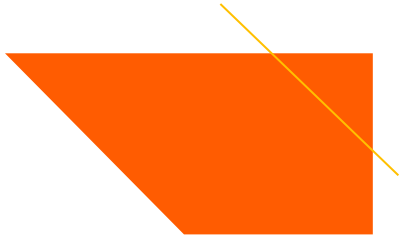
Advanced Programming Techniques in Java

COSI 12B



Class objectives

- Arrays (Chapter 7)

Review: Arrays and static methods

- **Method declaration**
- Syntax:

```
public static type methodName(type[] arrayName){
```

- Syntax:

```
public static type[] methodName(parameters) {
```

Review: Limitations of arrays

- You cannot resize an existing array

```
int[] A = new int[4];
A.length = 10;          // error
```

- An array does not know how to print itself

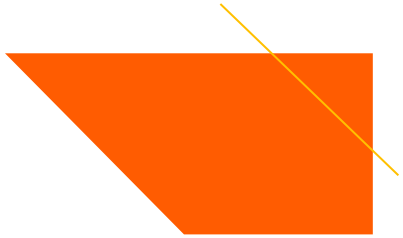
```
int[] A1 = {42, -7, 1, 15};
System.out.println(A1);
```

- You cannot compare arrays with == or `.equals` for Strings)

```
int[] A1 = {42, -7, 1, 15};
int[] A2 = {42, -7, 1, 15};
if (A1 == A2) {  ... }          // false!
if (A1.equals(A2)) {  ... }     // false!
```

Review: Limitations of arrays

```
public static void main(String[] args) {
        int[] A = {126, 167, 95};
        int[] B = A;
        int[] C = {126, 167, 95};

        System.out.println("A location = " + A);
        System.out.println("B location = " + B);
        System.out.println("C location = " + C);

        System.out.println(Arrays.toString(A));
        System.out.println(Arrays.toString(B));
        System.out.println(Arrays.toString(C));
}
```



Array question

- Write a method **increase** that accepts one array of integers and returns the same array with all the element values increased by 2

Array question

- Write a method **increase** that accepts one array of integers and returns the same array with all the element values increased by 2

```
public static void increase(int[] a) {
      for (int i = 0; i < a.length; i++) {
            a[i] = a[i] + 2;
      }
}
```

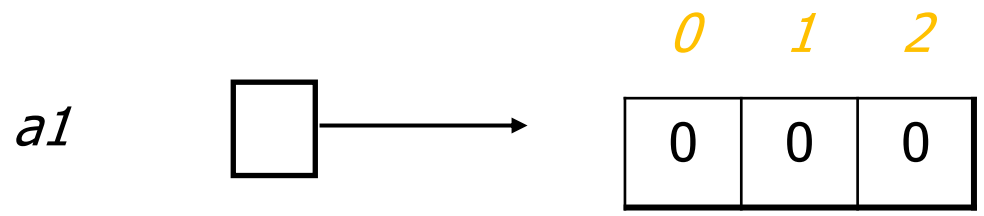
Does this look good?

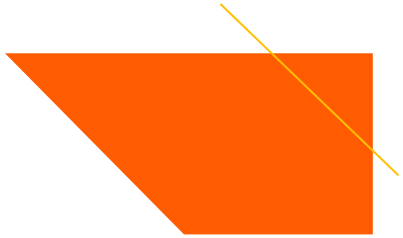
Arrays and references

- An array is a type of object
- An array variable is a reference variable
  - It stores a reference to the array

