Yet More on Arrays

GEEN163

"It has been said that man is a rational animal. All my life I have been searching for evidence which could support this."

Bertrand Russell

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

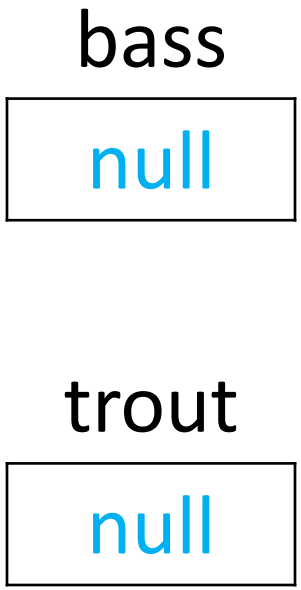
```
String[] holiday = { "Christmas",
                     "Thanksgiving", "Independence Day" };
```

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

Object References

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

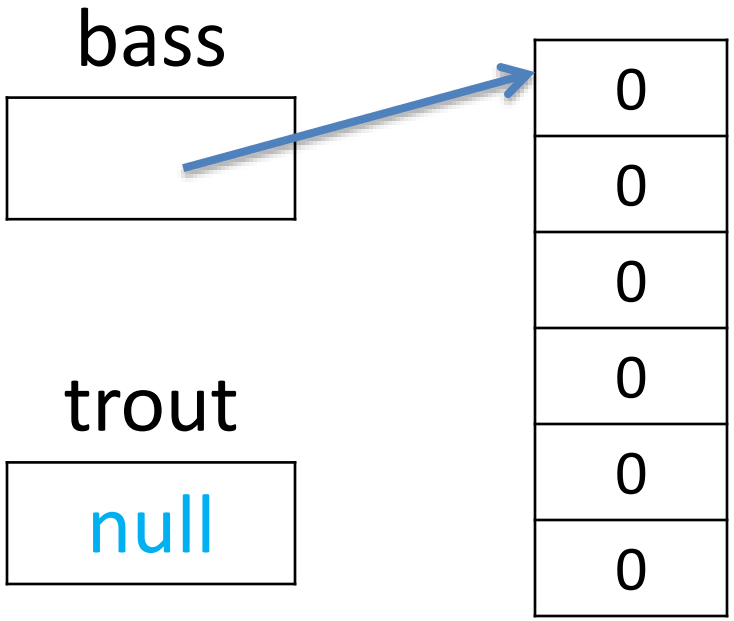
```
int[] bass;
int[] trout;
```



Creating an Array Object

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

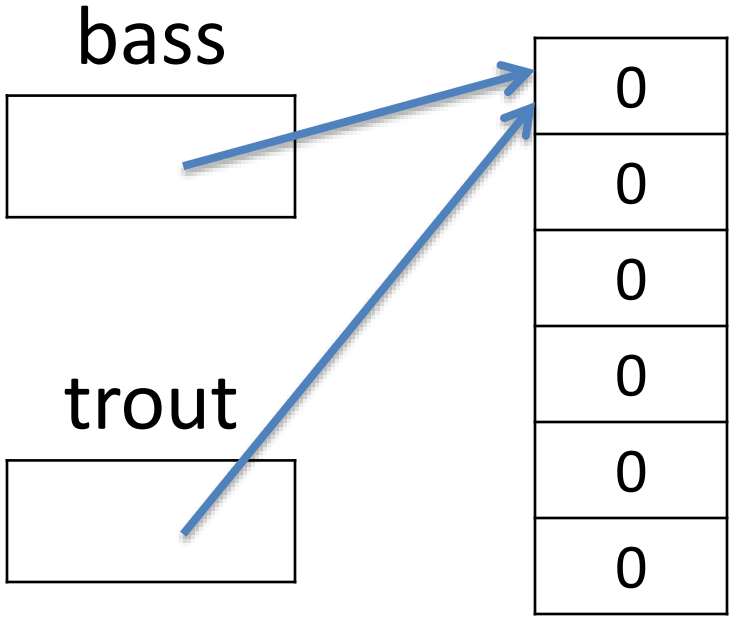
```
int[] bass;
int[] trout;
bass = new int[6];
```



