Laboratorio A.E.D. Ejercicio Individual Extra

Guillermo Román
guillermo.roman@upm.es
Lars-Åke Fredlund
lfredlund@fi.upm.es
Manuel Carro
manuel.carro@upm.es
Marina Álvarez
marina.alvarez@upm.es
Julio García
juliomanuel.garcia@upm.es
Tonghong Li
tonghong@fi.upm.es
José Ramón Sánchez
joseramon.sanchezp@fi.upm.es

Normas

- Fechas de entrega y la maxima puntuación obtenible:

  Hasta el Viernes 12 de enero, 23:59 horas 10
  Después la puntuación máxima será 0
- La entrega del ejercicio podrá sumar hasta 0.5 puntos extra sobre la nota de prácticas de la asignatura
- Se comprobará plagio y se actuará sobre los detectados

Entrega

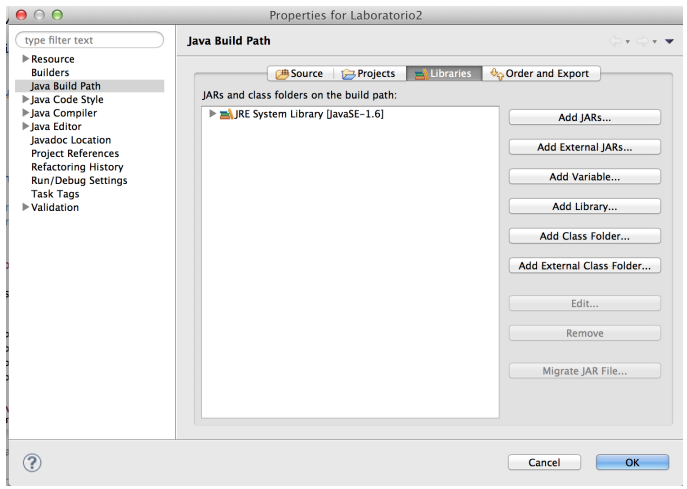
- Todos los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`.
- Los ficheros que hay que entregar son `TeseoExplorador.java`.

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.cnossos` en el proyecto `aed`, dentro de `src`
- Aula Virtual → AED → Laboratorios y Entregas Individuales → Individual Extra → `IndividualExtra.zip`; descomprimidlo
- Contenido de `IndividualExtra.zip`:
  - `TeseoExplorador.java`, `Punto.java`, `PuntoCardinal.java`, `RunOneTest.java`, `Laberinto.java`, `TesterIndExtra.java`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `aed.cnossos` los fuentes que habéis descargado (`TeseoExplorador.java`, `Punto.java`, `PuntoCardinal.java`, `RunOneTest.java`, `Laberinto.java`, `TesterIndExtra.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 esta:



- Usad la opción “Add External JARs...”.
- Intentad ejecutar `TesterIndExtra.java`

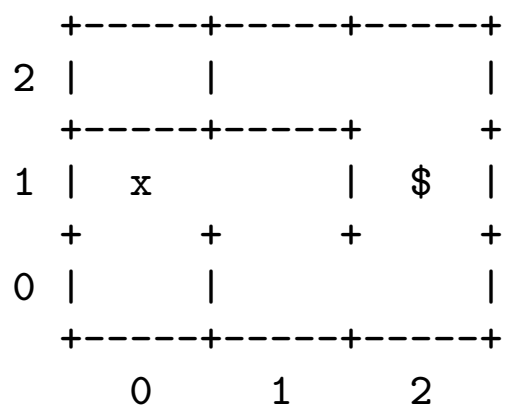
Documentación de la librería aedlib.jar

- 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: Explorar un laberinto usando movimientos "detallados"

- Las anteriores tareas de explorar un laberinto han asumido que, durante la exploración, la exploradora se puede trasladar immediatemente entre dos lugares cualquiera
- Ahora la tarea es mas "realista" – al explorar, los movimientos desde un lugar siempre son hacia un lugar vecino immediato situado al norte, sur, este u oeste

  Por ejemplo, desde el lugar (0,1) abajo solo se puede mover directamente al lugar (0,0) [sur] o (1,1) [este]



