More on Arrays

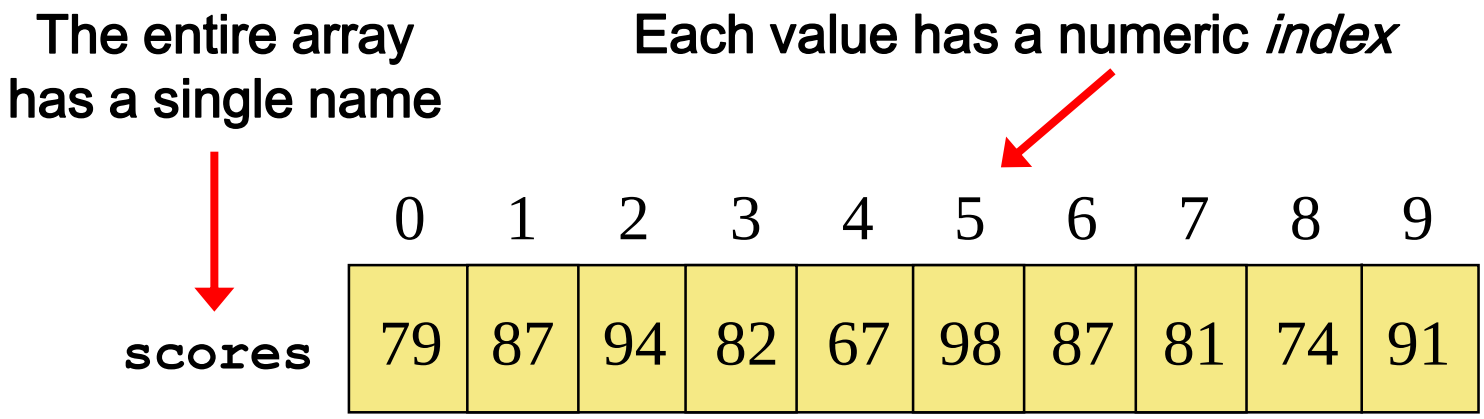
GEEN163

"I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."

Maurice Wilkes

Arrays

- An array is an ordered list of values:



An array of size N is indexed from zero to N-1

This array holds 10 values that are indexed from 0 to 9

Example Indexes

```
double[] friday = new double[25];
int cat = 3, dog = 4, cow = 5;

friday[ 1 ] = 47.2;                 // constant
friday[ cat ] = 12.3;               // variable
friday[ cow+dog ] = 32.23;          // expression
```

For Loop for Arrays

- A for loop is frequently used to go through an array

```
double sum = 0.0;
double[] dog = new double[1024];
```

- - - - - - - - - - - - - - - - - - -

```
for (int i = 0; i < dog.length; i++ ) {
    sum += dog[i];
}
```

Extended for Loop

- There is a special format for the for loop to go through an array

```
double sum = 0.0;
double[] dog = new double[1024];
- - - - - - - - - - - - - - - - - - -
for ( double cat : dog ) {
    sum += cat;
}
```

Extended `for` copy

- Each iteration of the loop, an extended **`for`** ***copies*** one element into the defined variable

```
double[] dog = new double[10];
for (int i = 0; i < 10; i++) {
        dog[i] = 303;       // sets all dogs to 303
}
for (double cat : dog) {
        cat = 270;          // does nothing
}
```

Initializing Arrays

- Like simple variables, arrays can be initialized to a value when they are declared
- When you initialize an array, you do not need to specify the size. The size is determined by how many values you use to initialize

```
int[] rabbit = {32, 14, 7, 26, 86};
```

- The initialization values must be the right type
- Note the semicolon after the curly bracket

Arrays of Strings

- An array of Strings can be initialized

```
String[] question = { "What is reality?",
                      "Is Santa real?", "How much is that?" };
```

- With any array initialization, the data values must be separated by commas

Which sums the values of the array?

```
double[] dog = new double[32], sum = 0.0; int index = 0;
```

```
while (index < 32) {          // A
    sum += dog[index];
    index++;
}
```

