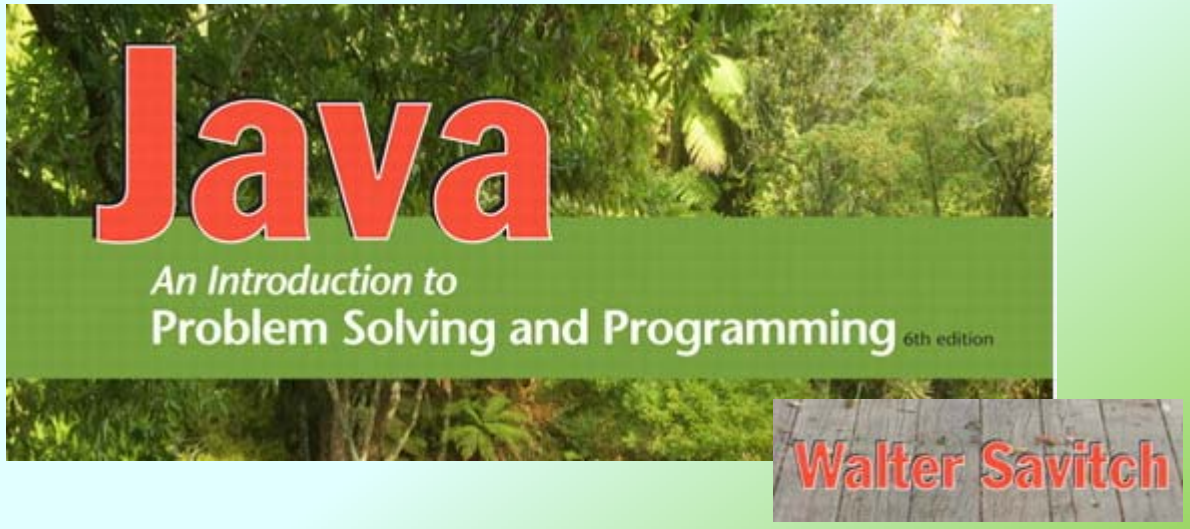


EECS168 Exam 4 Review

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709

Exam 4

- Time: 2pm-2:50pm Wednesday Nov 28
- Closed book, closed notes.
- Calculators or other electronic devices are not permitted or required.

- If you are unable to attend an exam for any reason, no make-up exam will be given. That exam will be dropped from your grade. If a second exam is missed, a make-up exam will only be granted under extenuating circumstances, with prior permission from the instructor.

Exam 4

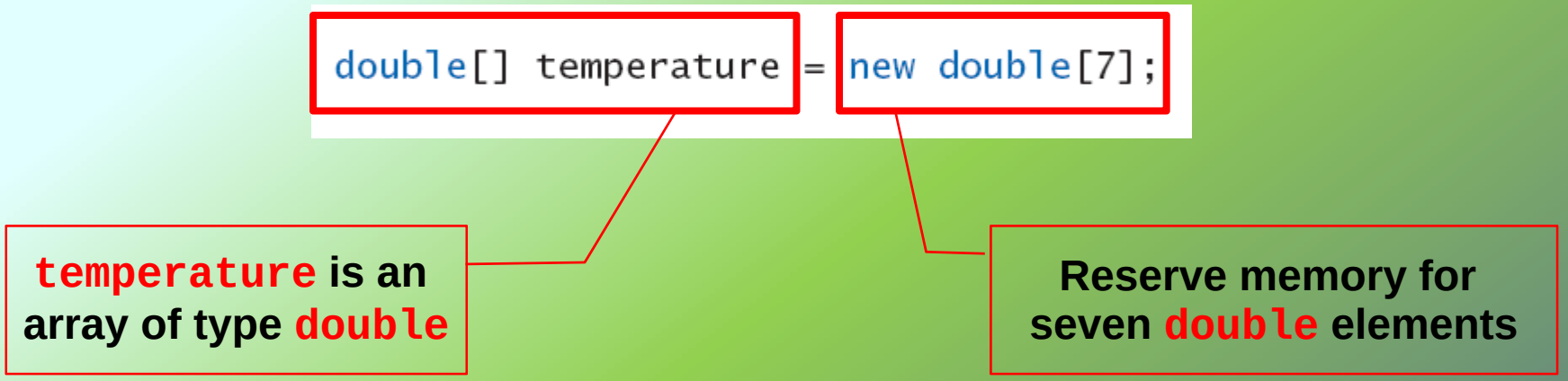
- Exam 4 covers:
  - **Mostly** Chapter 7
  - Arrays, selection sort
    - You will use a lot of loops and if-else statements.
  - No graphics
  - No Android