Example

```
int[] a1 = new int[3];
```





Reference and objects

- Arrays and objects use reference semantics

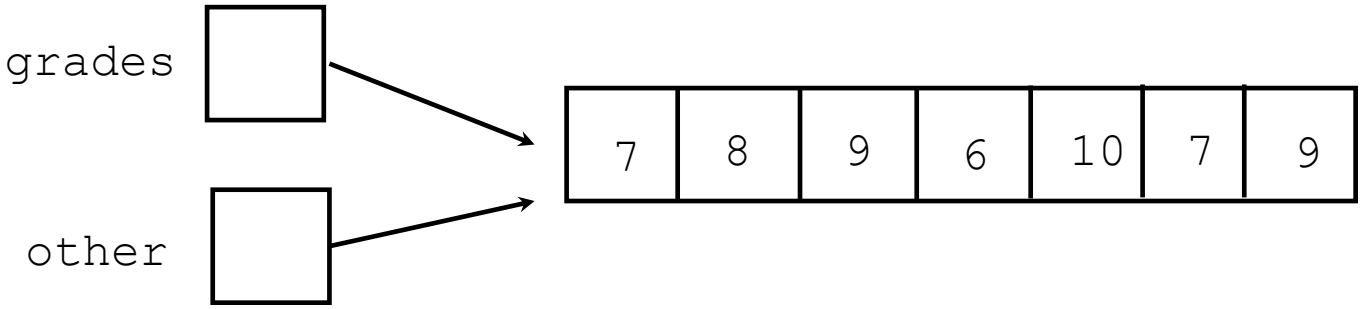
Why?

- Efficiency: copying large objects slows down a program
- Sharing: it's useful to share an object's data among methods

Copying references

- An example involving an array :

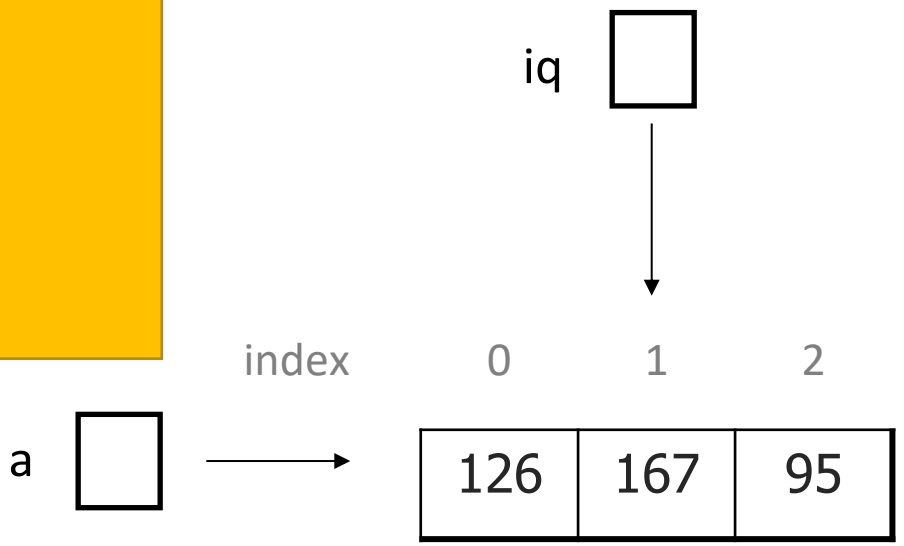
```
int[] grades = {7, 8, 9, 6, 10, 7, 9};
int[] other = grades;
```



Arrays passed by reference

- Arrays are passed as parameters by reference
  - Changes made in the method are also seen by the caller

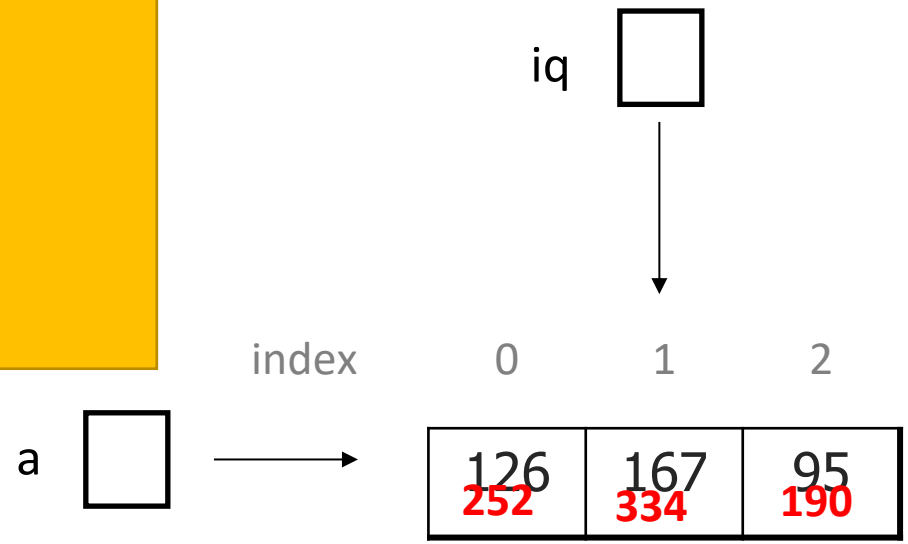
```
public static void main(String[] args) {
        int[] iq = {126, 167, 95};
        increase(iq);
        System.out.println(Arrays.toString(iq));
}

public static void increase(int[] a) {
        for (int i = 0; i < a.length; i++) {
                a[i] = a[i] * 2;
        }
}
```

