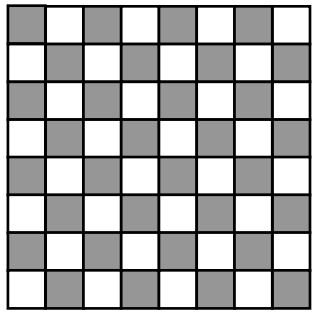
2D Data

- data in rows and columns

X	O	X
	O	



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

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

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

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.

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]~~

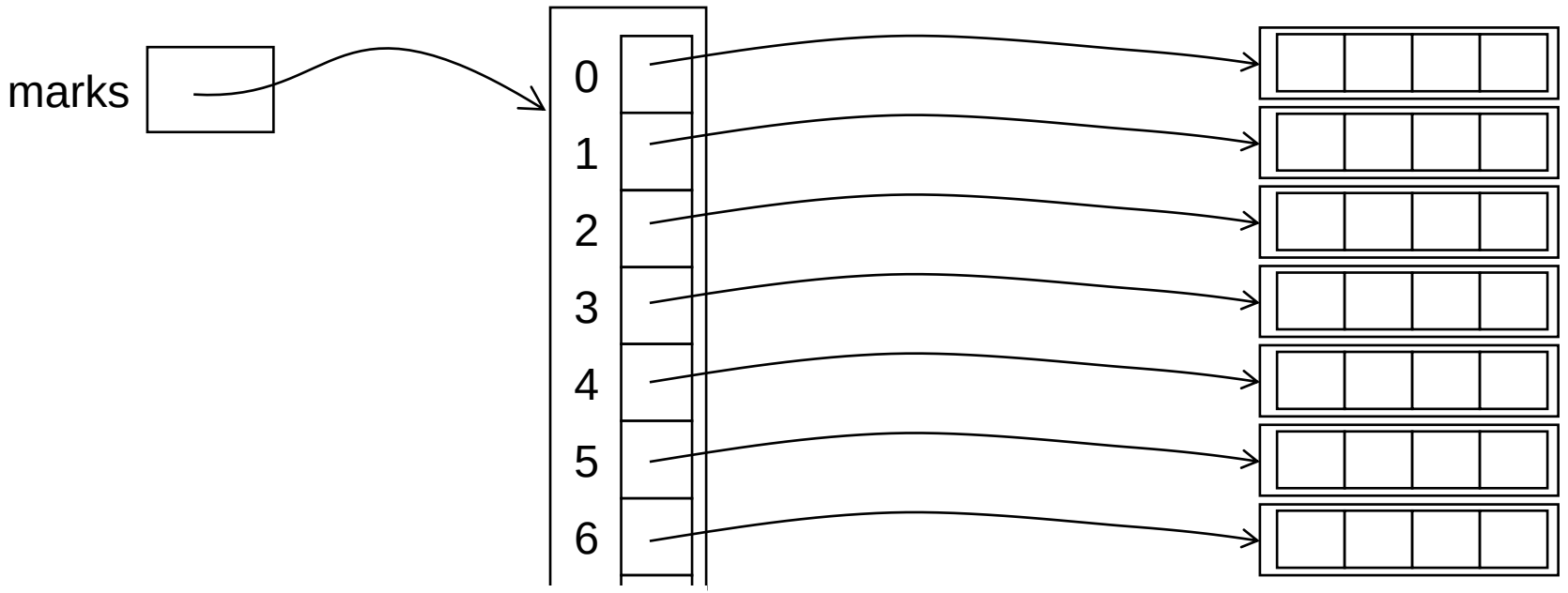
2D arrays

- 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];
}
```



2D arrays: length

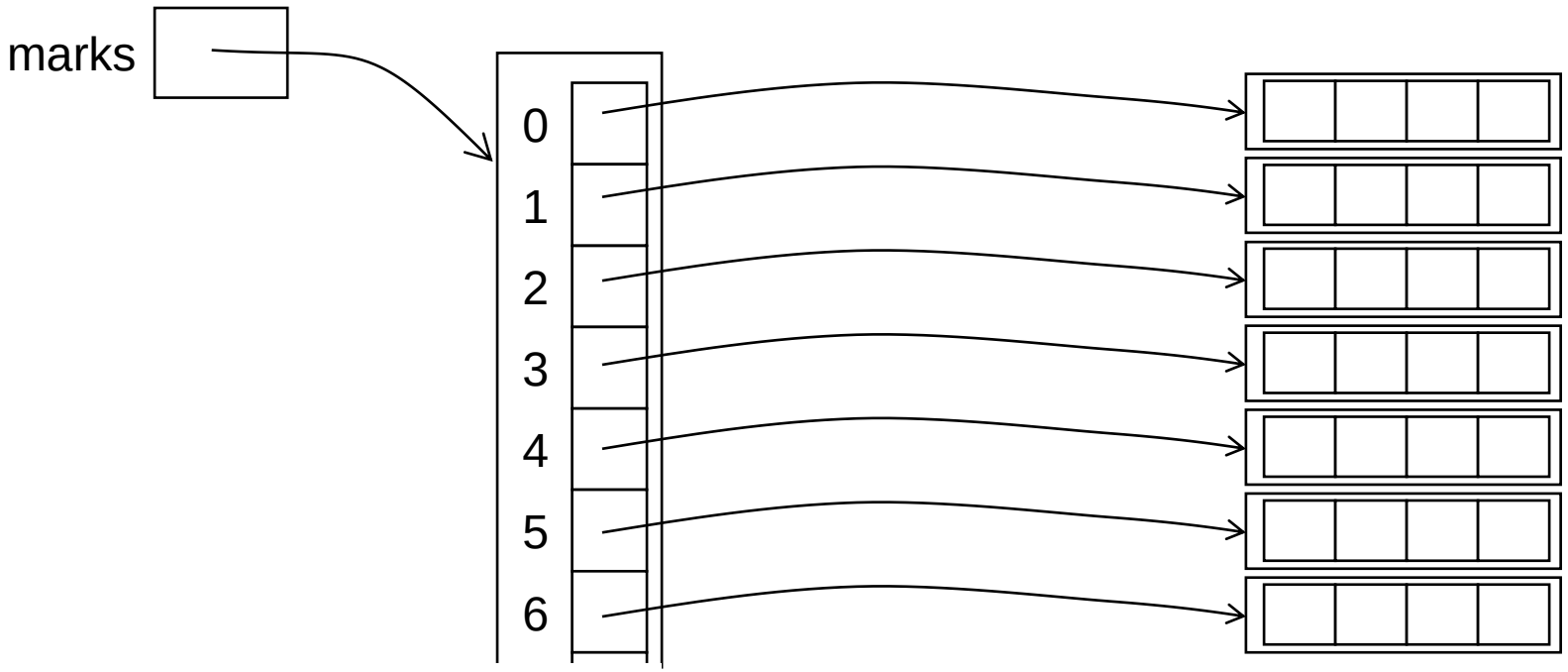
- Number of rows and columns in a 2D array?

```
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!