Exam 4

- Questions?
  - T/F, multiple choice, short answers
  - Read code, predict output
  - Debug code: identify syntax errors & logic bugs, fix them
  - **Write code**
  - Other…

Creating Arrays

- An array is an ordered collection, or numbered list, of values.
  - Values: primitive types, objects, even other arrays.
  - All of the values in an array must be of the same type
- Creating an array with 7 variables of type double



Formal Definition

- Syntax for declaring an array with **new**

```
Base_Type[] Array_Name = new Base_Type[Length];
```

- The number of elements in an array is its length
- The type of the array elements is the array's base type

Accessing Arrays

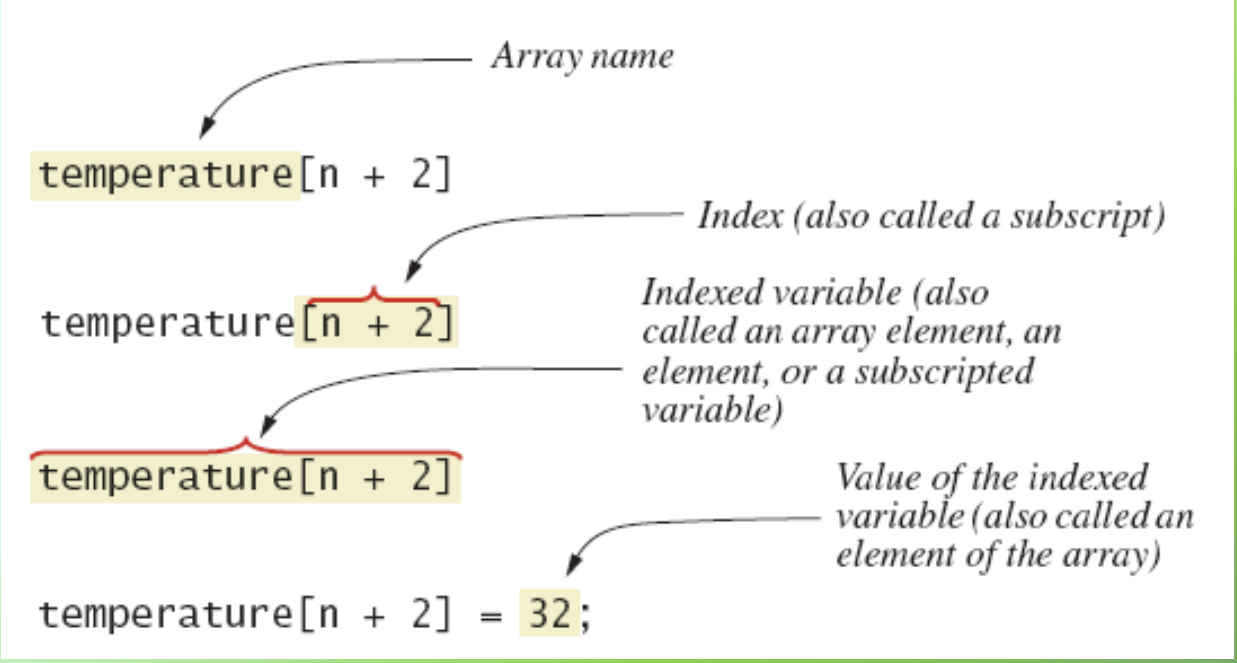
- To access an element use
  - The name of the array
  - An index number enclosed in braces

```
temperature[0]
temperature[1]
```

