Memoria práctica 3

Alejandro Pascual Pozo y Víctor Yrazusta Ibarra

Apartado 1:

Hemos desarrollado este apartado sin incidencias, implementando el camino tal y como viene especificado en el enunciado.

Apartado 2:

Para este apartado hemos decidido implementar la función existeCamino(), que nos dice directamente si un camino dado está presente en la lista de caminos que parten de una determinada posada.

Apartado 3:

Este apartado también lo hemos desarrollado sin incidencias, implementándolo tal y como dice el enunciado.

Apartado 4:

Para este apartado hemos desarrollado cuatro pruebas. La primera testea la función recorre() con una única posada en un caso en el que debería poder recorrerse. La segunda es similar pero esta vez no debería poder recorrerse. La tercera testea la función recorre() con varias posadas en un caso en el que deberían poder recorrerse todas. La cuarta es similar, pero esto vez no deberían poder recorrerse todas.

Para no repetir el mismo código tantas veces tenemos la función auxiliar intentaRecorrer(), que facilita las pruebas de recorre().

Apartado 5:

En este apartado hemos decidido devolver cualquier error en el formato de los archivos de prueba como una IOException. Estos errores pueden detectarse por un número de palabras imposible en una línea, un problema al convertir a entero o decimal, un error al abrir los archivos, etc.

Apartado 6:

Para la implementación de las luces de las posadas hemos usado una enumeración con valores asociados por defecto que nos permite comparar valores de luz cómodamente con una función propia. La clase Mago es abstracta ya que todo mago ha de ser de alguno de los tipos de magos reconocidos.

Apartado 7:

Al modificar la simulación en este apartado nos hemos asegurado de que los ficheros diseñados para su versión original sigan funcionando en este sistema. Además de los cambios necesarias para poder leer trampas, posadas con niveles de luz y magos de ficheros, también hemos modificado el funcionamiento de la función para que acepte tres tipos de líneas. Una línea en la que se defina un nuevo aventurero, concediéndole además el turno, una línea en la que se cambie el turno al aventurero i-ésimo (en orden de definición) o una línea en la que se mueva al aventurero con el turno a la posada establecida.

Para implementar la lista de exploradores en las posadas hemos creado la función privada setPosadaActual() en Explorador, que se encarga de actualizar las posadas además del explorador. Para que un mago pueda comprobar si hay magos de otros tipos en una posada, hemos creado las funciones hayHada() y hayHechicero().

Diagrama de clases: