UNIVERSIDAD ICESI

Departamento de TIC
Algoritmos y Programación II
2020-2 - Tarea Integradora 2

Objetivos

Unidad 3: Estructuras Lineales Enlazadas

OE3.1 Utilizar estructuras enlazadas de objetos para modelar grupos de atributos no primitivos de tamaño flexible.
OE3.2 Escribir los algoritmos necesarios para manipular estructuras lineales que almacenan sus elementos enlazándolos entre ellos.

Unidad 4: Estructuras y Algoritmos Recursivos

OE4.1 Emplear el concepto de recursividad como una alternativa a la estructura de control iterativa.
OE4.2 Aplicar la computación recursiva en la solución de problemas de naturaleza inherentemente autocontenida.
OE4.3 Utilizar árboles binarios de búsqueda para representar grupos de objetos que mantienen entre ellos una relación de orden.
OE4.4 Escribir algoritmos recursivos para manipular estructuras de información recursivas y explicar las ventajas que, en este caso, estos algoritmos tienen sobre los algoritmos iterativos.

Enunciado

El instituto de espejos, láseres y laberintos, lo ha contratado para el desarrollo de un programa que permita evaluar la capacidad de razonamiento del personal que es contratado. El programa es simple y puede ser visto como un juego. Se presenta al usuario una cuadrícula o tabla de n filas por m columnas, dentro de la cual hay k espejos que no son visibles.

Cuadrícula de 2 x 4

	A	B	C	D
1	1A	1B	1C	1D
2	2A	2B	2C	2D
3	3A	3B	3C	3D

Cada una de las casillas de la cuadrícula puede identificarse a través una nomenclatura en la cual la fila está dada por un número entero (la primera fila tiene asignado el número 1, la segunda fila el número 2, y así), y la columna está dada por una letra en mayúscula (la primera columna es A, la segunda B, y así). Por lo anterior, habrán máximo 26 columnas (por las 26 letras del alfabeto inglés).

A la izquierda se presenta una cuadrícula con los nombres de las filas, columnas y cada una de las celdas.

Al interior de la cuadrícula, aunque no son visibles para el usuario, los espejos pueden estar dispuestos únicamente de dos maneras, inclinado a la derecha así / o inclinado a la izquierda así \

Si pudiéramos ver los espejos, podrían verse como se aprecia a la derecha.

Cuadrícula de 2x6 con 5 espejos

	/		\		
	/		\	/	

El usuario del programa tiene la posibilidad de disparar un rayo láser de forma horizontal o vertical desde cualquier celda del borde de la cuadrícula. La dirección inicial del rayo disparado está dada por el lugar desde el cual se hace el disparo. Si se hace desde las celdas del borde izquierdo o derecho, el disparo es horizontal. Si se hace desde el borde superior o inferior, el disparo es vertical. Si se dispara desde una celda que está en una esquina de la cuadrícula, debe indicarse no sólo la celda sino la dirección en la cual se desea disparar horizontal (H) o vertical (V). Cuando se dispara, el programa indica por cuál celda de la cuadrícula sale el rayo.

Si por ejemplo, en la cuadrícula anterior disparamos un rayo desde la celda 1C, este saldrá por la celda 2C, ya que será un rayo láser vertical con dirección hacia abajo. Si el disparo sale de 1B, éste saldrá por la celda 1A, pues el espejo inclinado a la derecha, lo desviará a la izquierda. Si sale de 2AH, saldrá por 1E.

El programa debe iniciar con un sencillo menú con 3 opciones. La primera opción es para jugar, la segunda opción es para ver el tablero de posiciones y la tercera opción es salir del programa. Cuando se elige la primera opción, el programa esperará que sean digitados, en la misma línea, 4 valores separados por espacio, el primero es una cadena

indicando un nickname del usuario, seguido por 3 números enteros positivos indicando n, m y k respectivamente.

Cuando el usuario elige jugar, se crea un juego con una cuadrícula de tamaño m x n, con k espejos ubicados aleatoriamente en cualquiera de las casillas del tablero. También aleatoriamente se decide, en el momento de la creación del tablero, cuál es la disposición de cada espejo (si inclinado a la derecha / o a la izquierda \).

Cuando el usuario ingresa los parámetros del juego, el programa le mostrará una cuadrícula formada por guiones corchetes. Por ejemplo, si la cuadrícula es de 3 x 5 se verá como se presenta a la derecha.

```
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ]
```

Una vez se despliegue esta visualización de la cuadrícula, el programa queda esperando el comando para hacer un disparo de rayo láser. El usuario solo debe indicar la celda desde la cual se hará (y la dirección H o V, si es una esquina). Una vez se indica el disparo, el programa despliega la misma cuadrícula pero indicando con una S mayúscula (de Start) la celda desde la cual partió el rayo, y una E (de End) por donde el disparo sale de la cuadrícula.

```
[ ][ ][ ][ ][ ]
[S][ ][ ][ ][ ]
[ ][ ][ ][E][ ]
```

Estas letras solo se muestran cuando se acaba de hacer un disparo, y no se mostrarán en las visualizaciones siguientes, a menos que se haga otro disparo, en cuyo caso se mostrarán la S y E correspondiente de acuerdo con el inicio y salida del disparo del láser.

Cuando el usuario considera que conoce la ubicación de un espejo, puede indicarla digitando L (de Locate) seguido (sin espacios) de la celda donde considera que hay un espejo, seguido (sin espacio) de L o R (Left o Right), dependiendo de la inclinación que se indique. Si la ubicación indicada es errónea, se mostrará la cuadrícula, solo en esa ocasión, con una X en esa celda. Si la ubicación indicada es acertada, se mostrará en esa ubicación, y en adelante, el espejo correspondiente usando el símbolo / o \ según corresponda.

