FACULTY OF INFORMATION TECHNOLOGY

DATA STRUCTURES (CTDL)

Semester 1, 2021/2022

2-Dimensional Arrays

	Column 1	Column 2	Column 3	Column 4
Row 1	`a[0][0]`	`a[0][1]`	`a[0][2]`	`a[0][3]`
Row 2	`a[1][0]`	`a[1][1]`	`a[1][2]`	`a[1][3]`
Row 3	`a[2][0]`	`a[2][1]`	`a[2][2]`	`a[2][3]`

The Matrix

2-D Arrays: “Matrices”

Two Dimensional Arrays

- 1-D array: store a list of values.
- Many applications require us to store a table of values.

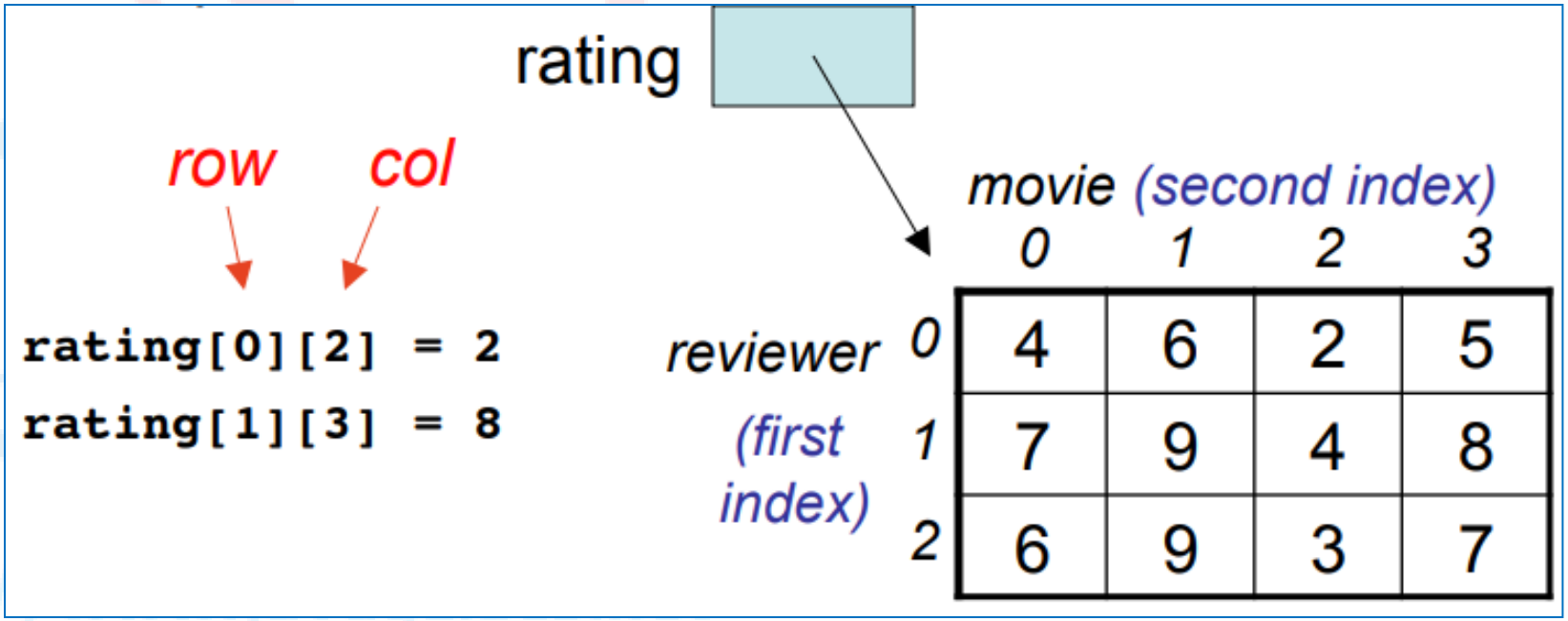
	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Student 1	7.2	8.5	7.9	6.7	7.1
Student 2	7.6	8.9	8.1	9.0	8.0
Student 3	8.2	9.0	7.6	9.2	9.2
Student 4	8.9	8.7	8.5	7.4	8.5

Two Dimensional Arrays

- The table contains a total of 20 values, five in each line
- The table can be regarded as a consisting of four rows
- Java allows us to define such tables of items by using two-dimensional arrays
- Each element in the 2D array must by the same type,
  - either a primitive type or object type.