Multiple Names for the Same Array

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

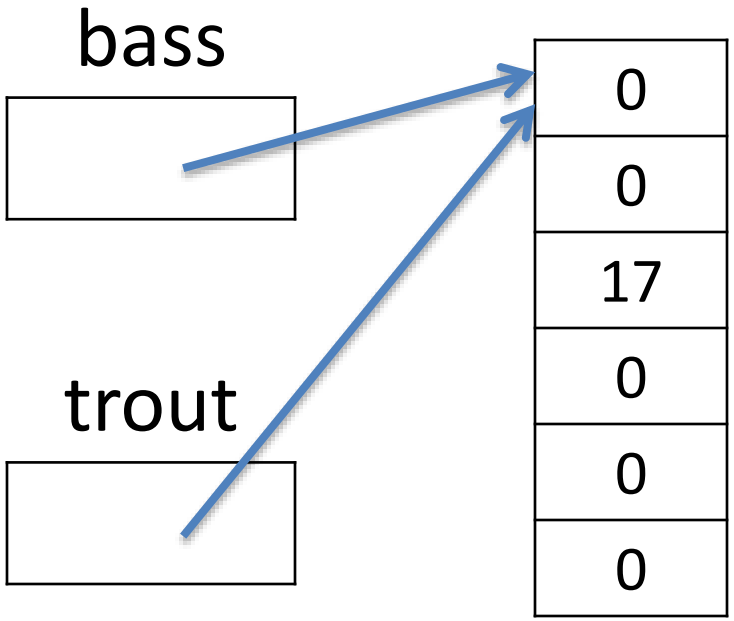
```
int[] bass;
int[] trout;
bass = new int[6];
trout = bass;
```



Changes to the Array

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

```
int[] bass;
int[] trout;
bass = new int[6];
trout = bass;
bass[2] = 17;
System.out.println(trout[2]); displays 17
```



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

```
Circle[] balls = new Circle[5];
```

- This creates an array, things, that can contain 5 objects of the class **`Circle`**

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.

Initial Values

- When you create an array, the elements of the array are initialized

int	0
double	0.0
String	null
Class	null

Creating an Array of Objects

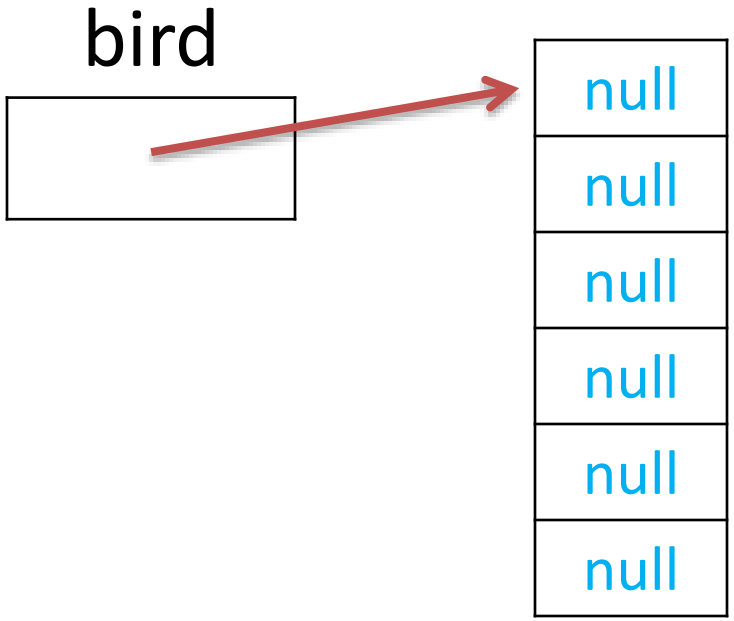
- When you declare an array of objects. The variable does not yet hold the array

```
Widget[] bird;
```

