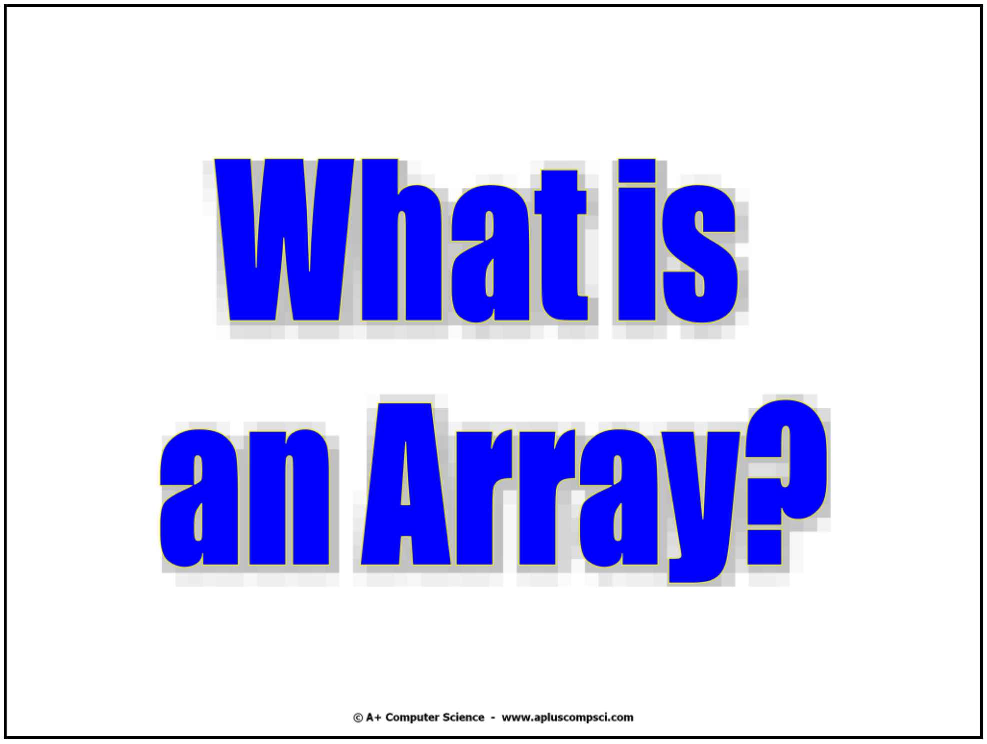
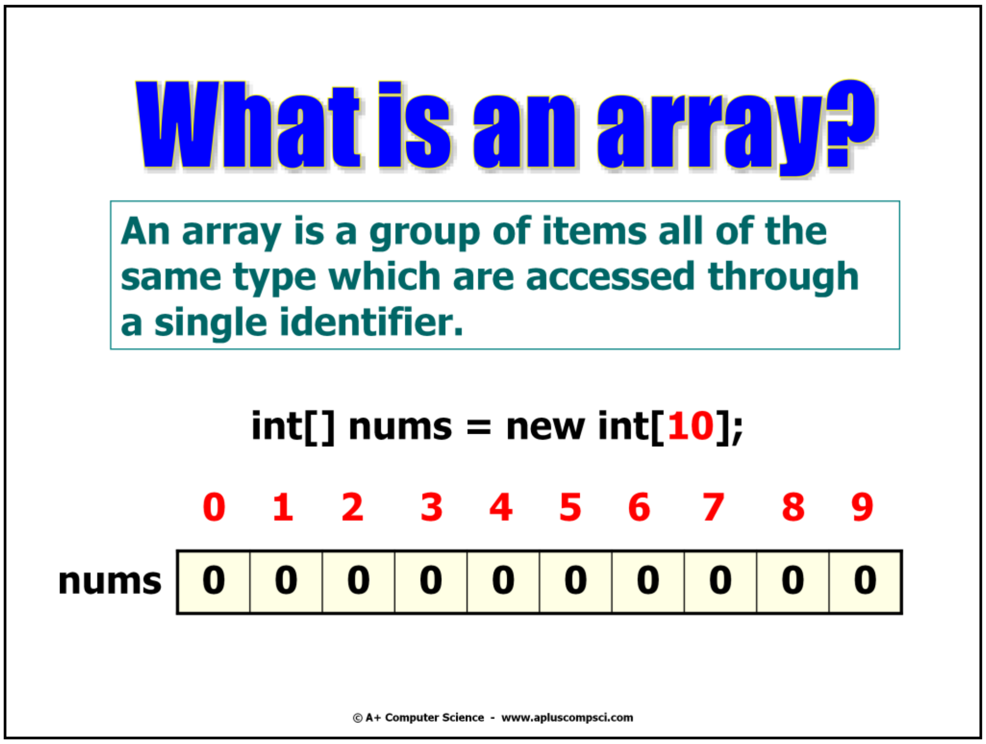
Arrays and Lists

What is a list?

	Name	Time	Artist	Album	Ge
5	☑ I Dare You to Move	4:08	Switchfoot	Learning to Breathe	
6	☑ I've Been Everywhere	3:20	Johnny Cash	Unchained	
7	☑ Brown Eyed Girl (Single Version)	3:05	Van Morrison	Super Hits	
8	☑ Born to Be Wild	3:31	Steppenwolf	Steppenwolf: All Time Greatest	
9	☑ Magic Carpet Ride	4:28	Steppenwolf	Steppenwolf: All Time Greatest	
10	☑ Crazy (Single Version)	2:42	Patsy Cline	Patsy Cline's Greatest Hits (Rer	
11	☑ Brick House	3:46	The Commodores	20th Century Masters - The Mil	
12	☑ Cleveland Rocks	2:33	The Presidents of the…	Pure Frosting	
13	☑ Chariots of Fire: Main Title Theme	3:32	Carl Davis & Royal Li…	Great Movie Themes	
14	☑ Dueling Banjos (From "Deliverance")	3:11	The Hit Crew	Smash Hit Dramas Movie Theme	
15	☑ Main Theme (From "Superman")	4:12	John Williams	The Music of John Williams - 40	
16	☑ Main Theme (From "Superman")	4:12	John Williams	The Music of John Williams - 40	
17	☑ I've Been Everywhere	3:20	Johnny Cash	Unchained	
18	☑ Born to Be Wild	3:31	Steppenwolf	Steppenwolf: All Time Greatest	



What is an Array?



What is an array?

An array is a group of items all of the same type which are accessed through a single identifier.

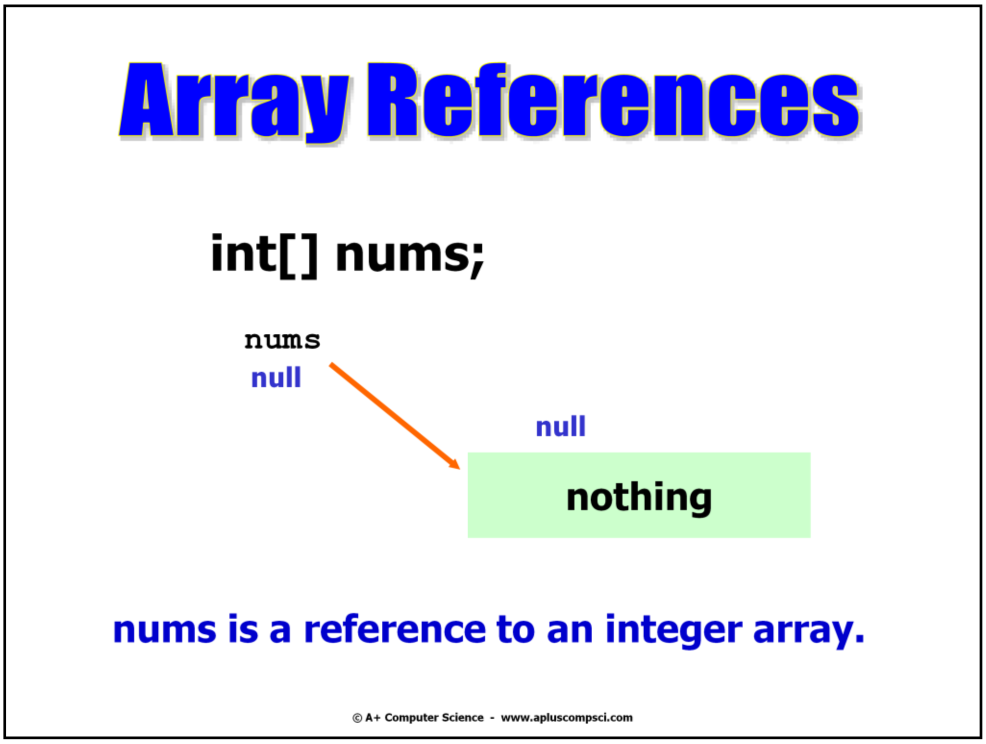
```
int[] nums = new int[10];
```

	0	1	2	3	4	5	6	7	8	9
nums	0	0	0	0	0	0	0	0	0	0

An array is a collection of boxes / spots / items that all store the same type of value. Each spot in the array stores a value of the same type.

