Laboratorio A.E.D. Laboratorio 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

Sergio Paraíso
sergio.paraiso@upm.es

Normas

- Fechas de entrega y penalización:

Hasta el Lunes 25 de Noviembre, 23:59 horas	0 %
Hasta el Martes 26 de Noviembre, 23:59 horas	20 %
Hasta el Miércoles 27 de Noviembre, 23:59 horas	40 %
Hasta el Jueves 28 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

  `http://costa.ls.fi.upm.es/entrega`

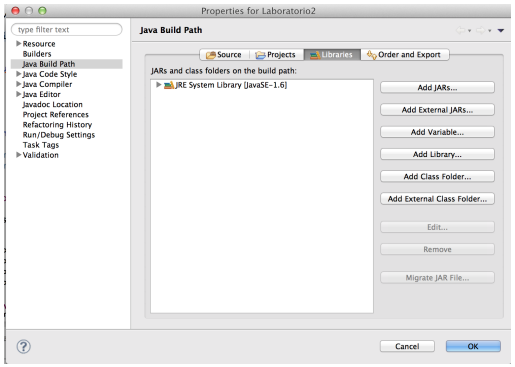
- El fichero que hay que subir es `TreeSearch.java`.

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.treesearch` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Laboratorios y Entregas Individuales → Laboratorio 5 → `Laboratorio5.zip`; descomprimidlo
- Contenido de `Laboratorio5.zip`:
  - `TreeSearch.java`, `TesterLab5.java`

Configuración previa

- Importad al paquete `aed.treesearch` los fuentes que habéis descargado (`TreeSearch.java, TesterLab5.java`)
- Añadid al proyecto `aed` la librería `aedlib.jar` que tenéis en Moodle (en Laboratorios y Entregas Individuales).

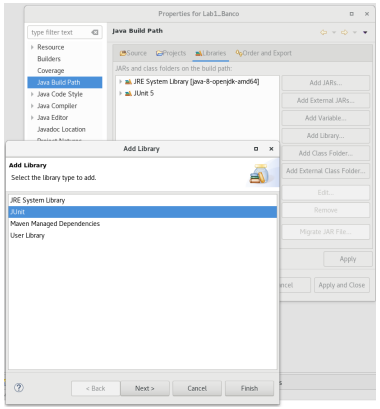


Para ello:

- Project → Properties → Java Build Path. Se abrirá una ventana como la de la izquierda
- Usad la opción “Add External JARs...”.
- Si vuestra instalacion 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 `TesterLab5` tenéis las pruebas, para ejecutarlas, abrid el fichero `TesterLab5`, 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 buton "Apply and Close"

Árboles y Conjuntos

- La practica se enfoca prinicipalmente en los árboles, pero también se usará la nueva estructura de datos "Set", que modela un conjunto:
  - No tiene elementos repetidos
  - No tiene orden entre los elementos
- El API del interfaz `Set` de *aedlib* está disponible en:

  `http://costa.ls.fi.upm.es/entrega/aed/docs/aedlib/es/upm/aedlib/set/Set.html`

Tarea 1: realizar búsquedas en arboles

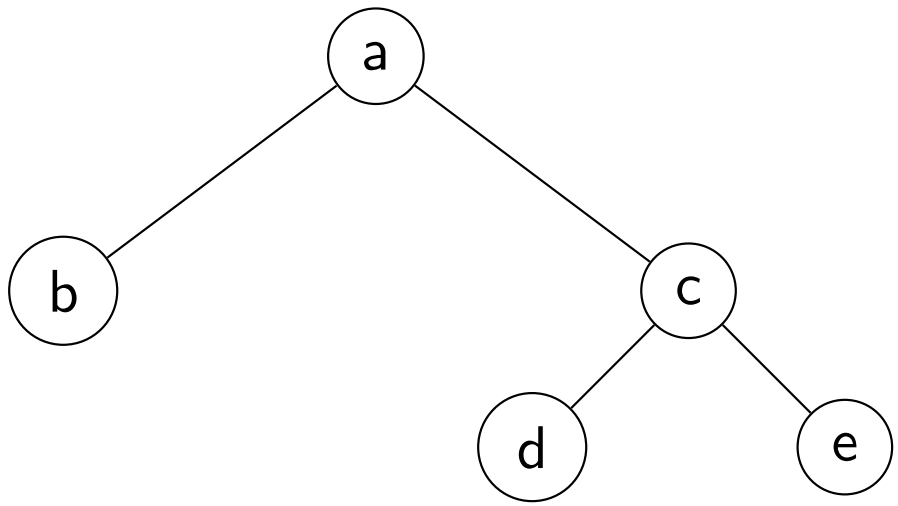
Hay que implementar el método

```
static Set<Position<String>> search
                    (Tree<String> t,
                     PositionList<String> searchExpr)
```

