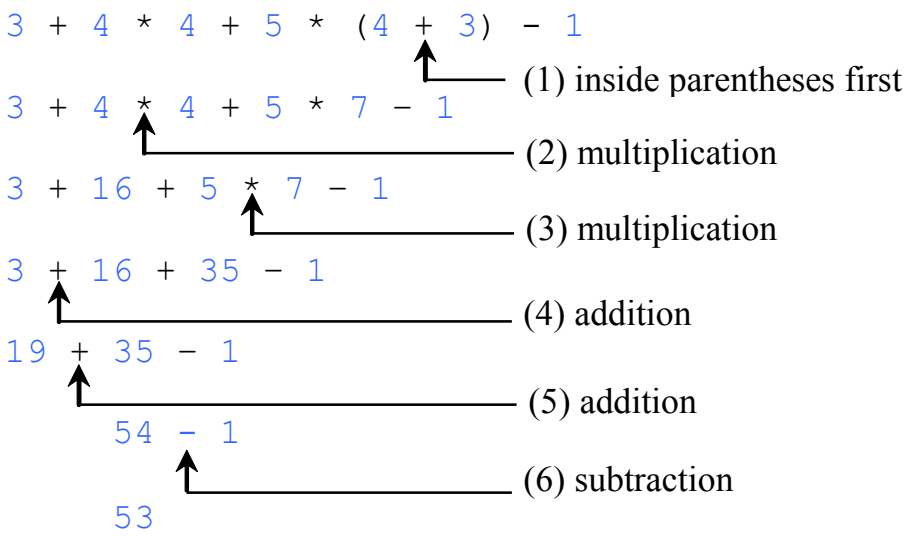
CS1 Review: Java Programming Colorado State University

Original slides by Daniel Liang
Modified slides by Chris Wilcox,
Russ Wakefield, Wim Bohm

Expressions

Remember operator precedence and associativity:



Java Logical and Arithmetic Operator Precedence Rules

1.	!	- (unary)		
2.	*	/	%	
3.	+	-		
4.	<	<=	>	>=
5.	==	!=		
6.	^	&	|	
7.	&&			
8.	||			

Division /and %

5 / 2 yields an integer, which?

5.0 / 2 yields a double, which?

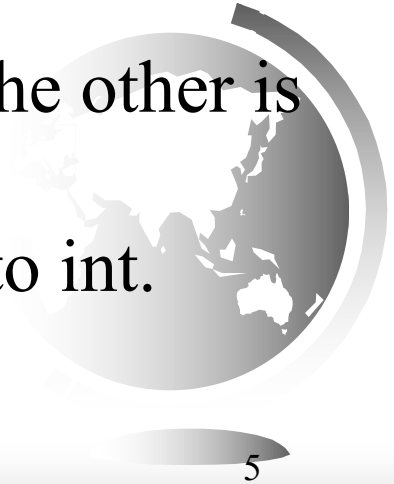
5 % 2 yields the integer remainder of the division, which?

p = (p/q)*q + p%q

Conversion Rules

When performing a binary operation involving two operands of different types, Java automatically converts the operand; promotes to wider type:

1. If one of the operands is double, the other is converted into double.
2. Otherwise, if one of the operands is float, the other is converted into float.
3. Otherwise, if one of the operands is long, the other is converted into long.
4. Otherwise, both operands are converted into int.



