**Cookie Monster**

Write a program in which Cookie Monster finds the optimal path from the upper left corner (0,0) to the lower right corner (SIZE - 1, SIZE - 1) in a "cookie grid" (a 2D array). The elements of the grid contain cookies (a non-negative integer) or barrels (-1). Each step, Cookie Monster can only go down or to the right. He is not allowed to step on barrels. The optimal path contains the largest number of cookies.

You will be provided with a two-dimensional array containing cookies, open spaces and barrels. You need to write code that finds the best path.

Your program will work with a randomly generated array containing data elements as shown below:

1 3 0 5 -1 7 -1 -1 0 4 2 1

-1 3 2 1 -1 4 -1 5 3 -1 1 0

5 4 8 -1 3 2 2 -1 4 -1 0 0

2 1 0 4 1 -1 8 0 2 -1 2 5

1 4 0 1 -1 0 3 2 2 4 1 4

0 1 4 1 1 6 1 4 5 2 1 0

3 2 5 2 0 7 -1 2 1 0 -1 3

0 -1 4 -1 -1 3 5 1 4 2 1 2

5 4 8 -1 3 2 2 -1 4 -1 0 0

2 1 0 4 1 -1 8 0 2 -1 2 5

1 3 0 5 -1 7 -1 -1 0 4 2 1

0 0 3 1 5 2 1 5 4 1 3 3

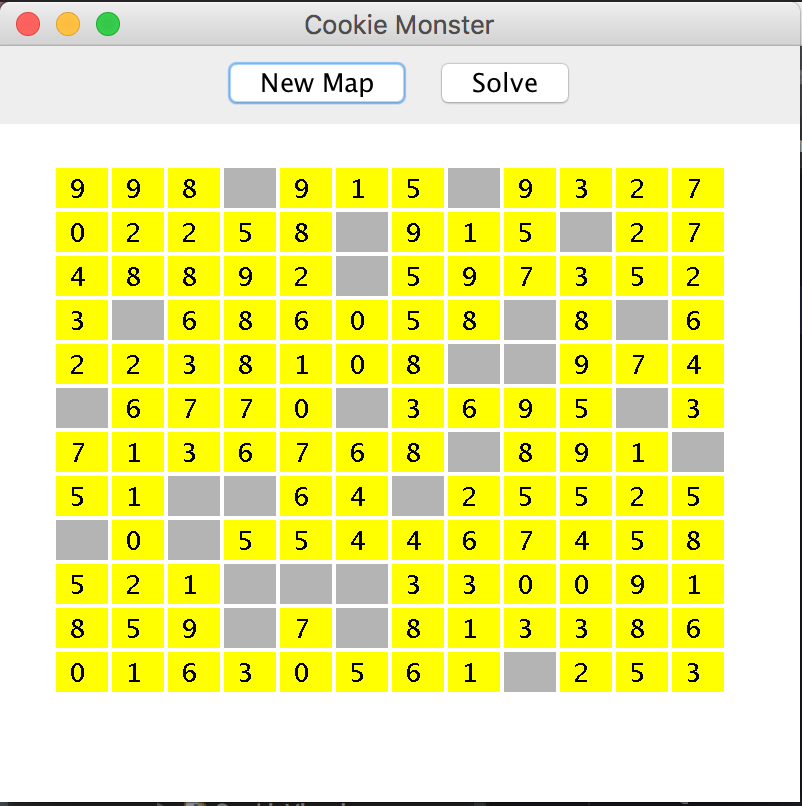
Before you start writing code, take the time to understand the algorithm. Do a pencil-trace of the following sequence using the above data grid and see if you are convinced it will work. If there is only one way to proceed from the current position, go there and update the total accumulated number of cookies.

If there are two ways to proceed, save one of the possible two points (and its total) on a stack and proceed to the other point. If you have reached the lower right corner, update the maximum if need be. If there is nowhere to go, examine the stack: pop a saved point, if any, and resume from there.

**The Assignment**

As described previously, find the number of cookies on the "optimal path" through the cookies; your results will be presented graphically.

This project follows the Model-View-Controller (MVC) model. You are provided with the Controller, **CookieMonster.java**, the View, **CookieView.java**, and a functional Model component, **CookieModel.java**. These files, as provided, are enough to compile and see the format of the output:



To complete the project, you must find the optimal path. You should use a stack to store intermediate steps in the solution. All your code is to be written in the CookieModel.java file. The entry point is the solve method. The optimum path solution changes each cookie count in the cells along the best path to their original value, plus 100. This allows the CookieView component to display the results: all cells over 100 are displayed in red, with the original value restored for display.

You may find it useful to create additional classes for this project, such as a Location class to store an (x, y) coordinate pair, and a CookieNode that includes a cookie count and a path to store on the stack. To store the path, you have many options, such as using a convenient data structure (e.g. ArrayList). Here is another:

* If you know linked lists, have each CookieNodestore a reference to its previous node (the node that added it to the path). Walking previous references backwards from the bottom-right node will give you the path.

Remember: there may be more than one path, and more than one cookie count, for most nodes in the grid. You are interested in the highest cookie count and the associated path.

Example of a finished path:



Based on the Cookie Monster lab

*http://www.rfrank.net/cslabs/1520-cookie/1520.html*