```
package hw3;

import static api.Orientation.*;
import static api.CellType.*;
import static api.Direction.*;

import java.util.ArrayList;

import api.Cell;
import api.CellType;
import api.Orientation;
/**
 * Utilities for parsing string descriptions of a grid.
 * @author Maxwell Skinner
 */
public class GridUtil {
      /**
       * Constructs a 2D grid of Cell objects given a 2D array of cell
descriptions.
       * String descriptions are a single character and have the following meaning.
       * <ul>
       * <li>"*" represents a wall.</li>
       * <li>"e" represents an exit.</li>
       * <li>"." represents a floor.</li>
       * <li>"[", "]", "^", "v", or "#" represent a part of a block. A block is not
a
       * type of cell, it is something placed on a cell floor. For these
descriptions
       * a cell is created with CellType of FLOOR. This method does not create any
       * blocks or set blocks on cells.</li>
       * </ul>
       * The method only creates cells and not blocks. See the other utility method
       * findBlocks which is used to create the blocks.
       *
       * @param desc a 2D array of strings describing the grid
       * @return a 2D array of cells the represent the grid without any blocks
present
       */
      public static Cell[][] createGrid(String[][] desc) {
            Cell [][] grid = new Cell [desc.length][desc[0].length];
            for(int i = 0; i< desc.length; i += 1) {
                  for(int j = 0; j< desc[i].length; j += 1) {
                        if (desc[i][j].equals("*")){
                              grid[i][j] = new Cell(CellType.WALL, i, j);
                        }
                        else if (desc[i][j].equals("e")) {
                              grid[i][j] = new Cell(CellType.EXIT, i, j);
                        }
                        else{
                              grid[i][j] = new Cell(CellType.FLOOR, i, j);
                        }
                  }
            }
            return grid;
      }

      /**
       * Returns a list of blocks that are constructed from a given 2D array of
cell
```

```
	 * descriptions. String descriptions are a single character and have the
	 * following meanings.
	 * <ul>
	 * <li>"[" the start (left most column) of a horizontal block</li>
	 * <li>"]" the end (right most column) of a horizontal block</li>
	 * <li>"^" the start (top most row) of a vertical block</li>
	 * <li>"v" the end (bottom most column) of a vertical block</li>
	 * <li>"#" inner segments of a block, these are always placed between the
start
	 * and end of the block</li>
	 * <li>"*", ".", and "e" symbols that describe cell types, meaning there is
not
	 * block currently over the cell</li>
	 * </ul>
	 *
	 * @param desc a 2D array of strings describing the grid
	 * @return a list of blocks found in the given grid description
	 */
	public static ArrayList<Block> findBlocks(String[][] desc) {
		int length = 0;
		ArrayList<Block> blocks = new ArrayList<Block>();
		for(int i = 0; i< desc.length; i += 1) {
			for(int j = 0; j< desc[i].length; j += 1) {
				if(desc[i][j].equals("[")) {
					length = 1;
					int k = j;
					while(!desc[i][k].equals("]")) {
						length += 1;
						k += 1;
					}
					blocks.add(new Block(i, j, length,
Orientation.HORIZONTAL));
				}
				else if(desc[i][j].equals("^")) {
						length = 1;
						int k = i;
						while(!desc[k][j].equals("v")) {
							length += 1;
							k += 1;
						}
					blocks.add(new Block(i, j, length,
Orientation.VERTICAL));
				}
			}
		}
		return blocks;
}
}
```