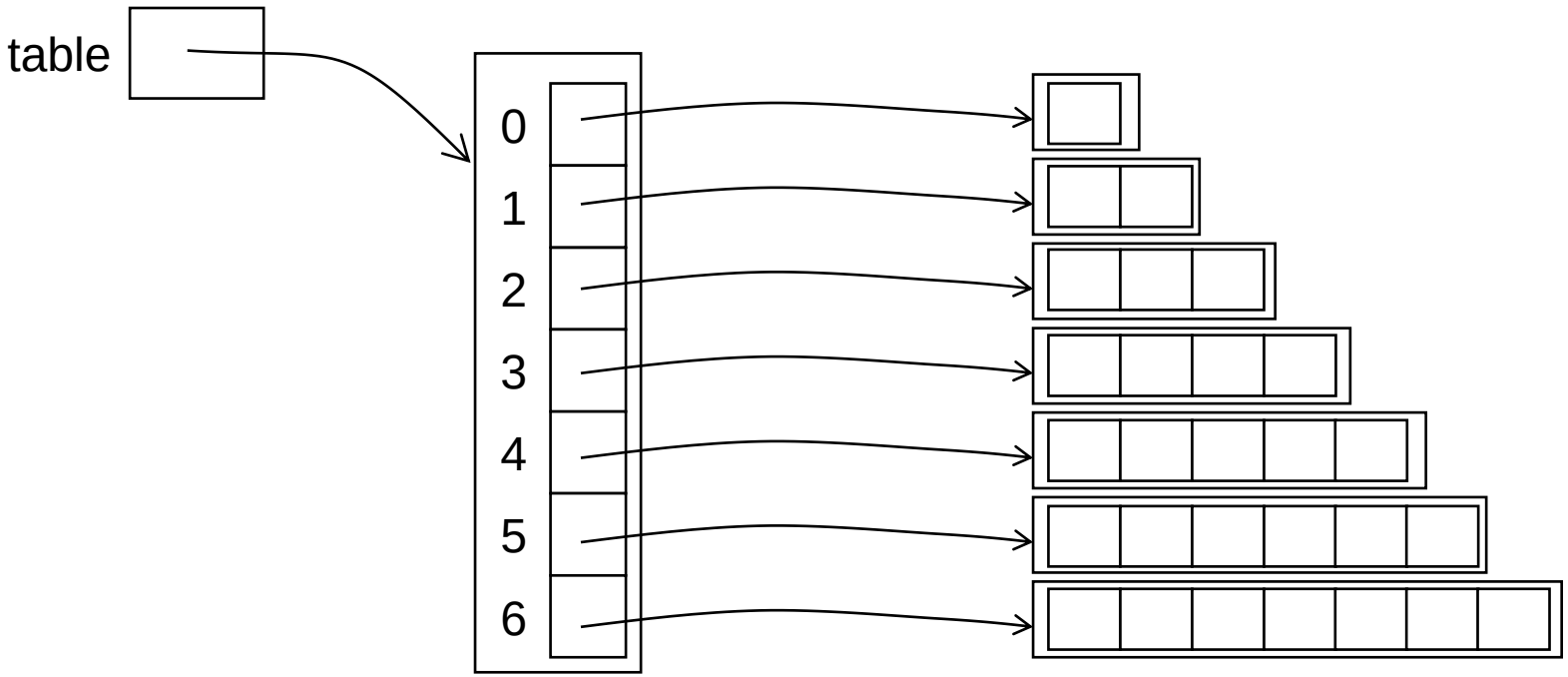


2D arrays

- 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!



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!

