Escaping the Maze

This is David.

He needs your help. David woke up and found himself in a maze. How? He has no clue. You can help him, though! All you have to do is program an algorithm to help him escape in the fewest amount of steps. He wants to get out of the maze as soon as possible and go back to his exciting life of working as a paper shredder salesman in a company named "!(Dunder Mifflin)".

Running the program

You can find out how to set up the project in Eclipse by following the instructions in the “Maze Escape Setup Instructions” document.

When you run it, you must first look at the console and select which map you would like to run. This can be either map 1,2,3,4, or 5. You can not add any more maps but you can modify the ones provided for you. However, know that these maps will not be the only ones your algorithm will be tried on when you submit it.

When you try to make a move that calls for David to move somewhere to which he can not move (i.e. moving into a wall), he will not move at all and it will be considered that he has taken a step. You will only be given 5 seconds to decide which way to go on each step. David needs to escape the map within 300 steps or else he will be trapped in the maze forever.

You will use the Runner.java file to tell David what moves to make. The char value you return in the nextMove() method will tell him what move to make. The different possible moves are shown below:
‘U’ = go up
‘R’ = go right
‘D’ = go down
‘L’ = go left
‘N’ = do nothing

The maze will be provided to you in the the 2d char array “maze”. In it:
 = ‘P’ = David
 = ‘ ‘ = Open space
 = ‘X’ = Wall


= ‘E’ = The end of the maze
The current position of David will be “maze[row][column]”. The top left corner of the maze will be 0,0.

Essentially, your goal is to get the ‘P’ onto the same spot in the maze as the ‘E’.

The program will end with a print to the console. It will either say you succeeded in however many moves it took for David to get to the end, or it will say that you have failed. You will have failed if you have exceeded the maximum number of allotted steps (300 steps). It will then tell you how long it took for your program to execute.

How the winner is decided

The provided maps are for testing your program, the submissions will be evaluated based on other maps. The first score is the number of maps a program solves in 300 steps. If there is a tie, the tiebreaker will be total the number of steps. If there is a tie again, the second tiebreaker is will be the total execution time.