The Black Box Challenge

The aim of this lab is to give you the opportunity to freely decide on architecture, algorithm, and data-structure decisions to create a game with interesting mechanics. Go into the challenge with the following advice: this program can be easy to write with some good initial planning and design decisions, but a very messy program if you just start coding.

A 1978 Parker Brothers board game called "Black Box" involved locating deflecting mirrors hidden within a closed box. The box can be easily represented by a 12 x 12 `char` array, or an array of `Cell` objects that store a `char` and a `boolean` (as to if the character should be hidden).
Have the array initialized to all '.' characters for the inside 10 rows and columns.

Randomly place 10 mirrors represented by either '/' or '\\'. Around the perimeter put laser firing stations, labeled by alphabetic characters. The playing field is 10 x 10, with the extra 2 rows and columns for storing the names of the firing stations. Note that the corners are unused.

Example:

```
*a b c d e f g h i j *
k . . . . . . . . . . K
l . . . . . . . . . . L
m . . . . . . . \ . . M
n . . . . . . . . . . N
o . . . . . . . . . . O
p . . . . . . . . . . P
q . . . . . . . . . . Q
r . . / . . . . / . . R
s . . . . . . . . . . S
t . . . . . . . . . . T
* A B C D E F G H I J *
```

Programming tip: The '\' character is the first part of an escape sequence where you can issue commands such as return carriages and sound events. To store this character in a variable, such as a `char` or array of `char`s, you must double it up, like this:

```
char slash = '\\';
```

The variable slash will store one '\' `char`.

Here, three mirrors have been found so far at coordinates ('m','h'), ('r','c') and ('r','h').
The mirrors will be hidden unless the player correctly guesses at their location.
The player will have three options:

```
(1) Shoot a laser
(2) Guess at a mirror location,
(3) Quit the game.
```

If the player were to choose to shoot a laser, then it would ask for a location to fire from. Shooting from station 'e' would result in a message stating that the laser ended up at station 'E'. Shooting from station 'r' would say that it ended up at station 'c' (after bouncing off the mirror). Shooting from station 'C' would say that it ended up at station 'm' (after bouncing off of the 3 known mirrors).
If the player were to guess at a mirror location correctly, then it will reveal the mirror. The object is to find all of the mirrors in under 20 guesses and 25 shots. You will need to keep track of the number of guesses and shots taken.

Suggestions:

- Create a Laser object that has an x value, y value (coordinates) and a direction (up-0, right-1, down-2, left-3). X values will increase when moving down (moving to a lower row in the array) and decrease when moving up. Y values will increase when moving right (moving to another column) and decrease when moving left.

- Have the Laser continue to move until it hits an alphabetic character (which will be the station it ends up at). If it hits a mirror, the Laser's direction will change. This is similar to Battleship, but requires more planning, so…plan before you program.

Extension: display the game using graphic panels instead of text.

Consider that if you have a Laser object with int data fields for the row and column for location and a third for the direction, where 0 is up, 1 is right, 2 is down and 3 is left.

One of the biggest challenges is to create a Laser at the correct position and direction given the starting location as a char. The following constructor will help you. Consider that the method assumes that the board is 12x12, where the valid board positions are from row 1, column 1 to row 10, column 10. Row 0, row 11, column 0 and column 11 are used to contain the station labels and mark which positions are considered out of bounds for a Laser.

```
public class Laser
{
   private int row, col, dir;
   public static final int UP    = 0;
   public static final int RIGHT = 1;
   public static final int DOWN  = 2;
   public static final int LEFT  = 3;

   public Laser (char station) {
      if(station>='a' && station<='j') {
         row = 1;
         col = (int)(station-'a')+1;
         dir = DOWN;
      }
      else if(station>='A' && station<='J') {
         row = 10;
         col = (int)(station-'A')+1;
         dir = UP;
      }
      else if(station>='k' && station<='t') {
         row = (int)(station-'k')+1;
         col = 1;
         dir = RIGHT;
      }
      else   //if(station>='K' && station<='T') {
         row = (int)(station-'K')+1;
         col = 10;
         dir = LEFT;
      }
   }

//other Laser methods defined here
}
```