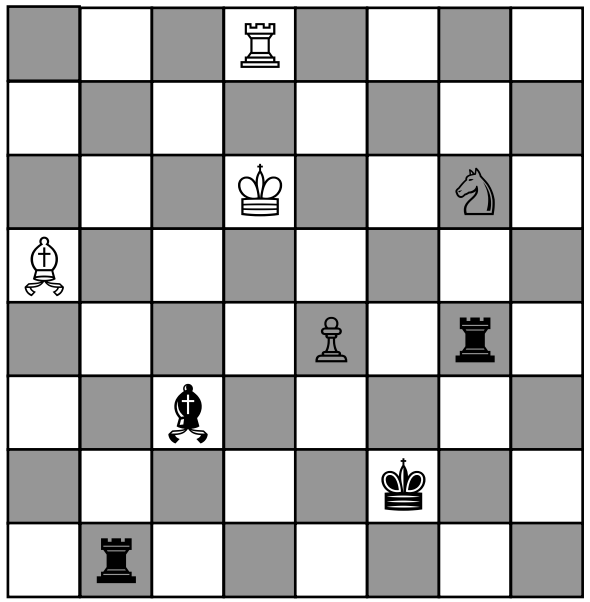
Drawing a 2D array

```
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:

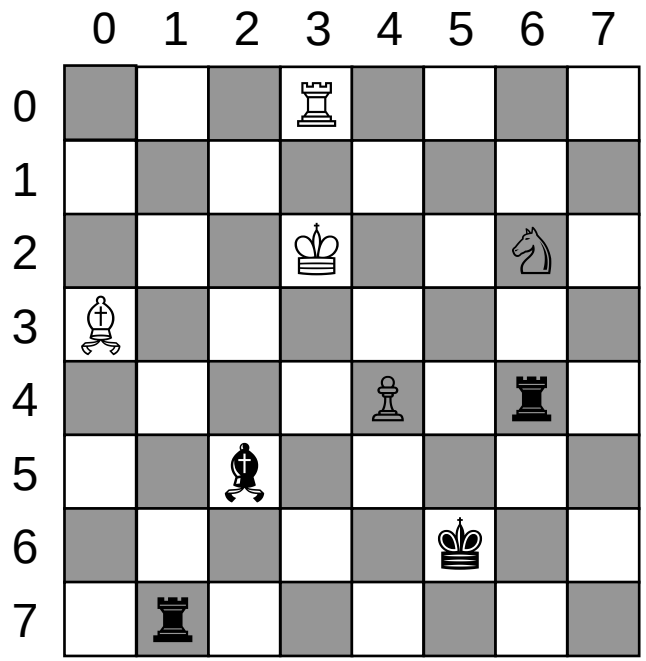
```
if (row%2==col%2) {
    UI.setColor(Color.gray);
}
else {
    UI.setColor(Color.white);
}
```

Make ChessPiece draw itself



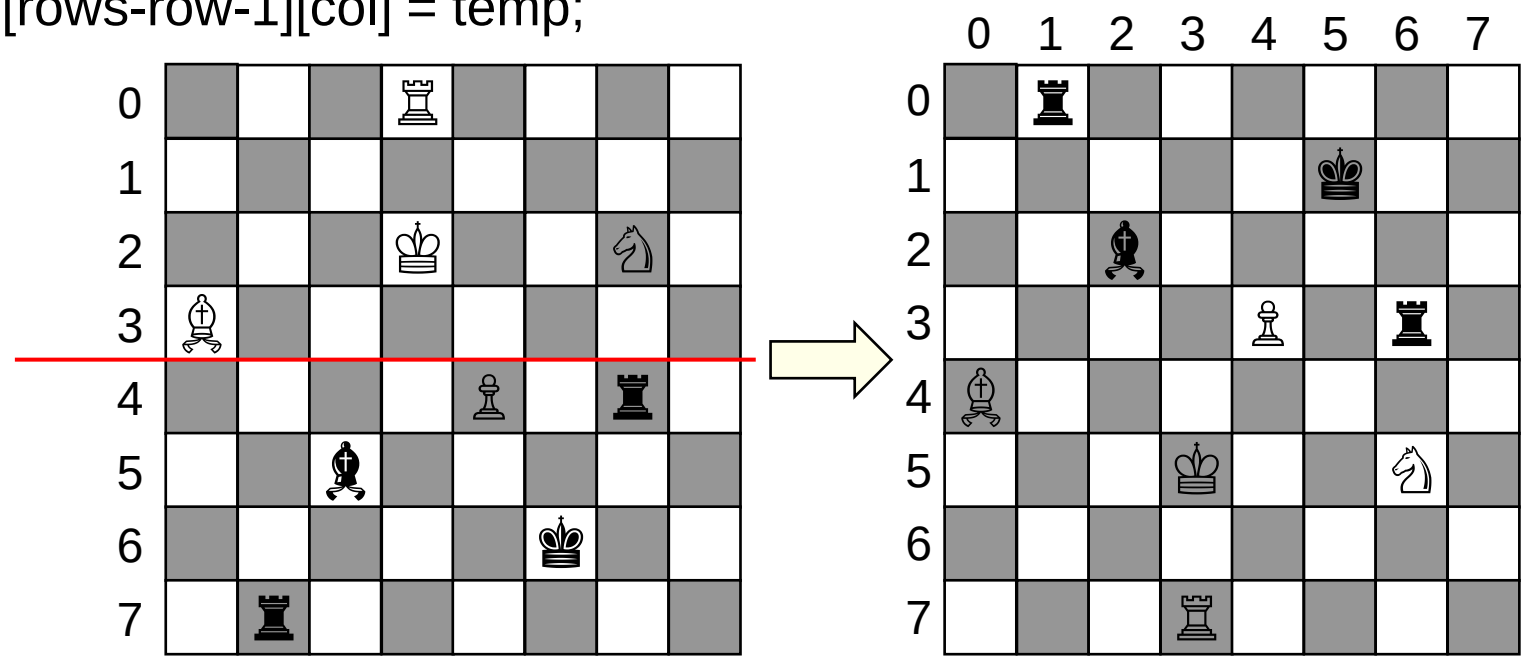
Moving value in a 2D array

```
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;
    }
}
```



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;
    }}
}
```