Each spot in the array is essentially a single variable of the type specified.

`int[] array = new int[10];` array can store 10 integers. array is basically a collection of 10 integer variables. Spot 0 stores the 1st integer, spot 1 stores the 2nd integer, and so on.



Array References

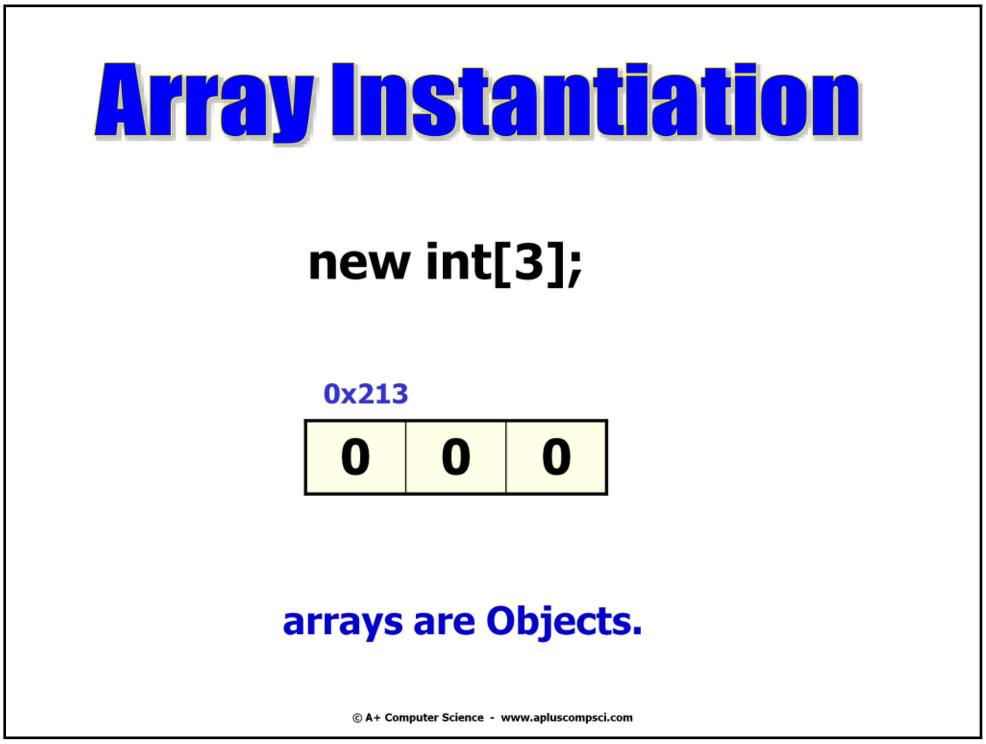
```
int[] nums;
```

nums
null
null
nothing

nums is a reference to an integer array.

A reference variable is used to store the location of an Object. In most situations, a reference stores the actual memory address of an Object.

`nums` stores the location / memory address of an integer array.



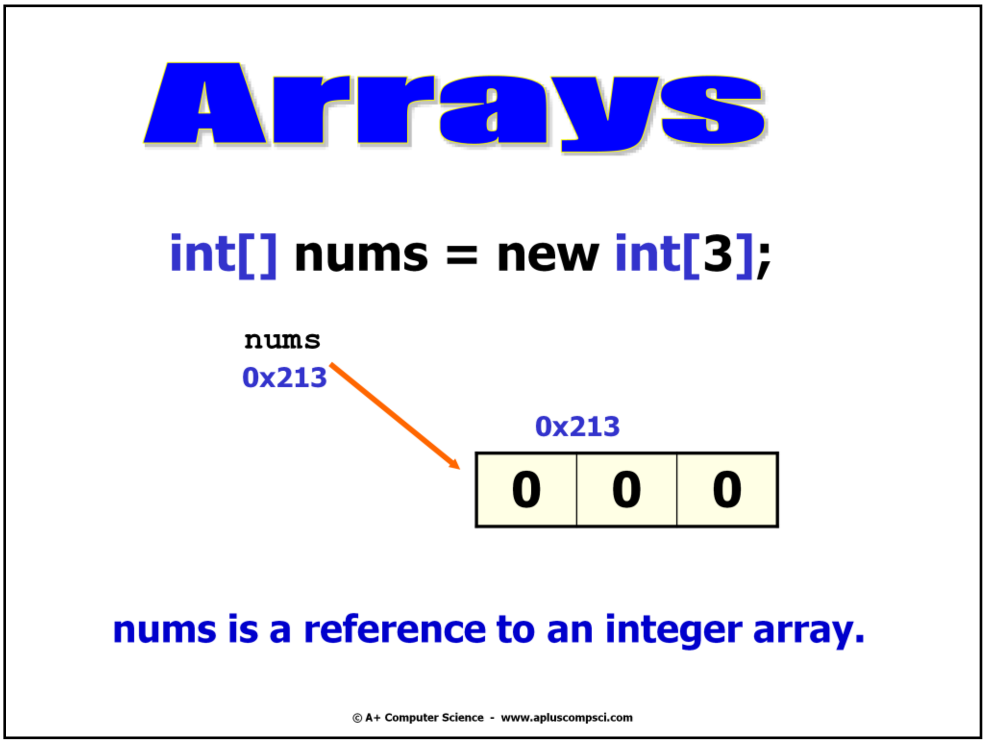
Array Instantiation

```
new int[3];
```

arrays are Objects.

A reference variable is used to store the location of an Object. In most situations, a reference stores the actual memory address of an Object.

In the example above, an array Object has been instantiated. There is nothing referring to the Object.



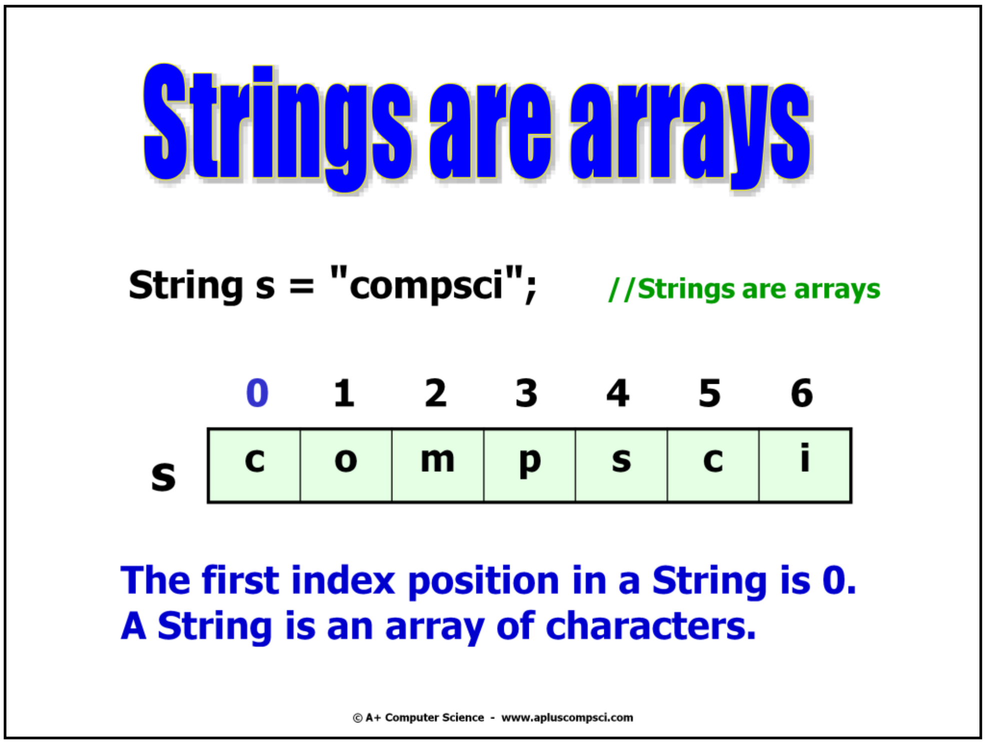
Arrays

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

nums is a reference to an integer array.