Two Dimensional Arrays

- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example:



Declaring 2-D Arrays

- Almost the same syntax as decraring a single dimensional array
- General form:

  Type[][] array_name;

  Type array_name[][];

- Examples:
  - int[][] marks;
  - float [][] sales;

Initializing 2-D arrays

- **Shortcut syntax:**

```
int[][] list = { {1,2,3},
                 {4,5,6} };
```

- **New operator:**

```
int[][] list = new int[row][col];
```

- **Skip second dimension:**

```
int[][] arr = new int[3][];
arr[0] = new int[4];
arr[1] = new int[4];
arr[2] = new int[4];
```

Initializing 2-D arrays (cont.)

- Assign values for each cell:

```
arr[0][0]=1;
arr[0][1]=2;
...
```

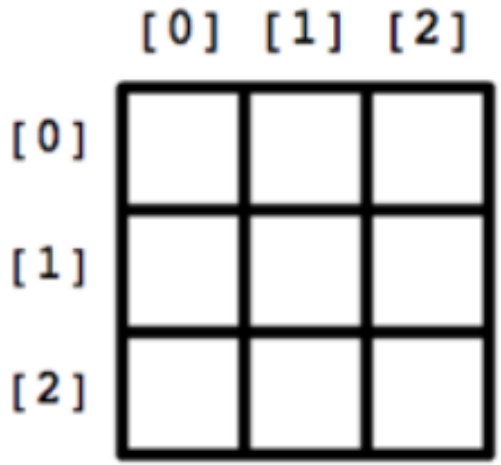
- Jagged array:

```
int[][] arr = new int[3][];
arr[0] = new int[1];
arr[1] = new int[2];
arr[2] = new int[3];
```

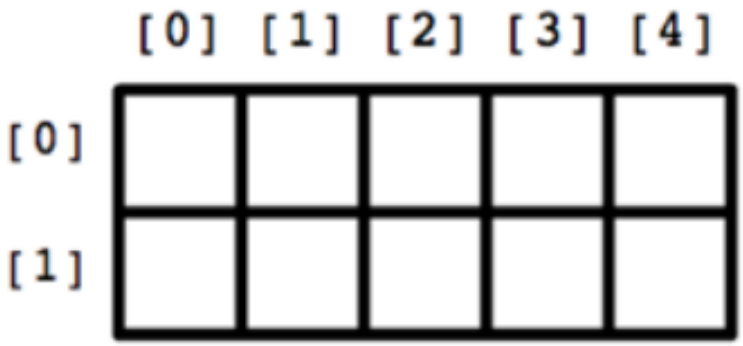
Example

- Initializing 2D array:

```
int[][] a = new int[3][3]
```




```
int[][] a = new int[2][5]
```



What is What?

- int[][] mat = new int[10][12];
- // mat
  - is a reference to the whole 2d array
- // mat[0] or mat[r]
  - are references to a single row
- // mat[0][1] or mat[r][c]
  - are references to single elements
- How to refer to a **single column**?

- // no way to refer to a **single column**

Methods

- Compare 2-D arrays with Arrays.equals.

➔ Arrays.deepEquals

- Print a 2-D array:

➔Arrays.deepToString(arr);

- clone: create a copy version of an array.