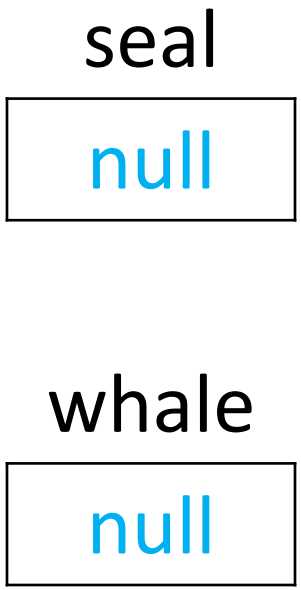
```
while (index  < 32) {         // B
    sum = sum + index;
    index++;
}
```

```
while (dog[index] < 32) {     // C
    sum = sum + dog[index];
    index++;
}
```

Object References

- When you declare an array, you are creating a reference to an array

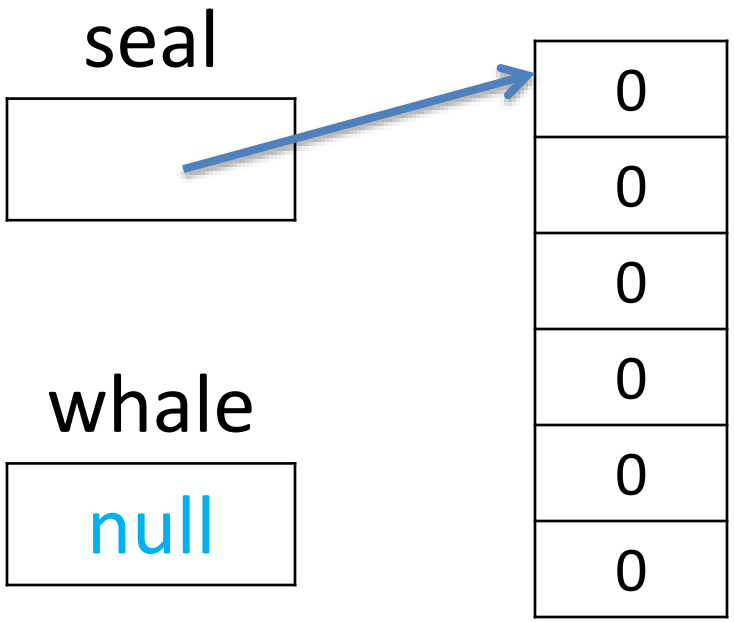
```
int[] seal;
int[] whale;
```



Creating an Array Object

- An array is created the same way you create other objects. The **new** create the array

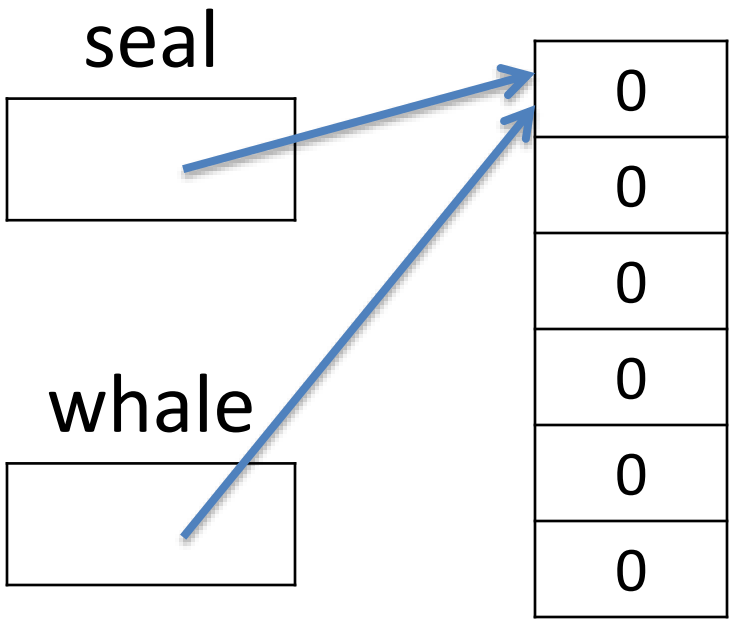
```
int[] seal;
int[] whale;
seal = new int[6];
```