A reference variable is used to store the location of an Object. In most situations, a reference stores the actual memory address of an Object.

`nums`stores the location / memory address of an integer array.



Strings are arrays

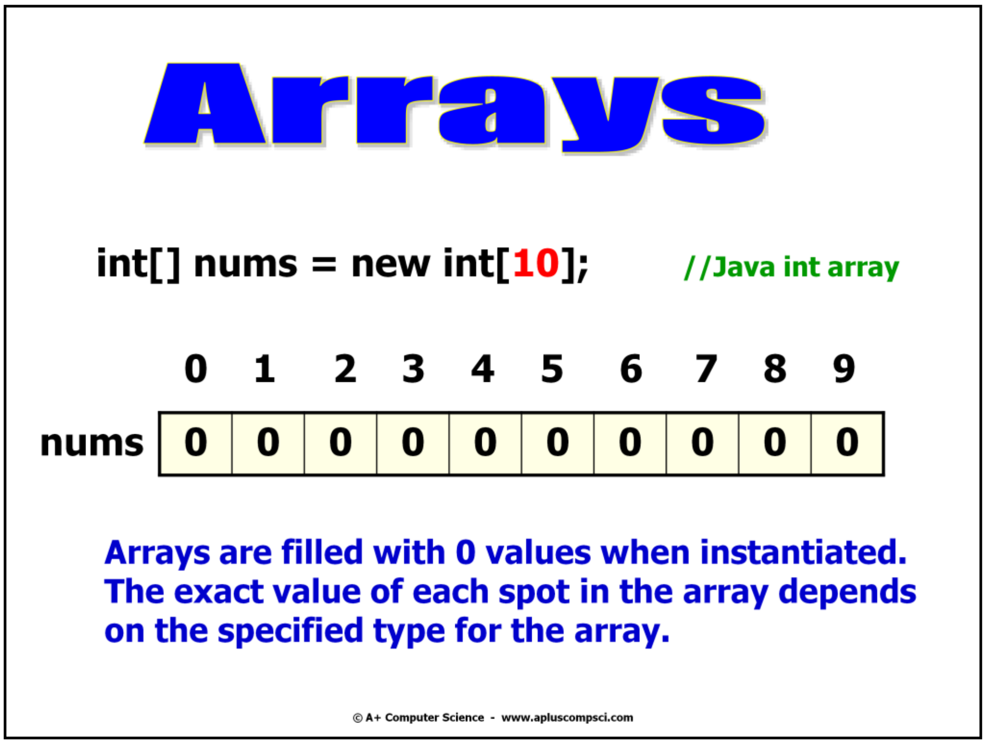
```
String s = "compsci";      //Strings are arrays
```

	0	1	2	3	4	5	6
s	c	o	m	p	s	c	i

The first index position in a String is 0.
A String is an array of characters.

© A+ Computer Science - www.apluscompsci.com

A String is a character array. String s is storing `"compsci"`. Spot [0] is storing character c and spot [length-1] is storing character i. Each spot in the String s is storing a single character.



Arrays

```
int[] nums = new int[10];        //Java int array
```

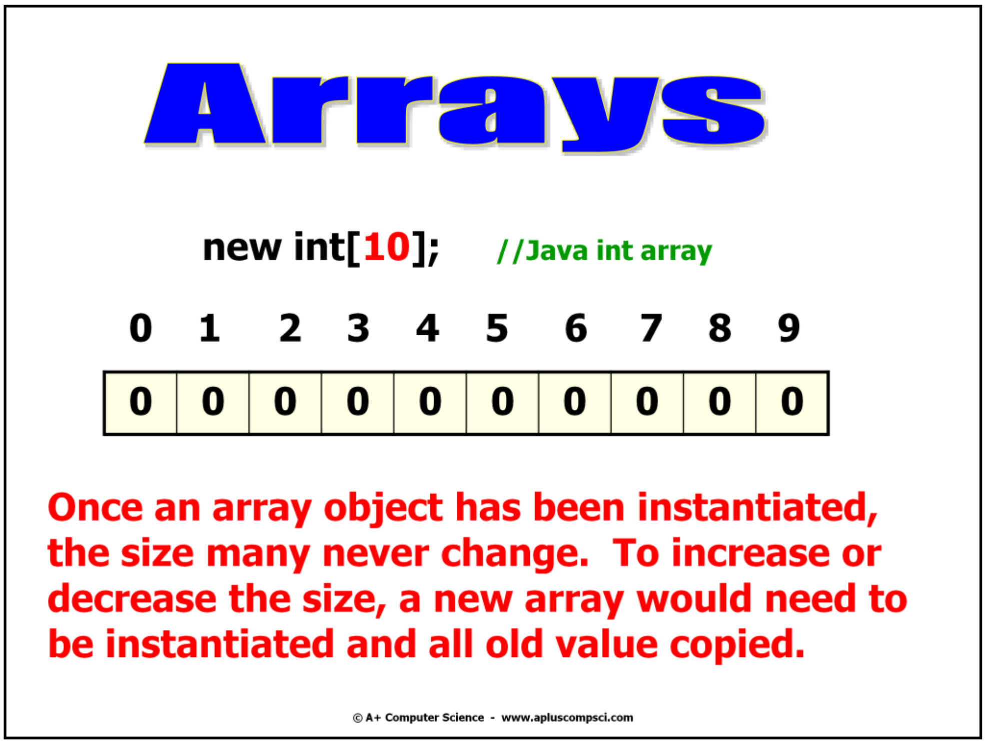
	0	1	2	3	4	5	6	7	8	9
nums	0	0	0	0	0	0	0	0	0	0

Arrays are filled with 0 values when instantiated. The exact value of each spot in the array depends on the specified type for the array.

When arrays are instantiated, each spot / box is filled with a zero value.

Integers have a zero value of 0, doubles have a zero value of 0.0, and characters have a zero value of 0 which happens to be a space.

A reference array would be filled with null.   Arrays of references will be discussed later.



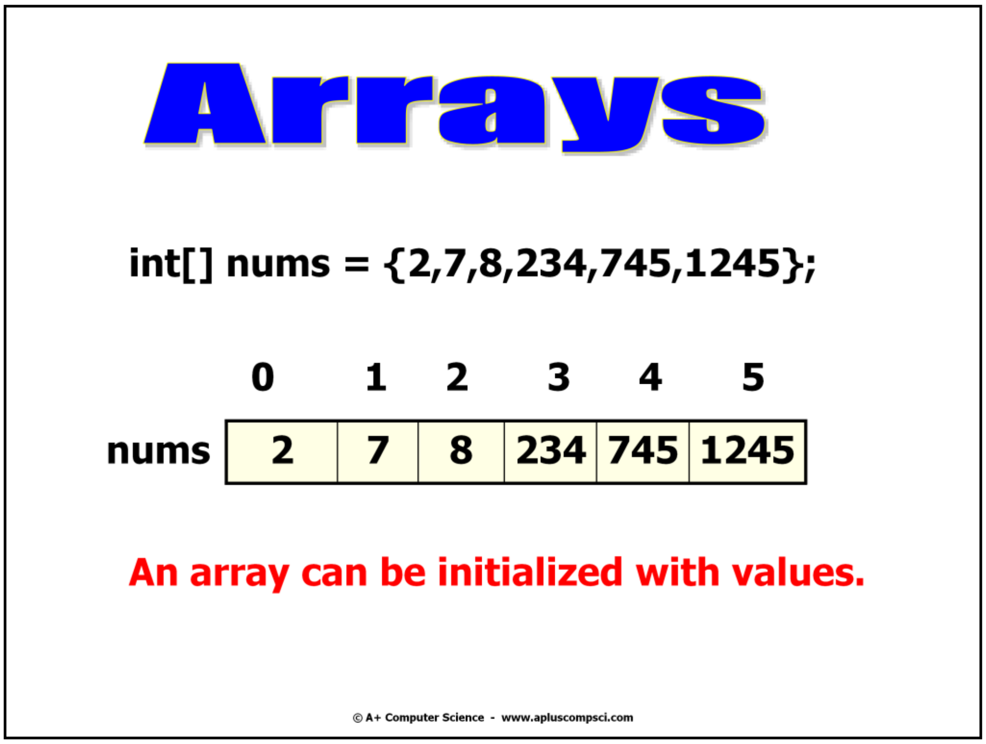
Arrays

```
new int[10];     //Java int array
```

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

Once an array object has been instantiated, the size many never change. To increase or decrease the size, a new array would need to be instantiated and all old value copied.

The size of an array object can never change. Arrays do not have methods that allow for the removal or addition of items. In order to add or remove items, a new array would be instantiated and all old values copied to the new array.