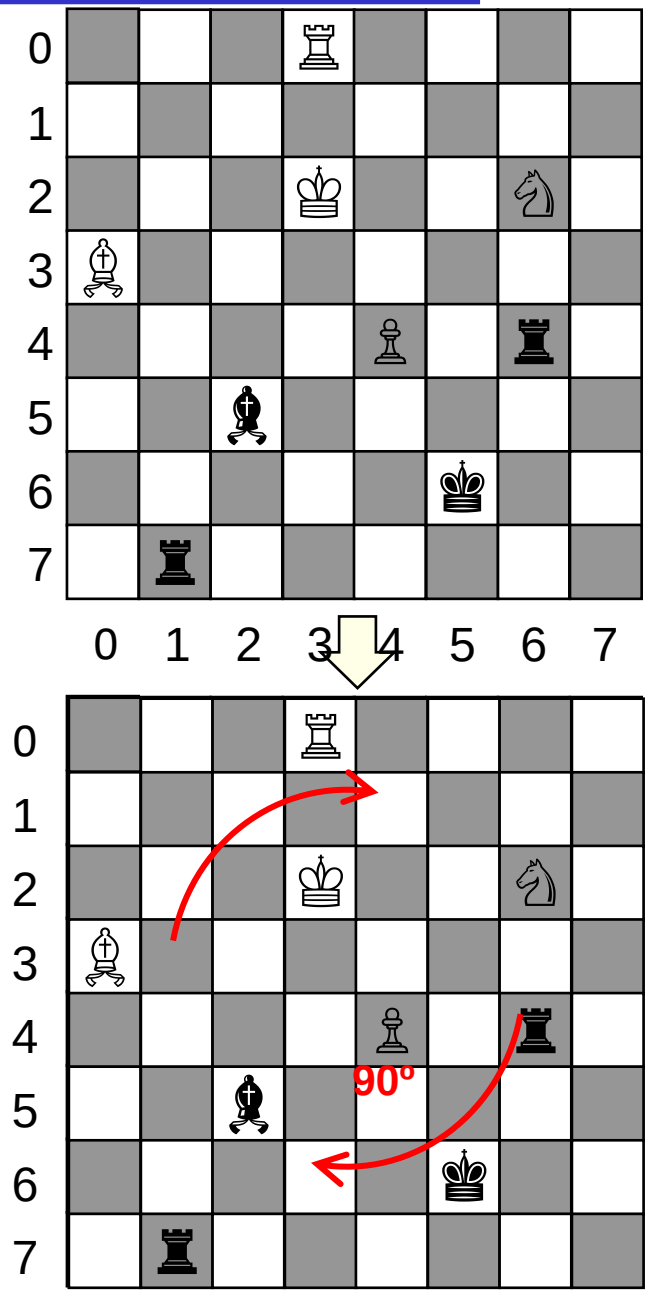


Rotating (cw) all values in a 2D array

```
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.



Processing multiple arrays

```
public double[ ][ ] matrixAdd(double[ ][ ] a, double[ ][ ] b){
    int rows = a.length;
    int cols = a[0].length;

    if (b.length!=rows || b[0].length!=cols) { return null;  }

    double[ ][ ] ans = new double[rows][cols];
    for (int row=0; row<rows; row++){
        for (int col=0; col<cols; col++){
            ans[row][col] = a[row][col] + b[row][col];
        }
    }
    return ans;
}
```

$$\begin{pmatrix} 2 & 4 & 2 \\ 4 & 5 & 4 \\ 2 & 4 & 2 \end{pmatrix} + \begin{pmatrix} 6 & 3 & 2 \\ 5 & 1 & 4 \\ 7 & 3 & 0 \end{pmatrix} = \begin{pmatrix} 8 & 7 & 4 \\ 9 & 6 & 8 \\ 9 & 7 & 2 \end{pmatrix}$$

Processing multiple arrays

```
public double[ ][ ] matrixMultiply(double[ ][ ] a, double[ ][ ] b){
    int rows = a.length;     int cols = b[0].length;

    if (cols != b.length) { return null;  }

    double[ ][ ] ans = new double[rows][cols];
    for (int row=0; row<rows; row++){
        for (int col=0; col<cols; col++){
            for (int i=0; i<b.length; i++){
                ans[row][col] = ans[row][col]+a[row][ i ] * b[ i ][col];
            }
        }
    }
    return ans;
}
```

$$\begin{pmatrix} 2 & 4 & 2 \\ 4 & 5 & 4 \\ 2 & 4 & 2 \end{pmatrix} * \begin{pmatrix} 6 & 3 & 2 \\ 5 & 1 & 4 \\ 7 & 3 & 0 \end{pmatrix} = \begin{pmatrix} 12+20+14 & 6+4+6 & 4+16+0 \\ 24+25+28 & 12+5+12 & 8+20+0 \\ 12+4+14 & 6+4+6 & 4+16+0 \end{pmatrix}$$

Saving a 2D array to a file

- Write the grade table to a file:

A
B
B+
A-
B
B+
B-
C+
A
A-
B-
B+

Three people with 4 assignments each
or
Four people with 3 assignments each
or
Six people with 2 assignments each

A	B	B+	A-
B	B+	B-	C+
A	A-	B-	B+

A	B	B+
A-	B	B+
B-	C+	A
A-	B-	B+

A	B
B+	A-
A	B+
B-	C+
A	A-
B-	B+

- Writing the dimensions of the array first will help.

Saving a 2D array to a file

- Writing the dimensions in the file helps:

