Panel 1

Multi-Dimensional Arrays

1-D Array:

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

a_0 a_1 a_2 — 3 elements

2-D Array:

```
int A[][] = new int [3][2]
```

rows, columns — 6 elements

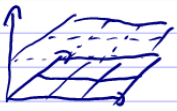
a_{00} a_{01}
a_{10} a_{11}
a_{20} a_{21}

2D Arrays are like "tables" or, in math lingo, like matrix.
Useful in specific situations

1

Panel 2

3D Array



are like a "box"

rarely used

```
int A[][][] = new int [3][4][5]
```

has 60 elements

4-D array

```
int A[][][][] = new int [3][4][5][6]
```

has 360 elements

not used

```
A[0][0][0][0] = 100
```

2

Panel 3

Ex: Declare 5×4 array and initialize each element to 42!

```
int A[][] = new int[5][4];
for (int i = 0; i < 5 ; i++)
{
    for (int j = 0; j < 4 ; j++)
    {
        A[i][j] = 42;
    }
}
```

poor programming!

1. poor choice of variables
2. Don't use constants

3

Panel 4

```
int A[][] = new int[5][4];
for (int row = 0; row < A.length; row++)
{
    for (int col = 0; col < A[row].length; col++)
    {
        A[row][col] = 42;
    }
}
```

Write code that adds upper-left and lower-right

a_{00} a_{01} a_{02} ... a_{10} a_{11} a_{12} ...

⇒ A[0][0] + A[4][3] ✓

A[0][0] + A[A.length-1][A[A.len

4

Panel 5

```
A[0][0] + A[A.length-1][A[A.length-1].length-1];
```

Difficult to figure, so use variables nRows, nCols for # of rows and columns.

Ex:

```
int nRows = 8;   10
int nCols = 4;   17
int A[][] = new int[nRows][nCols];
int sum = A[0][0] + A[nRows-1][nCols-1];
```

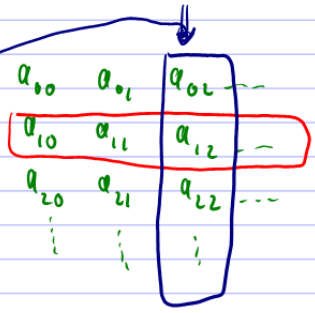
Rule: Avoid unnamed constants!

5

Panel 6

```
int A[][] = new int[nRows][nCols]
```

Find a) sum of row 1
b) sum of col 2
c) total sum



a)
```
int sum = 0;
for (int col = 0; col < nCols; col++)
    sum += A[1][col];
```

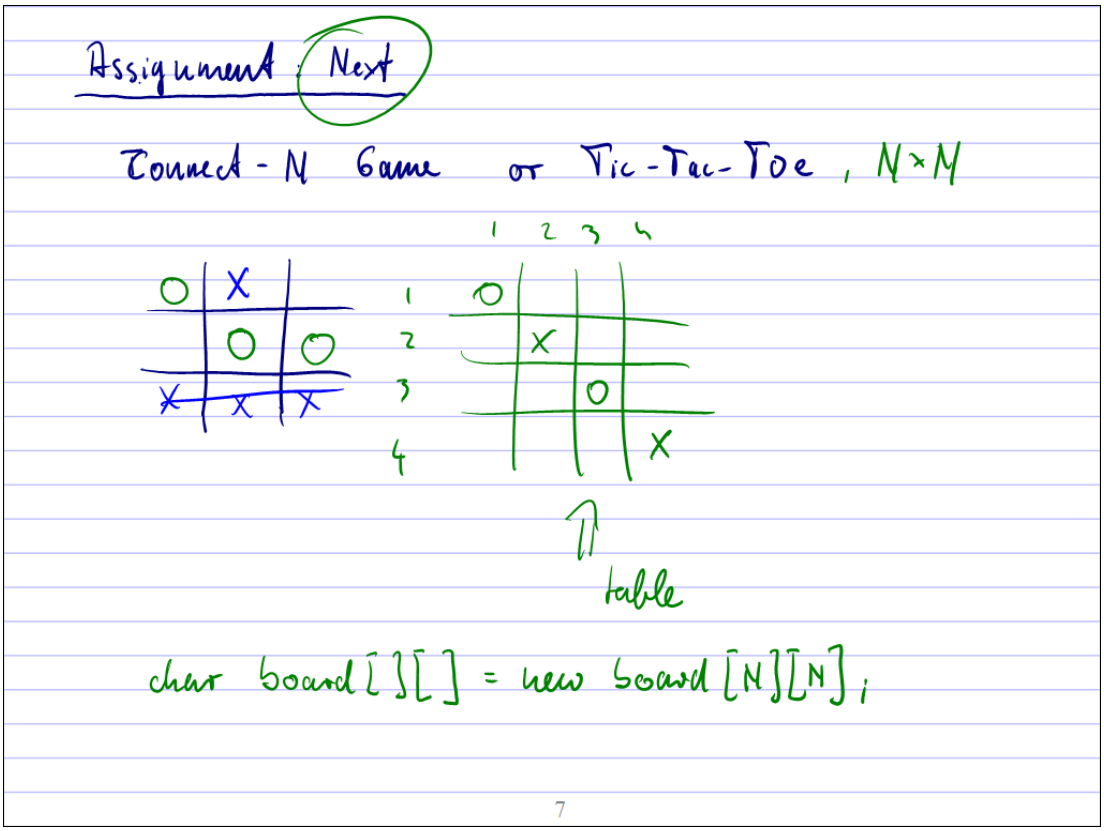
```
A[row][2]
```

```
for (row...
} for (col
  {
  }
```

```
A[row][col]
```

6

Panel 7


Assignment : Next
Connect - N Game or Tic-Tac-Toe, N×N
1 2 3 4
1
2
3
4
table
char board[][] = new board[N][N];
7