Arrays

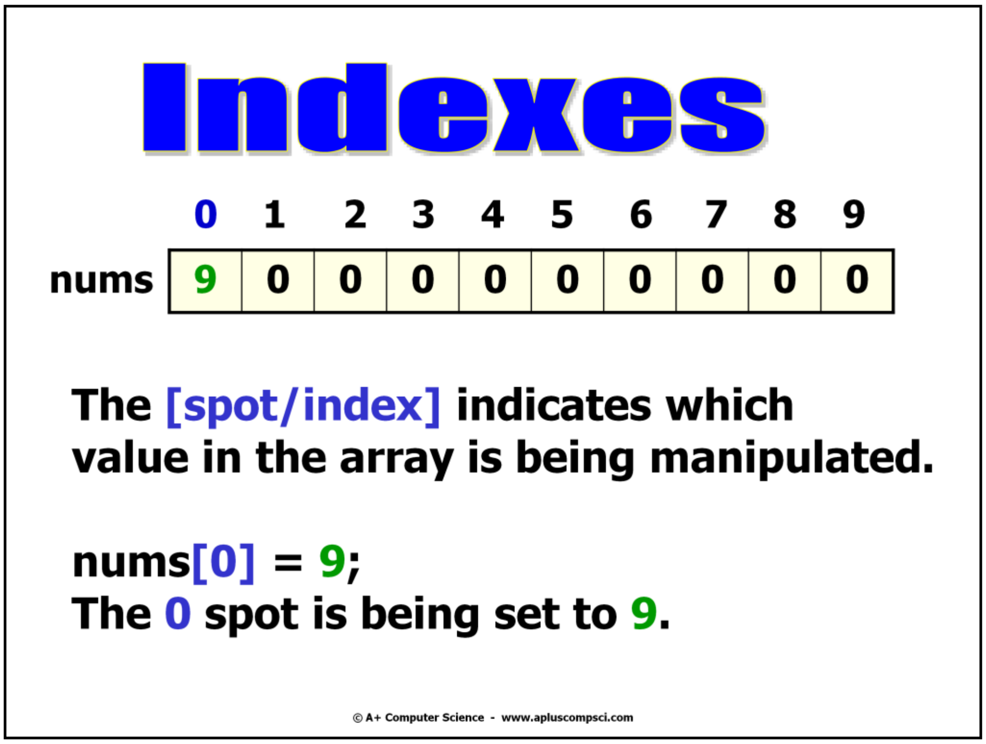
```
int[] nums = {2,7,8,234,745,1245};
```

	0	1	2	3	4	5
nums	2	7	8	234	745	1245

An array can be initialized with values.

Instantiating an array with a list of values is a great way to save some time if the values the array will store are known.

In the example above, nums is initialized with the value list 2,7,8,234,745,1245. Spot [0] is storing 2 and spot [length-1] is storing 1245.



Indexes

	0	1	2	3	4	5	6	7	8	9
nums	9	0	0	0	0	0	0	0	0	0

The [spot/index] indicates which value in the array is being manipulated.

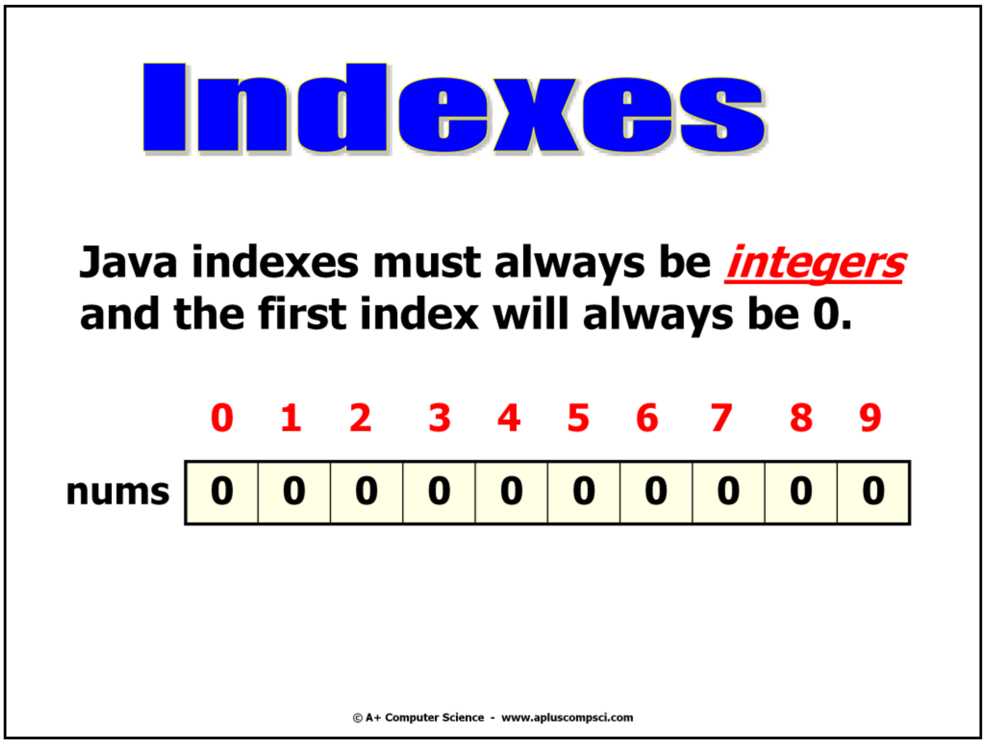
nums[0] = 9;
The 0 spot is being set to 9.

© A+ Computer Science - www.apluscompsci.com

Individual spots in an array are accessed by using a number. The number indicates which spot you are accessing. Only integer values can be used to [access] a spot in an array.

[int only]

```
out.println(nums[3]);   //outs 0
out.println(nums[0]);   //outs 9
```



Indexes

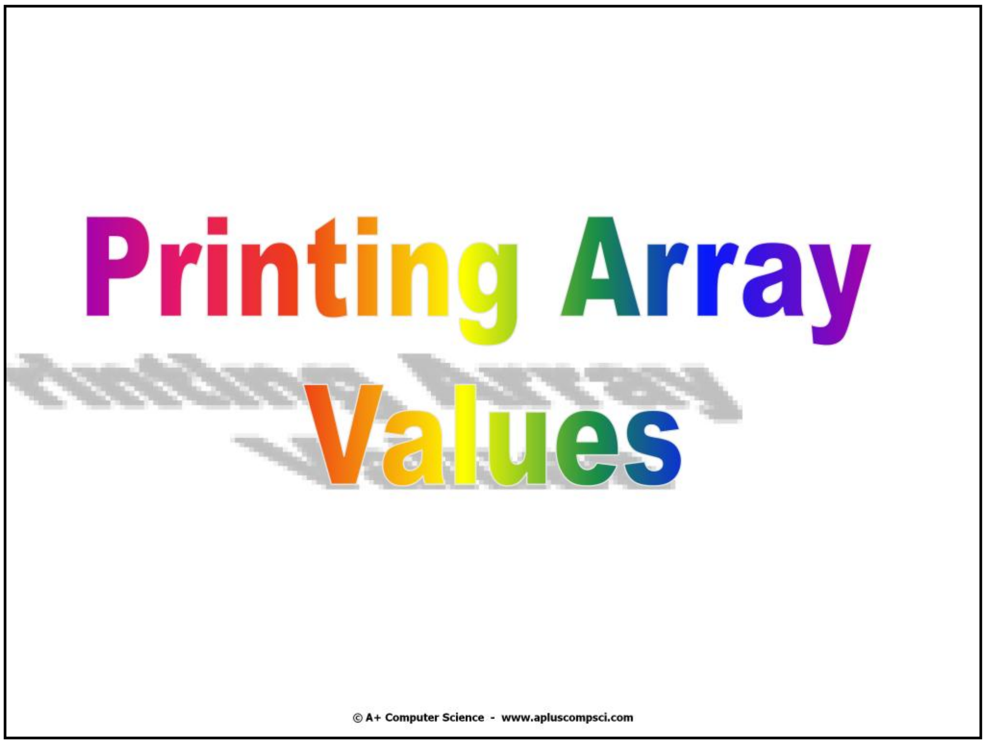
Java indexes must always be *integers* and the first index will always be 0.

