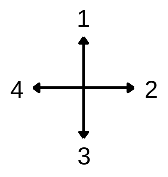
Java Computer Programming
Arrays
Mouse On The Island

The diagram below represents an island surrounded by water (on the border -1 represents the water and 0 on the border represents an escape bridge). A mouse is placed at the center of the island. Write a program to make the mouse take a 1000 walks through the island. The mouse is allowed to travel one square at a time, either horizontally or vertically. A random number in the range of 1 to 4 should be used to decide which direction the mouse is to take. The mouse escapes when hitting a bridge but drowns when hitting the water.
What are the mouse's chances? You may generate a random number up to 50 times. If the mouse does not find his way out by the 50th try, it will die of starvation. Have your program run the simulation 1000 times.

In the method **InitIsland**, you should initialize the 10 row by 15 column array **island** by following directions at the bottom of this page. In method **MoveAbout**, move the mouse around the island continuing until the mouse drowns, escapes, or starves. In **PrintIsland**, show the final contents of **island**, and indicate what happened to the mouse. This method should return a String variable of the entire island. A CheckStatus method will determine the status of the mouse after the random move. In method **GetRandom**, generate a random number in the range **high** to **low**. Note that **high** and **low** are parameters in the function heading for method **GetRandom** and do not appear outside of **GetRandom**.

-1	-1	0	-1	-1	-1	0	-1	0	-1	0	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	-1	-1	0	0	-1	-1	0	-1	-1	-1	0	-1	-1

A random number in the range of 1 to 4 causes a move of one unit as demonstrated in the diagram below.



Method **InitIsland**:

Create the table by setting all values in the array to zero unless you are on row 0 or 9 or on column 0 or 14. Next determine the border. Create a constant PERCENT_BRIDGES to represent a percent such as 30. If in row 0 or 9 or in column 0 or 14, generate a random number value in the range 1 to 100. If the random number ≤ PERCENT_BRIDGES, make the array element 0 (a bridge) and otherwise -1 (water).

After the island is initialized, assign the variable **mRow** to the middle row. Then assign another variable **mCol** to the middle column. The *mRow, mCol* element of the array marks the beginning of the simulation. The variables *mRow* and *mCol* keep track of the current mouse location.

So the method **InitIsland** initializes **island**, **mRow**, and **mCol**. Identify three Boolean variables to represent the status of the mouse – onboard, starved, and drowned.

Notes:

- If a random number causes a move to row 0, row 9, column 0, or column 14 then the condition of the mouse changes to *drowned* if the cell contains the number -1 or *escaped* if the cell contains a 0. If you move to any cell other than one in row 0 or row 9 and other than one in column 0 or column 14, then you should increment the number in the cell by 1. If you use all 50 random numbers without moving to the border, then the mouse starves.
- A random number of 1 causes **mRow** to decrement by one, a random number of 2 causes **mCol** to increment by one, etc.
- Run the simulation 1000 times and keep track of the number of times the mouse starves, escapes, or drowns. Use function **PrintIsland** to print the island for the first 3 times only. At the end, print the percent of times the mouse starves, the percent the mouse escapes, and the percent the mouse drowns.

Output: sample output is shown below. Note that by changing the contents of the array as you generate the random numbers, you can trace the path taken by the mouse. Also note that when printing contents of the array on the borders (row 0 or 9 or column 0 or 14) you should print a B if on a bridge (element row, column = 0) and a W if in the water (element row, column = -1).

```
The mouse has unfortunately drowned
   W   W   W   B   W   B   W   B   B   W   W   W   B   B   W
   W   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   B   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   W   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   W   0   0   0   4   3   0   0   0   0   0   0   0   0   B
   W   8   7   6   5   2   1   0   0   0   0   0   0   0   W
   W   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   W   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   B   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   W   W   W   B   W   W   W   W   W   B   W   W   W   B   W

The mouse has escaped
   B   W   W   W   W   B   W   B   W   W   W   W   W   W   B
   W   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   W   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   W   0   0   0   0   0  31  32   0   0   0   0   0   0   W
   B   0   0   0   0   7  16  33  26   0   0   0   0   0   W
   B   0   0   0   0   6  35  34  25   0   0   0   0   0   W
   W   0   0   0   0   5  36  37  38   0   0   0   0   0   B
   W   0   0   0   0   0   0   0  39  42   0   0   0   0   W
   W   0   0   0   0   0   0   0  40  43   0   0   0   0   W
   B   W   B   W   B   W   B   W   W   B   B   W   W   W   B

The mouse has run out of food and starved
   B   W   W   W   B   W   B   W   W   B   W   W   W   W   B
   W   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   B   0   0   0   0   0   0   0   0   0   0   0   0   0   W
   B  24  31  28   0  42   0   0   0   0   0   0   0   0   W
   W   0  32  37  38  43   0   0   0   0   0   0   0   0   W
   B   0   0   0  39  44   3   0   0   0   0   0   0   0   W
   W   0   0   0  14  45  10   0   0   0   0   0   0   0   W
   W   0   0   0   0  46   0  50   0   0   0   0   0   0   W
   B   0   0   0   0  47  48  49   0   0   0   0   0   0   B
   W   W   W   W   B   W   W   B   W   W   W   B   W   W   W


In 1000 simulations:
      percent escaped = 25.0
      percent drowned = 57.4
      percent starved = 17.6
```

Notes on global constants:

- The percentages you get at the end of the program are somewhat dependent on the value of constant PERCENT_BRIDGES. Use any value for this that seems reasonable.
- You should probably also use constants of NUM_ROWS and NUM_COLS. These values should be set to 10 and 15 respectively. Changing the size of the array changes the percentages at the end.
- A constant MAX_MOVES should be set to 50.
- A constant NUM_SIMULATIONS should be set to 1000.