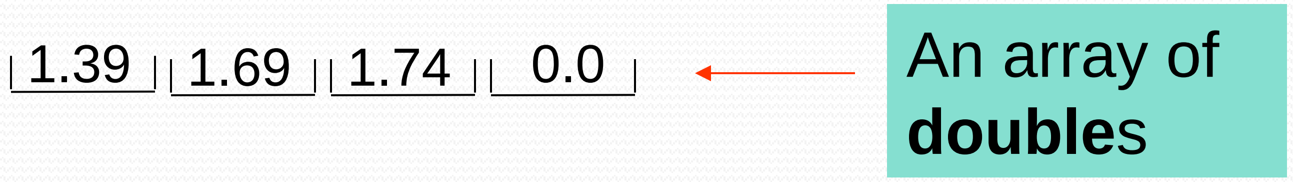
Objectives:

- Learn about one- and two-dimensional arrays and when to use them
- Learn the syntax for declaring and initializing arrays and how to access array’s size and elements
- Learn simple array algorithms

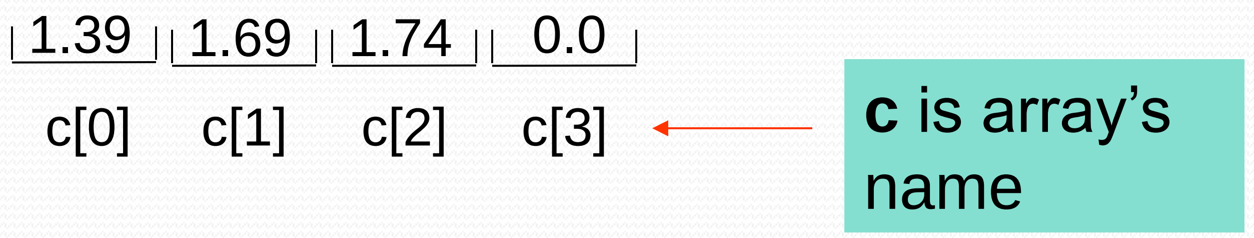
What is an Array

- An array is a block of consecutive memory locations that hold values of the same data type.
- Individual locations are called array's *elements*.
- When we say "element" we often mean the value stored in that element.



What is an Array (cont'd)

- Rather than treating each element as a separate named variable, the whole array gets one name.
- Specific array elements are referred to by using array's name and the element's number, called *index* or *subscript*.



Indices (Subscripts)

- In Java, an index is written within square brackets following array's name (for example, a[0]).
- <u>Indices start from 0</u>; the first element of an array a is referred to as a[0] and the *n*-th element as a[n−1].
- An index can have any int value from 0 to array's length – 1.

Indices (cont'd)

- We can use as an index an int variable or any expression that evaluates to an int value. For example:

```
a [3]
a [i]
a [i – 2]
a [ (int) (6 * Math.random()) ]
```

Indices (cont’d)

- In Java, an array is declared with fixed length that cannot be changed.
- Java interpreter checks the values of indices at run time and throws ArrayIndexOutOfBoundsException if an index is negative or if it is greater than the length of the array − 1.

Why Do We Need Arrays?

- The power of arrays comes from the fact that the value of an index can be computed and updated at run time.

No arrays:

1000 times!

```
int sum = 0;
sum += score0;
sum += score1;
…
sum += score999;
```

With arrays:

```
int n = 1000;
int sum = 0;

for (int i = 0;  i < n;  i++)
    sum += scores[i];
```

Why Arrays? (cont'd)

- Arrays give <u>direct access</u> to any element — no need to scan the array.

No arrays:

1000 times!

```
if (i == 0)
    display (score0);
else if (i == 1)
    display (score1);
else
…  // etc.
```

With arrays:

```
display (scores[i]);
```

Arrays as Objects)

- In Java, an array is an object
- As with other objects, the declaration creates only a reference, initially set to null. An array must be created before it can be used.
- One way to create an array:

```
arrName = new anyType [length] ;
```

Brackets

Declaration and Initialization

- When an array is created, space is allocated to hold its elements. If a list of values is not given, the elements get the default values. For example:

```
scores = new int [10] ;

words = new String [10000];
```

length 10, all values set to 0

length 10000, all values set to **null**

Initialization (cont'd)

- An array can be declared an initialized in one statement. For example:

```
int [ ]  scores = new int [10] ;

private double [ ]  gasPrices = { 3.05, 3.17, 3.59 };

String [ ]  words = new String [10000];

String [ ]  cities = {"Atlanta", "Boston", "Cincinnati" };
```

Initialization (cont'd)

- Otherwise, initialization can be postponed until later. For example:

```
String [ ]  words;
...
words = new String [ console.readInt() ];
private double[ ]  gasPrices;
…
gasPrices = new double[ ]  { 3.05, 3.17, 3.59 };
```

Not yet initialized

Not yet initialized

Array’s Length

- The length of an array is determined when that array is created.
- The length is either given explicitly or comes from the length of the {…} initialization list.
- The length of an array arrName is referred to in the code as arrName.length.
- length is like a public field (<u>not</u> a method) in an array object.