Multiple Names for the Same Array

- Setting one array to another copies the reference
  Both variable reference the same array

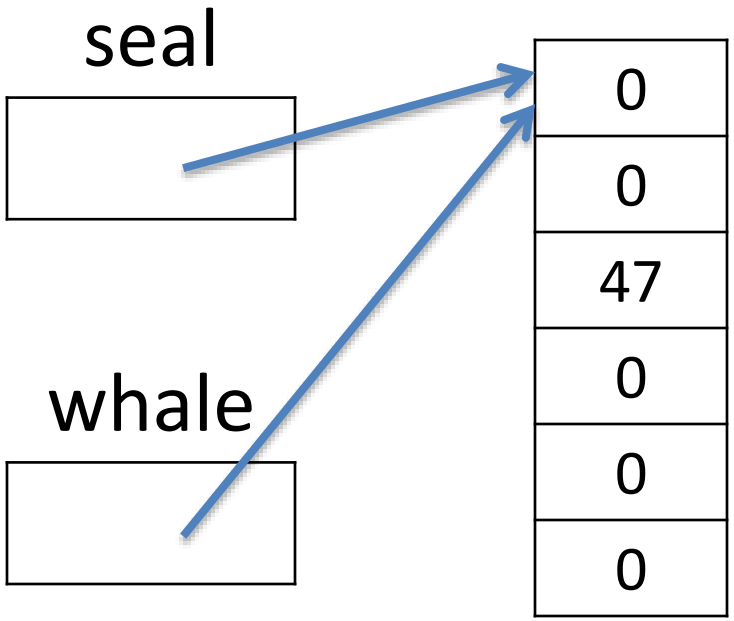
```
int[] seal;
int[] whale;
seal = new int[6];
whale = seal;
```



Changes to the Array

- Setting one array to another copies the reference
  Both variable reference the same array

```
int[] seal;
int[] whale;
seal = new int[6];
whale = seal;
seal[2] = 47;
```



System.out.println(whale[2]); displays 47

What is displayed?

```
String[] barn = {"cow",  "pig", "goat",   "chicken" };
System.out.println(barn.length);
```

A. cow
B. 3
C. 4
D. chicken

Array as a Parameter

- You can pass an array as a parameter to a method

```
void myMeth( double[] dog ){
… }
```

- Note that you do not have to specify the size of the array as a parameter
- You can always get the size from the **.length** variable

Passing Arrays

- When you pass the entire array to a method you do not use brackets

```
int[] dolphin = { 3, 5, 7, 11, 13, 17, 23 };
int otter;
otter = avgFirst( dolphin ) ;   // otter is 4
        …
int avgFirst( int[] porpoise ) {
        return (porpoise[0] + porpoise[1]) / 2;
}
```

What is displayed?

```
int[] dog = { 3, 5, 7 };
int[] cat = { 4, 8, 16};
dog[1] = 9;
cat = dog;
System.out.println(cat[1]);
```

A. 4 8 16

B. 3

C. 5

D. 8

E. 9

Write with your Team

- Complete this method that takes an array of doubles and returns the average of all the values in the array

```
double avg( double[] data )  {
```

Possible Solutions

```
double avg( double[] data ) {
   double sum = 0.0;
   for (int i = 0; i < data.length; i++) {
      sum = sum + data[i];
   }
   return sum / data.length;
}
```

or

```
double avg( double[] data ) {
   double sum = 0.0;
   for (double num : data) {
      sum = sum + num;
   }
   return sum / data.length;
}
```

Modify your method

- How would you change your method to return the average of only the **negative** values in the array?

