Laboratorio A.E.D. Ejercicio Individual 5

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 la penalización aplicada a la puntuación obtenida sobre 10:

Hasta el Lunes 30 de octubre, 23:59 horas	0 %
Hasta el Martes 31 de octubre, 23:59 horas	20 %
Hasta el Jueves 2 de noviembre, 23:59 horas	40 %
Hasta el Viernes 3 de noviembre, 23:59 horas	60 %
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

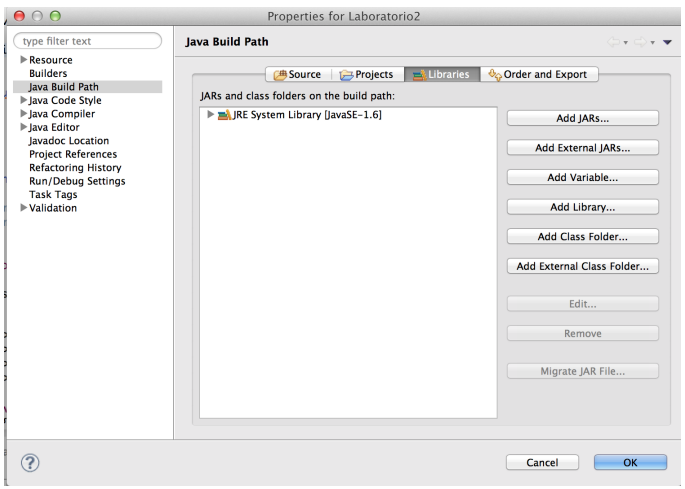
- Todos los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`.
- El fichero que hay que subir es `RecursiveUtils.java`.

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.recursion` en el proyecto `aed`, dentro de `src`
- Aula Virtual $\rightarrow$ AED $\rightarrow$ Laboratorios y Entregas Individuales $\rightarrow$ Individual 5 $\rightarrow$ `Individual5.zip`; descomprimidlo
- Contenido de `Individual5.zip`:
  - `RecursiveUtils.java`,`RunOneTest.java`,`ClassCFG.java`, `MethodCFG.java`,`bcel-5.2.jar`,`TesterInd5.java`

Configuración previa al desarrollo del ejercicio.

- Importad
  al paquete `aed.recursion` los fuentes que habéis descargado (`RecursiveUtils.java`,`RunOneTest.java`,`ClassCFG.java`, `MethodCFG.java`,`bcel-5.2.jar`,`TesterInd5.java`)
- **Añadid** al proyecto `aed` la librería `bcel-5.2.jar` que habéis descargado.
- 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...”.

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 para hoy: programar con `Recursion`

- La clase `RecursiveUtils` contiene tres métodos para completar: `power`, `allNonNull` y `countNonNull`.
- Está **prohibido** usar bucles `for`, `for-each`, `while`, `do-while`, o iteradores.
- Es **obligatorio** usar recursión en la implementacion de estos métodos.
- Está **permitido** (y **es necesario**) añadir métodos auxiliares para implementar correctamente métodos mencionados.
- Está **prohibido** añadir nuevos atributos a clases.

Notas

- **Es necesario añadir la libreria bcel-5.2.jar, que esta incluido en el fichero .zip, al Java Build Path del proyecto**. Consulta las instrucciones en la página "Configuración previa al desarrollo del ejercicio".
- Cuando se pide la implementación recursiva de un método, no tiene por qué hacerse mediante una llamada recursiva a ese método en concreto: por ejemplo el método `allNonNull` abajo no hará una llamada recursiva a si mismo, sino que llamará a un método auxilizar que será el método definido recursivamente:

```
public static <E> boolean allNonNull(PositionList<E> l) {
  return allNonNullRec(l, ...);
}

private static <E> boolean allNonNullRec(PositionList<E> l, ...) {
  if (...) { ... }
  else return allNonNullRec(l, ...);
}
```

La clase RecursiveUtils

```
public class RecursiveUtils {
  // devuelve a^n
  public static int power(int a, int n)

  // Devuelve true si todos los elementos dentro la
  // lista l son distintos de null
  public static <E> boolean allNonNull(PositionList<E> l)

  // Devuelve el numero de elementos dentro la lista l
  // que son distintos de null
  public static <E> int countNonNull(PositionList<E> l)
}
```

Tarea: implementar `power` (elevar a una potencia)

- Implementad
  `int power(int a, int n)`
  que calcula $a^n, n \geq 0$ usando recursion (sin bucles)

$$a^0 = 1$$
$$a^n = \underbrace{a * a * \ldots * a}_{n \text{ veces}} \text{ si } n > 0$$

- **No podéis** llamar a `Math.pow` u otros métodos de las librerías de Java. Usad solo aritmética básica (`+`, `-`, `*`, `/`, `%`).

- Ejemplo:

```
power(5,2)      =>  25
power(2,0)      =>   1
```

- **Sugerencia avanzada:** podéis intentar implementarlo en $O(\log n)$ usando:

$$a^n = \begin{cases} 1 & \text{si } n = 0 \\ a^{\frac{n}{2}} * a^{\frac{n}{2}} & \text{si } n \text{ par} \\ a * a^{\lfloor \frac{n}{2} \rfloor} * a^{\lfloor \frac{n}{2} \rfloor} & \text{si } n \text{ impar} \end{cases} .$$

  - No repitáis llamadas iguales.

Notas

- El Tester intenta a advertir si la implementación de un método no usa recursión:

```
WARNING: La implementacion de
aed.recursion.RecursiveUtils.power
no parece recursiva y es OBLIGATORIO -- compruebalo manualmente!
```

  Sin embargo, esta comprobación no es 100 % fiable, debe ser considerada como una advertencia.
- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar
- Debe ejecutar `TesterInd5.java` 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)