COMP 110-003
Introduction to Programming
Multidimensional Arrays

April 04, 2013

Haohan Li
TR 11:00 – 12:15, SN 011
Spring 2013

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

2D Arrays

- Arrays having more than one index are often useful
  - Tables
  - Grids
  - Board games

	0: Open	1: High	2: Low	3: Close
0: Apple Inc.	99.24	99.85	95.72	98.24
1: Walt Disney Co.	21.55	24.20	21.41	23.36
2: Google Inc.	333.12	341.15	325.33	331.14
3: Microsoft Corp.	21.32	21.54	21.00	21.50

Declaring and Creating 2D Arrays

- Two pairs of square brackets means 2D
  - **int[][] table = new int[3][4];**
- or
  - **int[][] table;**
  - **table = new int[3][4];**

Declaring and Creating 2D Arrays

- Array (or 1D array) gives you a list of variables
  - int[] score = new int[5] gives you score[0], score[1], … , score[5]
- 2D array gives you a table of variables
  - int[][] table = new int[3][4];

table[0][0]	**table[0][1]**	**table[0][2]**	**table[0][3]**
table[1][0]	**table[1][1]**	**table[1][2]**	**table[1][3]**
table[2][0]	**table[2][1]**	**table[2][2]**	**table[2][3]**

Using a 2D Array

- We use a loop to access 1D arrays

```
for (int i = 0; i < 5; i++) {
   scores[i] = keyboard.nextInt();
   scoreSum += scores[i];
}
```


Using a 2D Array

- We use nested loops for 2D arrays

```
int[][] table = new int[4][3];
for (int i = 0; i < 4; i++) {
   for (int j = 0; j < 3; j++) {
      table[i][j] = i * 3 + j;
      System.out.println(table[i][j]);
   }
}
```

Multidimensional Arrays

- You can have more than two dimensions
  - `int[][][] cube = new int[4][3][4];`
- Use more nested loops to access all elements
  - for (int i…)
    - for (int j…)
      - for (int k…)

Length of Multidimensional Arrays

```
public void print2DArray(int[][] arr) {
    for (int row = 0; row < arr.length; row++) {
        for (int column = 0; column < arr[row].length; column++) {
            System.out.print(arr[row][column] + " ");
        }
        System.out.println();
    }
}
```


Length of Multidimensional Arrays

```
public void print3DArray(int[][][] arr) {
    for (int page = 0; page < arr.length; page++) {
        for (int row = 0; row < arr[0].length; row++) {
            for (int column = 0; column < arr[0][0].length; column++) {
                System.out.print(arr[page][row][column] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }
}
```

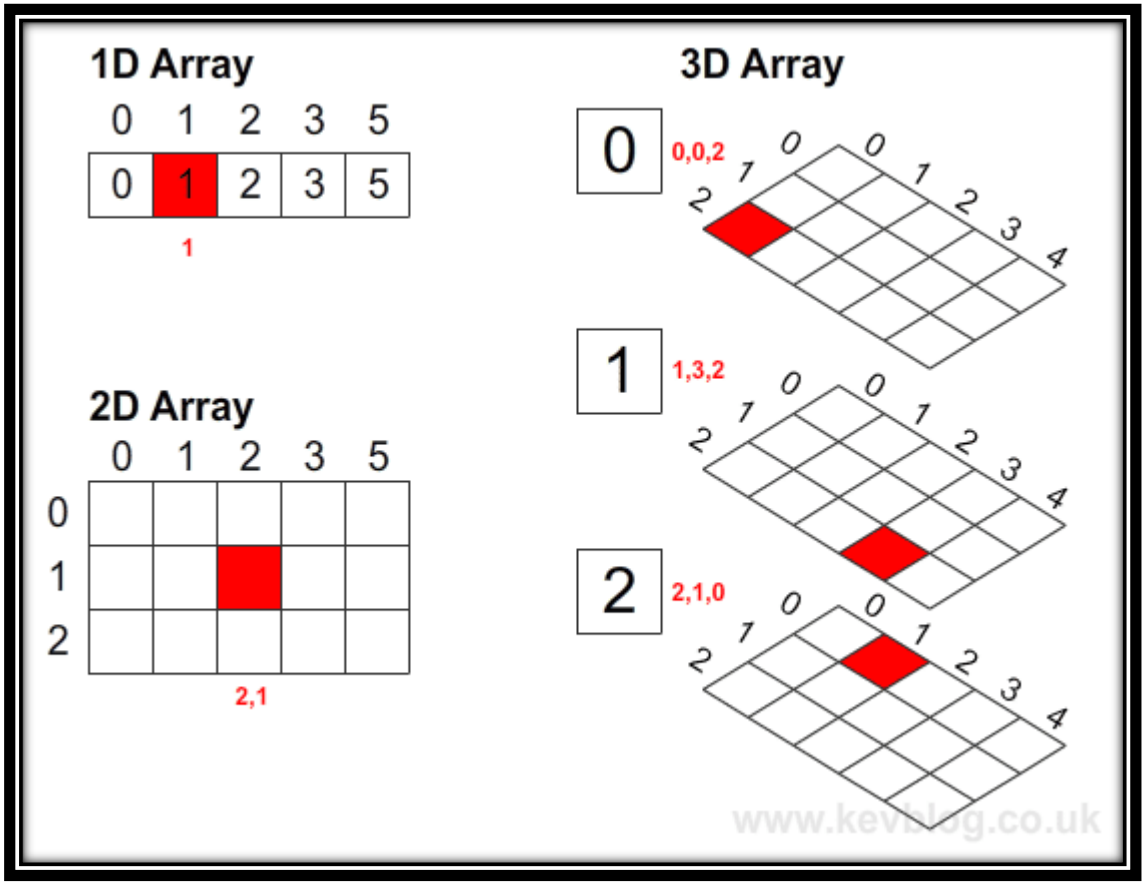
Length of a 2D Array

- **int[][] table = new int[4][3];**
- **table.length** is the number of rows, or the integer in the **first pair** of brackets (4)
- **table[i].length** or **table[0].length** is the number of columns, or the integer in the **second pair** of brackets (3)

Why? Array of Array!