Possible Solution

```
double avg( double[] data ) {
   double sum = 0.0;
   int numNeg = 0;
   for (double num : data) {
      if ( num < 0.0 ) {
         sum = sum + num;
         numNeg++;
      }
   }
   return sum / numNeg;
}
```

Possible Solution

```
double avg( double[] data ) {
   double sum = 0.0;
   int numNeg = 0;
   for (double num : data) {
      if ( num < 0.0 ) {
         sum = sum + num;
         numNeg++;
      }
   }
   return (numNeg==0) ? 0.0 : sum / numNeg;
}
```

Changing a Method's Parameters Does **Not** Change the Calling Arguments

- If a method changes the value of a **simple** parameter variable, it does not change the value of the variable in the calling program
- The values of simple argument variables are copied to the method parameters when the method is called, but are **not** copied back when the method returns

Parameter Values Not Changed

```
public class ModParm {
     public static void main(String[] args ){
        int   a = 5,;
        System.out.println("main a="+a);
        example( a );
        System.out.println("main a="+a);
     }
     static void example( int x ) {
         x = 13;      // parameter changed
         System.out.println("example x="+x);
     }
  }
```

Output

```
main a=5
example x=13
main a=5
```

Changing Array Parameters

- Unlike simple parameters, if a method changes a value in an array, the calling program will see the change

```
int[] seal = {2, 3, 5, 7, 11, 13};
aMethod( seal );
System.out.println( seal[2] );     // Prints 47
```

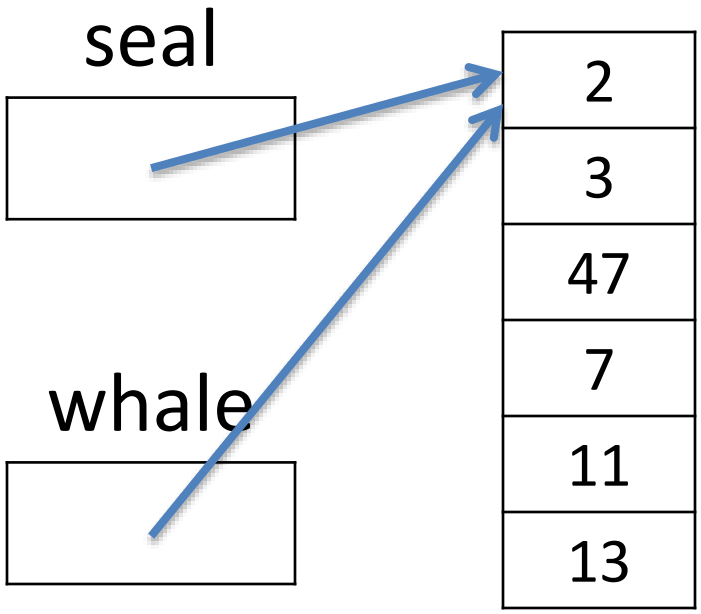
```
void aMethod(int[] whale) {
  whale[2] = 47;
}
```

Passing Array Reference

- When you pass an array to a method, the reference is copied to the method argument

```
int[] seal = {2, 3, 5, 7, 11, 13};
aMethod( seal );
System.out.println( seal[2] );

void aMethod(int[] whale) {
  whale[2] = 47;
}
```



Passing Individual Elements

- If you pass a single element of an array, it works just like passing a simple variable

```
int[] seal = {2, 3, 5, 7, 11, 13};
bMethod( seal[3] );
System.out.println( seal[3] );     // Prints 7
```

```
void bMethod(int polarBear) {
 polarBear = 51;
}
```

What is displayed?

```
int[] seal = {2, 3, 5, 7, 11, 13};
cMeth( seal[5] , seal );
System.out.println(seal[3]);

    . . .     . . .   . . .   . . .   . . .
void cMeth(int otter, int[] walrus ) {
   walrus[3] = otter;
}
```

A. 3

B. 5

C. 7

D. 13

E. none of the above

Arrays of Objects

- The elements of an array can be primitive type like int, double, long, float, char or boolean
- An array can also have objects as elements

```
Widget[] things = new Widget[47];
```

- This creates an array, things, that can contain 47 objects of the class Widget

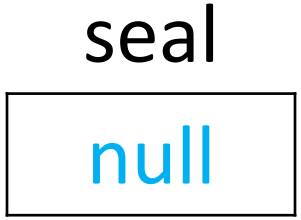
Empty Array

- When you create an array of double or int, the array initially contains many doubles or ints
- When you create an array of objects, the array is initially empty.

