Lecture 13: Two-Dimensional Arrays

Building Java Programs: A Back to Basics Approach
by Stuart Reges and Marty Stepp

Nested Loops

Nested loops

- **nested loop**: A loop placed inside another loop.

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.print("*");
    }
    System.out.println();   // to end the line
}
```

- Output:

```
**********
**********
**********
**********
**********
```

- The outer loop repeats 5 times; the inner one 10 times.
  - "sets and reps" exercise analogy

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

- Output:

```
*
**
***
****
*****
```

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(i);
    }
    System.out.println();
}
```

- Output:

```
1
22
333
4444
55555
```

Nested `for` loop exercise

- What is the output of the following nested `for` loops?

```
for (int i = 1; i <= 5; i++) {
    for (int j = i; j <= 5; j++) {
        System.out.print(i);
    }
    System.out.println();
}
```

- Output:

```
11111
2222
333
44
5
```

Common errors

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; i <= 10; j++) {
        System.out.print("*");
    }
    System.out.println();
}

for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 10; i++) {
        System.out.print("*");
    }
    System.out.println();
}
```

Both of the above sets of code produce *infinite loops*:

2-Dimensional Arrays

2D Array

0	0	0	0
0	0	0	0
0	0	0	0

```
int[][] matrix=new int[3][4]; //3 rows, 4 columns
                              //initialized to 0.
```

2D Array

2	0	0	0
0	0	-6	0
0	7	0	0

```
matrix[0][0]=2;
matrix[1][2]=-6;
matrix[2][1]=7;
```

Declare and Initialize

Declaring and initializing 2D arrays.

```
int[][] table; //2D array of ints, null reference


//one way to initialize 2D array.
double[][] matrix=new double[4][5];
//4 rows, 5 columns
//initialized all to 0.0

String[][] strs=new String[2][5];
//strs reference 2x5 array of
//String objects. Each element is
// null
```

Initializer List

```
//another way to initialize 2D array is through a list.

int[] array1={1,4,3};

int[][] mat={{3,4,5}, {6,7,8}}; //2 rows, 3 columns
```

3	4	5
6	7	8

Array of Arrays

- A matrix is implemented as an array of row arrays. Each row is a one-dimensional array of elements. Suppose that mat is the matrix

3	-4	1	2
6	0	8	1
-2	9	1	7

Then mat is an array of three arrays:

mat[0] is the one-dimensional array {3,-4,1,2}.

mat[1] is the one-dimensional array {6,0,8,1}.

mat[2] is the one-dimensional array {-2,9,1,7}.

mat.length is the number of rows.

Array of Arrays

3	-4	1	2
6	0	8	1
-2	9	1	7

- mat.length is the number of rows. In this case, it equals 3 because there are three row-arrays in mat.
- For each k, where 0<=k<mat.length, mat[k].length is the number of elements in that row, namely the number of columns. In this case, mat[k].length=4 for all k.
- Java allows “jagged arrays” where each row array may have different lengths. However, on the AP exam, assume all arrays are rectangular.

Initializer List

```
int[][] mat={{3,4,5},{1,2},{0,1,-3,5}};
```

```
mat[0]={3,4,5}
mat[1]={1,2}
mat[2]={0,1,-3,5}
```

```
mat.length=3
mat[0].length=3
mat[1].length=2
mat[2].length=4
```

Row-Column Traversal

Suppose that `mat` is a 2D array initialized with integers. Use nested for loop to print out the elements of the array.

```
for(int i=0;i<mat.length;i++){
 for(int j=0;j<mat[i].length;j++)
     System.out.print(mat[i][j]+ “ ”);
 System.out.println();
}
```

Row-by-Row

Suppose the following method has been implemented.

```
public void printArray(int[] array)
{/*implementation not shown*/}
```

Use it to print out the 2D array `mat`.

```
for(int i=0;i<mat.length;i++){
 printArray(mat[i]); //mat[i] is the ith row of mat
 System.out.println();
}
```

2D Arrays of Objects

```
Point[][] pointMatrix;
```

Suppose that pointMatrix is initialized with Point objects. Change the x-coordinate of each Point to 1.