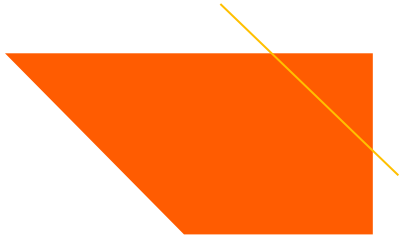


Arrays passed by reference

- Arrays are passed as parameters by reference
  - Changes made in the method are also seen by the caller

```
public static void main(String[] args) {
        int[] iq = {126, 167, 95};
        increase(iq);
        System.out.println(Arrays.toString(iq));
}

public static void increase(int[] a) {
        for (int i = 0; i < a.length; i++) {
                a[i] = a[i] * 2;
        }
}
```





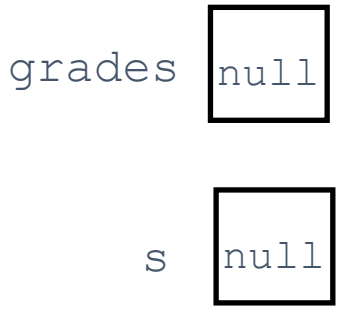
Reference semantics

- Reference semantics (or reference types): behavior where variables actually store the address of an object in memory

`Null` references

- To indicate that a reference variable doesn't yet refer to any object, we can assign it a special value called `null`

```
int[] grades = null;
String s = null;
```



- Attempting to use a `null` reference to access an object produce a `NullPointerException`
  - Pointer is another name for reference

```
grades[3] = 10; //NullPointerException
char ch = s.charAt(5); //NullPointerException
```

String vs. Array Objects

```
public class Test{

   public static void main(String[] args){

      int[] A = {1, 2, 3};
      int[] B = {1, 2, 3};
      int[] C = {1, 2, 3};

      if(A == B){
         System.out.println("true");
      }else{
         System.out.println("false");
      }
      if(A == C){
         System.out.println("true");
      }else{
        System.out.println("false");
      }
   }
}
```

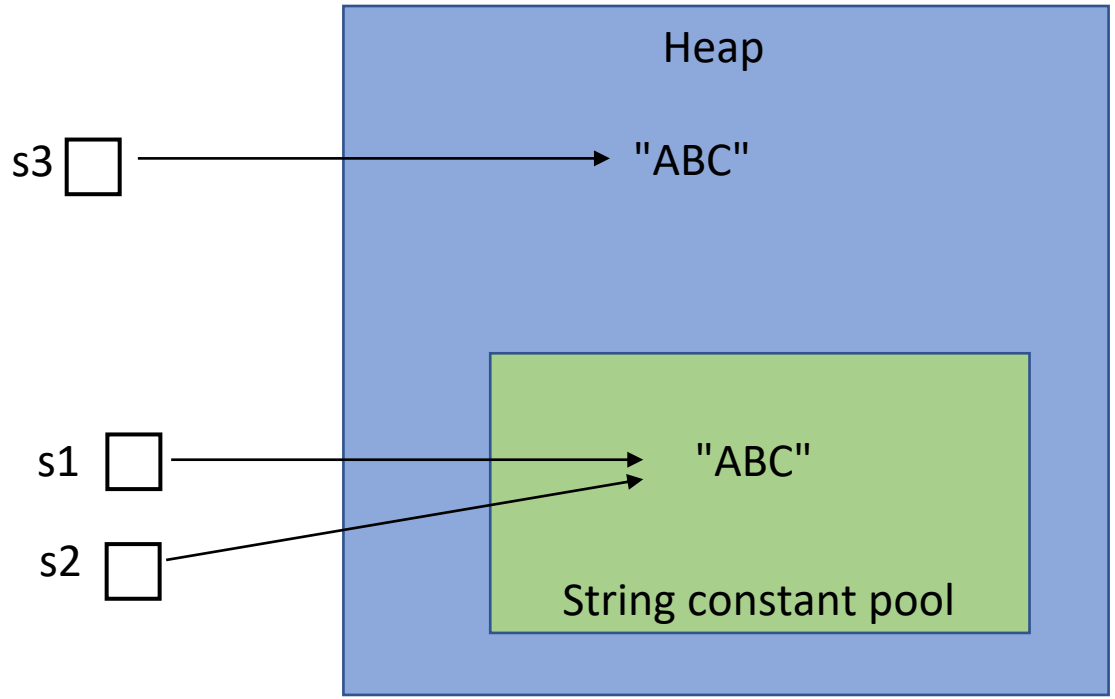
```
public class Test{

   public static void main(String[] args){

      String s1 = "ABC";
      String s2 = "ABC";
      String s3 = new String("ABC");

      if(s1 == s2){
         System.out.println("true");
      }else{
         System.out.println("false");
      }

      if(s1 == s3){
         System.out.println("true");
      }else{
         System.out.println("false");
      }
   }
}
```

