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

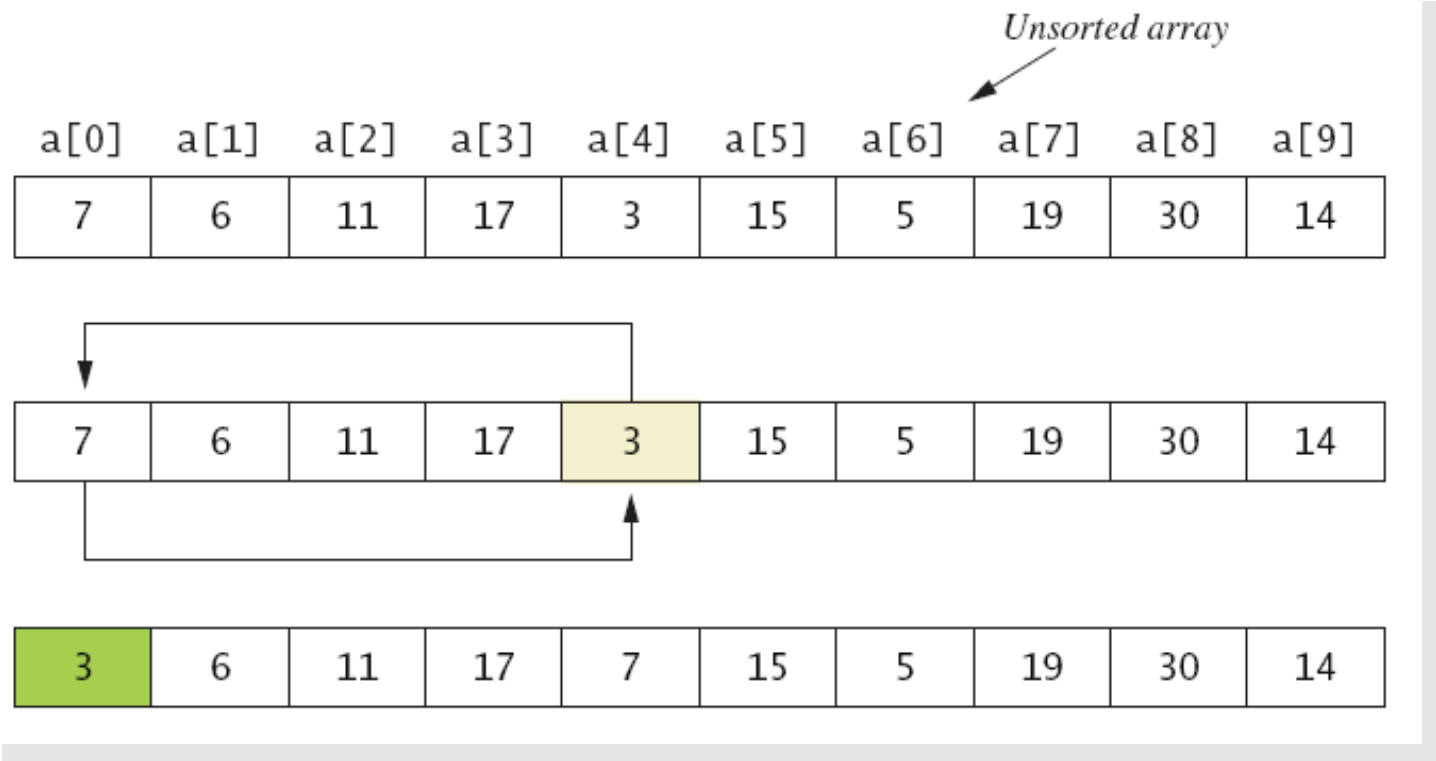
```
int[] count = new int[100];
for (int i = 0; i < 100; i++)
    count[i] = 0;
```

Selection Sort

- Consider arranging all elements of an array so they are ascending order
- Algorithm is to step through the array
  - Place smallest element in index 0
  - Swap elements as needed to accomplish this

