**Build instructions**

1. Ensure maven has been installed
2. Ensure java jdk 8 has been installed
3. Run ‘mvn clean package’ in the project root directory (where pom.xml is located)
4. Packaged jar file should be observed in the target subdirectory within thr project directory.

**Running instructions**

1. Run executable jar: **JuniferMaze-0.0.1-SNAPSHOT-jar-with-dependencies.jar**

from the target folder in the command line as follows:

**JuniferMaze-0.0.1-SNAPSHOT-jar-with-dependencies.jar -f <<abs. Path to test grid file>> *-d <<Delimiter if required>>***

**Design notes/plan**

When considering min steps: we don’t care about Thread execution order -> there are no race conditions whichever min will update at the appropriate time and interrupt other less efficient paths.

Apache CLI -> Params: filename, pattern delimeter

Pattern “ “

DirEnum - N,S,E,W

Treeable -> bean, which can be used by tree-graph search structure

Map<RoutePoint,Thread> => Node processing thread Map (Lock on Write, read)

Map<RoutePoint,State>

Final Int width

Final Int height

ImmutableGrid(ImmutableList<GridPoint>) Grid-Point Representation

=============================================

+ ImmutableGrid(ImmutableList<GridPoint> grid);

+ Optional<GridPoint> getAtIndex(int width, int height);

+ Optional<GridPoint> getNeighbouringToIndex(int width, int height, DirEnum dir);

DelimitedDataParseService<T> - Parse data from file

+ T parseData(File file, String pattern);

+ T parseData(String fileName, String pattern);

RoutePoint implements Observer extends Observable

- atomic<int> degOfFreedom

- Map<DirEnum,RootPoint> next

- final int posX

- final int posY

+ equals()

+ hashCode()

Treesearch<T Extends Treeable> extends Thread implements Observable

+ Optional<List<T>> determinePath();

State implements Comparable

- final int steps

- final LinkedList<RoutePoint> breadcrumbs

**determinePath pseudo code**

1. Goal has been achieved -> END -> O/P Path
2. No more moves -> END -> Observable.Empty()

NO MORE MOVES == Blocked on all sides taking into account previous node

1. For: Each Direction in enum:
   1. Get next RoutePoint in DirEnum
   2. Set RoutePoint to Map next
   3. If: RoutePoint not in Map<RoutePoint,Thread> with not null thread
      1. Synchronize(Map<Point,Thread>) {

Initialise Thread

Set Map<Point, Thread>

Execute Thread

}

* 1. Else: Already processing RoutePoint, R
     1. If this.steps + 1 > R.steps
        1. degOfFreedom--
        2. continue to next iteration
     2. Else: this is more efficient
        1. Synchronise(RoutePoint R) {

Kill other awaiting Thread(s)

Observe target Thread

Update steps in State

Update R.breadcrumbs to this.breadcrumbs + this (A copy of: ?)

}

1. While: degOfFreedom >= 0 || !APP\_END
   1. wait

**Interrupt Exception Handling**

1. Interrupt caught in catch block
2. degreesOfFreedom--
3. If degreesOfFreedom == 0
   1. Notify other awaiting threads “ITS\_OVER”
   2. Kill self

**Observer update**

1. If null object => do nothing
2. Switch on object received o.toString():
   1. CASE “ITS\_OVER” => degreesOfFreedom--