➔ array.clone()

Accessing 2-D Arrays

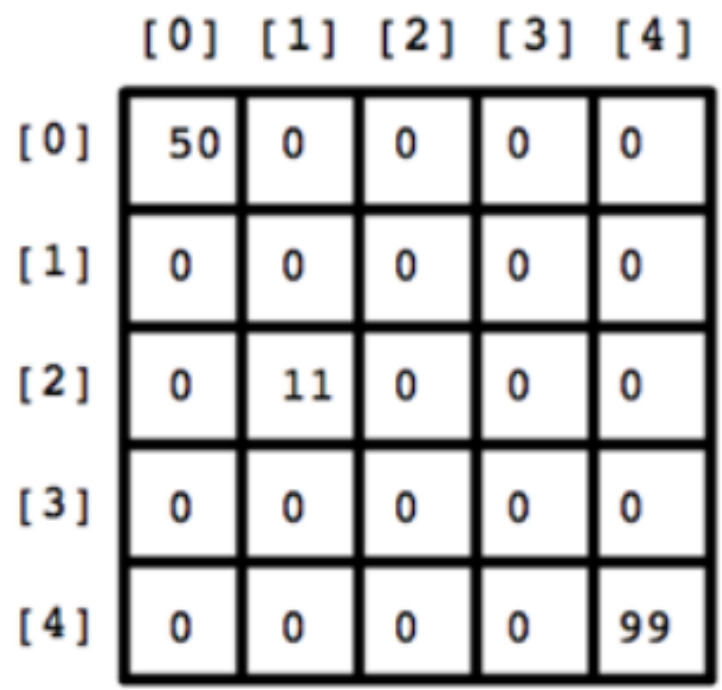
- Access by row and column indexes

```
int[][] list = new int[5][5];

list[0][0] = 50;

list[4][4] = 99;

list[2][1] = 11;
```