Type Casting

Implicit casting

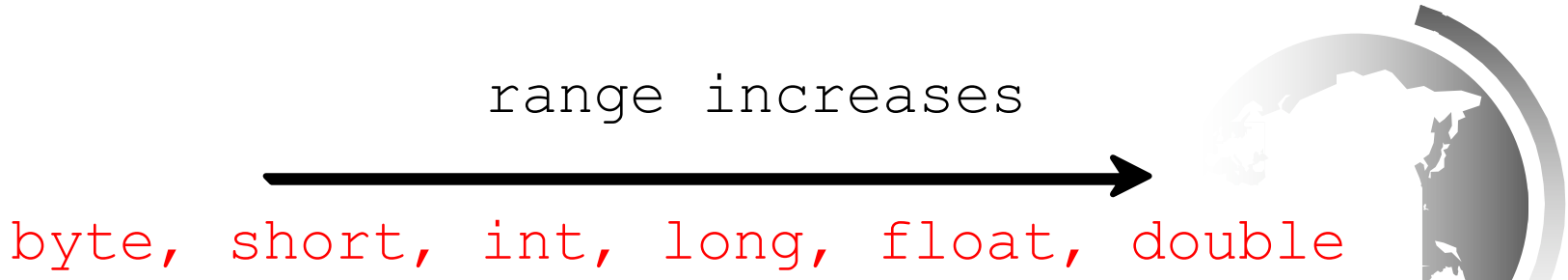
```
double d = 3;
```
(type widening)

Explicit casting for narrowing

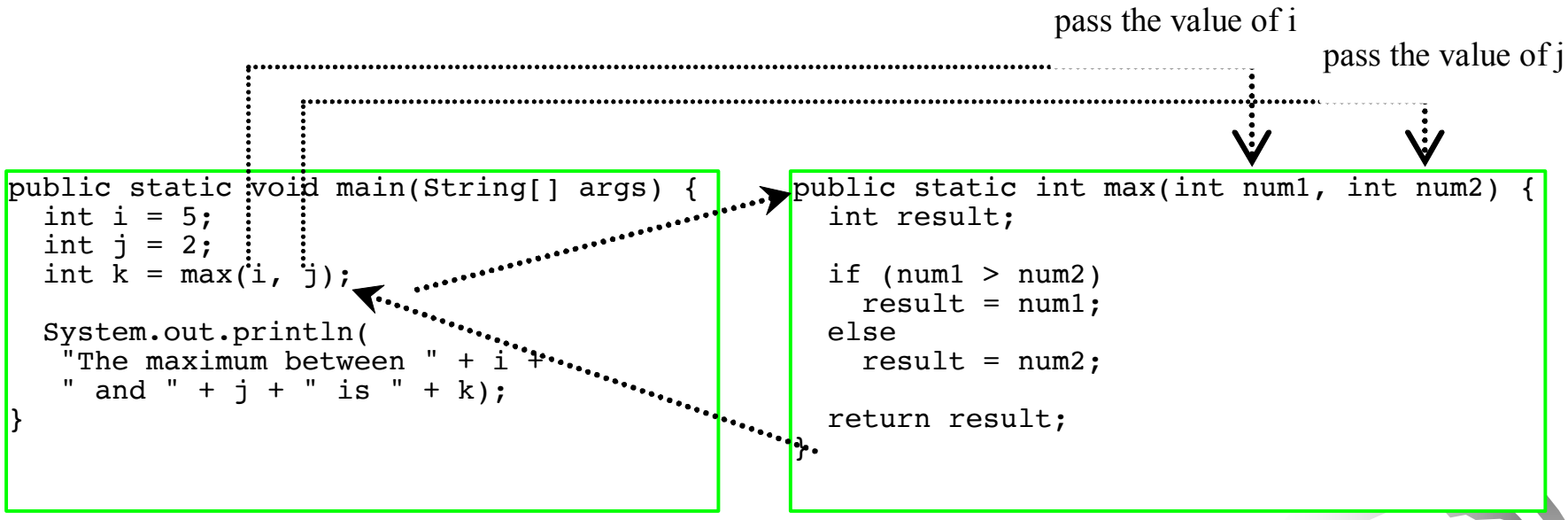
```
int i = (int)3.0;
```
(type narrowing)

```
int i = (int)3.9;
```
(Fraction part is truncated)

What is wrong? int x = 5 / 2.0;

