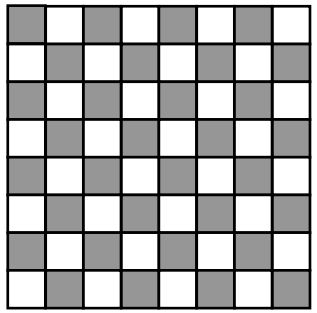
2D Data

- data in rows and columns

X	O	X
	O	



$$\begin{pmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{pmatrix}$$

ID	A1	A2	A3	A4
30012				
30031				
30048				
30056				
30080				
30118				
30185				
30302				
30345				
30382				
30495				
30515				
30545				

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
9-10							
10-11							
11-12							
12-1							
1-2							
2-3							
3-4							
4-5							

© Mohammad Nekooei and Peter Andreae

2D arrays: Creating

- 2D arrays require two indices, and two sizes
- Declaring and creating:

```
int[ ] [ ] marks =  new int [200][4];

ChessPiece[ ][ ] board = new ChessPiece [8][8];

Color[ ][ ] image = new Color [100][150];

int[ ][ ] matrix = new int [ ][ ]{{2, 4, 3},{5, 8, 4}, {8, 4, 9}};
```

- First index, second index : which is the row and which is the column?
  - You choose! Choose your variable names carefully.
  - Typically use first index as the row.

© Mohammad Nekooei and Peter Andreae

2D arrays: Accessing

- Assigning and accessing:

```
marks[10][3] = 72;

board[row][col] = board[row][col-1];
board[row][col-1] = null;


for (int row=0; row<height; row++){
    for (int col=0; col<width; col++){
        image[row][col] = image[row][col].darker();
    }
}
```