Creating an Array of Objects

- When you declare an array of objects. The variable does not yet hold the array

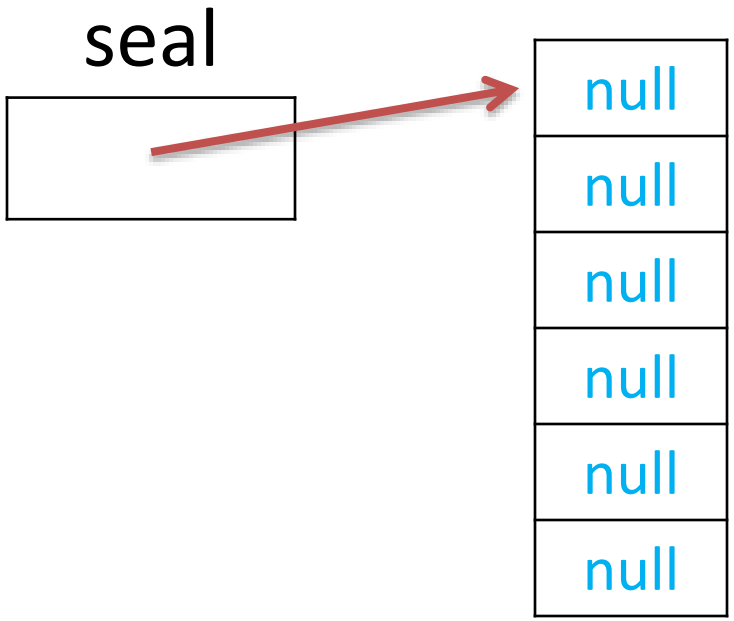
Widget[] seal;



Creating an Array of Objects

- An array is created the same way you create other objects. The **new** create the array

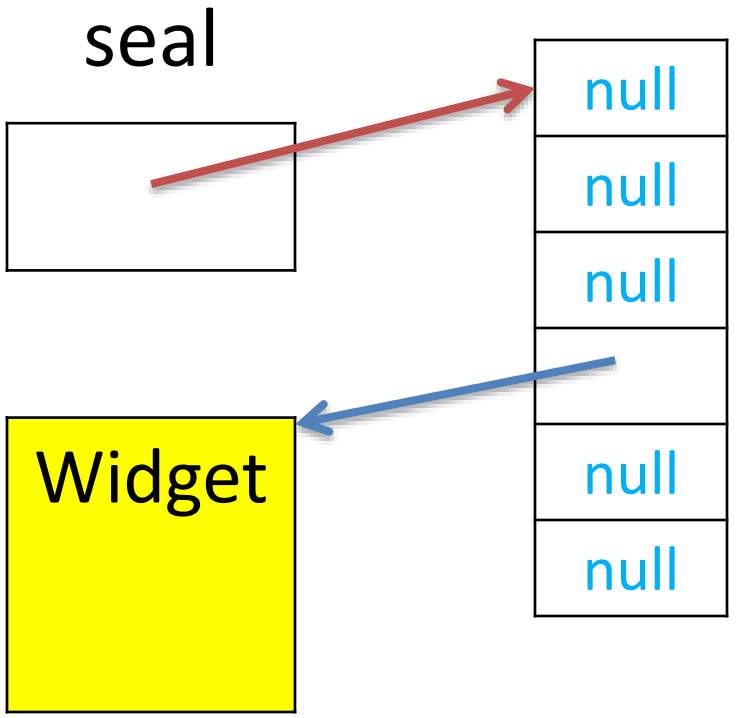
```
Widget[] seal;
seal = new Widget[6];
```