String Objects

- There are two ways to create string objects in Java
  - `String s1 = "ABC"; //string constant pool`
  - `String s3 = new String("ABC"); //heap`

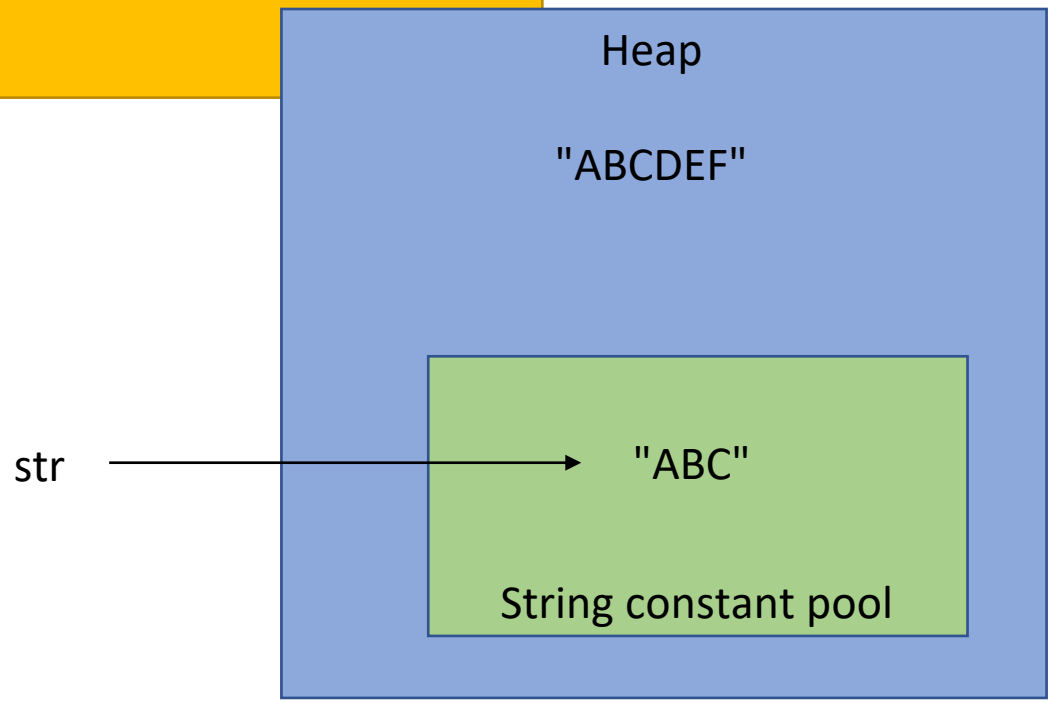
- The string constant pool is a separate place in the heap memory where the values of all the strings which are defined in the program are stored
- Duplicates are not allowed in the string constant pool

```
String s2 = "ABC";
```



String immutability

```
public class Test{

	public static void main(String[] args){
		String str = "ABC";
		str.concat("DEF");
		System.out.println(str);
	}
}
```

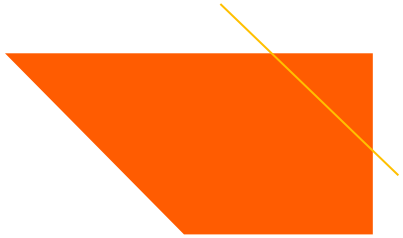


Java Constants

```
public static final int ARRAY_SIZE = 25;
```

```
private static final String URL = "tsekourakis.github.io";
```

- Constants in Java have to be initialized when declared!
- After that, they are read only.