- In Java, can't use commas

~~image[row, col]~~

© Mohammad Nekooei and Peter Andreae

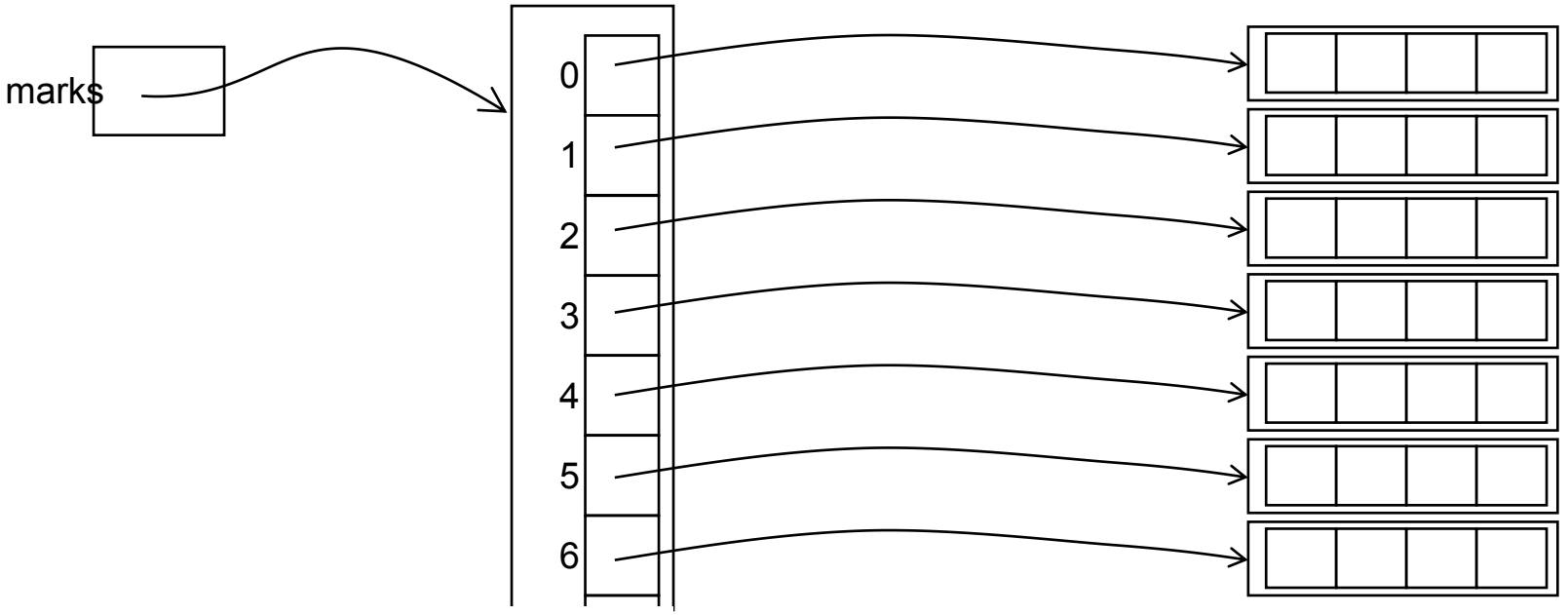
2D arrays

- 2D arrays are actually arrays of arrays:

```
int[ ] [ ] marks =  new int[200] [4];
```

is the same as

```
int[ ] [ ] marks =  new int[200] [ ];