Tarea: Explorar un laberinto usando movimientos “detallados

- La tarea es implementar el método:

```
public static Pair<Object,PositionList<PuntoCardinal>>
    explora(Laberinto cnossos) { ... }
```

  dentro la clase `TeseoExplorador`.
- El método devuelve el par con un tesoro encontrado y un camino que lleva a él desde el lugar inicial
- El camino es una secuencia de movimientos de la clase `PuntoCardinal`:

```
public enum PuntoCardinal { NORTE, SUR, ESTE, OESTE }
```

- La clase `Laberinto` contiene los métodos para poder navegar en el laberinto y localizar el tesoro

La clase Laberinto

```
public class Laberinto {
  public boolean tieneTesoro()        // Devuelve true si el lugar actual
                                       // contiene un tesoro

  public Object getTesoro()           // Devuelve el tesoro (un objeto) o null

  public void marcaSueloConTiza()     // Permite marcar el ''suelo'' en el lugar
                                       // actual con ''tiza''

  public boolean sueloMarcadoConTiza()   // Esta marcado el suelo con tiza?

  public Iterable<PuntoCardinal> caminos()  // Devuelve los caminos hacia
                                             // lugares vecinos

  public boolean ir(PuntoCardinal pc)     // Intenta mover al lugar vecino
                                           // en la direccion del parametro pc

  public String printLocation()           // Imprime el lugar actual

  public void printLaberinto()            // Imprime todo el laberinto
}
```

Consejos de la implementacion de la tarea

- Para poder volver de un lugar de un laberinto es aconsejable recordar el camino travesado, es decir, la secuencia de direcciones (sur, oeste, ...) desde el lugar inicial al lugar actual
- Un ejemplo clásico de este tipo de búsqueda en un laberinto es el uso del "Hilo de Ariadna":
  `https://es.wikipedia.org/wiki/Laberinto_de_Creta`
  `http://www.labolab.net/mitologia/el-laberinto-de-creta/`
- Tambien es aconsejable recordar, para cada lugar, las direcciones que quedan por probar. Por ejemplo, volviendo a un lugar X, si ya hemos probado ir hacia el sur no hace falta repetir este movimiento sino probar las otras direcciones (norte, este, oeste) que quedan por explorar

Reglas de la implementacion de la tarea dentro la clase TeseoExplorador

- Se puede usar tanto recursión como bucles para resolver el problema
- Se pueden usar PositionList, IndexedList, LIFO, FIFO, etc... para recordar el camino y las direcciones que quedan por probar
- Se pueden añadir nuevos metodos auxiliares, y nuevos atributos
- **NO** está permitido hacer *casting* ni usar **instanceof**
- Notad que la librería `aedlib` ahora contiene la clase `Pair` – está disponible como `es.upm.aedlib.Pair`. Es decir, no hace falta tener ningún fichero `Pair.java`

Consejos:

- Imprime un lugar usando, por ejemplo:

```
// Asumimos que cnossos es un objeto de la clase Laberinto
System.out.println("Estoy␣en␣el␣lugar␣"+cnossos.printLocation());
        ==> Estoy en el lugar <0,1>
```

- Se puede imprimir el laberinto entero llamando a:

```
cnossos.printLaberinto();
```

- Que imprime una representación textual del laberinto

```
  +-----+-----+-----+
2 |     |           |
  +-----+-----+     +
1 | .       x |  $  |
  +     +     +     +
0 | .   |           |
  +-----+-----+-----+
      0     1     2
```

El símbolo "$" marca el tesoro en el lugar (2,1), "x" es el lugar actual en el laberinto (1,1), y "." indica los lugares marcados con tiza

Cómo ejecutar una sola prueba

- Se puede ejecutar una sola prueba fácilmente cambiando la clase `RunOneTest.java` y ejecutándola
- Por defecto la clase contiene

```
package aed.cnossos;

public class RunOneTest {
  public static void main(String args[]) {
    TesterIndExtra.test_5();
  }
}
```

  para poder correr el "test_5".
- Otra alternativa para ejecutar un test es especificar su número como parámetro para el programa `TesterIndExtra` dentro de Eclipse: "Run As" $\Rightarrow$ "Run Configurations" $\Rightarrow$ "TesterIndExtra" $\Rightarrow$ "Arguments".

Notas

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