```
for(int row=0;row<pointMatrix.length;row++)
 for(int col=0;col<pointMatrix[0].length;col++)
     pointMatrix[row][col].setX(1);
```

Lab 1

Write the following methods.

`sum`: Write method `sum` which accepts a 2D array of integers and returns the sum of all of the elements. Use row-column traversal method. Nested Loop.

`rowSum`: `rowSum` accepts two parameters: a 2D array of integers and an integer `row`. `rowSum` returns the sum of the integers of elements in the row given by `row`.

`sum2`: This method is the same as `sum` above **but you must use `rowSum` in your code. One loop.**

Lab 1

Write the following methods.

`largest` accepts a 2D array of integers and returns the largest value. Use row-column traversal method to examine each value.

`largestByRow` accepts two parameters: a 2D array of integers and an integer `row`. `largestByRow` returns the largest value in the row given by `row`.

`largest2` accepts a 2D array of integers and returns the largest value. **You must call `largestByRow`. One loop.**

Lab 1

`printTranspose`: Given 2D array of integers, print the transpose of the array. The transpose of a 2D array is the array whose rows are the columns of the original array. **Do not create a new array, instead, use for loops to traverse the original array.**

If `mat={{1,2,3},{4,5,6}}; printTranspose(mat)` will print:

```
1 4
2 5
3 6
```

Lab 2

A magic square is an NxN array of numbers such that

1. Every number from 1 through N^2 must appear exactly once.
2. Every row, column, major and minor diagonal must add up to the same total.

Example: N=4

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Lab 2

Write the class `MagicSquare` with instance methods given in the next few slides. MagicSquare should have an instance 2D array variable. The methods `rowSum, colSum, diagSums` and `exactlyOnce` are intermediate methods to help you write the `isMagic` method, which determines whether a square is magic.

You must use the method headers indicated for each method. Write a driver class with a main method to test your MagicSquare class.

Lab 2

```
public int rowSum(int row){…}
```

Returns the row sum indicated by `row`.

```
public int colSum(int col){…}
```

Returns the column sum indicated by `col`.

Lab 2

```
public boolean diagSums(int sum){…}
```

Returns whether both the major and minor diagonal sums are equal to `sum`. The major and minor diagonal are highlighted below.

16	3	2	**13**
5	**10**	**11**	8
9	**6**	**7**	12
4	15	14	**1**

Lab 2

`public boolean exactlyOnce(){…}`

Returns true if the numbers 1 to N^2 occurs exactly once in `square` and false otherwise. N is the number of rows(and columns) in `square`.

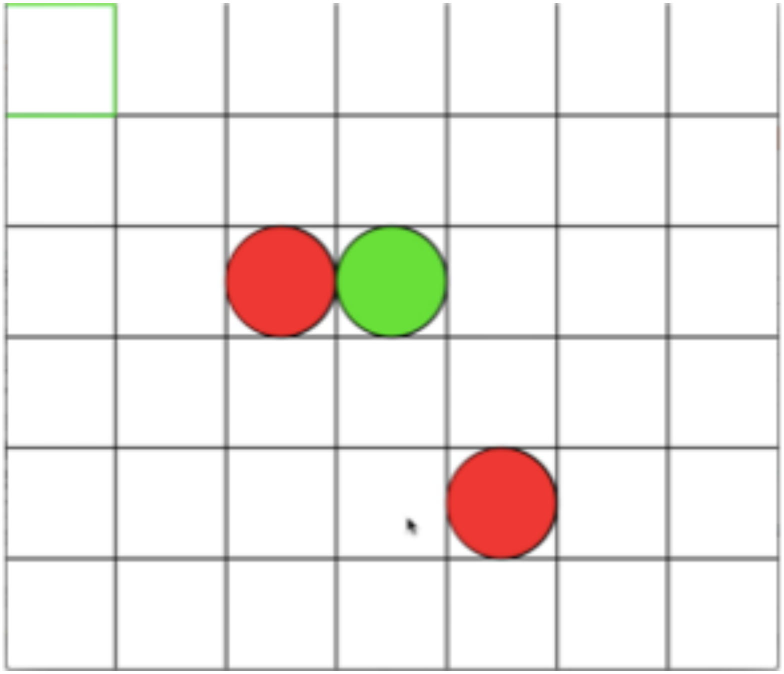
You must use the each of the above methods to write the following isMagic method.

`public boolean isMagic(){…}`

Returns true if `square` is magic and false otherwise.

Lab 3(Grid)

Write a program that creates a 2D(6 rows x 7 cols) grid as shown.



If a mouse is clicked inside of a Cell, a centered circle is drawn. The color of the circle alternates between red and green representing two players.

This program can be converted into Connect 4.

A template is provided on my website if you wish to get some help.

Lab 3

Add a transparent square(green square, top left in image below) to the grid that responds to keyboard inputs.

If a Cell is selected and ENTER is pressed, then that Cell displays the the circle as before.