Creating an Array of Objects

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

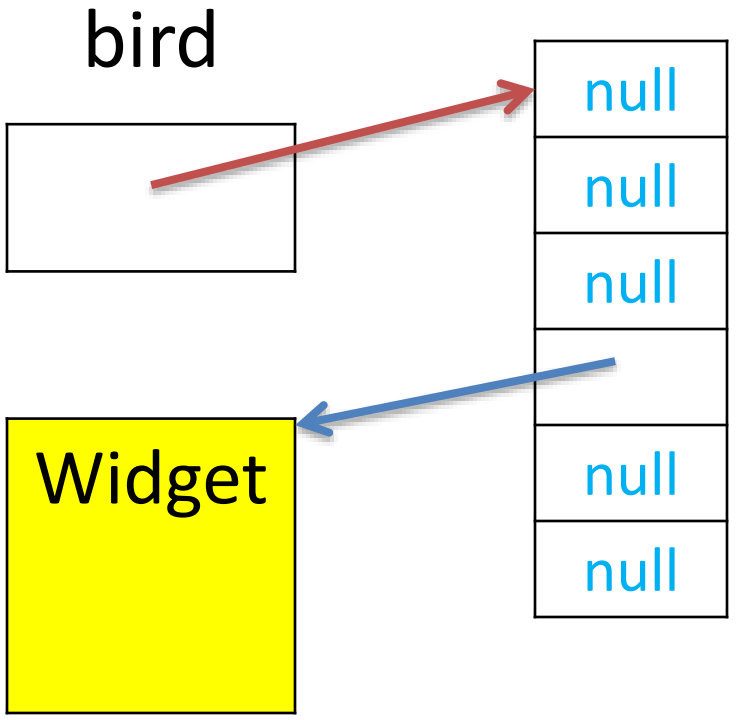
```
Widget[] bird;
bird = new Widget[6];
```



Creating an Array of Objects

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

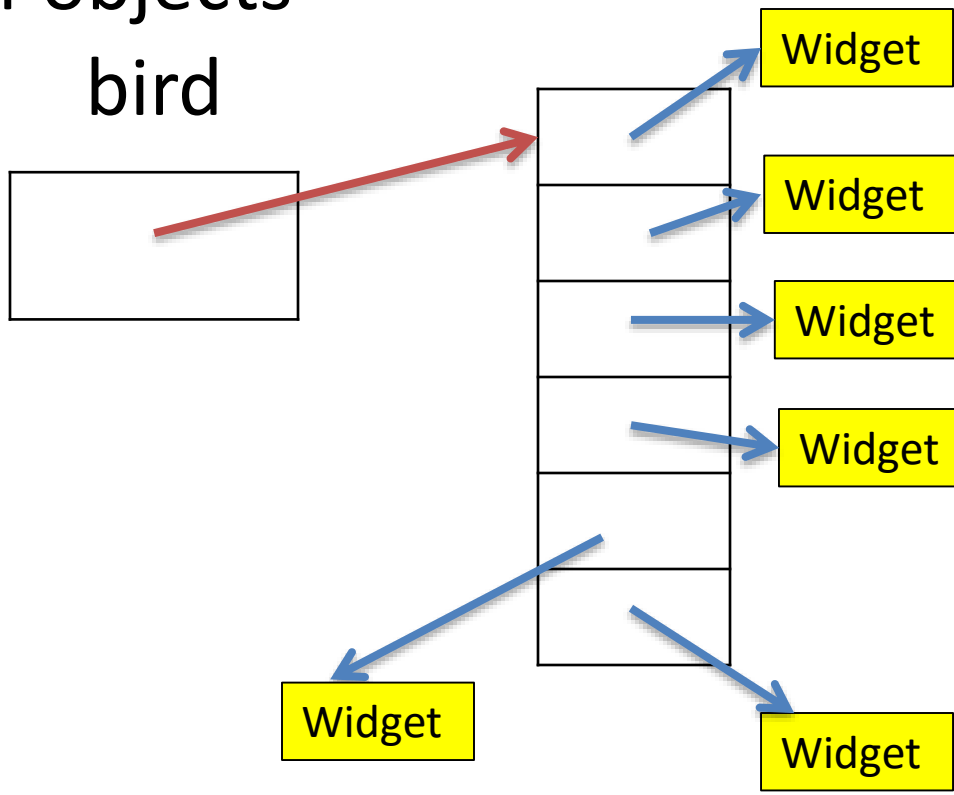
```
Widget[] bird;
bird = new Widget[6];
bird[3] = new Widget();
```