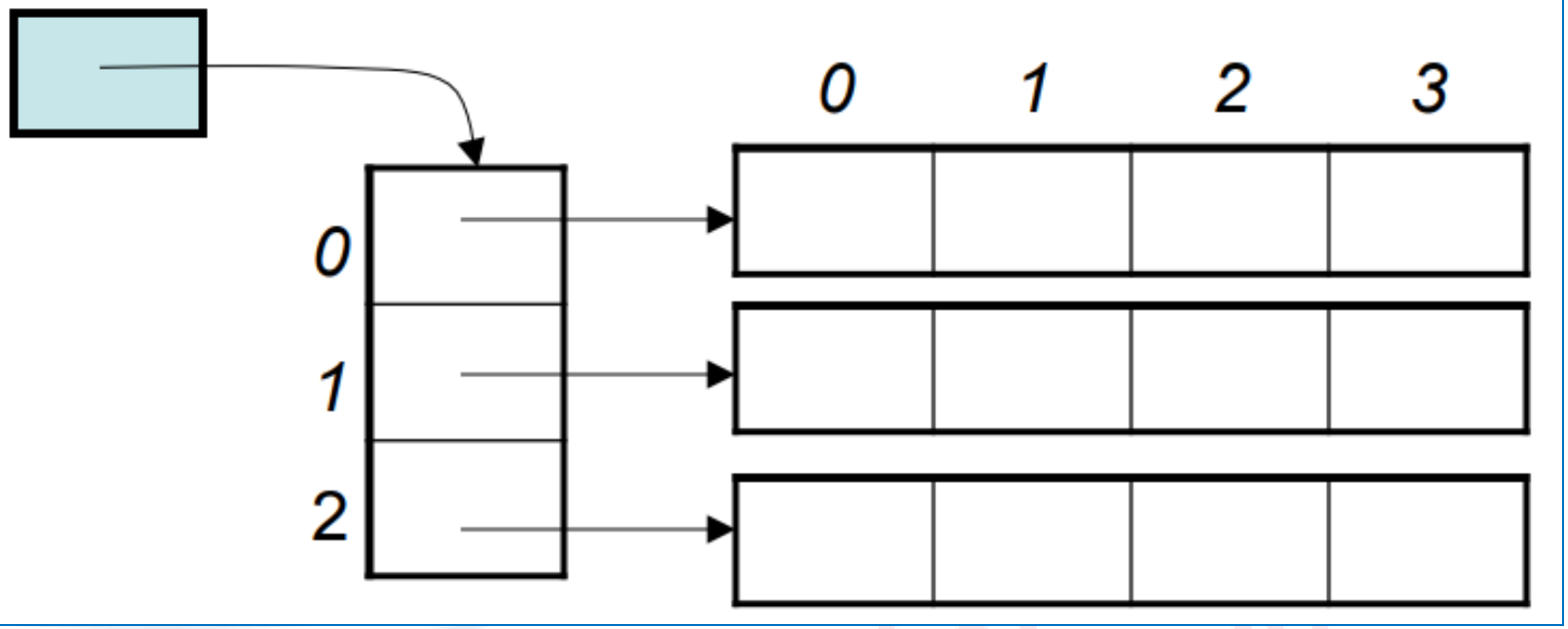


	[0]	[1]	[2]	[3]	[4]
[0]	50	0	0	0	0
[1]	0	0	0	0	0
[2]	0	11	