```
public void saveTable( String[][] grades, String fileName){
    try{
        PrintStream file =new PrintStream(new File(fileName));
        int rows = grades.length;
        int cols = grades[0].length;
        file.println(rows + " " + cols);
        for (int row=0; row<rows; row++){
            for (int col=0; col<cols; col++){
               file.println(grades[row][col]);
            }
        }
    }catch(IOException e){UI.println("Table saving failed: "+e);}
}
```

Print number of rows and columns on first line of file.

Saving a 2D array to a file

- Alternate design:
  - assume file has been opened elsewhere and passed as argument
  - use "foreach" because not modifying the array.

```
public void saveTable( String[ ][ ] grades, PrintStream file){
    file.println(grades.length + " " + grades[0].length);
    for (String[ ] row : grades){
        for (String grd : row){
            file.println(grd);
        }
    }
}
```

  - Note, you could pass System.out to the method to make it print to the terminal window!
    - (useful for debugging)

Loading 2D array from a file

- Assume first two tokens of file are the dimensions:

```
public String[ ][ ] loadTable( ){
    try {
        Scanner sc = new Scanner(new File(UIFileChooser.open()));
        int rows =sc.nextInt();
        int cols = sc.nextInt();
        String[ ][ ] ans = new String[ rows ][ cols ];
        for (int row=0; row<rows; row++){
            for (int col=0; col<cols; col++){
                ans[row][col] = sc.next();
            }
        }
        return ans;
    } catch(IOException e){UI.out.println("Table loading failed: "+e);}
    return null;
}
```

Loading 2D array from a file

- Alternate, assuming
  - scanner is passed as argument
  - array is stored in a field

```
public void loadTable(Scanner sc ){
        this.dataArray = new String[ sc.nextInt() ] [ sc.nextInt() ];
        for (int row=0; row<this.dataArray.length; row++){
            for (int col=0; col<this.dataArray[row].length; col++){
                this.dataArray[row][col] = sc.next();
            }
        }
}
```

Another file format for 2D arrays

- Suppose the array has only a few entries and many nulls
  - ie, a "sparse" array

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1			T1									
2					lects							
3												
4												
5	lects								lects			
6						Study						
7												
8												
9												
10						Exam						
11		end T3					T2					
12											end Ex	Grad
13												
14												
15												
16												
17					Grad					Study		
18				Break								
19												

Better to save just the non-null entries

File format

- size (months, days)
- month day entry

[note: first and second index, not rows and columns!]

```
12  32
0  5  lects
1  11  end T3
2  1  T2
3  18  break
 :
 :
```

- To load:
  - read size and create calendar array
  - read month, day, entry, and assign entry to calendar[month][day]
- To save:
  - print size
  - for each non-null entry, print month, day, entry

Load sparse array

```
public void loadSparse(){
    try{
        Scanner sc = new Scanner( new File(UIFileChooser.open()));
        int months= sc.nextInt();
        int days = sc.nextInt();
        this.calendar = new String[months][days];
        while (sc.hasNext()){
            int month = sc.nextInt();
            int day = sc.nextInt()
            String entry = sc.next();
            this.calendar[month][dayl] = entry;
        }
        sc.close();
    }
    catch (IOException e){UI.println("Fail: " + e);}
}
```

Don't know how many entries there will be!

Save sparse array

```
public void saveSparse(){
    try{
        PrintStream ps = new PrintStream(new File(UIFileChooser.save()));
        ps.println(this.calendar.length+" "+ this.calendar[0].length);
        for (int month = 0 ; month < this.calendar.length; month++){
            for (int day = 0 ; day< this.calendar[month].length; day++){
                if (this.calendar[month][dayl] != null){
                    ps.println(month+" "+day+" "+this.calendar[month][dayl]);
                }
            }
        }
        sc.close();
    }
    catch (IOException e){UI.println("Fail: " + e);}
}
```