Creating an Array of Objects

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

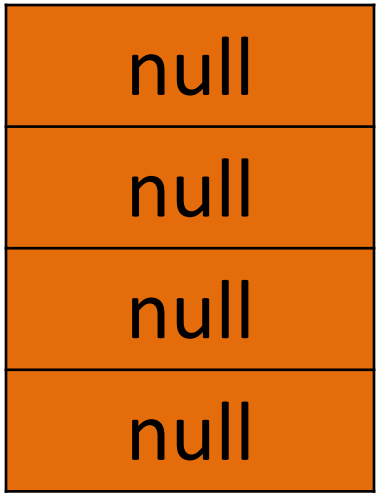
```
Widget[] bird;
bird = new Widget[6];
for(int i=0; i<6; i++)
   bird[i] = new Widget();
```



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
```



Consider a Box Class

```
public class Box {
   int serialNumber;        // serial number of box
   public Box( int sn ) {   // constructor
      serialNumber = sn;    // initialize serialNumber
   }
}
```

What creates 25 Box objects with unique serial numbers?

A.
```
Box[] crate = new Box[25];
for (int i=0; i<25; i++)
  crate[i] = new Box(i);
```

B.
```
Box crate = new Box[];
for (int i=0; i<25; i++)
  crate[i] = new Box(25);
```

C.
```
Box[] crate = new Box(25);
```

Simple vs. Objects

- When you create an array of 1000 ints or doubles, Java creates 1000 ints or doubles
- When you create an array of 1000 objects, Java creates the array, but you must also create the objects

Try it

- With your team, write a Java segment to create an array called **`sqr`** of 1000 doubles and initialize them to the square of their index, i.e. sqr[1] = 1.0, sqr[2] = 4.0;

Possible Solution

```
double[] sqr = new double[1000];
for (int ayeaye = 0; ayeaye < 1000; ayeaye ++) {
        sqr[ayeaye] = (double)(ayeaye * ayeaye);
}
```

Searching an Array

- A common task to do with an array is to search if a particular value is in the array
- A loop will be required. You do not know if the value is not in the array until you check all elements of the array

Search an Array

```
String[] pet = {"dog", "cat", "canary", "gerbil"};
String animal = keyboard.next();          // animal to find
boolean gotit = false;                    // true if found
look: for (int i = 0; i < pet.length; i++) {
        if ( animal.equalsIgnoreCase( pet[i] ) ) {
                gotit = true;             // mark found
                break look;               // stop looking
        }
}
if ( gotit ) {              // was the animal found?
        System.out.println("We have a "+animal);
} else {
         System.out.println("No "+animal+" here");
}
```

The previous program will work if we remove the break statement.

A. True it will work
B. False it will not work

Searching an Array of Objects

- Classes are used to keep information about an idea together
- A program may have an array of object where you want to find a specific object

Example Class

```
public class Student {
    private String name;
    private double gpa;
    public Student( String who, double grade){
        name = who;
        gpa = grade;
    }
    public String getName() {
        return name;
    }
}
```

Search an Array