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	99

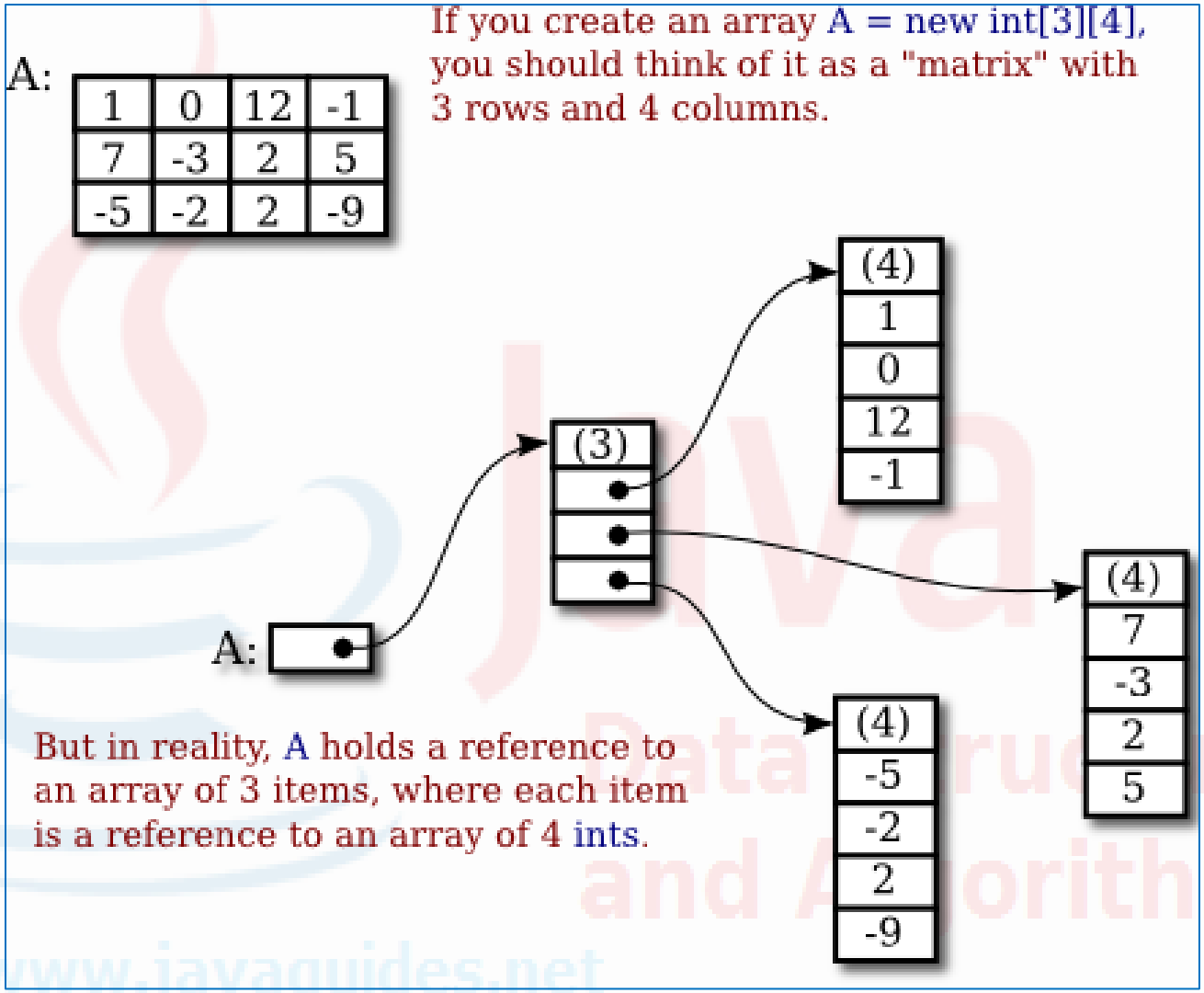
2-D arrays are arrays of arrays!

