

## Program 19 – Have a penny, leave a penny....

In the game of Penny Pitch, a two dimensional board is laid out as follows:

```
1 1 1 1 1
1 2 2 2 1
1 2 3 2 1
1 2 2 2 1
1 1 1 1 1
```

A player “tosses” five pennies on the board, trying for the number with the highest value. At the end of the game, the sum total of the tosses is returned. Develop a program that plays this game. The program should display the board, the toss number, the total score, and perform the following steps each time the user presses Enter:

- Generate two random numbers for the row and column of the toss.
- Add the number at this position to the running total.
- Display the board, replacing the numbers with \* where the pennies land.

(*Hint:* Use a two dimensional array of Square objects (a different class) for this problem. Each square contains a value number like those shown, and a Boolean flag that indicates whether or not a penny has landed on that square [see next page])

### Sample Output:

```
Welcome to penny toss.

Press <enter> to throw a penny/continue.

Throw #1 of 5

* 1 1 1 1
1 2 2 2 1
1 2 3 2 1
1 2 2 2 1
1 1 1 1 1

You got a 1.
Your total score so far is: 1
Press <enter> to throw a penny/continue.

Throw #2 of 5

* 1 1 1 1
1 2 2 2 1
1 2 3 2 1
1 * 2 2 1
1 1 1 1 1

You got a 2.
Your total score so far is: 3
Press <enter> to throw a penny/continue.

Throw #3 of 5

* 1 1 1 1
1 * 2 2 1
1 2 3 2 1
1 * 2 2 1
1 1 1 1 1
```

```
You got a 2.
Your total score so far is: 5
Press <enter> to throw a penny/continue.

Throw #4 of 5

* 1 1 1 1
1 * 2 2 1
1 2 3 2 1
1 * 2 2 1
1 1 1 1 1

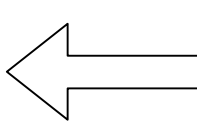
You got a 2.
Your total score so far is: 7
Press <enter> to throw a penny/continue.

Throw #5 of 5

* 1 1 1 1
1 * 2 2 1
1 2 * 2 1
1 * 2 2 1
1 1 1 1 1

You got a 3.

Your final score was 10.
Would you like to try again? (y/n)
```



*Penny landed on an  
occupied square.  
Just add in the  
value.*

## Class SquareObject

### Private instance variables

<code>int <i>squareValue</i></code>	<code>// holds value of current square</code>
<code>boolean <i>present</i></code>	<code>// tells if square holds a penny</code>

### Methods

<code>Default constructor</code>	<code>// sets <i>squareValue</i> to 1 and <i>present</i> to false</code>
<code>Constructor with parameter</code>	<code>// sets <i>squareValue</i> to parameter, <i>present</i> to false</code>
<code>getSquareValue</code>	<code>// returns value of <i>squareValue</i></code>
<code>setSquareValue</code>	<code>// sets value of <i>squareValue</i></code>
<code>hasPenny</code>	<code>// returns value of <i>present</i></code>
<code>setPenny</code>	<code>// sets value of <i>present</i></code>

## Class Program19

### Methods

<code>main</code>	<code>// main loop to get input, make random numbers</code> <code>// set the state of individual squares, etc.</code>
<code>displayBoard *</code>	<code>// displays the received board</code>
<code>resetBoard *</code>	<code>// fills/resets the received board by filling it with</code> <code>// square objects of the correct value/state</code>

\*Note: because main is a static method (more on this later) and methods/variables declared in the Program19 class that the main method wishes to access must either be declared locally or declared as static as well.



For a couple of extra points (2-3), ask the user for the size of the grid, and the number of pennies (tries) to toss. Follow the same pattern for determining the grid layout.

Ex: a grid that is 7x7 would like this:

```
1 1 1 1 1 1 1
1 2 2 2 2 2 1
1 2 3 3 3 2 1
1 2 3 4 3 2 1
1 2 3 3 3 2 1
1 2 2 2 2 2 1
1 1 1 1 1 1 1
```