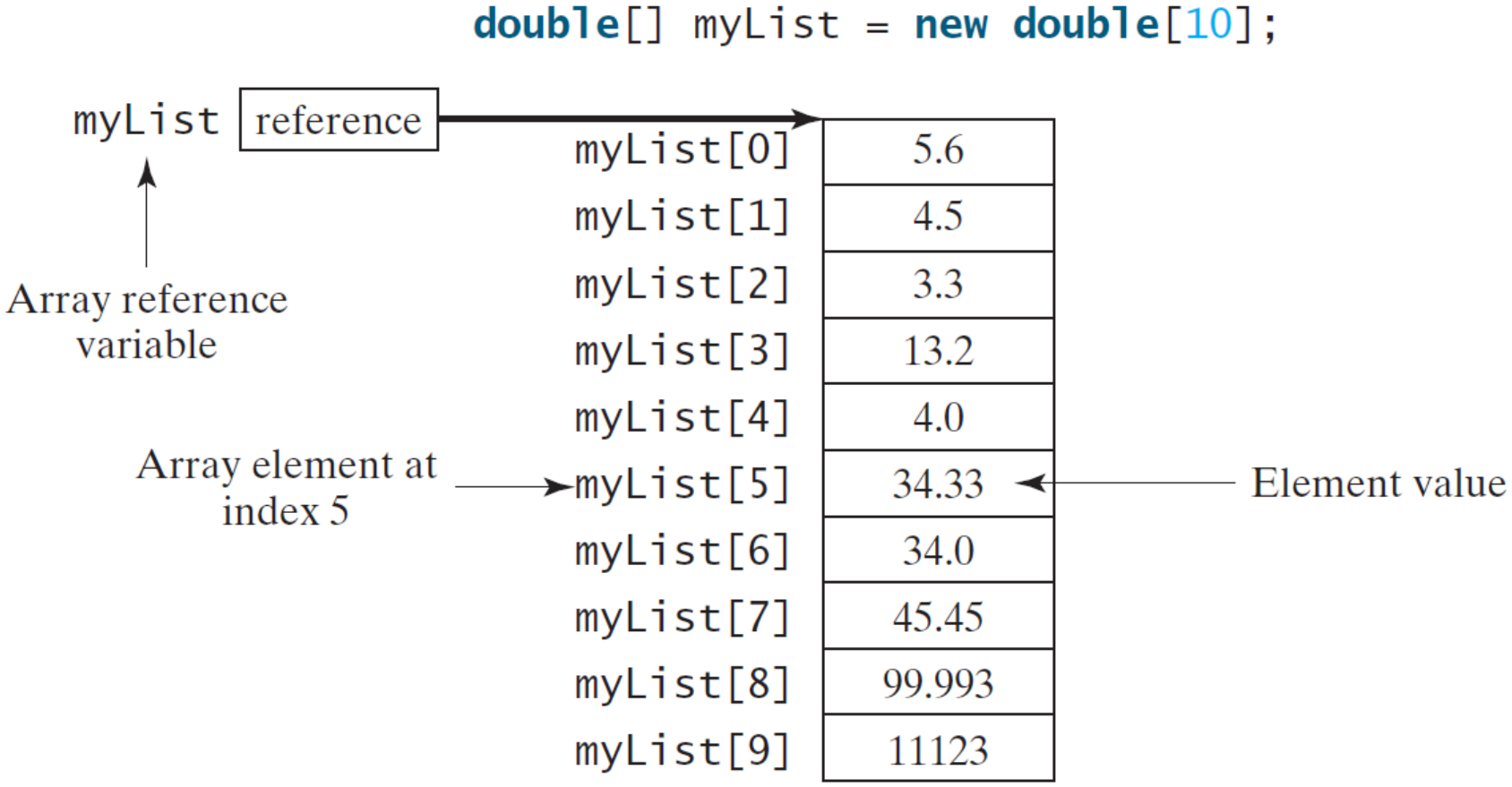


Calling Methods, cont.



Introducing Arrays

Array is a data structure that represents a collection of the same types of data.