0 1 2 3 4 5 6 7 8 9
nums
0 0 0 0 0 0 0 0 0 0

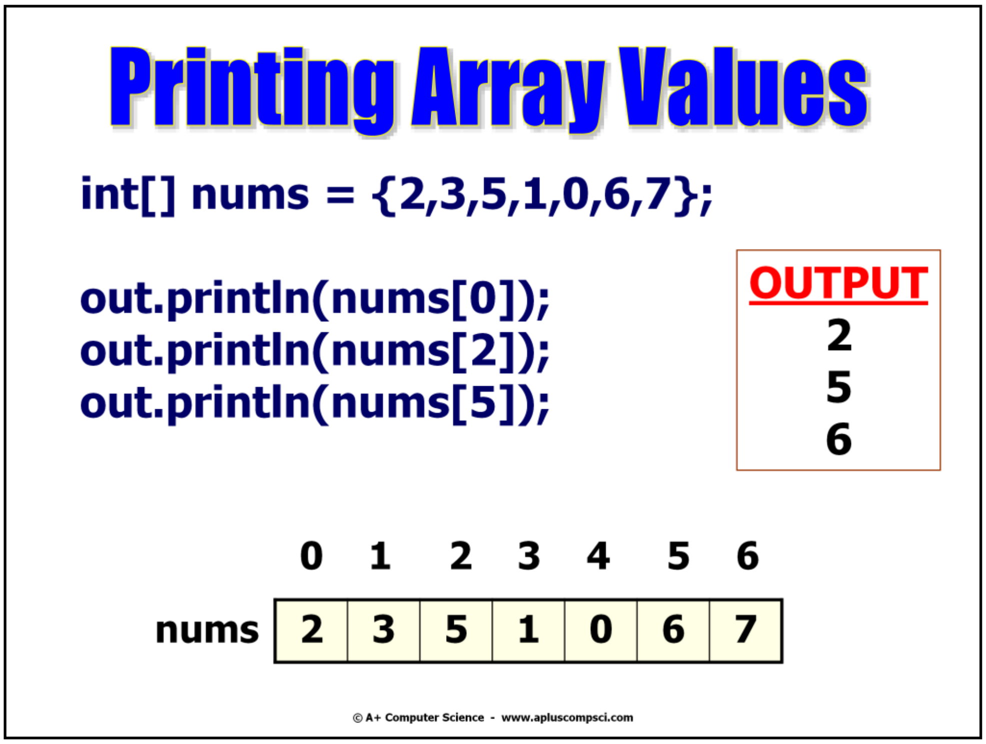
Individual spots in an array are accessed by using a number. The number indicates which spot you are accessing. Only integer values can be used to [access] a spot in an array.

[int only]

arrayinit.java



Printing Array Values



Printing Array Values

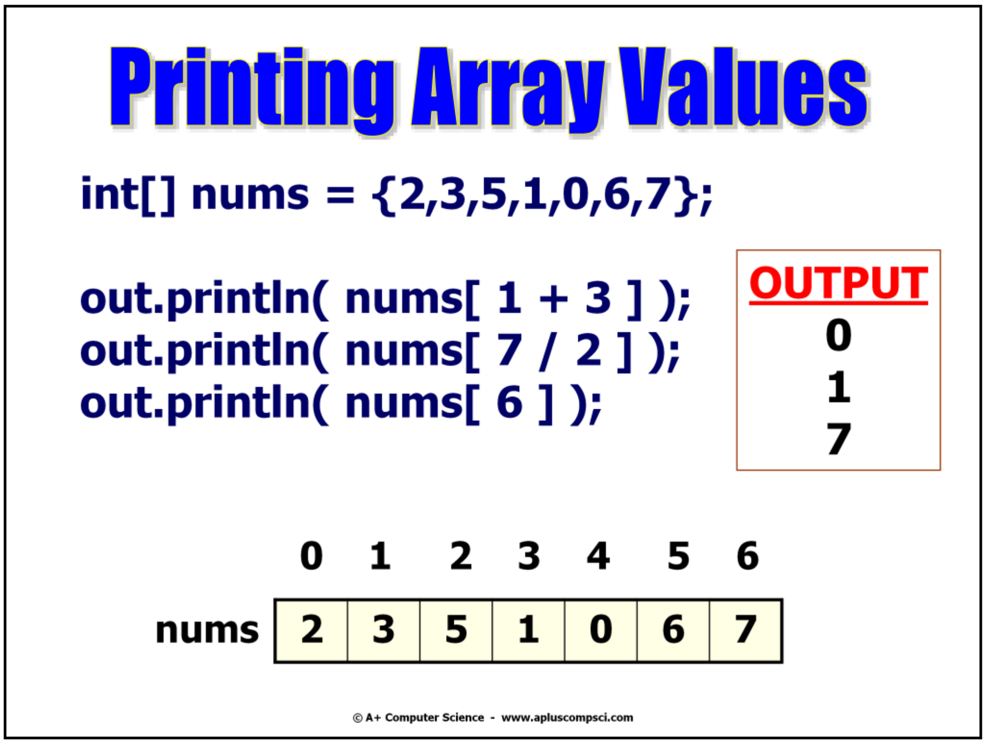
```
int[] nums = {2,3,5,1,0,6,7};

out.println(nums[0]);
out.println(nums[2]);
out.println(nums[5]);
```

OUTPUT

```
2
5
6
```

Once the array has been instantiated and has values, it is very simple to print/access a particular spot. An integer value must be provided to indicate which [spot] will be accessed.

```
int[] thoseNums = {5,7,3,6,9};
out.println(thoseNums[3]);     //outs 6


out.println(thoseNums[1/2]);    //outs 5
       // 1/2 is 0


out.println(thoseNums[2+2]);    //outs 9
out.println(thoseNums[5/2]);    //outs 3
       //  5/2 is 2
```



Printing Array Values

```
int[] nums = {2,3,5,1,0,6,7};

out.println( nums[ 1 + 3 ] );
out.println( nums[ 7 / 2 ] );
out.println( nums[ 6 ] );
```

OUTPUT

```
0
1
7
```

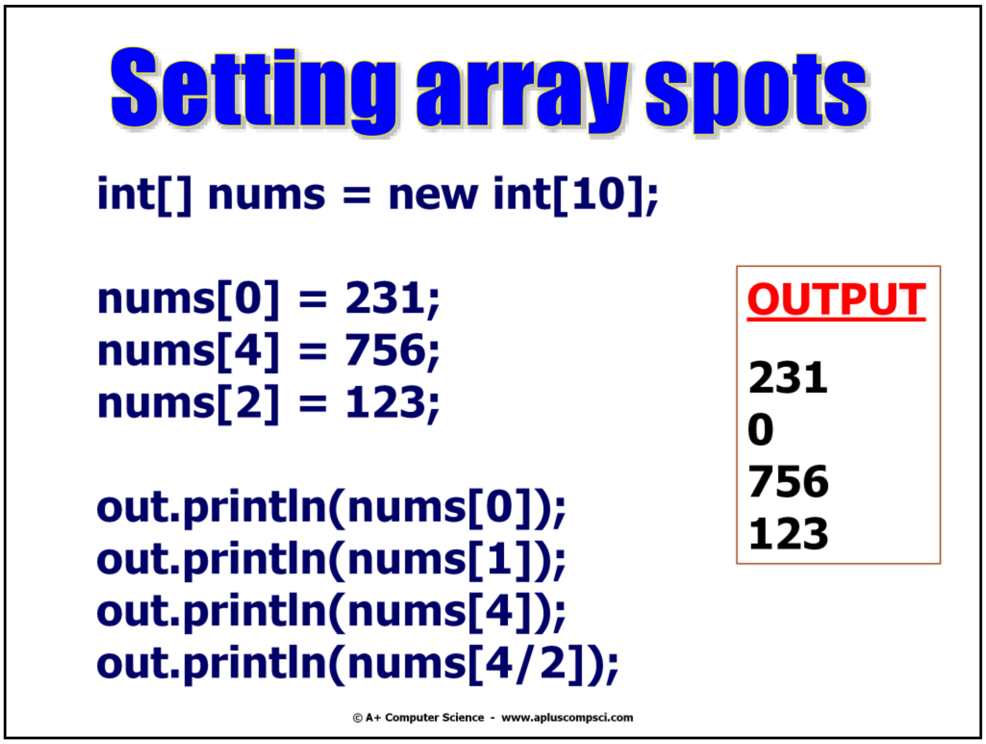
0 1 2 3 4 5 6
nums
2 3 5 1 0 6 7

Once the array has been instantiated and has values, it is very simple to print/access a particular spot. An integer value must be provided to indicate which [spot] will be accessed.

```
int[] thoseNums = {5,7,3,6,9};
out.println(thoseNums[3]);     //outs 6


out.println(thoseNums[1/2]);    //outs 5
       // 1/2 is 0


out.println(thoseNums[2+2]);    //outs 9
out.println(thoseNums[5/2]);    //outs 3
       //  5/2 is 2
```

open arrayprintone.java arrayprinttwo.java

Setting Array Spots



Setting array spots

```
int[] nums = new int[10];

nums[0] = 231;
nums[4] = 756;
nums[2] = 123;

out.println(nums[0]);
out.println(nums[1]);
out.println(nums[4]);
out.println(nums[4/2]);
```