Siempre que se visualiza la cuadrícula, aparece previamente (en la línea superior) una frase indicando el nickname del usuario seguido de cuántos espejos faltan por ubicar. Por ejemplo: seyerman: 4 mirrors remaining. Para volver al menú sin terminar el juego se puede digitar *menu*. Al regresar al menú principal, ya sea por digitar *menu* o por ganar el juego, se calcula un puntaje para el usuario que debe ser almacenado en un árbol binario de búsqueda ordenado por puntaje. La opción 2 mostrará un listado de los nicknames de los usuarios con sus respectivos puntajes, resultado de recorrer el árbol binario de búsqueda en inorden.

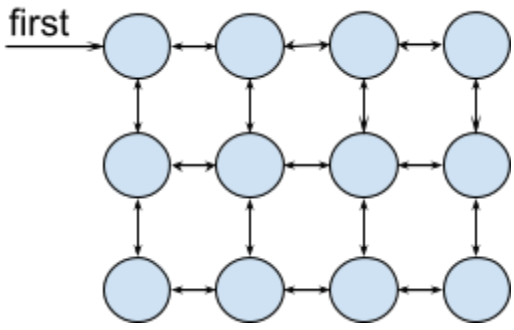
Condiciones:

i. La cuadrícula debe ser modelada e implementada utilizando listas enlazadas. Puede ser una lista enlazada de listas enlazadas.

ii. No es posible utilizar ningún arreglo (de ninguna dimensión) ni arraylist (ni ninguna colección de Java) en este programa. La **única excepción** es el arreglo que devuelve el split que se le hace al String que contiene el nick, m, n y k.

iii. No es posible utilizar ciclos en este programa. Todos las repeticiones deben hacerse utilizando **recursión**.

Se sugiere que su lista enlazada de listas enlazadas tenga el siguiente enlazamiento:



UNIVERSIDAD ICESI

Departamento de TIC
Algoritmos y Programación II
2020-2 - Tarea Integradora 2

Usted debe entregar los siguientes artefactos y condiciones:

1. **[10%]** Especificación de Requerimientos Funcionales.
2. **[25%]** Diseño completo del diagrama de clases, incluyendo el paquete del modelo y la interfaz con el usuario.
3. **[50%]** Implementación completa y correcta del modelo, y la ui.
4. **[15%]** Usted debe entregar el enlace del repositorio en GitHub o GitLab con los elementos anteriores. El nombre del repositorio debe estar en inglés, en minúsculas y si tiene varias palabras, éstas van separadas por un guión. Su repositorio debe corresponder con un proyecto de eclipse. Debe tener al menos 10 commits con diferencia de 1 hora entre cada uno de ellos. En el repositorio o proyecto de eclipse debe haber un directorio llamado **docs/** en el cual deberán ir cada uno de los documentos del diseño.

   El **readme.md** del repositorio debe explicar brevemente (en inglés) de qué se trata el proyecto. Es importante que ni en el nombre, ni la explicación del readme hagan referencia a una "tarea" de un curso ni nada parecido, sino que explique de forma independiente al curso, de que se trata el problema y la solución del mismo. Deben enlazar los archivos que documentan el proyecto (en formato pdf) y deben especificar las condiciones técnicas del mismo (lenguaje, sistema operativo, ambiente de desarrollo e instalación). **El último commit en la rama master debe estar marcado con un tag llamado Milestone1**. La fecha del commit marcado con este tag debe ser previo a la fecha y hora límite de la entrega. Usted podrá seguir haciendo commits posteriores a la fecha de entrega, pero la revisión se hará sobre la versión del proyecto marcada con esta etiqueta.

   Recuperación:
   Después de la sustentación de la primera entrega, su equipo tendrá un tiempo (unos días, estipulados en su momento por el profesor), para hacer ajustes y completar los elementos del proyecto que desee si desea que sea evaluado nuevamente. Esa segunda entrega deberá ser etiquetada como **Milestone2**.
   En esta versión recuperatoria del proyecto, al **readme.md** del repositorio debe agregársele una sección que indique concretamente qué elementos fueron agregados o modificados en esta versión (para que quien califique conozca qué es lo nuevo que debe revisar). El título de la sección puede ser changelog. También está la opción de hacer esta descripción de cambios de la nueva versión en un archivo aparte, **enlazado en el readme**. Así lo haga en un archivo aparte o dentro del mismo readme, por favor siga las recomendaciones en https://keepachangelog.com.

Los requerimientos funcionales y el diagrama de clases deben entregarse en un mismo archivo en formato **pdf**, bien organizado por secciones y títulos, con hoja de portada.

Importante: su repositorio debe ser privado hasta la hora y fecha de entrega, después de la cual usted debe hacerlo público para que pueda ser revisado.

Nota: La rúbrica con la que se evaluará esta tarea se encuentra en el listado de notas de seguimientos y tareas integradoras en la pestaña TI2. Se recomienda revisar la rúbrica con la que será evaluada su entrega.

Fecha Máxima de Entrega: 4 de Noviembre de 2020. 23:55. La tarea se califica sobre 5.0.

Bonificación por Pronta Entrega: 28 de Octubre de 2020. 23:55. La tarea se califica sobre **6.0**.

VIDEO EXPLICATIVO DE UNA MATRIZ DOBLEMENTE ENLAZADA BASE - PARTE 1

VIDEO EXPLICATIVO DE UNA MATRIZ DOBLEMENTE ENLAZADA BASE - PARTE 2

El código explicado en este video puede ser utilizado por ustedes en el desarrollo de su proyecto. El código funciona, de acuerdo como se muestra en el video, pero entrega sin ninguna garantía. Usted debe dar los créditos al autor en caso de utilizarlo en su propio desarrollo.