Passing Arrays to Methods

```
public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

```
Invoke the method

int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

```
Invoke the method
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

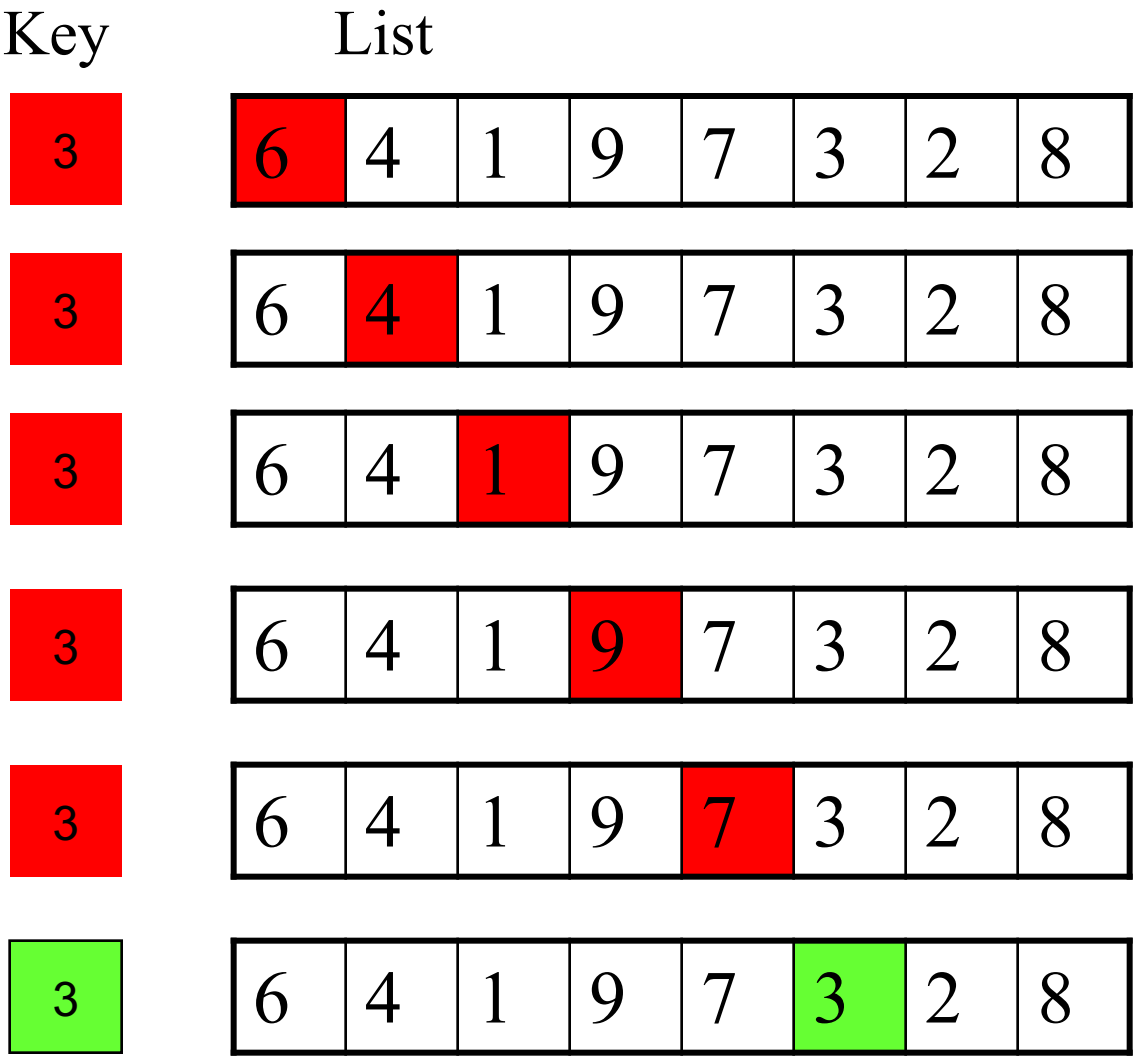
Anonymous array

Linear Search

The linear search approach compares the key element, key, *sequentially* with each element in the array list. The method continues to do so until the key matches an element in the list or the list is exhausted without a match being found. If a match is made, the linear search