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 viernes 18 de diciembre, 15:00 horas	10
Hasta el lunes 21 de diciembre, 15:00 horas	8
Hasta el martes 22 de diciembre, 15:00 horas	6
Hasta el miércoles 23 de diciembre, 15:00 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

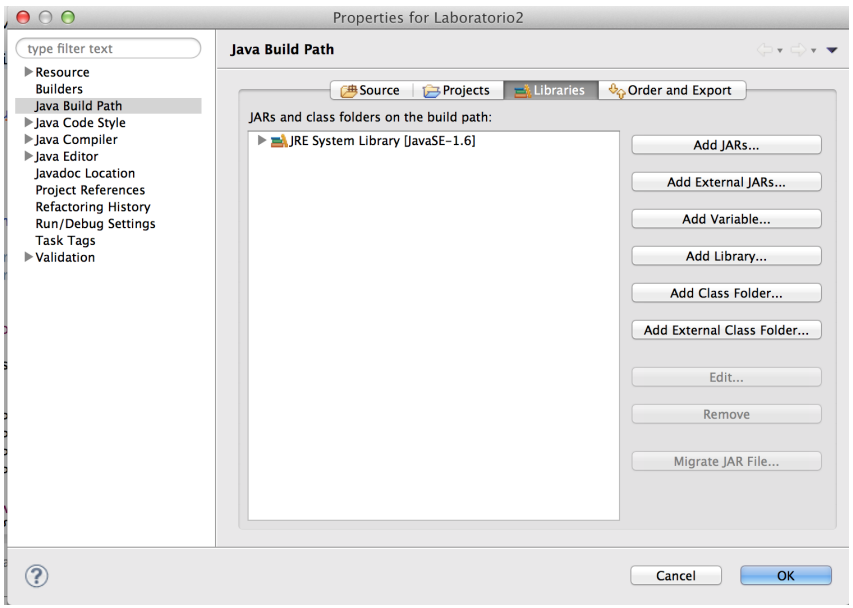
- Todas los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`.
- El fichero a subir hoy es `ComparadorExpArith.java`.

Configuración previa al desarrollo del ejercicio.

- Arrancad Eclipse. Debéis tener un acceso directo.
- Si trabajáis en portátil, podeis utilizar cualquier version relativamente reciente de Eclipse. Debería valer cualquier versión entre la versión 3.7 (Indigo) o 4.3 (Kepler). Es suficiente con que instaleis 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* `CompareExpArith` en el proyecto `aed`, dentro de `src`.
- Aula Virtual → AED → Sesiones de laboratorio → `Laboratorio10` → `Laboratorio10.zip`; descomprimidlo.
- Contenido de `Laboratorio10.zip`
  - `Tester.java`, `TipoExpresion.java`, `ElementoExpresion.java`, `ComparadorExpArith.java`
  - `net-datastructures-5-0.jar`

Configuración previa al desarrollo del ejercicio.

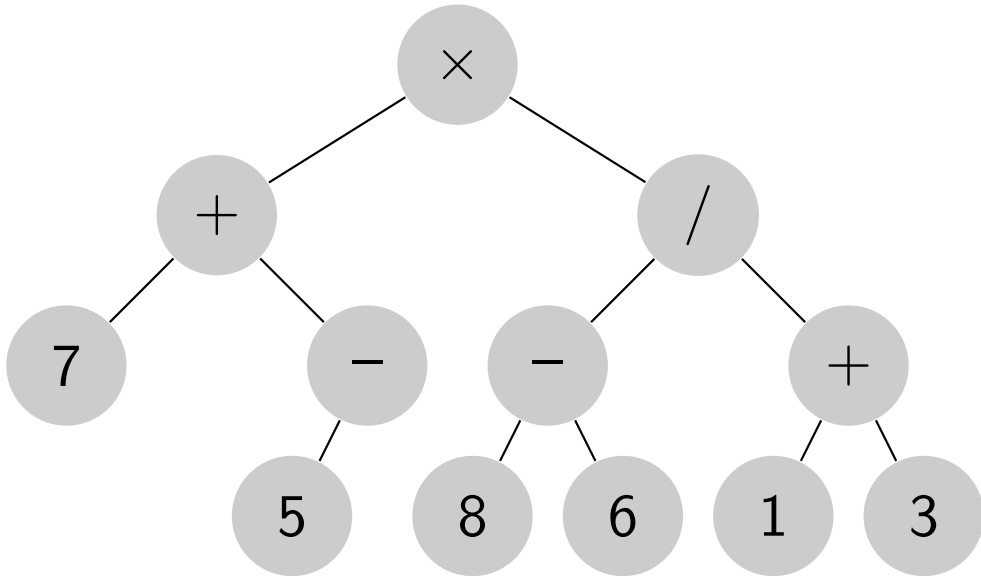
- Importad al paquete `CompareExpArith` las fuentes que habéis descargado (`Tester.java`, `TipoExpresion.java`, `ElementoExpresion.java`, `ComparadorExpArith.java`).
- Añadid al proyecto aed la librería `net-datastructures-5-0.jar` que habéis descargado. Para ello:
- Project → Properties. Se abrirá una ventana como esta:



- Java Build Path → Libraries → Add external JARs → Seleccionad el fichero `net-datastructures-5-0.jar` que os habéis descargado
- Ejecutad `Tester`. Veréis que imprime un mensaje de error.