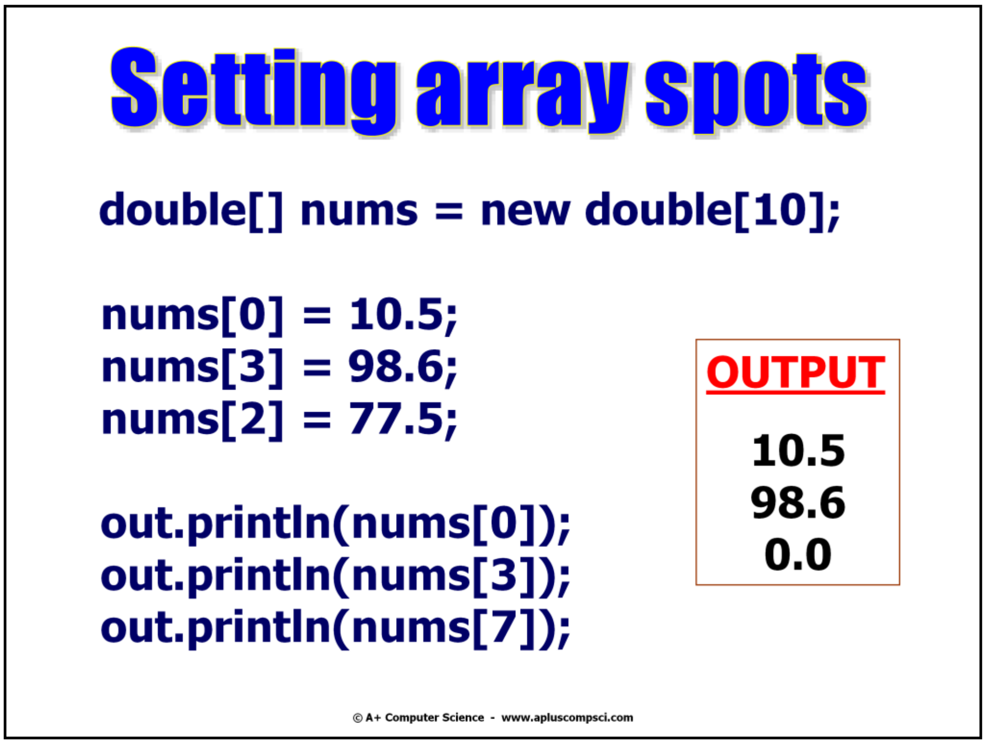
OUTPUT

```
231
0
756
123
```

© A+ Computer Science - www.apluscompsci.com

An integer value must be provided when accessing a [spot] in an array.

`nums[0]=231;` is setting spot 0 to the value 231.



Setting array spots

```
double[] nums = new double[10];

nums[0] = 10.5;
nums[3] = 98.6;
nums[2] = 77.5;

out.println(nums[0]);
out.println(nums[3]);
out.println(nums[7]);
```

OUTPUT

```
10.5
98.6
0.0
```

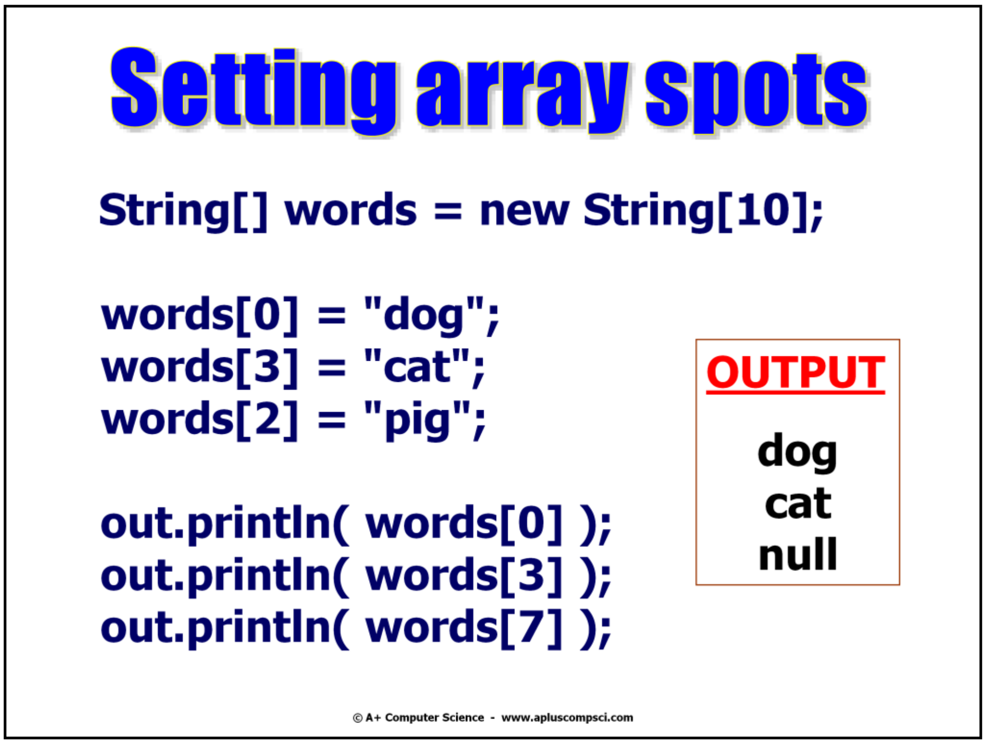
© A+ Computer Science - www.apluscompsci.com

`nums` has been instantiated with the capacity to store 10 doubles. All spots are set to 0.0 to start.

An integer value must be provided when accessing a [spot] in an array.

`nums` is storing double values, but the index/spot value must be an integer.

`nums[0]=10.5;` //sets spot 0 to the value 10.5.



Setting array spots

```
String[] words = new String[10];

words[0] = "dog";
words[3] = "cat";
words[2] = "pig";

out.println( words[0] );
out.println( words[3] );
out.println( words[7] );
```

OUTPUT

```
dog
cat
null
```

© A+ Computer Science - www.apluscompsci.com

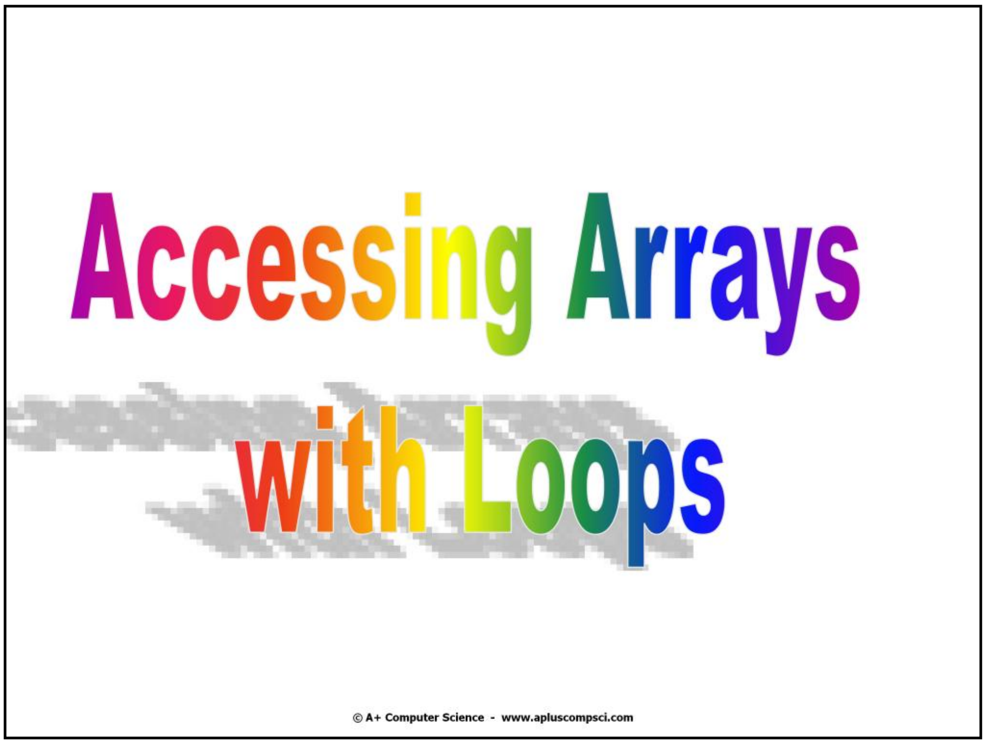
`words` has been instantiated with the capacity to store 10 String references.