Using an array to count things

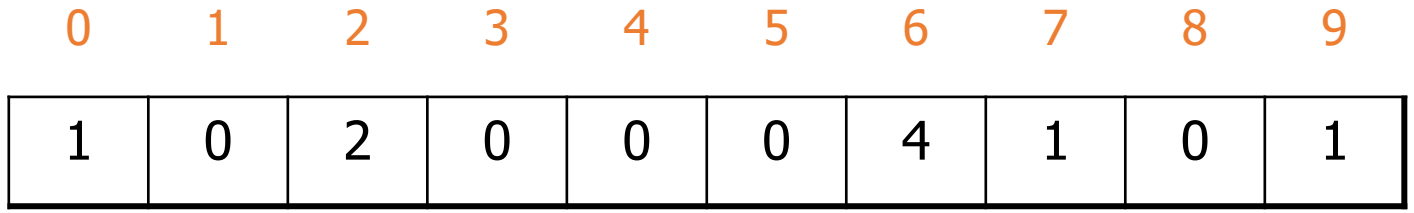
- Write a method `mostFrequentDigit` that returns the digit value that occurs most frequently in a number
- Example
  - The number 669260267 contains one 0, two 2s, four 6es, one 7, and one 9
    `mostFrequentDigit(669260267)` returns 6
  - If there is a tie, return the digit with the lower value `mostFrequentDigit(57135203)` returns 3

Using an array to count things

- We could declare 10 counter variables …

```
int counter0, counter1, counter2, counter3, counter4, …, counter9;
```

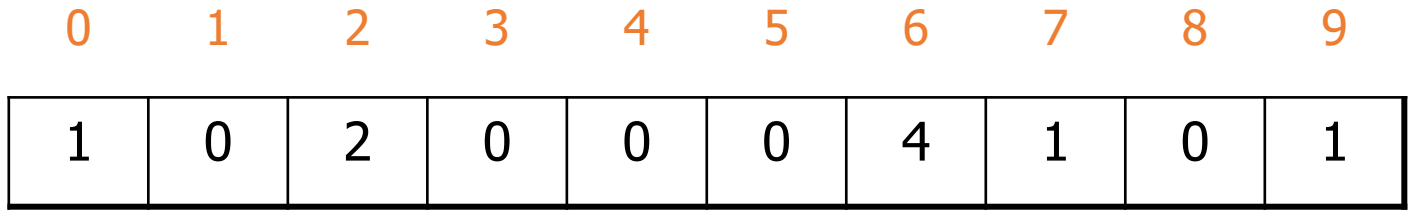
- But a better solution is to use an array of size 10
  - The element at index `i` will store the counter for digit value `i`
  - Example for 669260267



- How do we build such an array? And how does it help?

Creating an array of tallies

```
// assume n = 669260267
int[] counts = new int[10];
int digit = 0;
while (n > 0) {
    // pluck off a digit and add to proper counter
    digit = n % 10;
    counts[digit]++;
    n = n / 10;
}
```

