |

  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.

# Sistema de Entrega

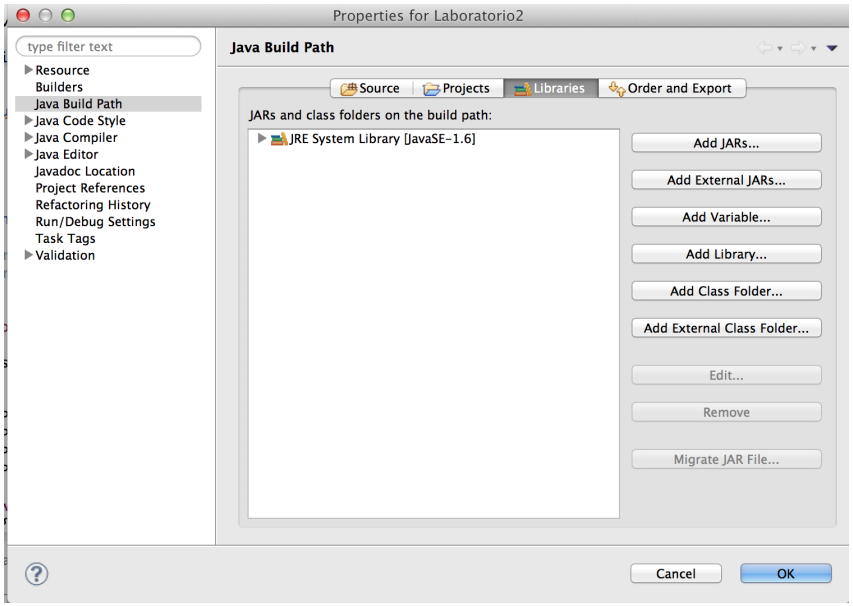
- Todas los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`.
- Hoy, el fichero que hay que subir es `Flatten.java`.

# Configuración previa al desarrollo del ejercicio.

- Arrancad Eclipse. Debéis tener un acceso directo.
- Si trabajáis en portátil, podéis utilizar cualquier versión relativamente reciente de Eclipse. Debería valer cualquier versión entre la versión 3.7 (Indigo) o 4.3 (Kepler). 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* `flatten` en el proyecto `aed`, dentro de `src`.
- Aula Virtual → AED → Sesiones de laboratorio → Laboratorio 6 → `Laboratorio6.zip`; descomprimidlo.
- Contenido de Laboratorio6.zip:
  - `Tester.java` y `Flatten.java`
  - `positionList.jar`

# Configuración previa al desarrollo del ejercicio.

- Importad al paquete `flatten` las fuentes que habéis descargado (`Tester.java` y `Flatten.java`).
- Añadid al proyecto `aed` la librería `positionList.jar` que habéis descargado. Para ello:
- Project → Properties. Se abrirá una ventana como esta:




- Java Build Path → Libraries → Add external JARs → Seleccionad el fichero `positionList.jar` que os habéis descargado.
- Ejecutad `Tester`. Veréis que imprime un mensaje de error.

# Tarea para hoy

- Realizar una implementación de los métodos:

  ```
  PositionList<E> flatNub(PositionList<PositionList<E>> listOfLists)
  boolean member(E elem, PositionList<E> list)
  ```

  dentro la clase `Flatten<E>`.
- `flatNub` recibe una lista de listas (`listOfLists`), y devuelve una lista *nueva* con los elementos, no repetidos, y en orden, que aparecen en las sublistas de `listOfLists`.
- `member` devuelve `true` si `list` contiene un elemento que es igual que `elem`, y `false` si no.
- Se puede asumir que `listOfLists`, `elem` y `list` no son `null`. Además, las sublistas de `listOfList` no son `null`, y todos los elementos de las sublistas son distintos a `null`.

# Importante

- El uso de iteradores (`java.util.iterator`) para implementar ambos métodos es **obligatorio**.
- **No está permitido** usar los métodos de la interfaz `PositionList`, excepto `iterator`, `addLast`, y `addBefore`.
- Es **obligatorio** usar el método `member` en la implementación de `flatNub`.

# Tarea para hoy (2)

- Resultado del método para distintos valores de entrada:

```
flatNub([]) ⇝ []
flatNub([[]]) ⇝ []
flatNub([[1,2]]) ⇝ [1,2]
flatNub([[2,1,2]]) ⇝ [2,1]
flatNub([[2],[2],[1]]) ⇝ [2,1]
flatNub([[],["hola"],["hi"]]) ⇝ ["hola","hi"]
flatNub([[1,2],[],[2,3,4],[8,4,]]) ⇝ [1,2,3,4,8]
```

# Tareas para hoy (3)

- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar.
- Debe ejecutar `Tester` 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.