Initializing Elements

- Unless specific values are given in a {…} list, all the elements are initialized to the default value: 0 for numbers, false for booleans, null for objects.
- If its elements are objects, the array holds references to objects, which are initially set to null.
- <u>Each object-type element must be initialized before it is used</u>.

Initializing Elements (cont’d)

- Example:

```
Color[ ]  pens;
...
pens = new Color [ 3 ];
...
pens [0] = Color.BLUE;
pens [1] = new Color (15, 255, 255);
pens [2] = g.getColor();
```

Array not created yet

Array is created; all three elements are set to **null**

Now all three elements are initialized

Passing Arrays to Methods

- As other objects, an array is passed to a method as a reference.
- The elements of the original array are not copied and are accessible in the method's code.

```
//  Swaps a [ i ] and a [ j ]
public void swap (int [ ] a, int  i, int  j)
{
    int temp = a [ i ];
    a [ i ] = a [ j ];
    a [ j ] = temp;
}
```

Returning Arrays from Methods

- As any object, an array can be returned from a method.
- The returned array is usually constructed within the method or obtained from calls to other methods.
- The return type of a method that returns an array with *someType* elements is designated as *someType* [ ].

Returning Arrays from Methods (cont'd)

```
public double[ ] solveQuadratic
     (double a, double b, double c)
{
  double d = b * b – 4 * a * c;
  if (d < 0) return null;

  d = Math.sqrt(d);

  double[ ] roots = new double[2];
  roots[0] = (–b – d) / (2*a);
  roots[1] = (–b + d) / (2*a);
  return roots;
}
```

Or simply:

```
  return new double [ ]
    { (–b – d) / (2*a),
      (–b + d) / (2*a) };
```

Two-Dimensional Arrays

- 2-D arrays are used to represent tables, matrices, game boards, images, etc.
- An element of a 2-D array is addressed using a pair of indices, “row” and “column.” For example:

```
board [ r ] [ c ] = 'x';
```

2-D Arrays: Declaration

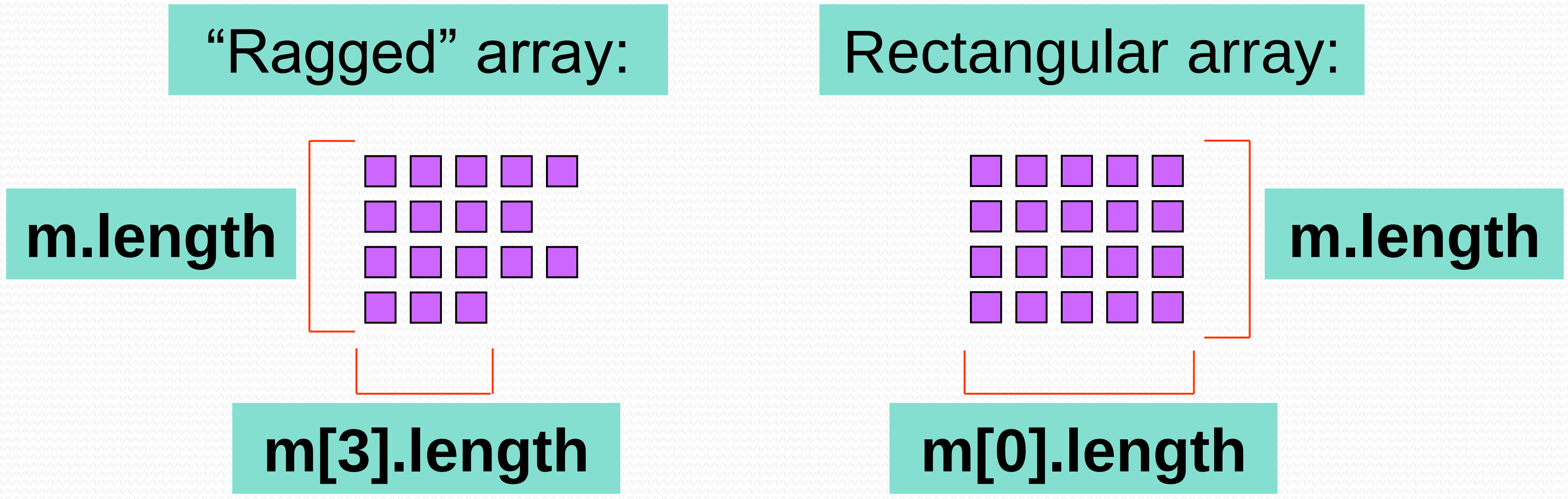
```
// 2-D array of char with 5 rows, 7 cols:
char[ ] [ ]  letterGrid = new char [5][7];

// 2-D array of Color with 1024 rows, 768 cols:
Color[ ] [ ]  image = new Color [1024][768];

// 2-D array of double with 2 rows and 3 cols:
double [ ] [ ]  sample =
    { { 0.0, 0.1, 0.2 },
      { 1.0, 1.1, 1.2 } };
```