- **Base_Type[] Array_Name = new Base_Type[Length];**
- **int[] scores = new int[5];**
  - scores is a one-dimensional array
  - base type is **int**
- **int[][] table = new int[4][3];**
  - table is still in fact a one-dimensional array
  - base type is **int[]**
- Actually, the syntax rule is not strictly followed here. Just try to understand the basic idea.

Dimensions of Arrays



Array as a Class Type Variable

```
public static void changeArray(int[] arr)
{
    int[] newArray = new int[arr.length];
    newArray[0] = 12;
    arr = newArray;
}

public static void main(String[] args)
{
    int[] arr = { 3, 6, 15 };
    changeArray(arr);
    for (int i = 0; i < arr.length; i++)
    {
        System.out.println(arr[i]);
    }
}
```

Output:
3
6
15

The parameter is local to changeArray, Only makes it point to some other address

Array as a Class Type Variable

```
public static void changeArray(int[] arr)
{
    arr[0] = 12;
}

public static void main(String[] args)
{
    int[] arr = { 3, 6, 15 };
    changeArray(arr);
    for (int i = 0; i < arr.length; i++)
    {
        System.out.println(arr[i]);
    }
}
```

Output:
12
6
15

The parameter is local to changeArray, but it contains the *address* of the array passed in, so we can change its elements

Array as a Class Type Variable

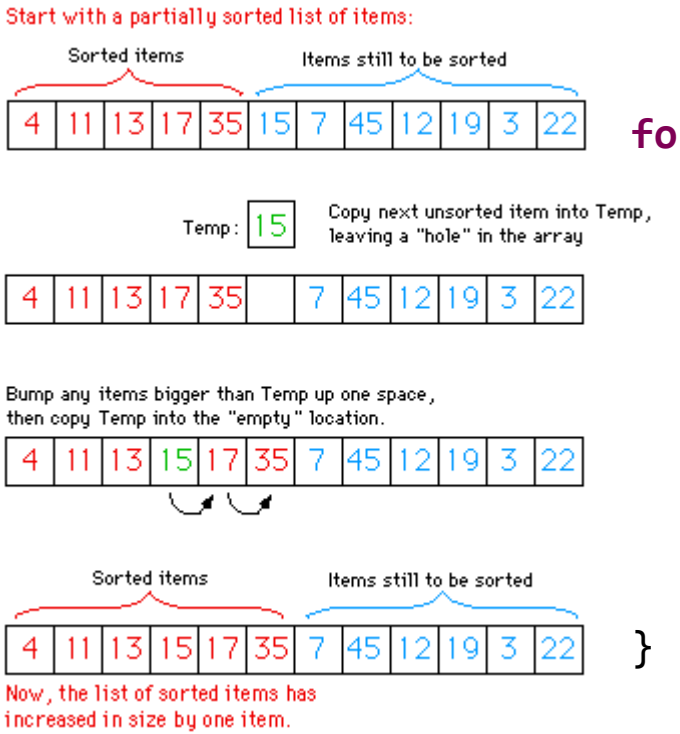
```
public static void changeArray(int[] arr)
{
    arr[0] = 12;
}

public static void main(String[] args)
{
    int[] arr = { 3, 6, 15 };
    int[] newArray = arr;
    changeArray(newArray);
    for (int i = 0; i < arr.length; i++)
    {
        System.out.println(arr[i]);
    }
}
```

Output:
12
6
15

arr and *newArray* both contain the same address, and therefore refer to the same data in memory

Insertion Sort




```
for (int index = 1; index < num.length; index++) {
    int temp = num[index];
    for (int i = index - 1; i >= -1; i--) {
        if (i >= 0 && num[i] > temp) {
            num[i + 1] = num[i];
        } else {
            num[i + 1] = temp;
            break;
        }
    }
}
```

Random Number: Try and Test

- What if you want to find a random prime number?
  - Generate a number and test if it is prime

```
public static void main(String[] args) {
    int range = 1000000;
    Random generator = new Random();
    int p = generator.nextInt(range) + 2;
    while (!isPrime(p))
        p = generator.nextInt(range) + 2;
    System.out.println("A random prime number is " + p);
}
private static boolean isPrime(int N) {
    for (int i = 2; i <= Math.sqrt(N); i++) {
        if (N % i == 0)
            return false;
    }
    return true;
}
```

TicTacToe.java

- Start with these methods:
  - setMouseSquare()
  - getIconDisplayXLocation()
  - buildBoard()
  - legalSquare()
  - setMovePosition()
  - **gameDraw()**
- After you finish these methods, your game will work and end (when the board is full)

TicTacToe.java

- GameEnd() and markWinningLine() are hard
  - Think about some helper methods
  - checkRow()
  - checkColumn()
  - checkDiagonal()
  - …