- Dado un árbol, y una expresión de búsqueda `searchExpr`, el método debe devolver el conjunto (de tipo `Set`) de *posiciones* cuyos *caminos*, empezando en la raíz, sean válidos para la expresión de búsqueda

Caminos en Arboles

- Un camino hasta una posición p es la secuencia de nodos (padre-hijo) que empieza en la raíz y termina en p.
- Ejemplo:



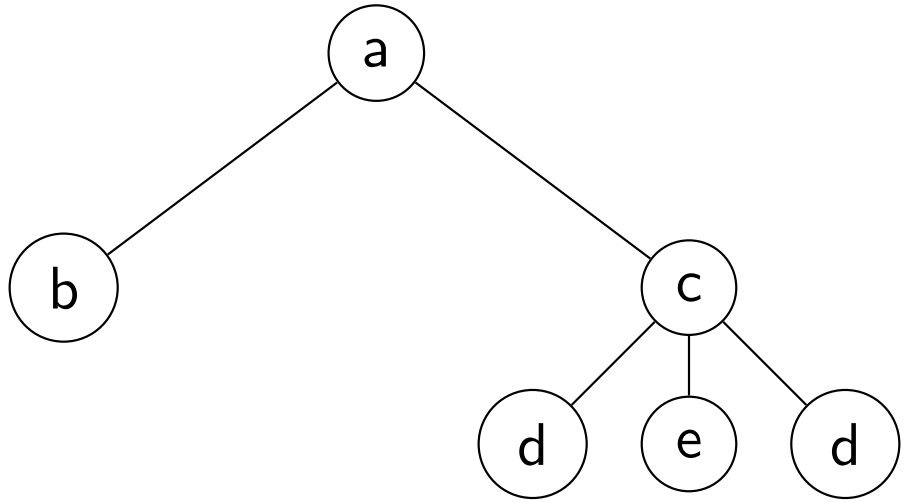
  - El camino a la raíz es [a].
  - El camino que va al nodo d es [a,c,d].
  - El camino que va al nodo c es [a,c].

Comprobando caminos con expresiones de búsqueda

Decimos que una expresión de búsqueda $[e_1, \ldots, e_n]$ acepta una posición n_n del árbol (de tipo `Position<E>`) si hay un camino $[n_1, \ldots, n_n]$, y, para cada pareja e_i y n_i:

- $e_i = n_i.element()$ o
- $e_i =$ "$*$" // el símbolo "$*$" acepta cualquier elemento

Ejemplo



- Dado una expresión de búsqueda $[a]$, `search` devuelve un conjunto con la posición del nodo a.
- Dado $[*]$, `search` también devuelve un conjunto con la posición del nodo a.
- Dado $[a, *]$, `search` devuelve un conjunto con los nodos b y c (los dos hijos del nodo a).
- Dado $[*, c, d]$, `search` devuelve un conjunto con los dos hijos del nodo c con elemento d.

Consejos

- Implementad un método privado `search(...)`, de forma *recursiva*, que reciba dos cursores, uno para la posición actual de la expresión de búsqueda, y otro cursor para la posición actual del árbol

Tarea 2: construir un árbol desde un conjunto de caminos

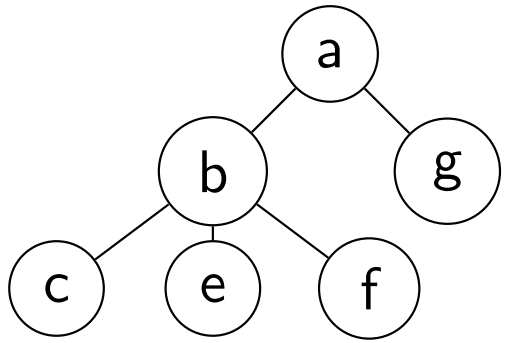
- Hay que completar el método

```
static Tree<String> constructDeterministicTree
                (Set<PositionList<String>> paths)
```

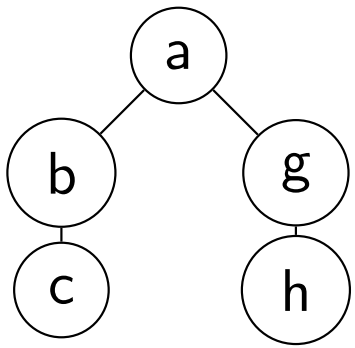
- El método construye un **nuevo** árbol **determinista** que contiene todos los caminos que están en el parámetro `paths`
- Un árbol *determinista* es un árbol donde ningún nodo tiene dos o mas hijos con el mismo elemento
- Se puede asumir que todos los caminos que no son vacíos empiezan con el mismo elemento
- Para crear el árbol podéis utilizar la clase `LinkedGeneralTree` que dispone de un constructor sin parámetros y tendréis que usar los métodos del interfaz `GeneralTree<E>`