```
Student[] group = new Student[120];

        // assume Student objects created

String target = "Fred";
boolean gotit = false;                          // true if found
look: for (int i = 0; i < group.length; i++) {
        if ( target.equals( group[i].getName() ) ) {
                gotit = true;                   // mark found
                break look;                     // stop looking
        }
}
```

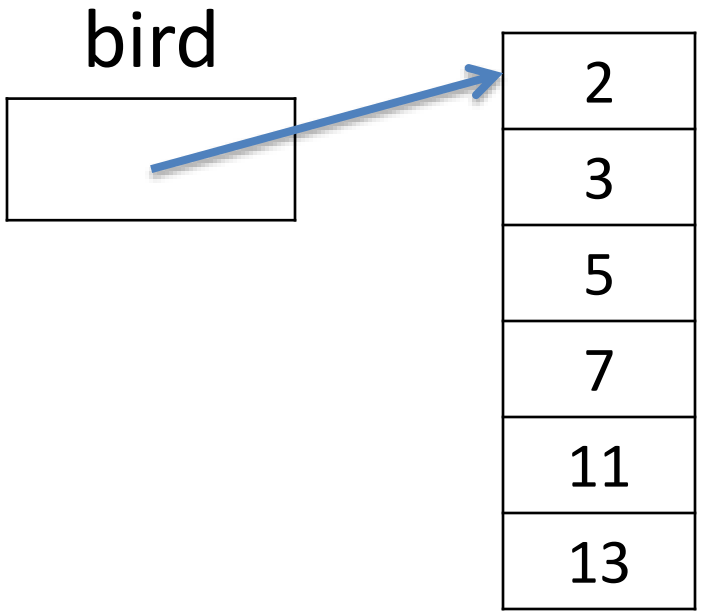
Search an Array Extended for

```
Student[] group = new Student[120];