for (int i = 0; i<200; i++){
    marks[i] = new int[4];
}
```



© Mohammad Nekooei and Peter Andreae

2D arrays: length

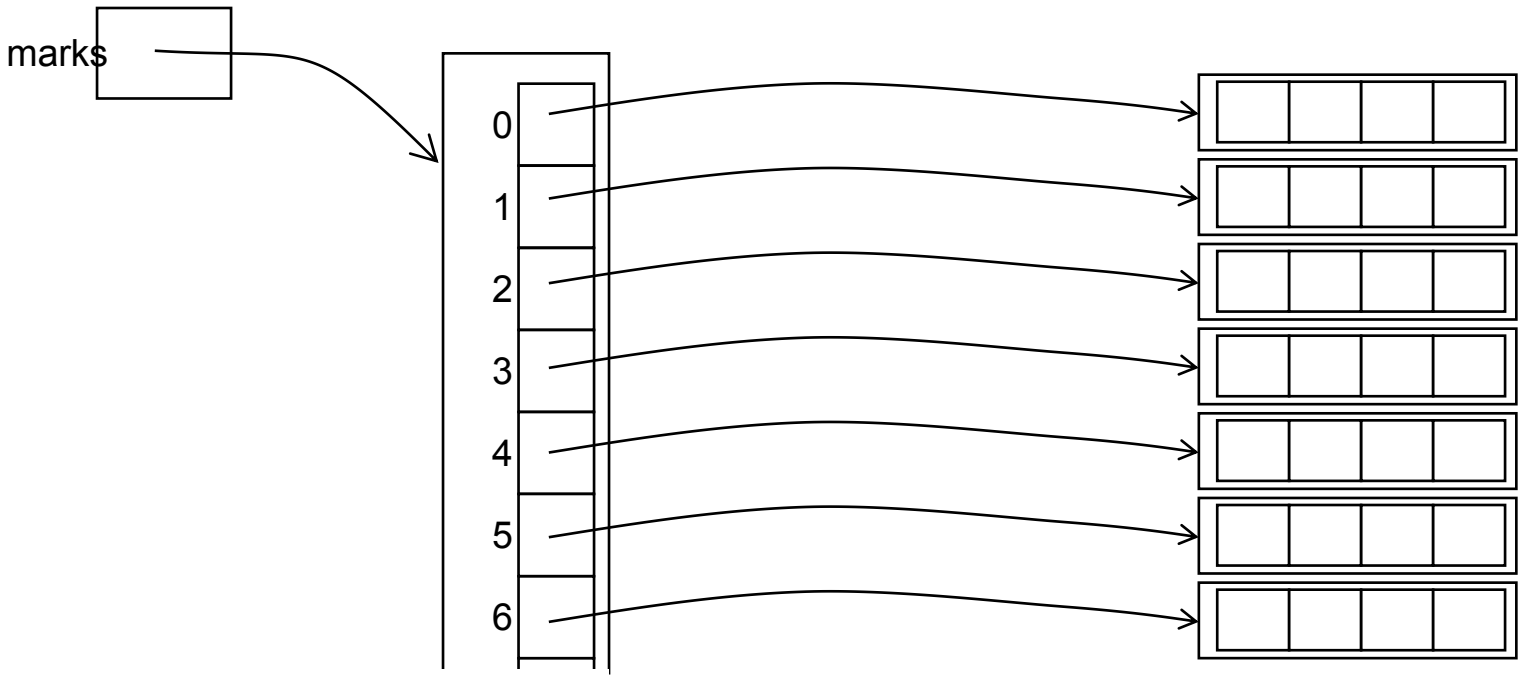
- Number of rows and columns in a 2D array?

```
int[ ] [ ] marks = new int[200] [4];
```