Ejemplo

- Dado un conjunto de caminos $\{[a, b, c], [a, b, e], [a, b, f], [a, g]\}$ el árbol resultante es:

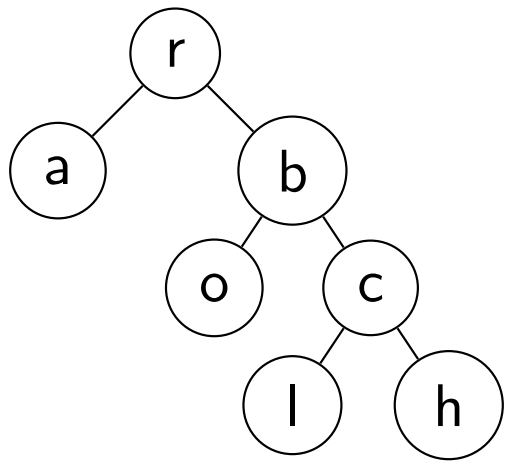


- Dado un conjunto de caminos $\{[a, b, c], [a, g, h], [a]\}$ el arból resultante es:



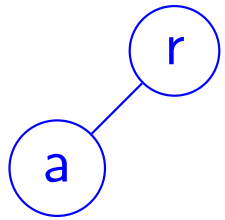
Ejemplo paso a paso

- Dado un conjunto $\{[r, a], [r, b, o], [r, b, c, h], [r, b, c, l]\}$ el método debe devolver el árbol:



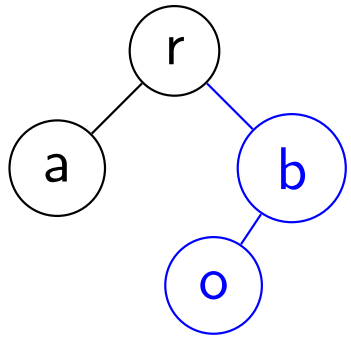
- Lo veremos paso a paso

Ejemplo paso a paso



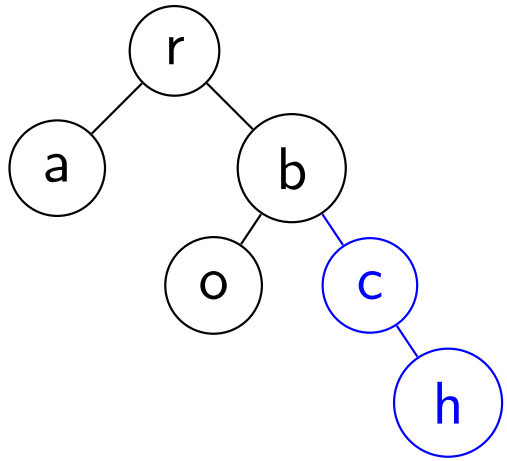
Con $[r, a]$ creamos la raíz con nombre r y un hijo con nombre a

Ejemplo paso a paso



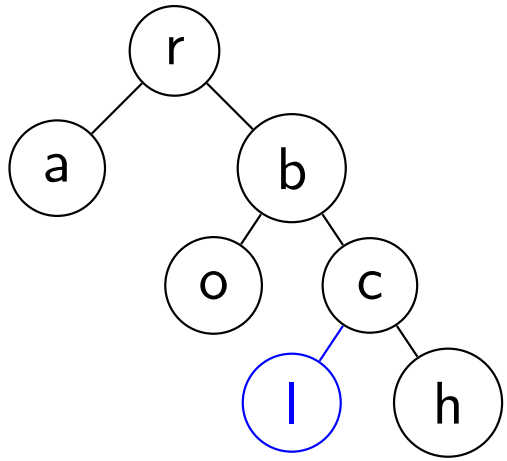
Con $[r, b, o]$ pasamos la raíz y creamos el nodo b,
y creamos un hijo o de b

Ejemplo paso a paso



Con $[r, b, c, h]$ pasamos la raíz y el nodo b,
y creamos otro hijo c de b, y su hijo h

Ejemplo paso a paso



Con $[r, b, c, l]$ pasamos la raíz y los nodos b y c,
y creamos otro hijo l de c

Notas

- El proyecto debe compilar sin errores, cumplir la especificación de los métodos a completar y ejecutar `TesterLab5` 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.
- Está **prohibido** añadir nuevos atributos a clases.