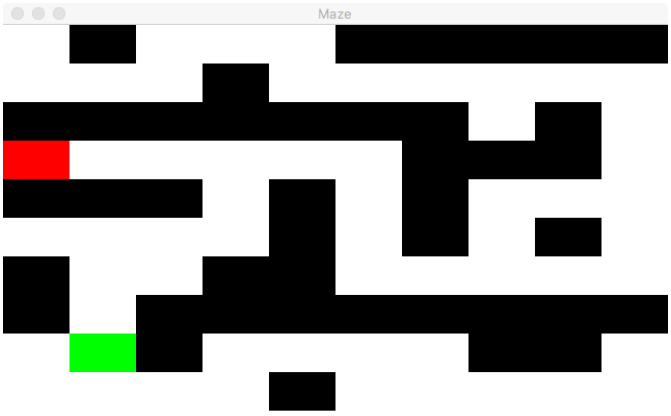
COM6516 Assignment: shortest path in a maze.

Consider a maze 10x10 defined in a .csv file. An example of such a maze is in map1.csv shown below, with a visualisation of it on the right:

```
,W,,,,W,W,W,W,W
,,,W,,,,,,
W,W,W,W,W,W,W,,W,
T,,,,,,W,W,W,
W,W,W,,W,,W,,,
,,,,W,,W,,W,
W,,,W,W,,,,,
W,,W,W,W,W,W,W,W,W
,L,W,,,,,W,W,
,,,,W,,,,,
```



In the .csv file commas separate cells. An empty cell is where a comma immediately follows another comma; two other cases are where a line starts with a comma or ends with a comma. Empty cells can be walked through, W means an impassable wall, T is a teleporter (displayed as a red square) and L is where the teleporter lands a player (depicted green). When a player steps on a teleporter, they are immediately transported to the 'landing spot' L. Teleporter is one-way only. You are expected to define a constant `public final int SIZE=10` and use `SIZE` in your work to make it capable of handling mazes of different size.
The starting cell is (0,0) (top left) and the exit is (`SIZE-1`, `SIZE-1`) (bottom-right).
Movement is left/right/up/down, there are no diagonal moves. There is exactly one teleporter and one landing cell.

You are expected to load the layout of a maze from a .csv file and plot it using rectangles in a similar way to the presented screenshots. You are also expected to plot a shortest path from the start to the end. Such a path is not necessarily unique and you should consider using a teleporter to shorten the travel time. Whenever you change a maze by clicking on different cells, the shortest path should be recomputed and a new path displayed.

Let the direction of movement be provided by enumeration

```
enum DIR { DIR_UP, DIR_DOWN, DIR_LEFT, DIR_RIGHT }
```

A maze is expected to be represented by a map from Position to CELL, with CELL defined as an enumeration:

```
enum CELL { CELL_W, CELL_E, CELL_T, CELL_L }
```

`CELL_W` is a solid wall,
`CELL_E` is an empty room,
`CELL_T` teleporter,
`CELL_L` where a teleporter places (lands) a player.

The maze is defined by

```
Map<Position,CELL> cell = new HashMap<Position,CELL>();
```

For a position p, `cell.get(p)` returns an instance of `CELL` reflecting what can be found at that position.