        // assume Student objects created

String target = "Fred";
boolean gotit = false;                          // true if found
look: for (Student who : group) {
        if ( target.equals( who.getName() ) ) {
                gotit = true;                   // mark found
                break look;                     // stop looking
        }
}
```

Passing Array Reference

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

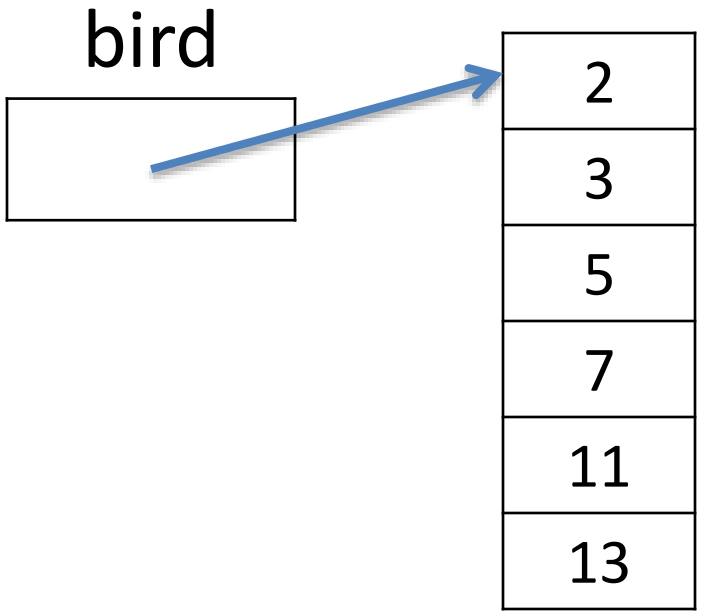
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Passing Array Reference

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

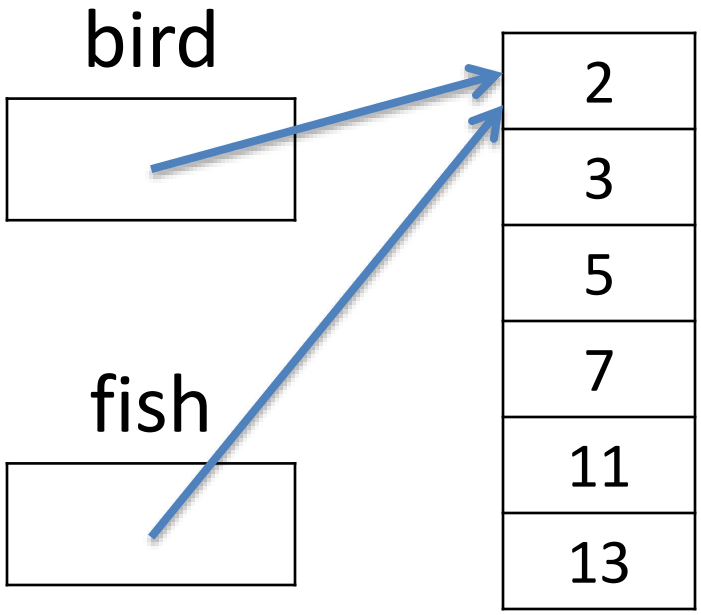
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Passing Array Reference

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

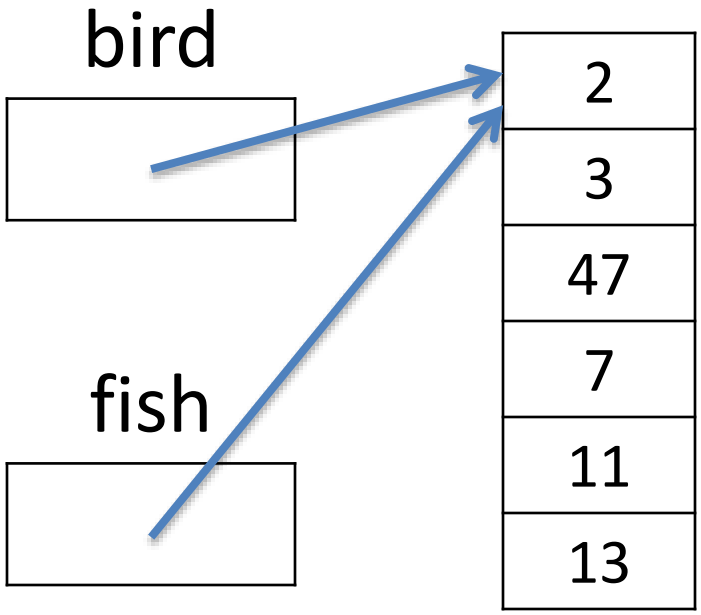
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Passing Array Reference

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

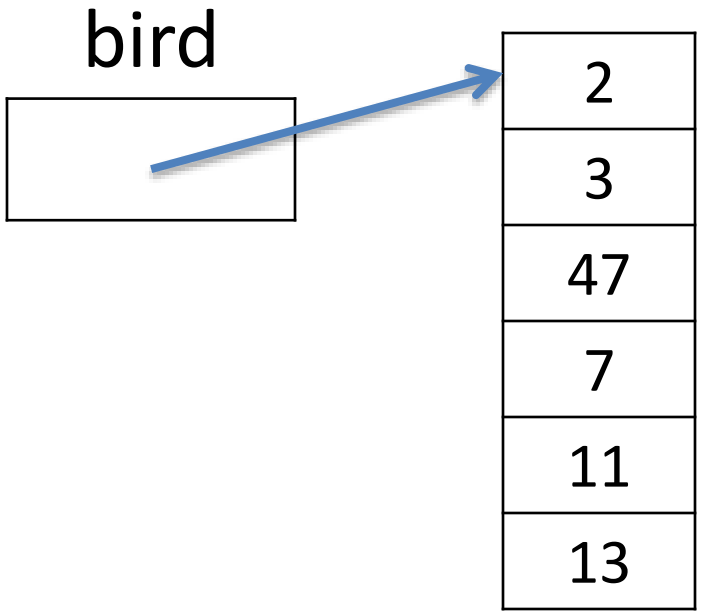
```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Passing Array Reference