All spots are set to null to start.

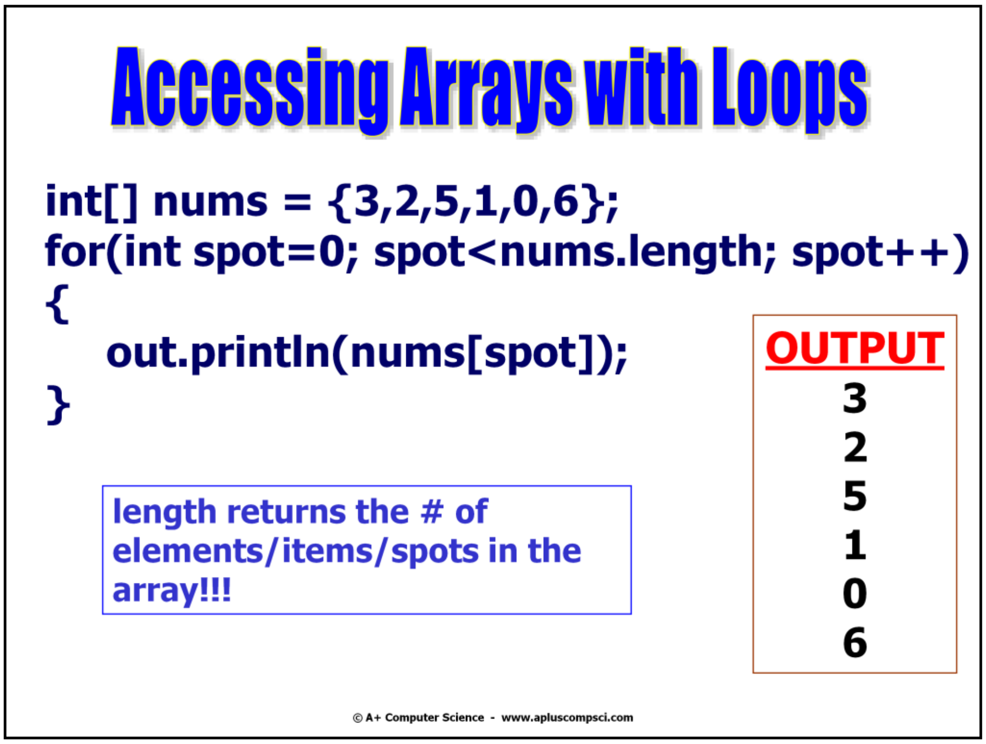
An integer value must be provided when accessing a [spot] in an array.

words is storing String references, but the index/spot value must be an integer.

open
arraysetone.java
arraysettwo.java



Accessing Arrays with Loops



Accessing Arrays with Loops

```
int[] nums = {3,2,5,1,0,6};
for(int spot=0; spot<nums.length; spot++)
{
   out.println(nums[spot]);
}
```

length returns the # of elements/items/spots in the array!!!

OUTPUT

```
3
2
5
1
0
6
```

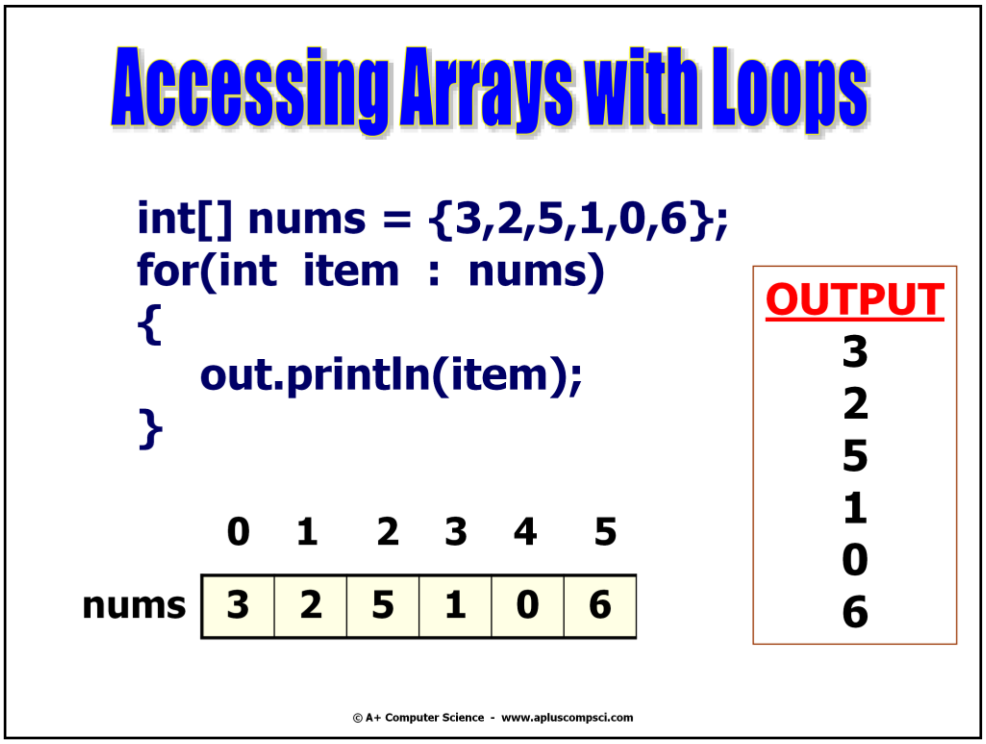
Using loops to print all spots in an array is a necessary approach.

As array lengths could change with different input values, it is

good to use a for loop based on length. If length changes, the loop will change accordingly.

The loop variable will start at 0 and go up to the array length.

The loop variable will be used to access each [spot] in the array.



Accessing Arrays with Loops

```
int[] nums = {3,2,5,1,0,6};
for(int  item  :  nums)
{
    out.println(item);
}
```

0 1 2 3 4 5
nums 3 2 5 1 0 6

OUTPUT

```
3
2
5
1
0
6
```

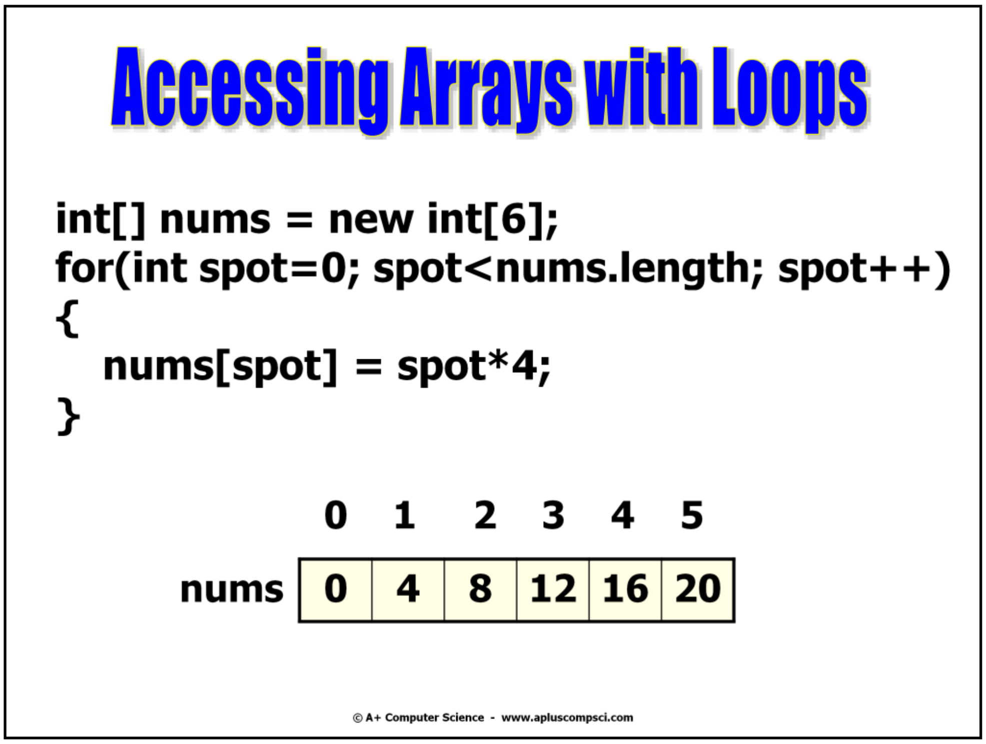
The for each loop is a great tool to use when accessing array values if a spot/index variable is not needed.