- Array indices begin at zero

Accessing Arrays

- Figure 7.2 Array terminology



The Instance Variable `length`

- As an object an array has only one public instance variable
  - Variable **`length`**
  - Contains number of elements in the array
  - It is final, value cannot be changed

```
int [] n=new int[20];
String s="this is a string";

System.out.println(n.length);
System.out.println(s.length());
```

More About Array Indices

- Index of first array element is 0
- Last valid Index is `arrayName.length – 1`
- Array indices must be within bounds to be valid
  - When program tries to access outside bounds, run time error occurs
- OK to "waste" element 0
  - Program easier to manage and understand
  - Yet, get used to using index 0

Initializing Arrays

- Possible to initialize at declaration time

```
double[] reading = {3.3, 15.8, 9.7};
```

- Also may use normal assignment statements
  - One at a time
  - In a loop

```
int[] count = new int[100];
for (int i = 0; i < 100; i++)
    count[i] = 0;
```

Arrays as Method Arguments

- Passing array elements as arguments
  - Example … **`a[i]`**
  - Can be used anywhere variable of array base type can be used
- Entire arrays as arguments
  - Formal parameter: use square brackets to
  - Actual parameter: do not use square brackets; the array could be any length.

```
public class SampleClass
{
    public static void incrementArrayBy2(double[] anArray)
    {
        for (int i = 0; i < anArray.length; i++)
            anArray[i] = anArray[i] + 2;
    }
    <The rest of the class definition goes here.>
}
```

Array Assignment and Equality

- Arrays are objects, array types are reference types
  - Assignment and equality operators behave (misbehave) as specified in previous chapter
- Variable for the array object contains memory address of the object
  - Assignment operator **=** copies this address
  - Equality operator **==** tests whether two arrays are stored in same place in memory
  - Define (static) methods (equals()) to check equality
  - Array class has static methods to compare two arrays.

Methods that Return Arrays

- A Java method may return an array
- Definition of return type as an array
  - public int[] theArray()
- To return the array value
  - Declare a local array
  - Use that identifier in the **`return`** statement

Partially Filled Arrays

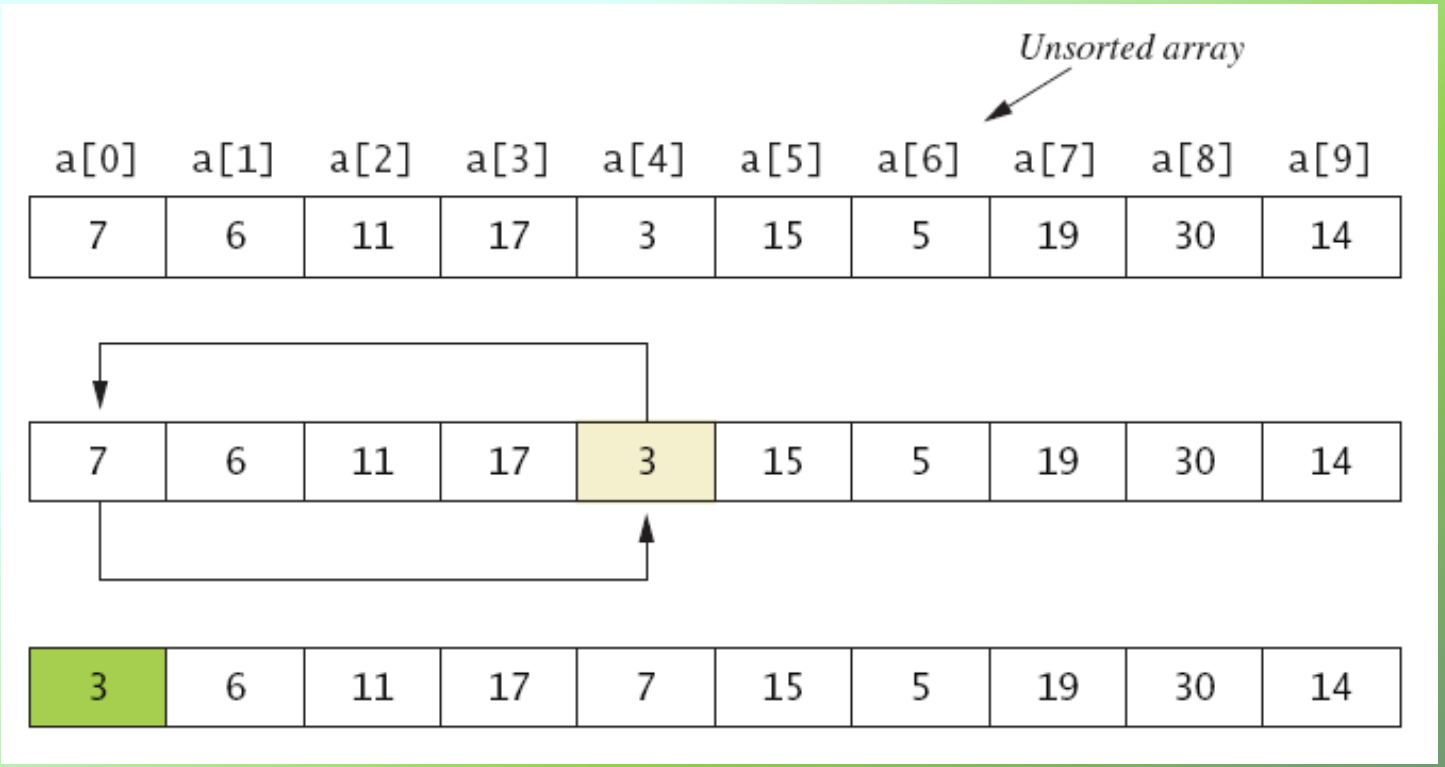
- Array size specified at definition
- Not all elements of the array might receive values
  - This is termed a *partially filled array*
- Programmer must keep track of how much of array is used