marks.length ➔ 200 (number of rows)

marks[row].length ➔ 4 (number of columns)

if the first index is the row!



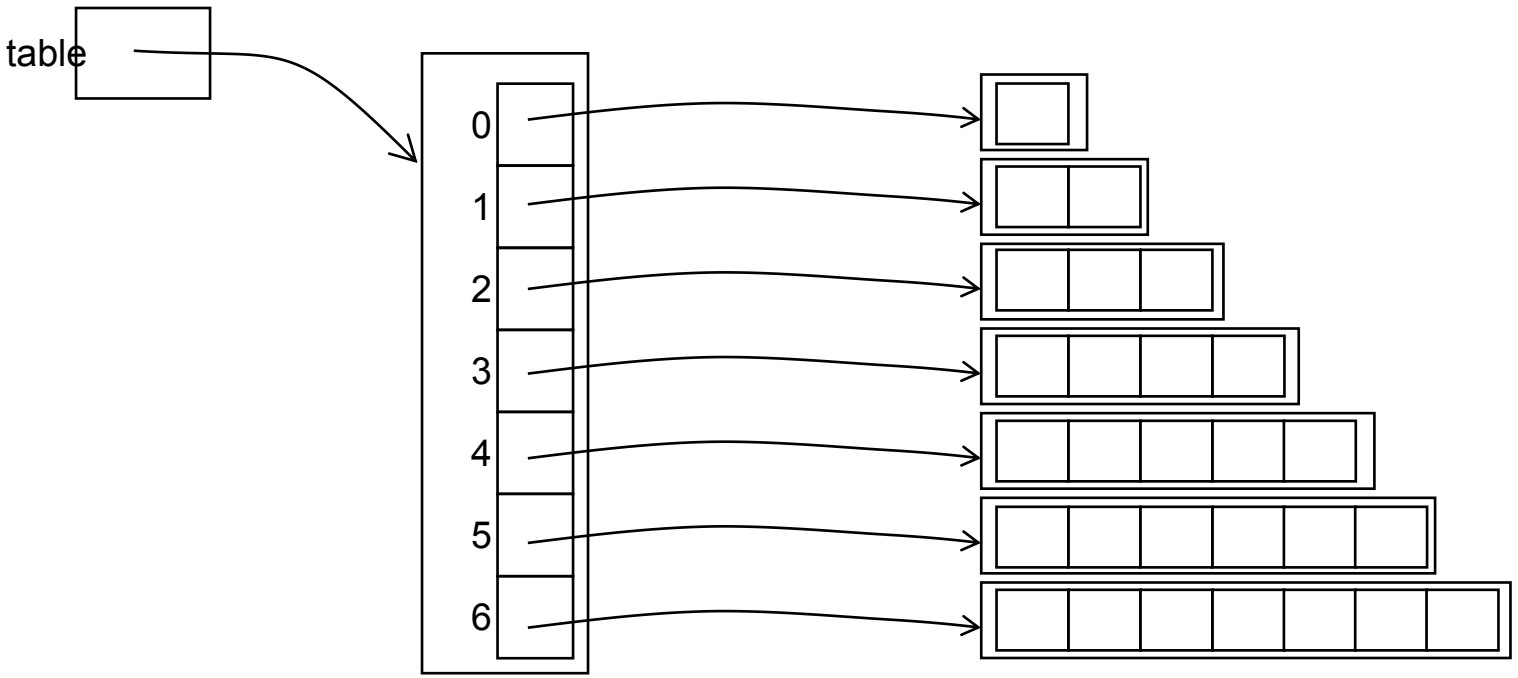
© Mohammad Nekooei and Peter Andreae

2D arrays

- Can have non-square arrays:

```
int [ ] [ ] table =  new int [7] [ ];
for (int row = 0; row<7; row++){
      table[row] = new int [row+1];
}
```

We don't use these in COMP102!



© Mohammad Nekooei and Peter Andreae

Processing 2D arrays

- Typically use nested **for** loops to process each item

```
public void printTable( String[ ][ ] grades){
    for (int row=0; row< grades.length; row++){
        for (int col=0; col< grades[row].length; col++){
            UI.printf(" %-2s ", grades[row][col]);
        }
        UI.println();
    }
}
```

'-' flag means left justified

```
A+   B-   A-   B
B+   C    A    B-
A    D    A+   A
A-   B+   B+   B
A    A-   C+   C+
```

```
public void printTable( String[ ][ ] grades){
    for (String[ ] row  : grades){
        for (String grade : row){
            UI.printf(" %-2s ", grade);
        }
        UI.println();
    }
}
```

If not modifying the array, can use foreach loops, but must be careful!

© Mohammad Nekooei and Peter Andreae

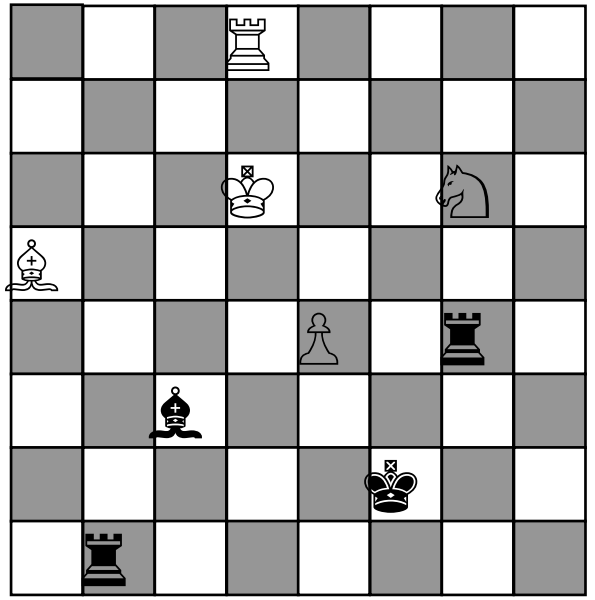
Drawing a 2D array

```
public void drawBoard(ChessPiece[ ][ ] board){
    int rows = board.length;
    int cols = board[0].length;
    for (int row=0; row<rows; row++){
        int y = TOP + SIZE*row;
        for (int col=0; col<cols; col++) {
            int x = LEFT + SIZE*col;
            UI.setColor( (row%2==col%2) ? Color.gray : Color.white);
            UI.fillRect(x, y, SIZE, SIZE);
            if (board[row][col] !=null) {
                board[row][col] .draw(x, y);
            }
        }
    }
    UI.setColor(Color.black);
    UI.drawRect(LEFT, TOP, SIZE * rows, SIZE * cols);
}
```