2-D Arrays: Dimensions

- In Java, a 2-D array is basically a 1-D array of 1-D arrays, its rows. Each row is stored in a separate block of consecutive memory locations.
- If m is a 2-D array, then m[k] is a 1-D array, the *k*-th row.
- m.length is the number of rows.
- m[k].length is the length of the k-th row.

Dimensions (cont'd)

- Java allows "ragged" arrays, in which different rows have different lengths.
- In a rectangular array, m[o].length can be used to represent the number of columns.



2-D Arrays and Nested Loops

- A 2-D array can be traversed using nested loops:

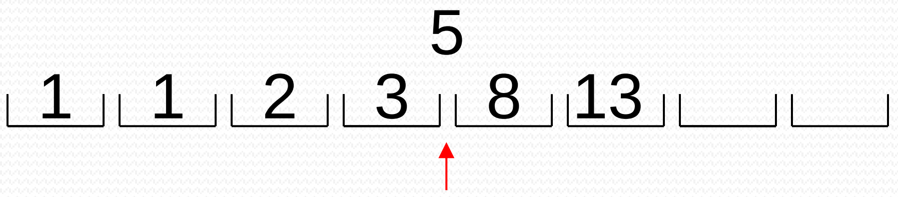
```
for (int  r = 0;  r < m.length;  r++)
{
    for (int  c = 0;  c < m[0].length;  c++)
    {
        ...   //  process m[ r ][ c ]
    }
}
```

Inserting a Value into a Sorted Array

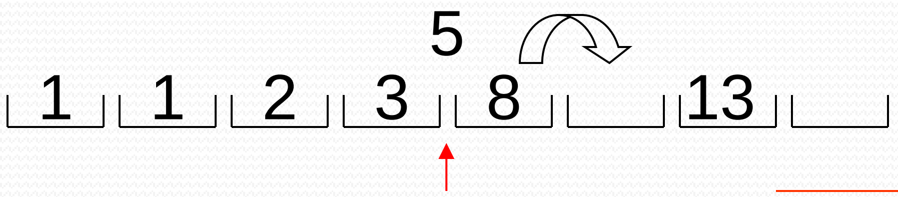
- Given: an array, sorted in ascending order. The number of values stored in the array is smaller than array's length: there are some unused elements at the end.
- Task: insert a value while preserving the order.

Inserting a Value (cont'd)

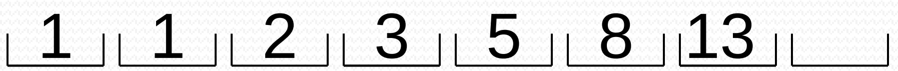
1. Find the right place to insert:



2. Shift elements to the right, starting from the last one:



3. Insert the value in its proper place:



Can be combined together in one loop: look for the place to insert while shifting.

Inserting a Value (cont’d)

```
//  Returns true if inserted successfully, false otherwise
public boolean insert(double[ ] arr, int count, double value)
{
    if (count >= arr.length)
        return false;

    int k = count – 1;
    while ( k >= 0  && arr [ k ] > value )
    {
        arr [ k + 1 ] = arr [ k ];
        k––;
    }
    arr [ k + 1] = value;

    return true;
}
```

Review:

- Why are arrays useful?
- What types of elements can an array have?
- How do we refer to an array’s element in Java?
- What happens if an index has an invalid value?
- How do we refer to the length of an array?

Review (cont’d):

- Can we resize an array after it has been created?
- Are arrays in Java treated as primitive data types or as objects?
- What values do array’s elements get when the array is created?
- Are the array’s elements copied when an array is passed to a method?
- Can a method return an array?

Review (cont'd):

- Name a few applications of two-dimensional arrays.
- If m is a 2-D array of ints, what is the type of m[0]?
- How do we get the numbers of rows and cols in a 2-D array?
- Describe an algorithm for inserting a value into a sorted array.