Selection Sort

- Sort: a very very important operation in all programming languages!
- Consider arranging all elements of an array so they are ascending order
- Selection sort:
  - Scan through the array (element 0 to length-1), move the smallest element to element 0
  - Scan through the rest of the array (element 1 to length-1), move the smallest element to element 1
  - Go on…

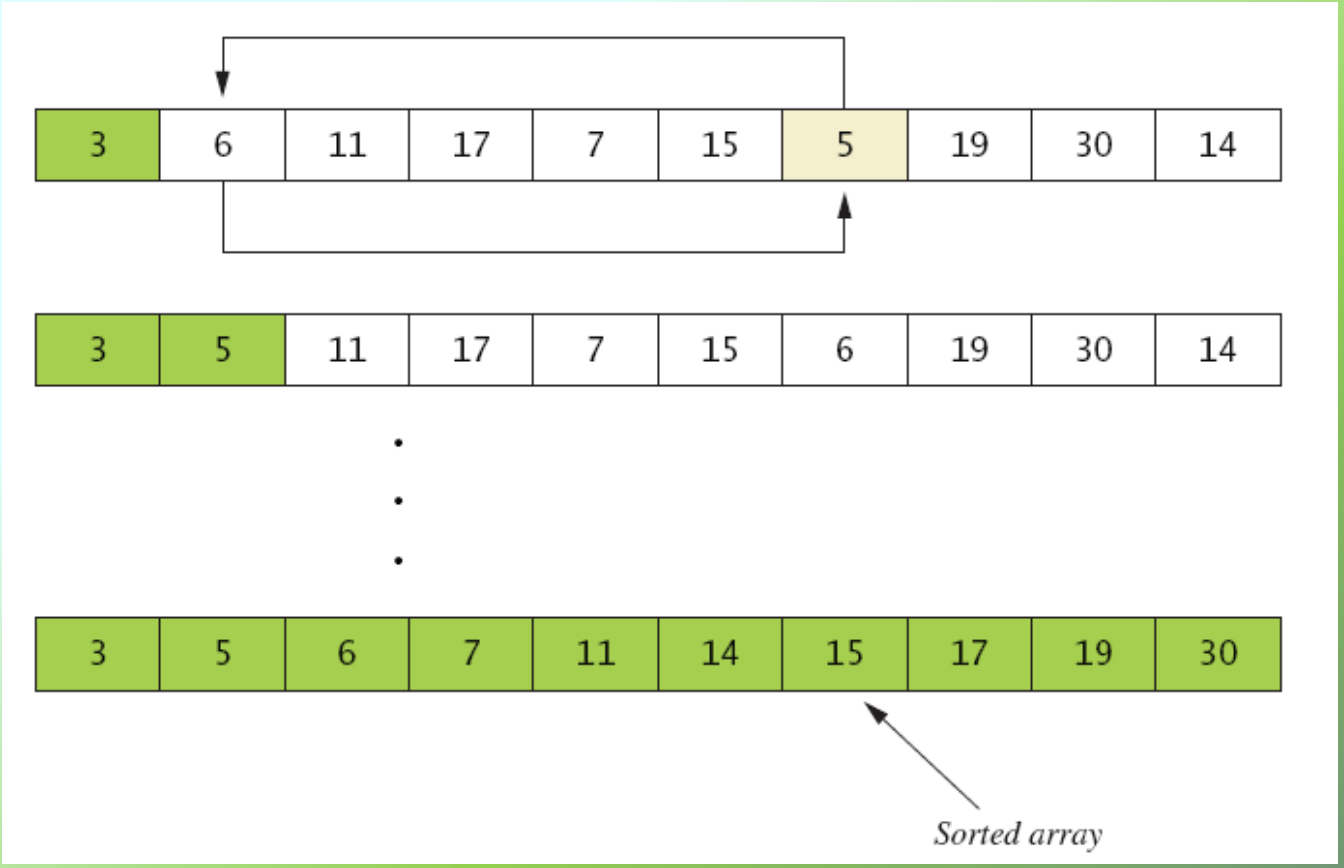
Selection Sort

- Figure 7.5a



Selection Sort

- Figure 7.5b



Multidimensional Arrays

- Multidimensional array represented as several one-dimensional arrays
- Given
  ```
  int [][] table = new int [10][6];
  ```
- Array table is actually 1 dimensional of type `int[]`
  - **It is an array of arrays**
- Important when sequencing through multidimensional array

Ragged Arrays

- Not necessary for all rows to be of the same length
- Example:

```
int[][] b;
b = new int[3][];