2-D Arrays Implementation

```
int[][] rating = new int[3][4];
```



Example



Size of 2-D Arrays

- Given

```
int[][] rating = new int[3][4];
```

- What is the value of **rating.length**?
- Answer: 3, the number of rows (first dimension)

- What is the value of **rating[0].length**?
- Answer: 4, the number of columns (second dimension)

Looping Through 2-D Arrays

- In order to iterate over all elements in a two dimensional array you will need to maintain two indexes – one for the row and one for the column
- This is usually done by setting up a nested for loop:

```
for (int row = 0; row < list.length; row++) {
        for (int col = 0; col < list[row].length; row++) {
                // statements
                // list[row][col] = …
        }
 }
```

Example 1

- Find the number of ratings above the value of the parameter.

```
public int countAbove(int[][] rating, int num) {
//TODO
}
```

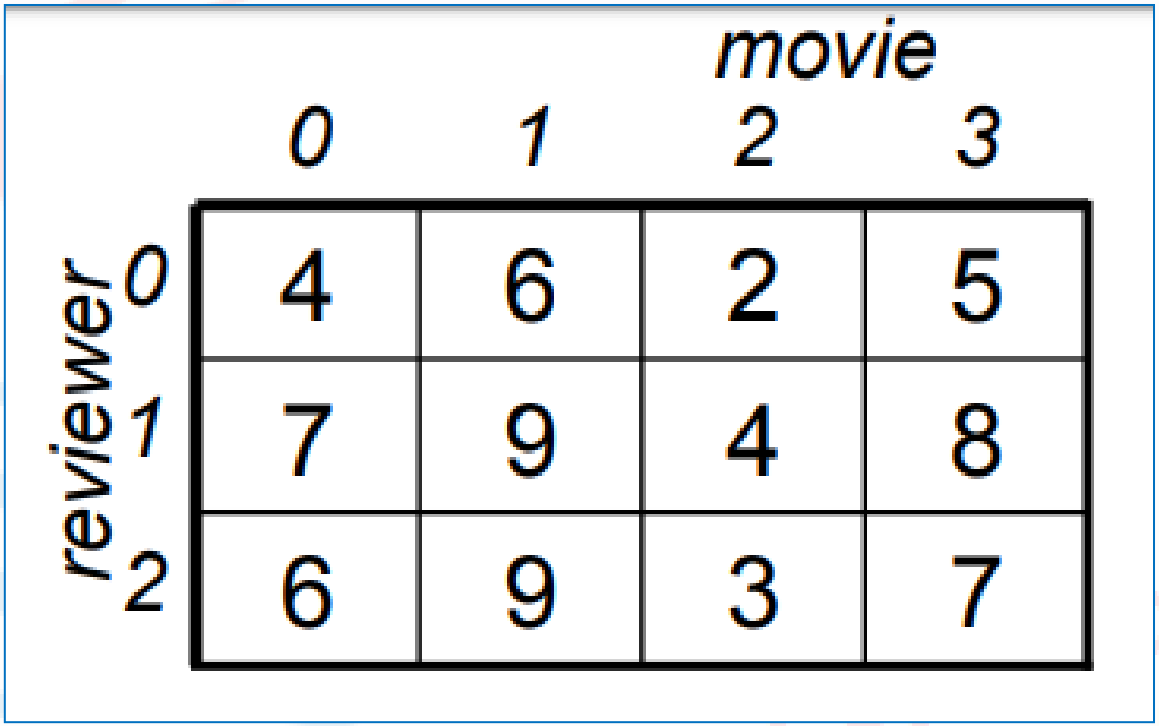
Example 1 (cont.)

