Laboratorio A.E.D. — ejercicio Laboratorio 4

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

Normas

- Fechas de entrega y penalización aplicada a la puntuación obtenida:

Hasta el Lunes 6 de noviembre, 23:59 horas	0 %
Hasta el Martes 7 de noviembre, 23:59 horas	20 %
Hasta el Miércoles 8 de noviembre, 23:59 horas	40 %
Hasta el Jueves 9 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

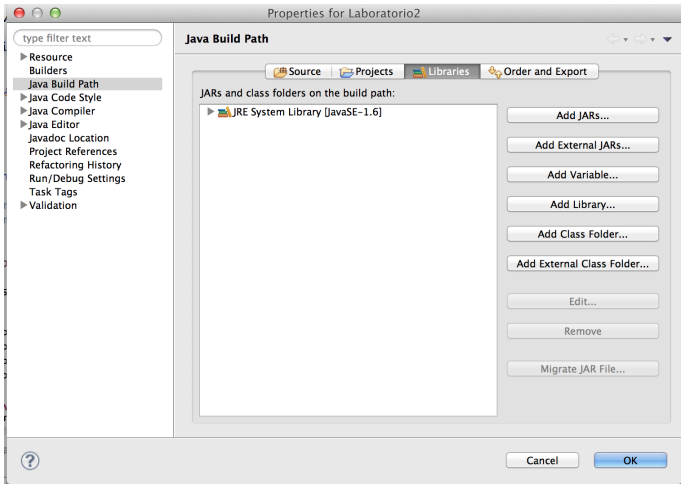
- Todos los ejercicios de laboratorio se deben entregar a través de la web `http://lml.ls.fi.upm.es/~entrega`.
- Los ficheros que hay que entregar son: `Explorador.java`.

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.zork` en el proyecto `aed`, dentro de `src`
- Aula Virtual $\rightarrow$ AED $\rightarrow$ Laboratorios y Entregas Individuales $\rightarrow$ Laboratorio 4 $\rightarrow$ `Laboratorio4.zip`; descomprimidlo
- Contenido de `Laboratorio4.zip`:
  - `Explorador.java`, `Lugar.java`, `Punto.java`, `PuntoCardinal.java`, `RunOneTest.java`, `TesterLab4.java`

Configuración previa al desarrollo del ejercicio.

- Importad al paquete `aed.zork` las fuentes que habéis descargado (`Explorador.java`, `Lugar.java`, `Punto.java`, `PuntoCardinal.java`, `RunOneTest.java`, `TesterLab4.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 `TesterLab4.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 recursión

- Objetivo: desarrollar un método `explora` en la clase `Explorador` que sea capaz de explorar un laberinto de forma sistemática y encontrar un "tesoro" en él
  - El laberinto esta compuesto por "lugares", implementados como objetos de la clase `Lugar`
  - En un lugar:
    - puede haber un tesoro (un objeto)
    - el "suelo" puede estar marcado (para detectar que un lugar ya ha sido visitado)
    - puede haber caminos hacia otros lugares.
- Es similar a la práctica individual 4, pero:
  - Es obligatorio usar *recursion* para efectuar la busqueda
  - El metodo `explora` debe devolver un par con el tesoro encontrado y UN *camino* (una secuencia de lugares) que lleva al tesoro:

```
Pair<Object,PositionList<Lugar>> explora(Lugar inicialLugar) { ... }
```

  - Si no hay ningún tesoro alcanzable, el método devolverá **null**

La clase Lugar

```
public class Lugar {
  public boolean tieneTesoro()       // Devuelve true si el lugar tiene un tesoro

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

  public Iterable<Lugar> caminos()   // Devuelve los lugares conectados
                                     // con el lugar (del objeto)

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

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

  public String toString()               // Para imprimir el lugar

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

Reglas de implementación de la clase `Explorador`

- Modificad solo `Explorador.java`
- El camino devuelto debe empezar con el lugar inicial y terminar con el lugar que contiene el tesoro
- Es **obligatorio** usar recursión
- Está permitido usar un iterador **únicamente** para iterar sobre los lugares devueltos por el metodo `caminos`
  - En el resto del código **NO** esta permitido usar bucles `for`, `for-each`, `while`, `do-while`
- **NO** esta permitido hacer *casting* ni usar **instanceof**
- Está permitido añadir métodos auxiliares
- **NO** está permitido añadir nuevos atributos
- **NO** se debe modificar el contenido de las estructuras de datos recibidas como parámetros en los métodos

Consejos

- Se puede imprimir la información sobre un lugar:

```
// Asumimos que lugar es un objeto de la clase Lugar
System.out.println("Estoy␣en␣el␣lugar␣"+lugar);
        ==> Estoy en el lugar <<{x=0,y=1}, esta marcado con tiza>>
```

- Se puede imprimir el laberinto entero llamando:

```
// Asumimos que lugar es un variable de tipo Lugar
lugar.printLaberinto();
```

- Imprime una representación textual del laberinto:

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

El símbolo "$" marca el tesoro (en la posición (2,1)). "x" es el lugar actual en el laberinto (posición (1,1)) y "." señala los lugares marcados.

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.laberinto;

public class RunOneTest {
  public static void main(String args[]) {
    TesterLab4.test_5();
  }
}
```

  para poder correr el "test_5".
- Otra alternativa para ejecutar un test nada más es especificar su número como parámetro para el programa `TesterLab4` dentro Eclipse: "Run As" $\Rightarrow$ "Run Configurations" $\Rightarrow$ "TesterLab4" $\Rightarrow$ "Arguments".

Notas

- El proyecto debe compilar sin errores y debe cumplirse la especificación de los métodos a completar
- Debe ejecutar `TesterLab4.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)
- Todos los ejercicios se comprueban manualmente antes de dar la nota final