b[0] = new int[5]; //First row,  5 elements
b[1] = new int[7]; //Second row, 7 elements
b[2] = new int[4]; //Third row,  4 elements
```

- Q: how to determine if two multidimensional arrays are identical?

Example 1

- Your friend has designed a static method to display the three largest elements from an array of integers. However, the code appears to be inefficient. Please identify and fix the problem.

```
public static void printTop3(int[] a) {
    for (int i = 0; i < a.length; i++) {
        int max = a[i], maxindex = i;
        for (int j = i+1; j < a.length; j++) {
            if (a[j] > max) {
                max = a[j];
                maxindex = j;
            }
        }
        a[maxindex]=a[i];
        a[i]=max;
    }
    for (int i = 0; i < 3; i++)
        System.out.print(a[i] + ", ");
}
```

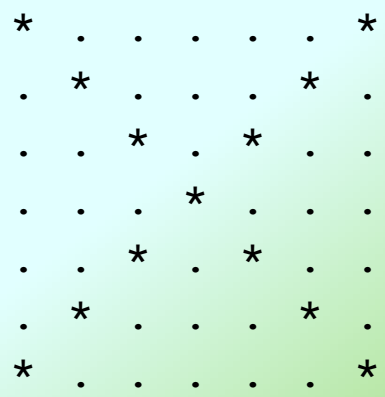
Example 1

- There's no need to sort the entire array, when you only need the top 3.

