Laboratorio A.E.D. Ejercicio Individual 6

Guillermo Román
guillermo.roman@upm.es

Julio García
juliomanuel.garcia@upm.es

Lars-Åke Fredlund
lfredlund@fi.upm.es

Tonghong Li
tonghong@fi.upm.es

Manuel Carro
mcarro@fi.upm.es

Clara Benac Earle
cbenac@fi.upm.es

Marina Álvarez
marina.alvarez@upm.es

Sergio Paraiso
sergio.paraiso@upm.es

Normas

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

Hasta el Miércoles 12 de Diciembre, 23:59 horas	10
Hasta el Jueves 13 de Diciembre, 23:59 horas	8
Hasta el Viernes 14 de Diciembre, 23:59 horas	6

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

Sistema de Entrega

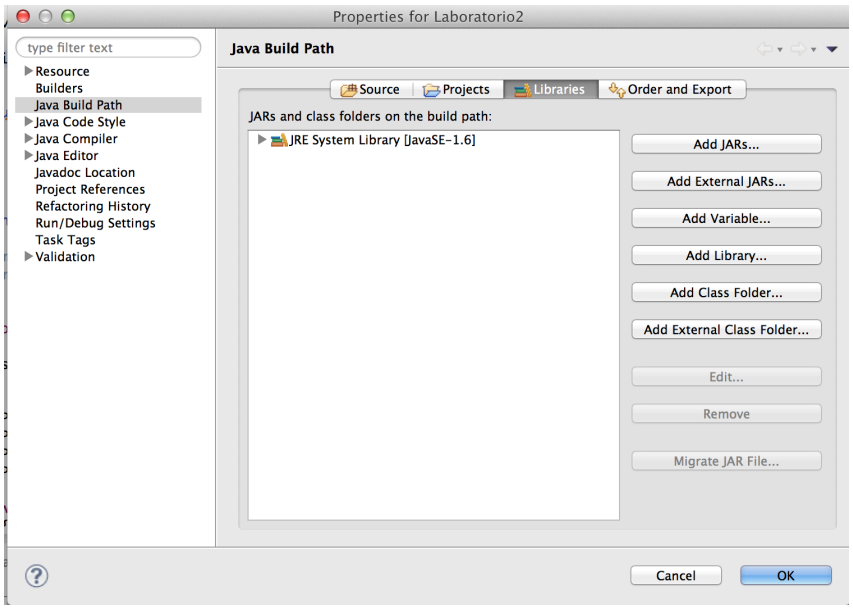
- Todas los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`
- El fichero a subir es `Suma.java`

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.invididual6` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Laboratorios y Entregas Individuales → Individual 6 → `Individual6.zip`; descomprimidlo
- Contenido de Individual6.zip
  - `Suma.java`,`TesterInd6.java`
- Descargad también el fichero `aedlib.jar`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `aed.invididual6` los fuentes que habéis descargado (`Suma.java,TesterInd6.java`)
- Añadid al proyecto `aed` la librería `aedlib.jar` que habéis descargado. Para ello:
- Project → Properties. Se abrirá una ventana como esta:



- Java Build Path → Libraries → Add external JARs → Seleccionad el fichero `aedlib.jar` que os habéis descargado
- Ejecutad `TesterInd6`. Veréis que imprimen un mensaje de error

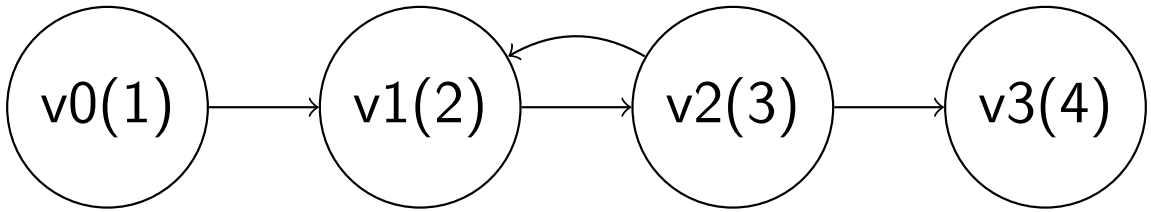
Tarea: implementar el método `sumVertices`

```
static <E> Map<Vertex<Integer>,Integer>
                    sumVertices(DirectedGraph<Integer,E> g) { ... }
```

Dado un grafo dirigido `G`, cuyos vértices contienen elementos de tipo `Integer`, el método devuelve un objeto de tipo `Map<Vertex<Integer>,Integer>` donde:

- las claves serán todos los vértices del grafo
- el valor asociado a cada clave (es decir, a cada vértice) v será la suma de los elementos – que son de tipo `Integer` – de los vértices alcanzables ("reachable") desde v

Ejemplo



- En la figura, el vértice `v0(1)` contiene el elemento 1, el vértice `v1` contiene 2, el vértice `v2` contiene 3, y `v3` contiene 4.
- El map devuelto por el método debe contener:

v0	10
v1	9
v2	9
v3	4

  - v0, v1, v2 y v3 son alcanzables desde v0
  - sólo v1, v2 y v3 son alcanzables desde v1 y v2
  - sólo v3 es alcanzable desde v3.

El Tester

- Notad que para ayudaros, el tester imprime los grafos en un formato sencillo, mostrando todos los vértices y los nodos a los que llegan las aristas que salen de cada uno.
- Por ejemplo, el grafo de la figura anterior se imprime de la siguiente forma:

```
v0(1): -->v1
v1(2): -->v2
v2(3): -->v1, -->v3
v3(4):
```

Notas

- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar.
- Debe ejecutar `TesterInd6` correctamente y 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).