Creating an Array of Objects

- You have to create each of the element objects in the array The **new** creates each element

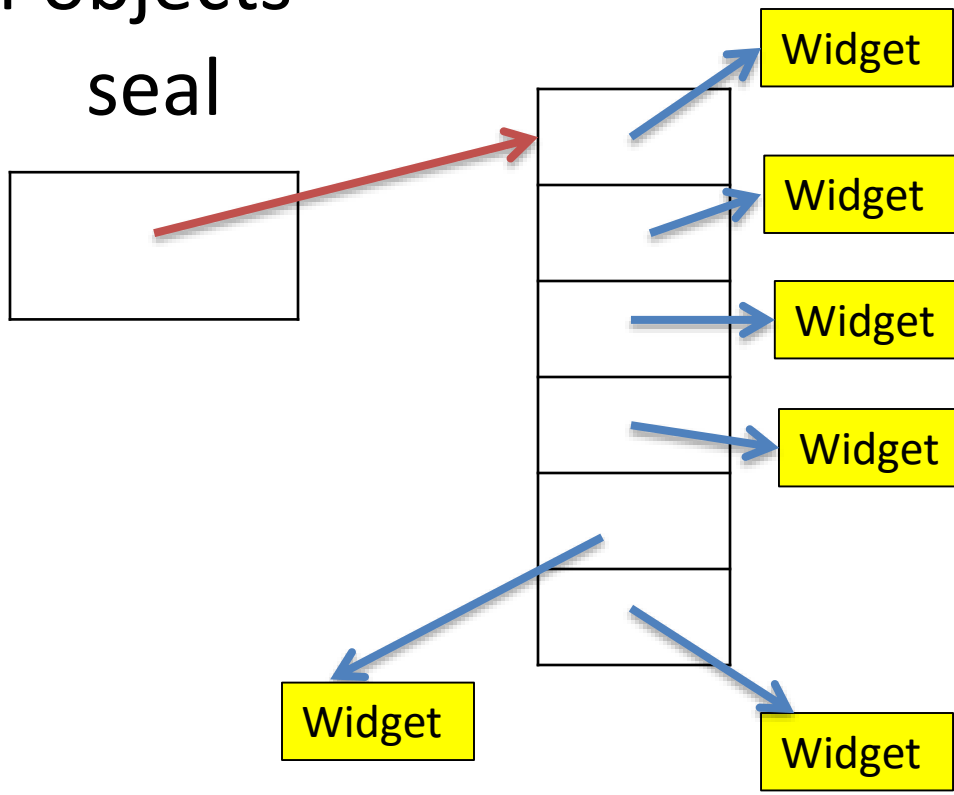
```
Widget[] seal;
seal = new Widget[6];
seal[3] = new Widget();
```



Creating an Array of Objects

- You should create a new object for each element of the array of objects

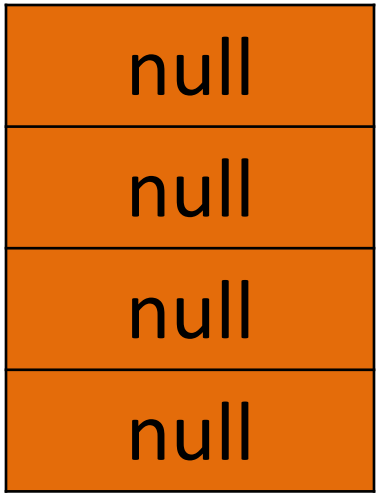
```
Widget[] seal;
seal = new Widget[6];
for(int i=0; i<6; i++)
   seal[i] = new Widget();
```



NullPointerException

- If you do not create the objects of an array, you may get a NullPointerException error

```
String[] stuff = new String[4];
stuff[2].length;            // error
```



Try it

- Create an array of 6 Rooster objects and fill each element with an object

Possible Solution

```
Roster[] panther = new Rooster[6];
for (int i = 0; i < 6; i++) {
    panther[i] = new Rooster();
}
```