2D Arrays

Multi-dimensional arrays are easy to work with in Java. We will here work with 2D arrays that you can think of as having rows and columns. Java allows arrays to have 3, 4, or even more dimensions, but 2D arrays are the most common of the multi-dimensional structures.

The type of a 2D array with base type E is

```
E [ ][ ]
```

The constructor for such an array with N rows and M columns is

```
new E[N][M];
```

For example, we might make an array of ints with 5 rows and 3 columns with

```
int[ ][ ] A = new int[5][3];
```

When you are working with an array of objects, note that

```
new E[5][5];
```

doesn't construct objects of class E; it just makes an array with 25 slots that can hold pointers to E objects.

A 2D array is not laid out spatially in memory, but we usually think of the first index as referring to the rows and the second index as referring to columns. Note that the organization in memory (which is inherently linear) is <first row><second row><third row>.... This means that the elements of any row are consecutive in memory and can be treated as a 1D array. **If M is a 2D array then we can refer to M[0] as its first row, M[1] is its second. There is no easy way to refer to the columns of M**.

We usually process a 2D array with a double for-loop: an outer loop over its rows and an inner loop on the column entries of a specific row.
For example, we might find if array M contains element e with

```
public boolean contains(E[][] M, E e) {
        for (int row = 0; row < M.length; row++)
                for (int col = 0; col < M[row].length; col++)
                        if (M[row][col].equals(e))
                                return true;
        return false;
}
```

For example, in Lab 3 we work with arrays that represent mazes.

Here is a typical maze file:

```
3 5
0 0 1 0 0
2 0 1 0 3
0 0 1 0 0
```

The first two numbers are the number of rows and the number of columns of the maze. The rest is a 2D array of integers that indicate, walls, entraces, etc.

In Lab 3 we make a Square class that represents one square of this array. It is useful for Squares to know their location in the maze, so the constructor for Square asks for the square's row, column and type. Here is the code I use to read such a file:

```
Scanner reader = new Scanner( new File(fname));
numRows =  reader.nextInt();
numCols = reader.nextInt();
Square[ ][ ] maze = new Square[numRows][numCols];
for (int row = 0; row < numRows; row++)
        for (int col = 0; col < numCols; col++) {
                int type = reader.nextInt();
                maze[row][col] = new Square(row, col, type);
        }
}
```