- Find the number of ratings above the value of the parameter.

```
public int countAbove(int[][] rating, int num) {
   int count = 0;
   for (int row = 0; row <rating.length ; row++) {
         for (int col = 0; col < rating[row].length; col++) {
                  if (rating[row][col] > num)
                           count++;
         }
   }
   return count;
}
```

Example 2

- Print the average rating for the movie in column 3.



reviewer \ movie	0	1	2	3
0	4	6	2	5
1	7	9	4	8
2	6	9	3	7

Example 3

- How to sum up all elements in a 2-D array?

Jagged Array in Java

```
//declaring a 2D array with odd columns
int arr[][] = new int[3][];
arr[0] = new int[3];
arr[1] = new int[4];
arr[2] = new int[2];
//initializing a jagged array
int count = 0;
for (int i=0; i<arr.length; i++)
    for(int j=0; j<arr[i].length; j++)
        arr[i][j] = count++;
```

```
//printing the data of a jagged array
for (int i=0; i<arr.length; i++){
    for (int j=0; j<arr[i].length; j++){
        System.out.print(arr[i][j]+" ");
    }
    System.out.println();//new line
}
```

Output:

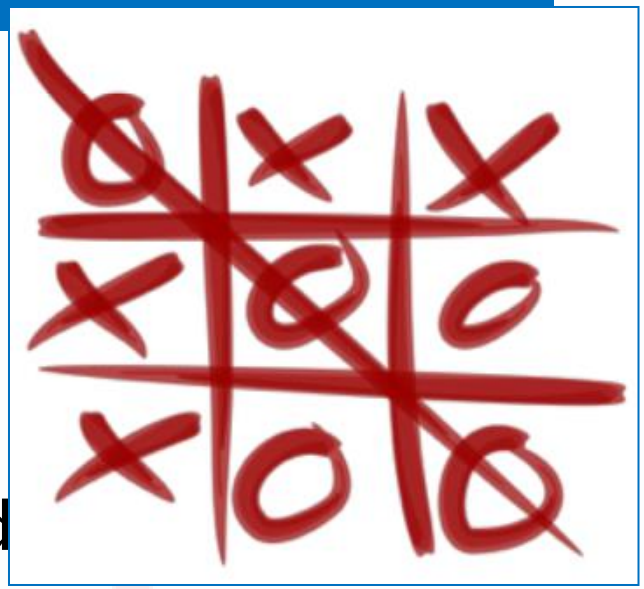
```
0 1 2
3 4 5 6
7 8
```

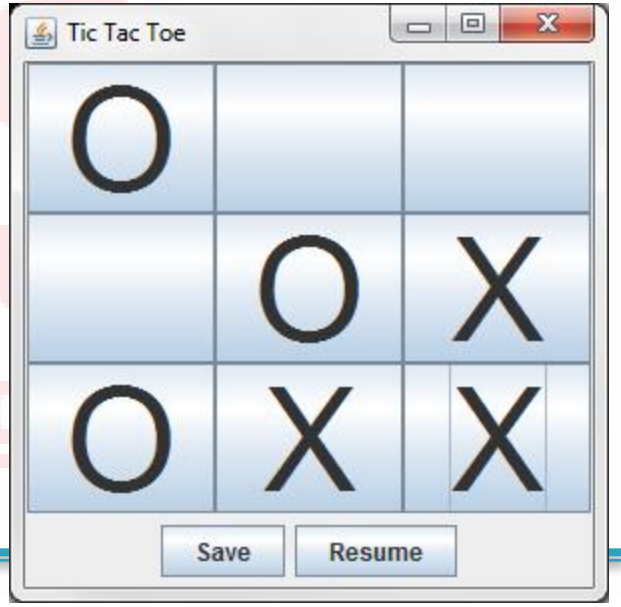
Other problems

- Write a method to find the **max/min value** in a 2-d array of integers
- Write a method that finds the sum of values in each column of a 2-d array of doubles
- Write a method to print out the elements of a 2d array of integers in row order.
  - row 0, then row 1, then row 2 ...
- Write a method to print out the elements of a 2d array of integers in column order
  - column 0, then column 1, then column 2 ...
- Addition, multiplication two 2-d arrrays.

Programming Exercise

- Write the game Tic Tac Toe
- Players begin with an empty board of 9 cells
- Players can elect to place their token (X or O) in a cell. If the cell is not taken the token is assigned to that cell.
- Players take turns until either
  - (a) all cells are occupied or
  - (b) one player's token is found to occupy all three cells along any row, column or diagonal

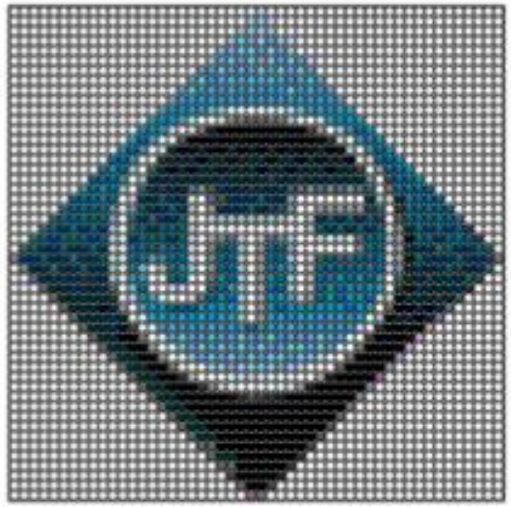




2-D arrays and Images

2-D Arrays and Images

- Images are just grids (2D arrays!) of pixels!
  - Pixels are just integer values from 0-255.
- One of the best examples of multidimensional arrays is a Java image, which is logically a twodimensional array of pixels.
- The GImage class allows you to convert the data for the image into a two-dimensional array of pixel values.
  - Once you have this array, you can work with the data to change the image.



How to use GImage

```
GImage img = new GImage("res/daisy.jpg");
int[][] pixels = img.getPixelArray();
int pixel = pixels[0][0]; // top-left pixel
```

GImage Pixel Methods

GImage img = new GImage("res/daisy.jpg");

Method name	Description
img.getPixelArray()	returns pixels as 2D array of ints, where each int in the array contains all 3 of Red, Green, and Blue merged into a single integer
img.setPixelArray(*array*);	updates pixels using the given 2D array of ints
GImage.createRGBPixel(*r*, *g*, *b*)	returns an int that merges the given amounts of red, green and blue (each 0-255)
GImage.getRed(*px*) GImage.getGreen(*px*) GImage.getBlue(*px*)	returns the redness, greenness, or blueness of the given pixel as an integer from 0-255