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

```
int[] bird = {2, 3, 5, 7, 11, 13};
aMethod( bird );
System.out.println( bird[2] );
      . . .
void aMethod(int[] fish) {
  fish[2] = 47;
}
```



Histogram

- The drunk driver program requires you to have two arrays of 24 integers initialized to zero
  - number of accidents
  - number of accidents with drunks
- For each line of data in the file, use the hour as an index into the arrays
- Increment the accident counts

```
accident[hour]++;
```

What is displayed?

```
int[] hare= {8, 11, 4};
int  bunny = 22;
qMeth( bunny , hare);
for(int i = 0; i < 3; i++)
   System.out.print(hare[i]);
     . . .      . . .    . . .    . . .
void qMeth(int pika, int[] rabbit ) {
   rabbit[1] = pika;
}
```

A. 8 11 4

B. 8 22 4

C. 22 11 4

D. none of the above

Write a method with your team

- Write a method that returns the index of where a string is in an array or -1 if the target is not in the list

```
int where(String target, String[] list)
```

Example:

- target = "dog"
- list = "cat", "dog", "cow"
- where(dog, list) returns 1

Possible Solution

```
int where(String target, String[] list) {
   int index = -1;
   for (int loc = 0; loc < list.length; loc++) {
      if (target.equals( list[loc] )) {
         index = loc;
         break;
      }
   }
   return index;
}
```

Another Possible Solution

```
int where(String target, String[] list) {
   for (int loc = 0; loc < list.length; loc++) {
      if (target.equals( list[loc] )) {
         return loc;
      }
   }
   return -1;
}
```

Method Review

- Follow the flow of execution. Execution starts in the main method
- Constructors can be used to initialize objects

What is displayed?

```
public class ClickQ {
    int lion = 1;

    public ClickQ ( int tiger) {
        lion = tiger;
    }

    public static void main(String[] args) {
        ClickQ ohmy = new ClickQ( 2 );
        System.out.println( ohmy.lion );
    }
}
```

A. 1 B. 2 C. 5 D. 6 E. 7

What is displayed?

```
public class ClickQ {
    int lion = 1;

    public void forest( int bear) {
        lion = lion + bear;
    }

    public static void main(String[] args) {
        ClickQ ohmy = new ClickQ();
        ohmy.forest( 5 );
        System.out.println( ohmy.lion );
    }
}
```

A. 1 B. 2 C. 5 D. 6 E. 7

What is displayed?

```
public class ClickQ {
    int lion = 1;
    public ClickQ ( int tiger) {
        lion = tiger;
    }
    public void forest( int bear) {
        lion = lion + bear;
    }
    public static void main(String[] args) {
        ClickQ ohmy = new ClickQ( 2 );
        ohmy.forest( 5 );
        System.out.println( ohmy.lion );
    }
}
```

A. 1 B. 2 C. 5 D. 6 E. 7

Static Variables

- Class variables can be specified as static

```
public class electric {
   static  int cat;   // static class variable
   int dog;           // non- static class variable
}
```

- There is only one copy of a static variable shared by all objects of that class
- There is a separate copy of non-static variables for each object

Array Summary

- Arrays are declared by

```
int[] dog = new int[size];
```

- When you create an array of objects, you have to create the objects separately
- Array elements are always used with brackets
- If you pass an array element to a method, you cannot change its value in the calling program
- If you pass an entire array to a method, you can change its value in the calling program

Array Summary (cont.)

- Arrays are almost always used with loops
- Array indexes range from zero to one less than the size of the array
- You can get the maximum array size with arrayname.length