Tareas para hoy

árboles que representan expresiones aritméticas.

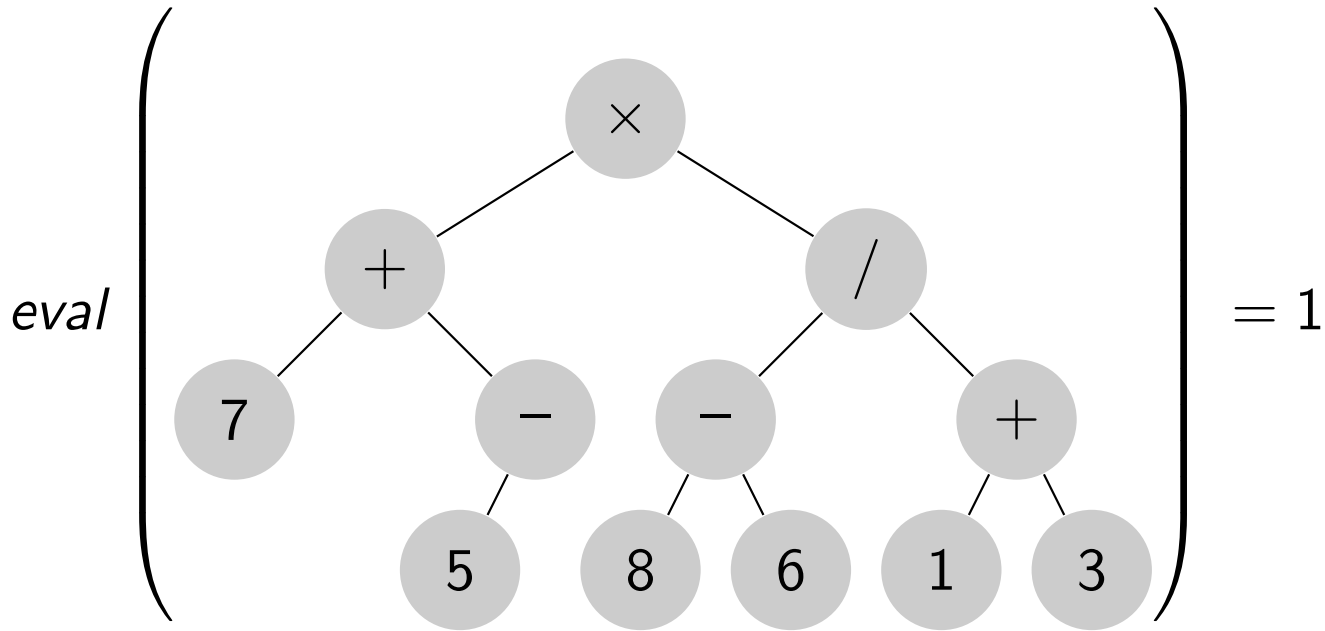


$$((7 + (-5)) \times ((8 - 6) / (1 + 3)))$$

- Números en nodos hoja.
- Operaciones en nodos internos.

Tareas para hoy

evaluación de árboles que representan expresiones aritméticas.



$$((7 + (-5)) \times ((8 - 6) / (1 + 3))) = 1$$

- Números en nodos hoja.
- Operaciones en nodos internos.

Tarea para hoy

- `ComparadorExpArith` debe implementar un comparador de árboles binarios que representen expresiones aritméticas sobre enteros.
- El comparador debe evaluar los árboles con el método auxiliar `eval` y comparar los resultados.
- Dados dos árboles `a1` y `a2`, el método `compare` debe devolver un entero i tal que:

$$
\begin{array}{ll}
i < 0 & \text{si } \texttt{eval(a1)} < \texttt{eval(a2)} \\
i = 0 & \text{si } \texttt{eval(a1)} = \texttt{eval(a2)} \\
i > 0 & \text{si } \texttt{eval(a1)} > \texttt{eval(a2)}
\end{array}
$$

- Hay que completar el código de los métodos `compare` y `eval`.

Tarea para hoy

Los nodos del árbol binario tienen como elementos objetos de clase `ElementoExpresion` que representan enteros y símbolos de operador.

- `getTipo` devuelve un valor de `TipoExpresion` que nos permite saber qué tenemos en el elemento.
- Si `a1.root().element().getTipo() == Lit`, el elemento en la raíz es un entero cuyo valor es `a1.root().element().getLiteral()` .
- Si `a1.root().element().getTipo() == Sum`, el elemento es un símbolo de operador binario de suma (cuyos operandos son los hijos).
- Similar para `Res` (resta), `Div` (división), `Mul` (multiplicación) **sobre enteros**.
- Y para el operador unario de cambio de signo `Neg` (p.e. -5): se aplica a la expresión correspondiente al subárbol de **uno de los dos hijos** (izquierdo o derecho, indistintamente).

Casos especiales a tratar

- Casos especiales a tratar en la clase `ComparadorExpArith`:
  - Se debe lanzar `IllegalArgumentException` si alguno de los dos árboles a comparar es `null` o vacío.
  - Se debe lanzar `RuntimeException` si:
    - Un entero no está en una hoja.
    - Un operador binario ($+$, $-$, $/$, $\times$) no tiene ambos hijos.
    - Una división ($/$) tiene 0 como denominador.
    - Un operador unario (p.e. -5) tiene cero o dos hijos.

Tarea para hoy

- 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).
- Corregimos los ejercicios a mano antes de dar la nota final.