Shorthand for:

```
if (row%2==col%2) {
    UI.setColor(Color.gray);
}
else {
    UI.setColor(Color.white);
}
```

Make ChessPiece draw itself



© Mohammad Nekooei and Peter Andreae

Printing a table with headers

```
public void printTable(long[ ] IDs, String[ ][ ] grades){
    int rows = grades.length;
    int cols = grades[0].length;
    ---------------------------------------
    UI.print(" ID |");
    for (int col=0; col<cols; col++) {
        UI.printf(" A%d |", col); }
    UI.println();
    ---------------------------------------
    for (int col=-1; col<cols; col++)  {
        UI.print("----+"); }
    UI.println();
    ---------------------------------------
    for (int row=0; row<rows; row++){
        UI.printf("%4d|", IDs[row])
        for (int col=0; col<cols; col++) {
            UI.printf(" %-2s |", grades[row][col]); }
        UI.println();
    }
    ---------------------------------------
    for (int col=-1; col<cols; col++)  {
        UI.print("----+"); }
    UI.println();
}
```

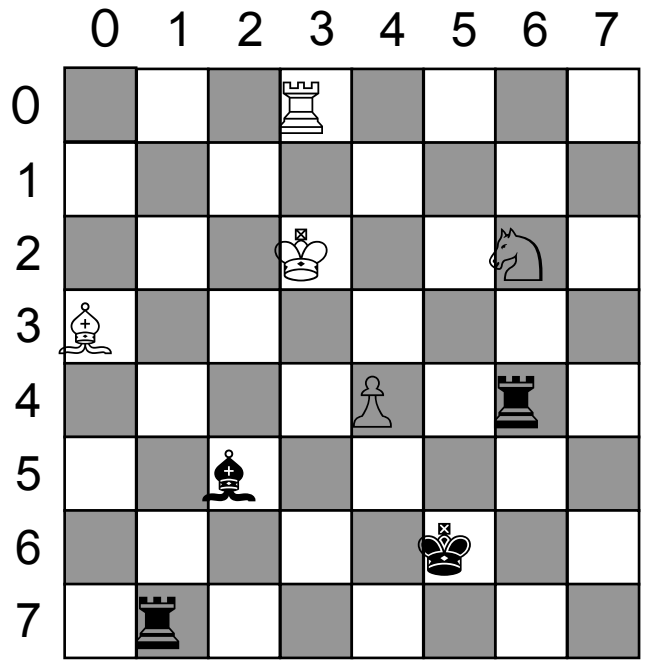
Assumes all rows same length

```
ID   | A1 | A2 | A3 | A4 |
----+----+----+----+----+
3012| A+ | B- | A- | B  |
3052| B+ | C  | A  | B- |
3029| A  | D  | A+ | A  |
3172| A- | B+ | B+ | B  |
3094| A  | A- | C+ | C+ |
----+----+----+----+----+
```