returns the index of the element in the array that matches the key. If no match is found, the search returns -1.

animation

Linear Search Animation



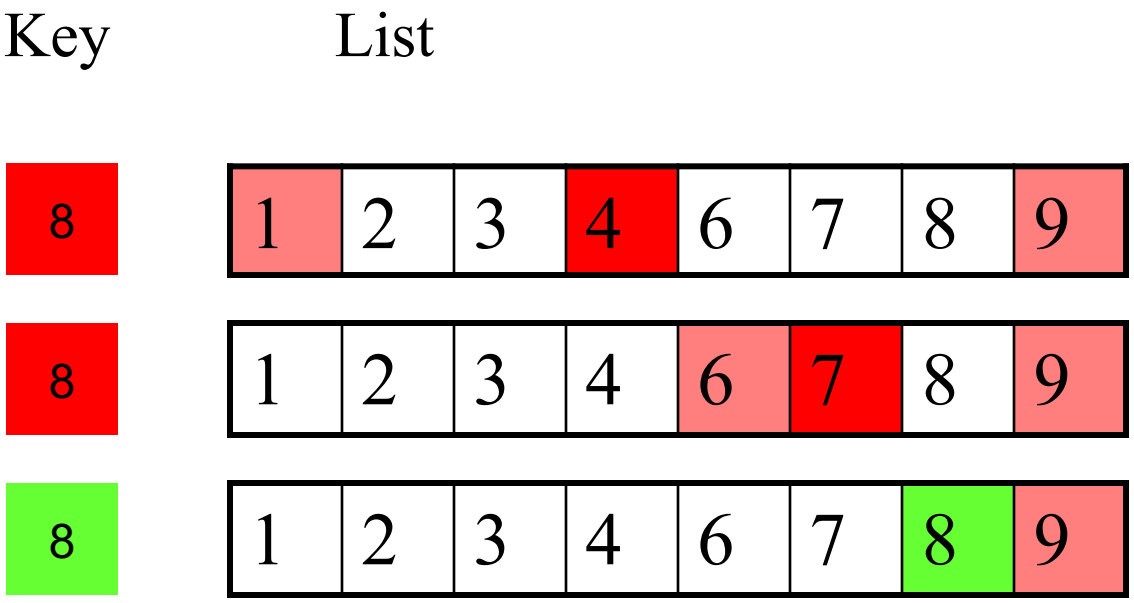
Binary Search

Consider the middle element in a sorted array:

- If the key is less than the middle element, you only need to search the key in the first half of the array.
- If the key is equal to the middle element, the search ends with a match.
- If the key is greater than the middle element, you only need to search the key in the second half of the array.

animation

Binary Search



Two-dimensional Arrays