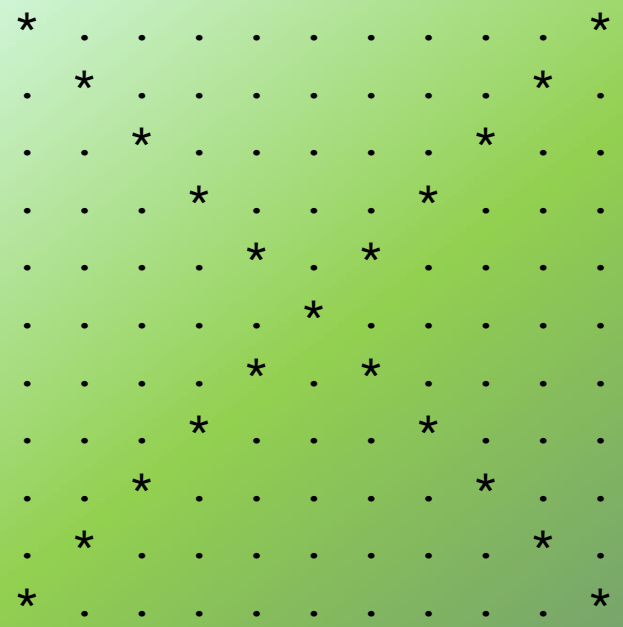
```
public static void printTop3(int[] a) {
    for (int i = 0; i < 3; i++) {
        int max = a[i], maxindex = i;
        for (int j = i+1; j < a.length; j++) {
            if (a[j] > max) {
                max = a[j];
                maxindex = j;
            }
        }
        a[maxindex]=a[i];
        a[i]=max;
    }
    for (int i = 0; i < 3; i++)
        System.out.print(a[i] + ", ");
}
```

Example 2

```
Enter n
3
```



```
Enter n
5
```

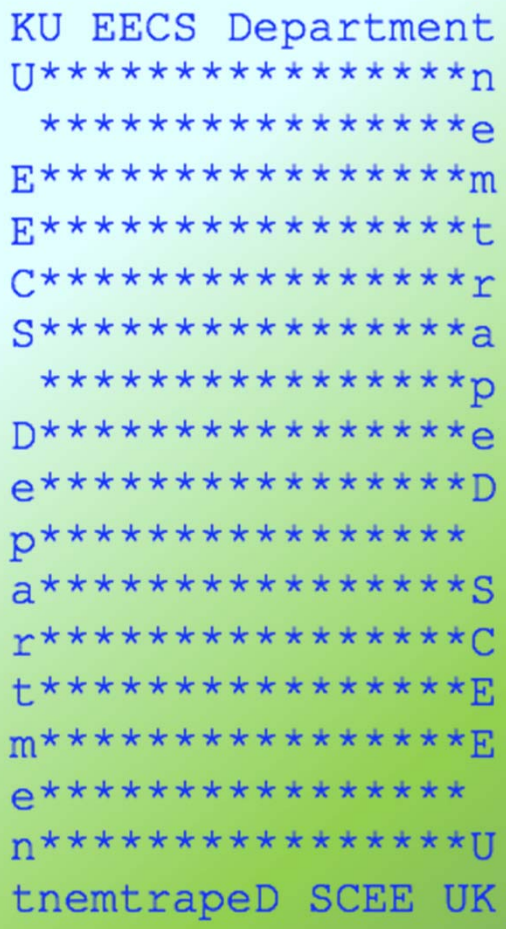


Example 2

```
public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    System.out.println("Enter n");
    int n = kb.nextInt();
    char[][] x = new char[2 * n + 1][2 * n + 1];
    for (int i = 0; i < 2 * n + 1; i++)
        for (int j = 0; j < 2 * n + 1; j++)
            x[i][j] = '.';
    for (int i = 0; i < 2 * n + 1; i++) {
        x[i][i] = '*';
        x[i][2 * n - i] = '*';
    }
    for (int i = 0; i < 2 * n + 1; i++) {
        for (int j = 0; j < 2 * n + 1; j++)
            System.out.print(x[i][j]);
        System.out.println();
    }

}
```

Example 3



```
public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    System.out.println("Enter a string");
    String s = kb.next();
    int n = s.length();
    char[][] x = new char[n][n];
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            x[i][j] = '*';

    for (int i = 0; i < n; i++)
        x[0][i] = s.charAt(i);
    for (int i = 0; i < n; i++)
        x[n - 1][i] = s.charAt(n - i - 1);
    for (int i = 0; i < n; i++)
        x[i][0] = s.charAt(i);
    for (int i = 0; i < n; i++)
        x[i][n - 1] = s.charAt(n - i - 1);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            System.out.print(x[i][j]);
        System.out.println();
    }
}
```