© Mohammad Nekooei and Peter Andreae

Moving value in a 2D array

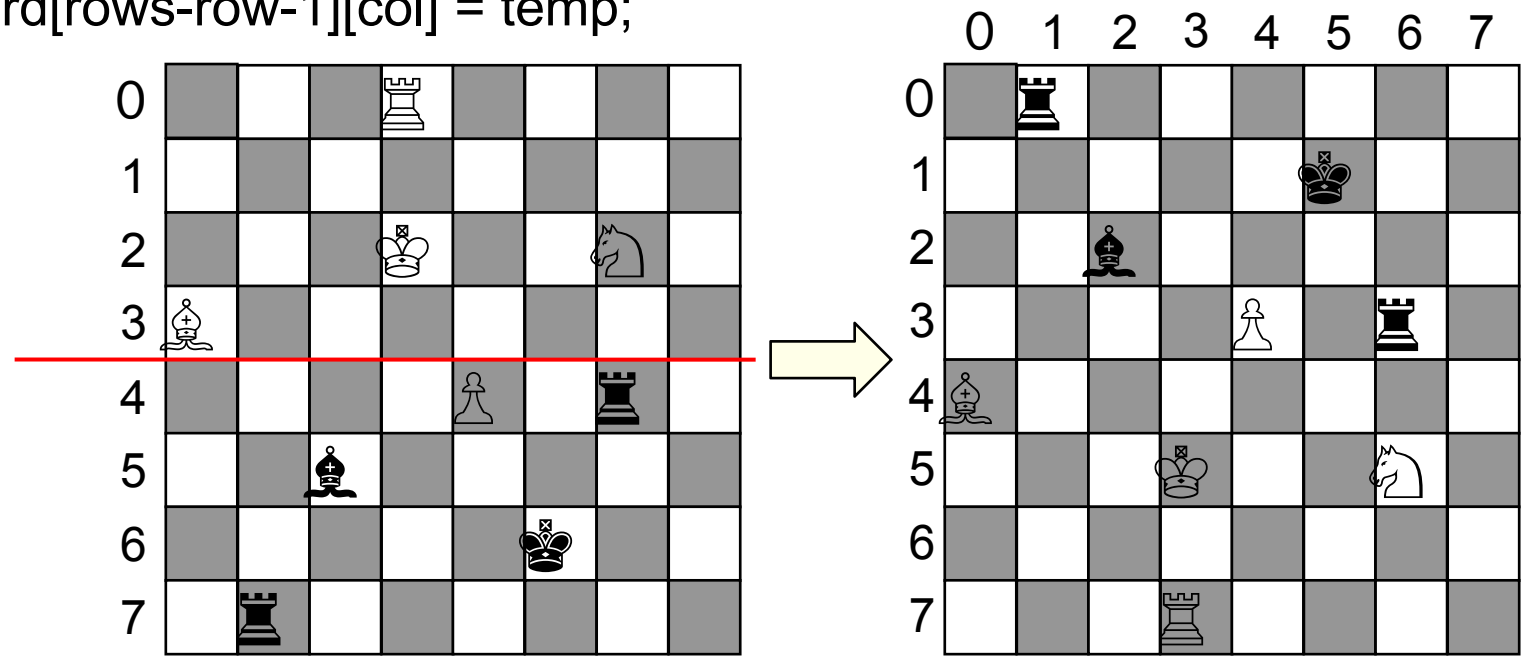
```
public void moveUp(ChessPiece[ ][ ] board, int row, int col){
    if ( row > 0  && row < board.length
        && col >= 0 && col  < board[row].length
        &&  board[row][col] != null
        &&  board[row-1][col] == null ) {
       board[row-1][col] = board[row][col];
       board[row][col] = null;
    }
}
```



© Mohammad Nekooei and Peter Andreae

Moving all values in a 2D array

```
public void flipBoard(ChessPiece[ ][ ] board){
    int rows = board.length;
    int cols = board[0].length;
    for (int row=0; row<rows/2; row++){
        for (int col=0; col<cols; col++) {
            ChessPiece temp = board[row][col];
            board[row][col] = board[rows-row-1][col];
            board[rows-row-1][col] = temp;
    }}
}
```



© Mohammad Nekooei and Peter Andreae

Rotating (cw) all values in a 2D array

```
public void rotateBoard(ChessPiece[ ][ ] board){
   int rows = board.length;
   int cols = board[0].length;
   ChessPiece[ ][ ] temp= new ChessPiece[rows][cols];
   for (int row=0; row<rows; row++){
      for (int col=0; col<cols; col++) {
          temp[row][col] = board[row][col];
   }}
   for (int row=0; row<rows; row++){
      for (int col=0; col<cols; col++) {
          int srow = cols-col-1;
          int scol = row;
          board[row][col] = temp[srow][scol];
   }}
}
```

Make a temporary copy of the source, to avoid "losing" values.

For each cell in the result, work out where its value comes from.