```
int[][] array = {
  {1, 2},
  {3, 4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

what is array[2]?

what are the array bounds ?
(there are many)

array.length ?

array[0].length ?

array[1].length ?

array[2].length ?

array[3].length ?

Interface

An interface declares the public methods and constants as a contract:

Implementing an interface demands implementing the methods in the interface

```
public interface InterfaceName {
  constant declarations;
  method signatures;
}
```

Interfaces as a contract

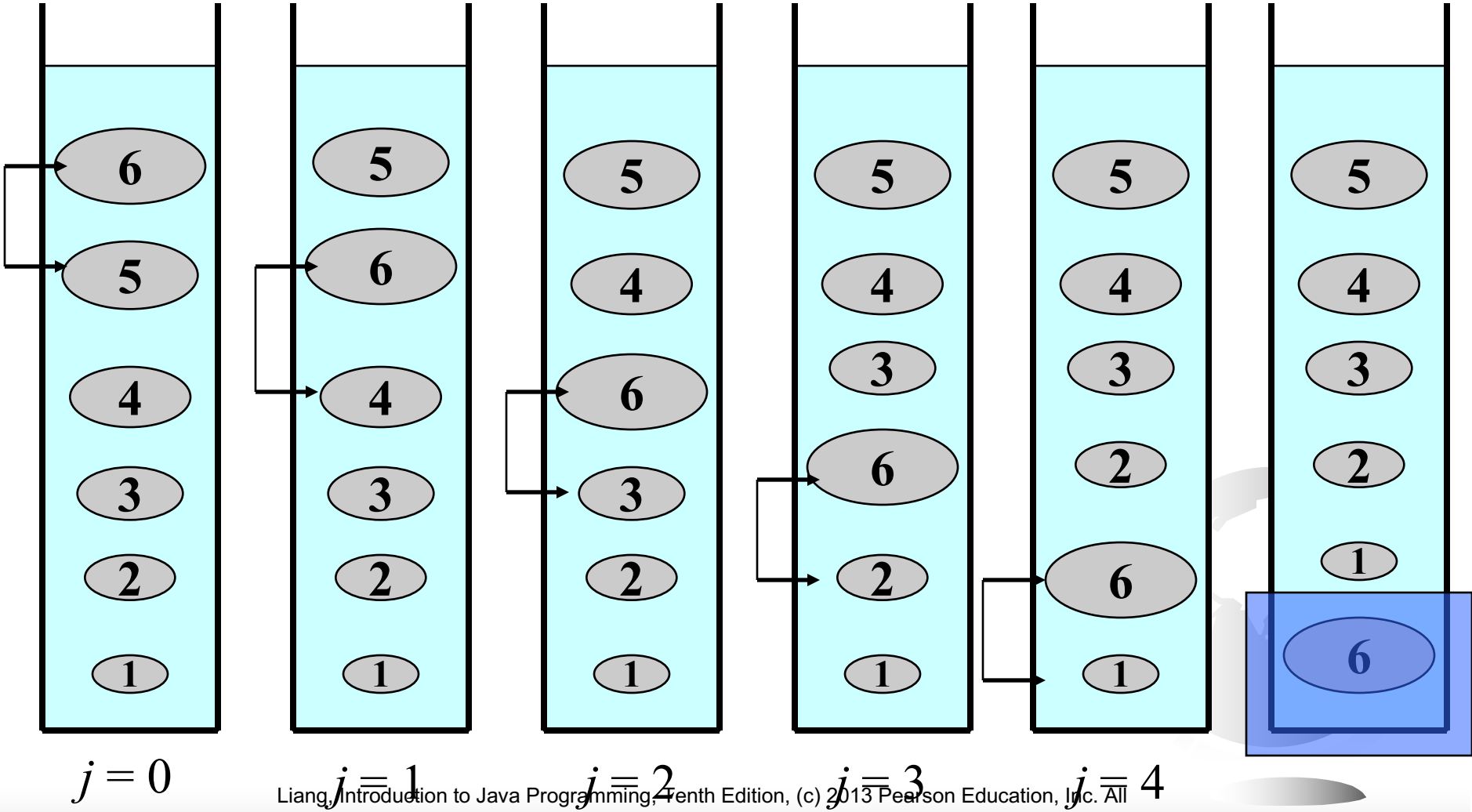
- Specifying what each method does
  - Specify it in a comment before method's header
- Precondition
  - What is assumed to be true before the method is executed
  - **Caller obligation**
- Postcondition
  - Specifies what will happen if the preconditions are met – what the method guarantees to the caller
  - **Method obligation**

Bubble Sort

- Compares neighboring elements, and swaps them if they are not in order
  - Effect: the largest value will "bubble" to the last position in the array.
  - Repeating the process will bubble the 2nd to largest value to the 2nd to last position in the array