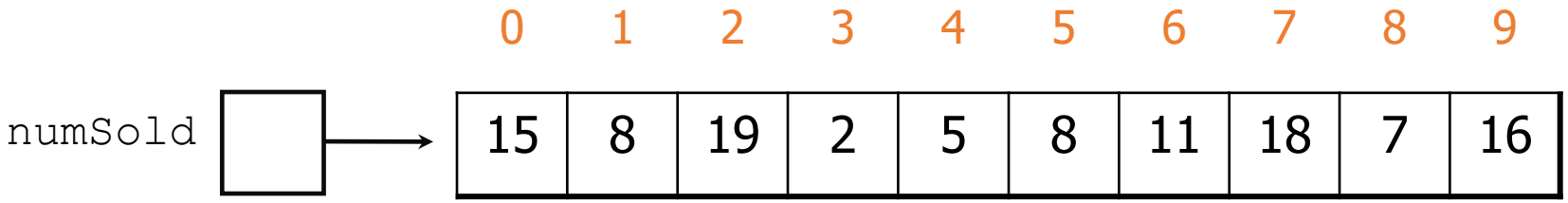


Creating an array of tallies

```
// Returns the digit value that occurs most frequently in n
// Breaks ties by choosing the smaller value
public static int mostFrequentDigit(int n) {
    int[] counts = new int[10];
    int digit = 0;
    while (n > 0) {
        digit = n % 10;  // pluck off a digit and tally it
        counts[digit]++;
        n = n / 10;
    }
    // find the most frequently occurring digit
    int bestIndex = 0;
    for (int i = 1; i < counts.length; i++) {
        if (counts[i] > counts[bestIndex]) {
            bestIndex = i;
        }
    }
    return bestIndex;
}
```

Shifting values in an array

- A small business is using an array to store the number of items sold over a 10-day period



`numSold[0]` gives the number of items sold today

`numSold[1]` gives the number of items sold 1 day ago

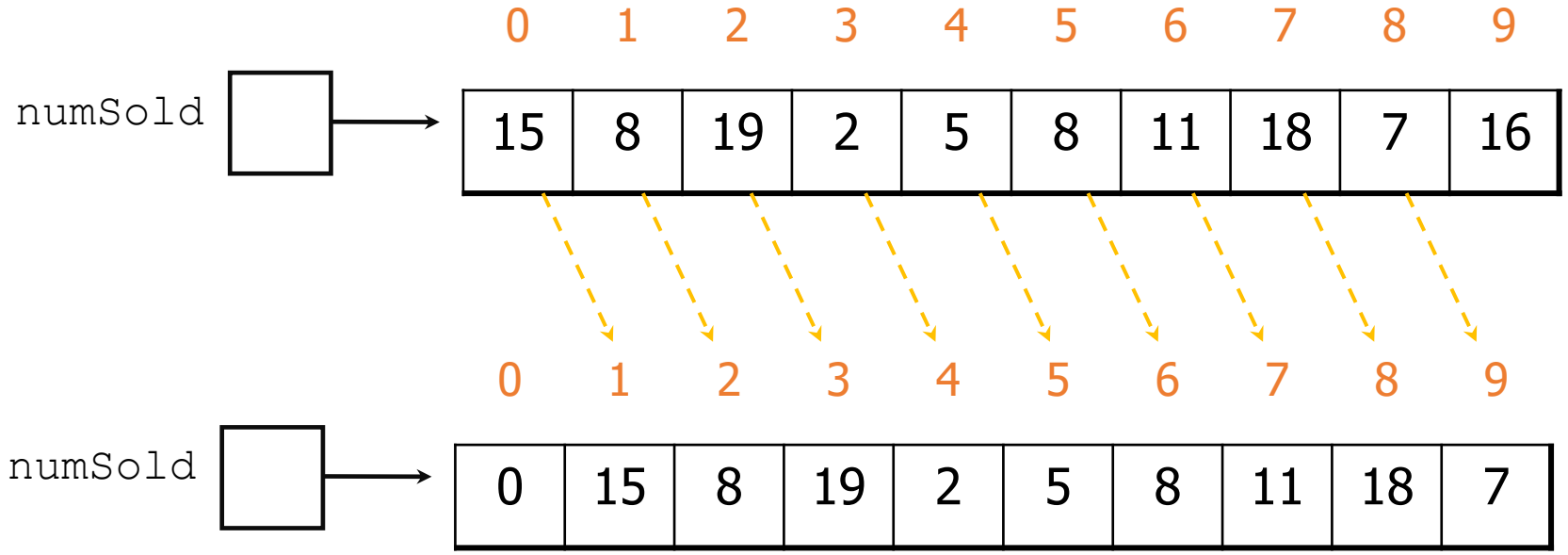
`numSold[2]` gives the number of items sold 2 days ago

…

`numSold[9]` gives the number of items sold 9 days ago

Shifting values in an array

- At the start of each day, it's necessary to shift the values over to make room for the new day's sales



- The last value is lost, since it's now 10 days old

Shifting values in an array (cont.)

```
for (int i = 0; i < numSold.length; i++) {
                numSold[i] = numSold[i - 1];
}
```

- Does this work?

