The method we designed last time

Hopefully I've remembered to pack my adapter and we can actually look at the code, otherwise I'll print it out and hand it out.

Walk through methods, see what they do, figure out how to incorporate each one into the primary `int2string()` method.

Call stacks

Let's look at the call stack that occurs when our program executes.

1. Start up called method: allocate AR, parameters
2. Execute method
3. Return to caller: remove AR, replace call with return value
4. Continue executing calling method

Let's review arrays!

We've talked briefly about how arrays can behave strangely (at least compared to primitives) when you pass them to a method. $\Rightarrow$ "Pass-by-value"

```
public static int zeroOut(int[] arr) {
    for (int i=0; i<arr.length; i++) {
        arr[i] = 0; // does this persist?
    }
}
```

What's the result of calling this method?

More array shenanigans

All right, so how about this one?

```
public static void resize(int[] arr) {
    arr = new int[arr.length*2];
    for (int i=0; i<arr.length; i++) {
        arr[i] = 0;
    }
}
```

Returning arrays from methods

You can return an array from a method, with no fuss:

```
public static int[] resize(int[] arr) {
    newArr = new int[arr.length*2];
    for (int i=0; i<newArr.length; i++) {
        newArr[i] = 0;
    }
    return newArr;
}
```

Arrays of Arrays

So, remember when I said you can make an array out of *any* variable type?

And then how I said arrays are variable types?

...do you see where this is going?

2D Arrays

In databases, for example, we might have an arrangement of rows and columns. Or, if you're mathy, you might have dealt with matrices. Or maybe you're just wasting time in the back of the room.

X		O
O	X	
		X

Using 2D Arrays

Like the arrays we've been using, 2D arrays have a special syntax:

```
char[][] tictactoe = new char[3][3];

for (int i=0; i<tictactoe.length; i++) {
    for (int j=0; j<tictactoe[i].length; j++) {
        tictactoe[i][j] = ' ';
    }
}
```

We traverse entire 2D arrays with *nested* `for` loops.

Initializing 2D arrays with {}

You can also explicitly give an array's initial values using the {} syntax we used before:

```
char[][] tictactoe =
    {
        {'x',' ','o'},  // row 0
        {'o','x',' '},  // row 1
        {' ',' ','x'}   // row 2
    };
```

Each row needs to have its own {} declaration, and each element is separated by a comma.

Accessing elements by index

How do we check the neighboring elements of a particular element `[i][j]` of an array?

Remember that the row index is `i`, and the column index is `j`.

Checking a tic-tac-toe game for a win

If we've got time - let's talk about how to conceptually check if there's a win in a tic-tac-toe array like the one we were using before.

Pseudocode only!