3. Word Search Puzzle Document Write-Up

High-Level Description:

To solve this problem I approached it in a three step process:

- 1. Establish origin: I needed to find if/where the first letter of the word was in the board, this was my starting point.
- 2. Establish Direction: Next, I needed to find the direction the word was going in, so I looked for the second letter of the word in all four directions. If found, I marked which direction it was pointing towards.
- 3. Search Direction: Finally, I would search for the remainder of the word in the direction that was given. If the word was found, I would return true, if one of the directions was a 'dead end' (meaning the word could not be found in that direction) I would backtrack to the original point and try the next direction. If all directions were tried and the word was not found, I would return false, backtrack to the original point, and continue searching the board for the first character (Essentially repeating from step 1).

Model:

I passed in the board and the word I was looking for. The word was passed in as a string, and every recursive call would reduce the word by one character until the word was empty. If the word was empty (base case), that meant the word was found. I also passed in the row and column of the current point on the board, this would correspond to the first letter of the *reduced* word I passed in. I also passed in the direction I was moving towards. I did this by passing in the change in row and column that would be made to move in that direction. (For example, if I was moving right, I would pass in (0, 1) because I would not change the row, but I would increment the column by 1).

Java Source Code (Recursive Backtracking Implementation)

```
public static boolean wordSearch(char[][] board, String word)
        {

                // 1. Establish origin
                // This function will search through the board until it
finds a char that
                //matches the first letter of the word we are looking for,
then calls helpers to do the rest
                char firstChar = word.charAt(0);
                String restOfWord = word.substring(1);
                boolean result = false;

                for(int row = 0; row < board.length; row++)
                {
                        for(int col = 0; col < board[0].length; col++)
                        {
                                if(board[row][col] == firstChar)
                                {

                                        result = searchDirectional(board,
restOfWord, row, col);

                                        if (result) return true;
                                }
                        }
                }
                return result;
        }

public static boolean searchDirectional(char[][] board, String word, int
row, int col)
        {
                // 2. Establish Direction
                // This one uses the second character of the word and tracks
in which direction it goes
                int[] dx = {0, 1, 0, -1};
                int[] dy = {1, 0, -1, 0};
                int nextRow = row;
                int nextCol = col;
                char secondChar = word.charAt(0); // Use second char because
I want to find the direction the search should go

                for(int i = 0; i < dx.length; i++)
                {
                        nextRow = row + dy[i];
                        nextCol = col + dx[i];
                        if(nextRow >= 0 && nextRow < board.length && nextCol
>= 0 && nextCol < board[0].length)
                        {
                                if(board[nextRow][nextCol] == secondChar)
                                {
                                        if(completeWord(board, word, row,
col, dx[i], dy[i]))
                                        {
                                                // System.out.println("TRUE
COMPLETE WHYARE YOU NOT RETURNING TRUe?????");
                                                return true;
                                        }
                                }
                        }
                }

                return false;

        }

public static boolean completeWord(char[][] board, String word, int row, int
col, int dirX, int dirY)
        {
                // 3. Search Direction
                // Base case: if we went through every character already,
that means we found the word
                if(word.length() == 0)
                {
                        // System.out.println("TRue");
                        return true;
                }

                // Edge cases: if on the sides of the board, cannot extend
past the borders, so return false
                //if we're going in that direction
                if(row == 0 && dirY == -1) return false;
                if(col == 0 && dirX == -1) return false;
                if(row == board.length-1 && dirY == 1) return false;
                if(col == board[0].length-1 && dirX == 1) return false;

                String reducedWord = word.substring(1);
                char searchChar = word.charAt(0);
                char searchLocation = board[row+dirY][col+dirX];

                if(searchChar != searchLocation)
                {
                        // System.out.println("Word_*"+word+"*_");
                        return false;
                }
                return completeWord(board, reducedWord, row+dirY, col+dirX,
dirX, dirY);
        }
```

Testing Program

```java
public static void main(String[] args)
{
        //Create a board to test with, find one on google/online
        char[][] board = //Board from Thompson's Teachings - - - Amanda 
Thompson on TPT
        {{'P', 'S', 'U', 'Z', 'Z', 'S', 'C', 'S'},
        {'L', 'K', 'T', 'W', 'R', 'L', 'O', 'E'},
        {'T', 'I', 'G', 'R', 'J', 'I', 'R', 'E'},
        {'M', 'N', 'L', 'Z', 'E', 'C', 'E', 'D'},
        {'C', 'B', 'U', 'S', 'H', 'E', 'L', 'S'}};
        //Words: Tree, Core, Slice, Bushel, Skin, Seeds

        String findStr1 = "BUSHEL"; //Horizontal Check
        String findStr2 = "SEEDS";  //Vertical Check
        String findStr3 = "BUG";    //False check


        boolean wordInBoard = wordSearch(board, findStr2);
        System.out.println("Word "+findStr2+" is in the board? "
+wordInBoard);

}
```

Sample Input and Output:

board //Board from Thompson's Teachings - - - Amanda Thompson on TPT
{{'P', 'S', 'U', 'Z', 'Z', 'S', 'C', 'S'},
{'L', 'K', 'T', 'W', 'R', 'L', 'O', 'E'},
{'T', 'I', 'G', 'R', 'J', 'I', 'R', 'E'},
{'M', 'N', 'L', 'Z', 'E', 'C', 'E', 'D'},
{'C', 'B', 'U', 'S', 'H', 'E', 'L', 'S'}};

words:
findStr1 = "BUSHEL"; //Horizontal Check (Expect True)
findStr2 = "SEEDS"; //Vertical Check (Expect True)
findStr3 = "BUG"; //False check (Expect False)