Shifting values in an array (cont.)

```
for (int i = 0; i < numSold.length; i++) {
                numSold[i] = numSold[i - 1];
}
```

- No, If we run this, we get an `ArrayIndexOutofBoundsException`

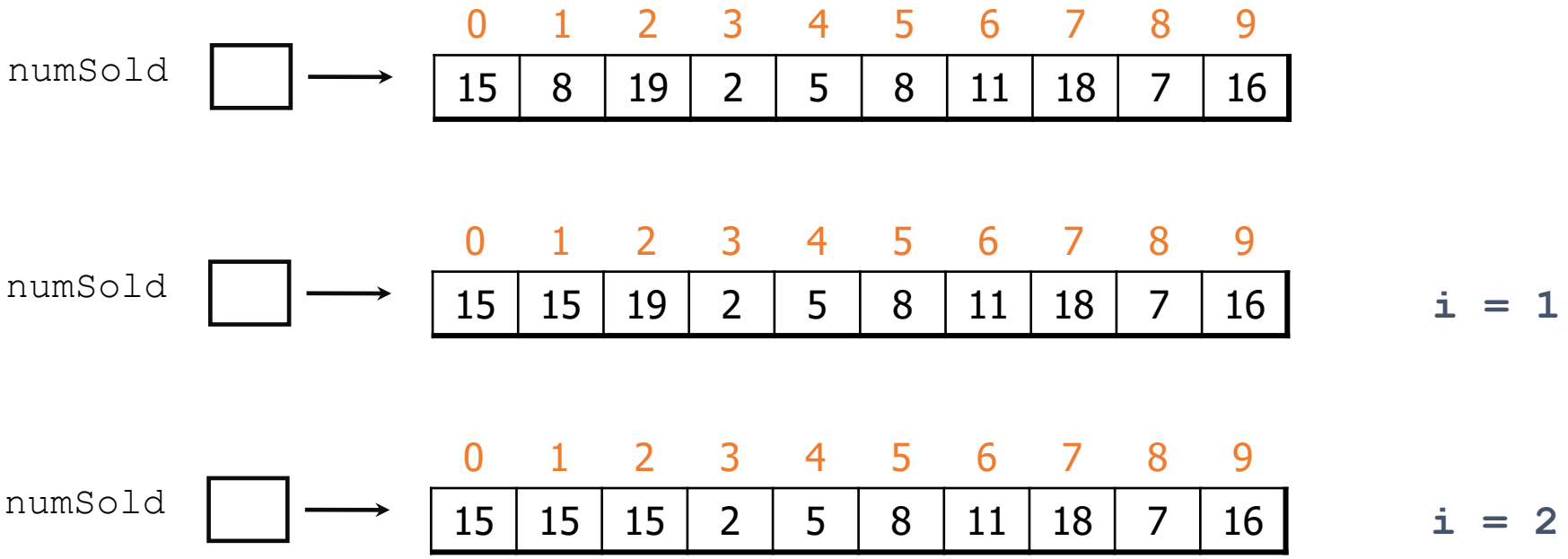
Shifting values in an array (cont.)

```
for (int i = 1; i < numSold.length; i++) {
            numSold[i] = numSold[i - 1];
}
```

- Does this work?

Shifting values in an array (cont.)

```
for (int i = 1; i < numSold.length; i++) {
            numSold[i] = numSold[i - 1];
}
```



- It doesn't work!

Shifting values in an array (cont.)

```
for (int i = 1; i < numSold.length; i++) {
            numSold[i] = numSold[i - 1];
}
```

- How can we fix the code below so that it does the right thing?

```
for (int i = numSold.length - 1; i >= 1; i--) {
      numSold[i] = numSold[i - 1];
}
```

- Are we done?

Shifting values in an array (cont.)

```
for (int i = 1; i < numSold.length; i++) {
            numSold[i] = numSold[i - 1];
}
```

- How can we fix the code below so that it does the right thing?

```
for (int i = numSold.length - 1; i >= 1; i--) {
      numSold[i] = numSold[i - 1];
}
```

- After performing all the shifts, we would do: `numSold[0] = 0;`

“Growing” an array

- Once we have created an array, we can’t increase its size
- Instead, we need to do the following:
  - Create a new, larger array
  - Copy the contents of the original array into the new array
  - Assign the new array to the original array variable

```
int[] a1 = {42, -7, 1, 15};
…
int[] tmp = new int[10];
for (int i = 0; i < a1.length; i++){
        tmp[i] = a1[i];
}
a1 = tmp;
```