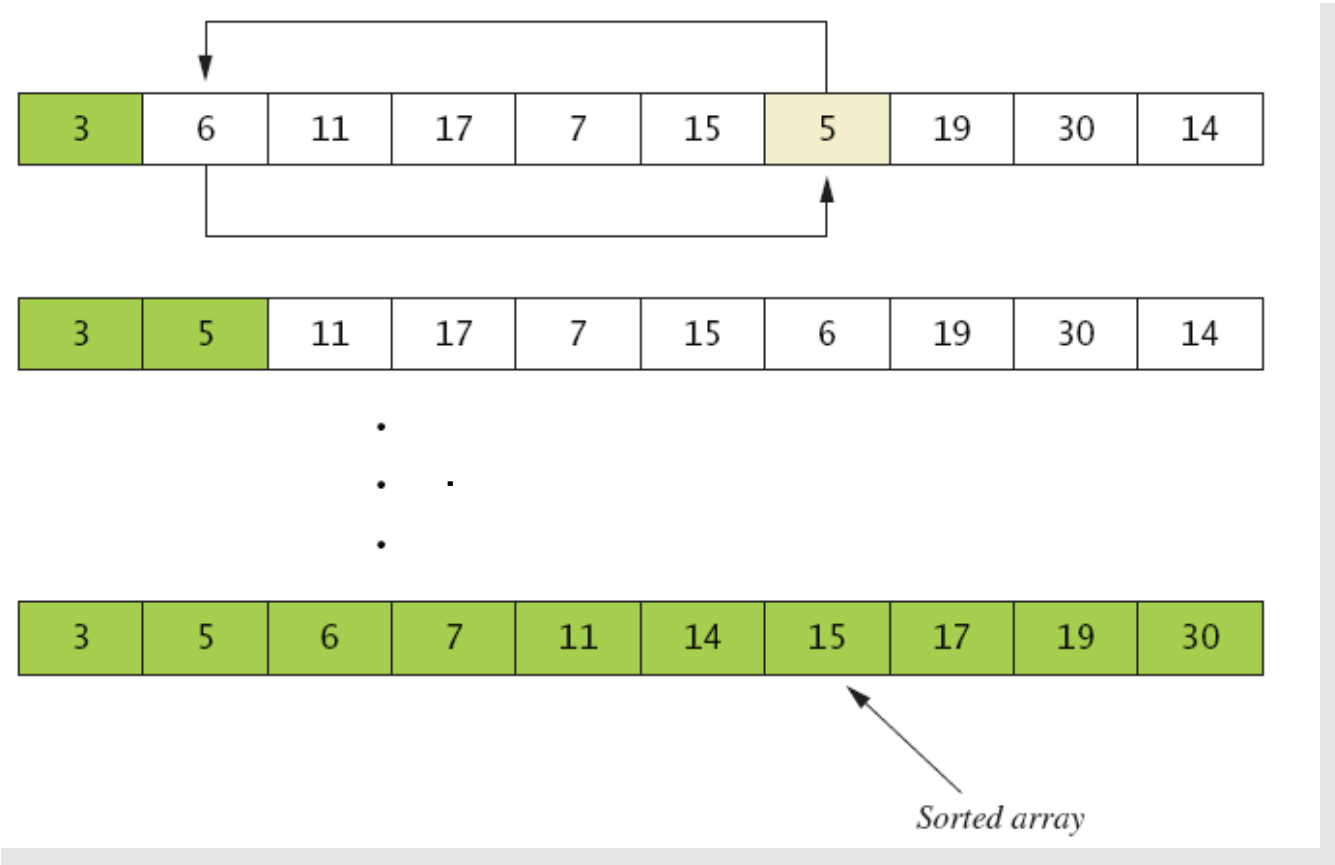
Selection Sort

- Figure 7.5a



Selection Sort

- Figure 7.5b



Gotcha – Don’t Exceed Array Bounds

- The code below fails if the user enters a number like 4. Use input validation.

```
 Scanner kbd = new Scanner(System.in);
int[] count = {0,0,0,0};

System.out.println("Enter ten numbers between 0 and 3.");
for (int i = 0; i < 10; i++)
{
 int num = kbd.nextInt();
 count[num]++;
}
for (int i = 0; i < count.length; i++)
   System.out.println("You entered " + count[i] + " " + i + "'s");
```

Array Assignment and Equality

- Arrays are objects
  - 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

Java's Representation of 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