Image Manipulation

- Executing the following steps:
  - 1. **Read an existing image** from a file into a GImage object.
  - 2. Call **getPixelArray** to get the pixels.
  - 3. Write the code to **manipulate the pixel values** in the array.
  - 4. Call the **GImage constructor to create a new image.**

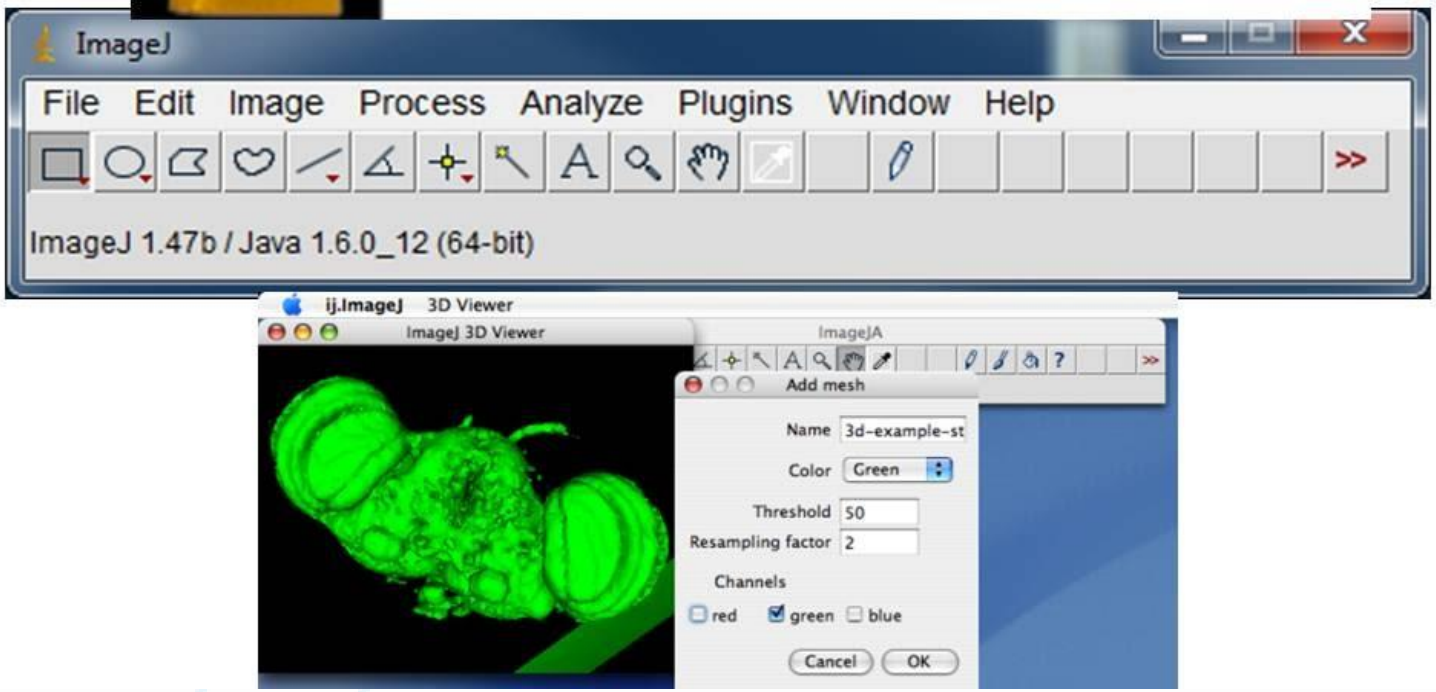
Some selected problems

- Modifying Pixels of an image
- Changing Image Size
- Brighten, Grayscale
- Flip Vertical/Horizontal
- …

ImageJ API

ImageJ

Image Processing & Analysis in Java



FACULTY OF INFORMATION TECHNOLOGY

Java
Data Structures