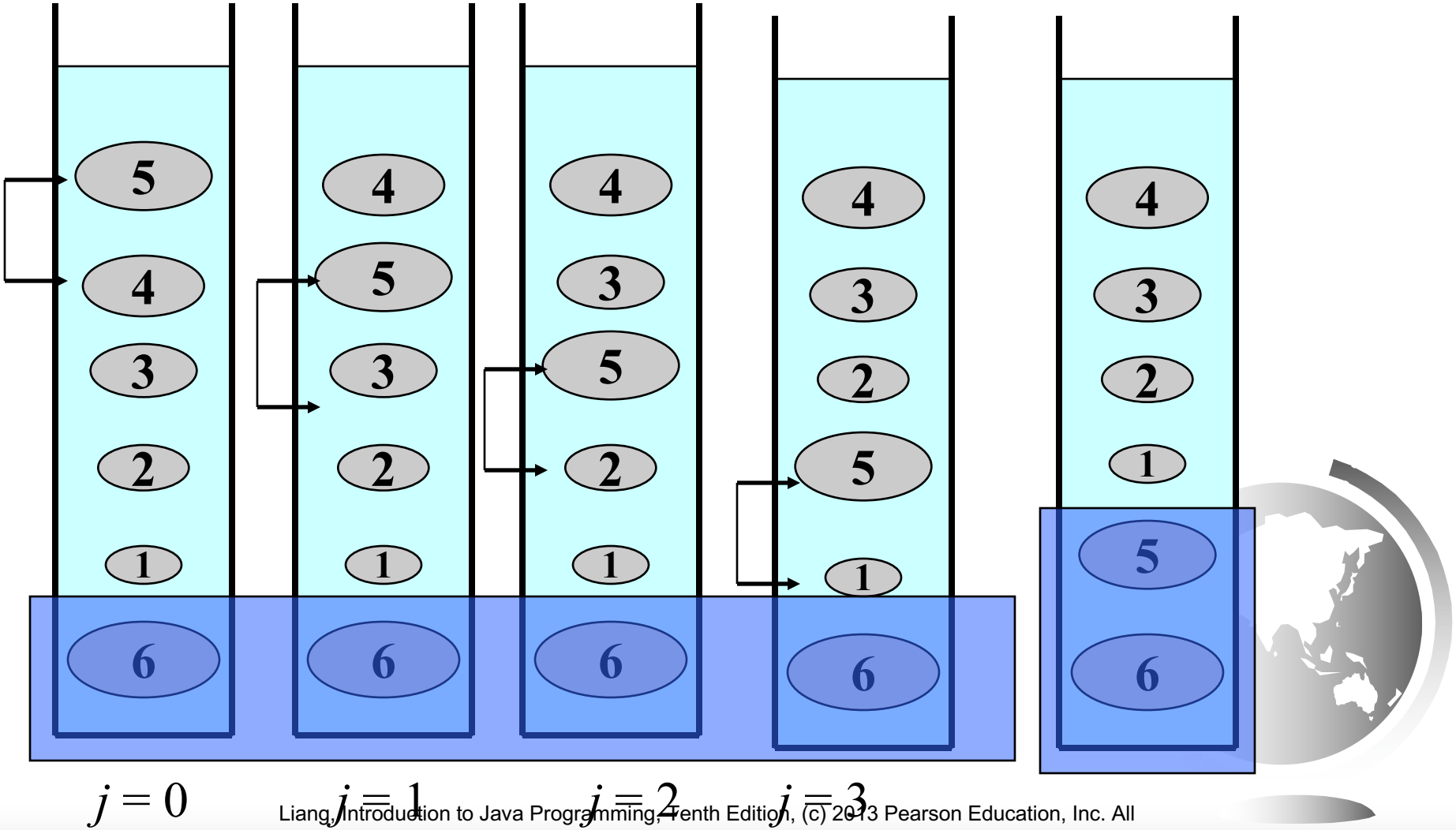
Bubble sort (First pass)

$i = 0$



Bubble sort (Second pass)

$i = 1$



Bubble Sort

```
public void bubbleSort (Comparable [] array) {
   for (int position = array.length-1; position>=0;
       position--) {
       for (int i = 0 ; i < position; i++) {
              if (array[i].compareTo(array[i+1]) > 0)
                     swap(array, i, i+1);
       }
   }
}
```

Inner Invariant: array[i] is the largest element in the first i elements in the array

Outer Invariant: After i iterations the largest i elements are sorted

Wrapper Classes

- Boolean
- Character
- Short
- Byte
- Integer
- Long
- Float
- Double

NOTE: (1) The wrapper classes do not have no-arg constructors. (2) The instances of all wrapper classes are immutable, i.e., their internal values cannot be changed once the objects are created.