The for each loop above accesses all values in nums and prints each one.

Each time the loop iterates, the next value from nums is pasted into item.

The for each loop will iterate as long as the structure it is connected to has values.

```
int[] nums = {1,2,3,4,5,6,7};
for(int  item  :  nums)
{
     out.print(item + " ");
}
//outs 1 2 3 4 5 6 7
```



Accessing Arrays with Loops

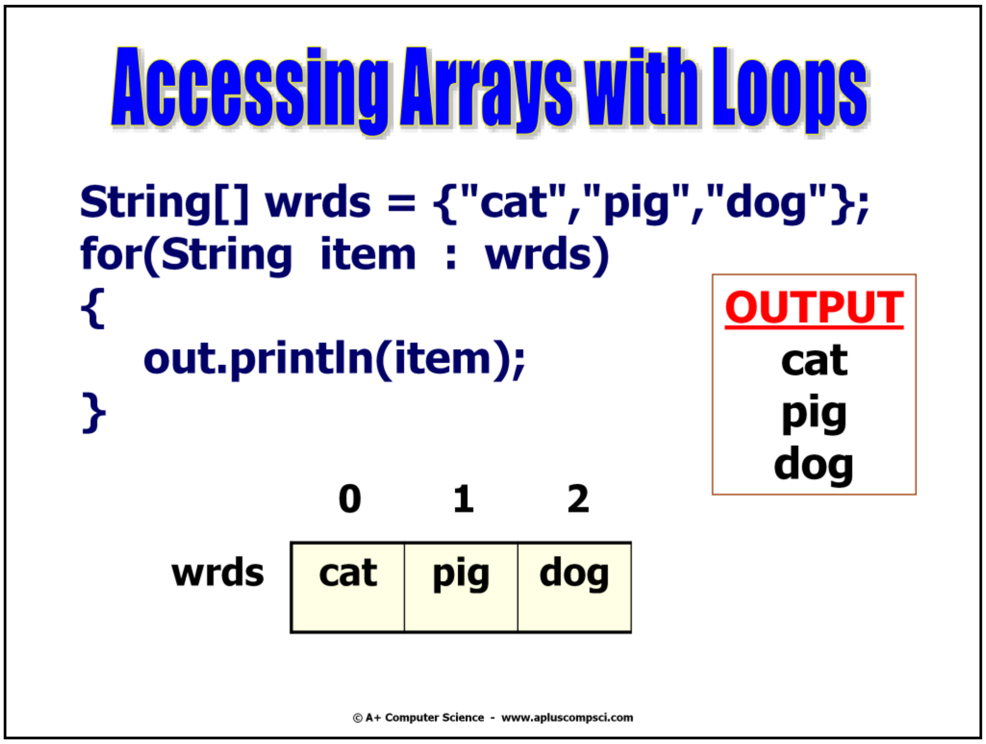
```
int[] nums = new int[6];
for(int spot=0; spot<nums.length; spot++)
{
   nums[spot] = spot*4;
}
```

	0	1	2	3	4	5
nums	0	4	8	12	16	20

© A+ Computer Science - www.apluscompsci.com

Using loops to print all spots in an array is a necessary approach.
As array lengths could change with different input values, it is
good to use a for loop based on length. If length changes, the loop will change accordingly.

The loop variable will start at 0 and go up to the array length.
The loop variable will be used to access each [spot] in the array.



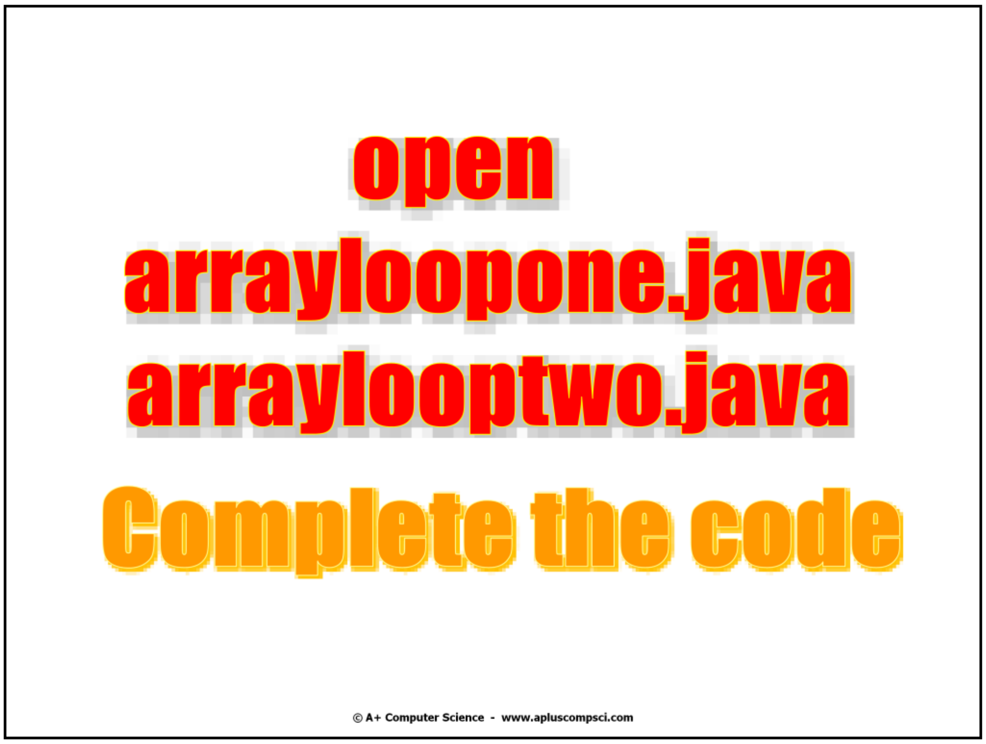
Accessing Arrays with Loops

```
String[] wrds = {"cat","pig","dog"};
for(String  item  :  wrds)
{
    out.println(item);
}
```

OUTPUT

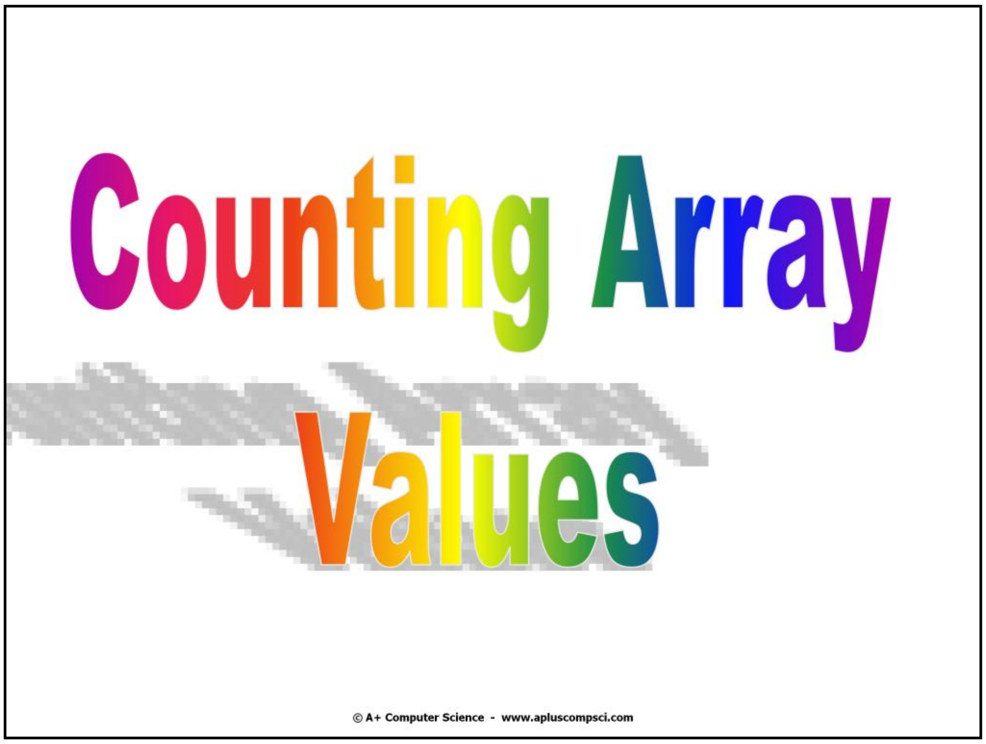
```
cat
pig
dog
```

	0	1	2
wrds	cat	pig	dog



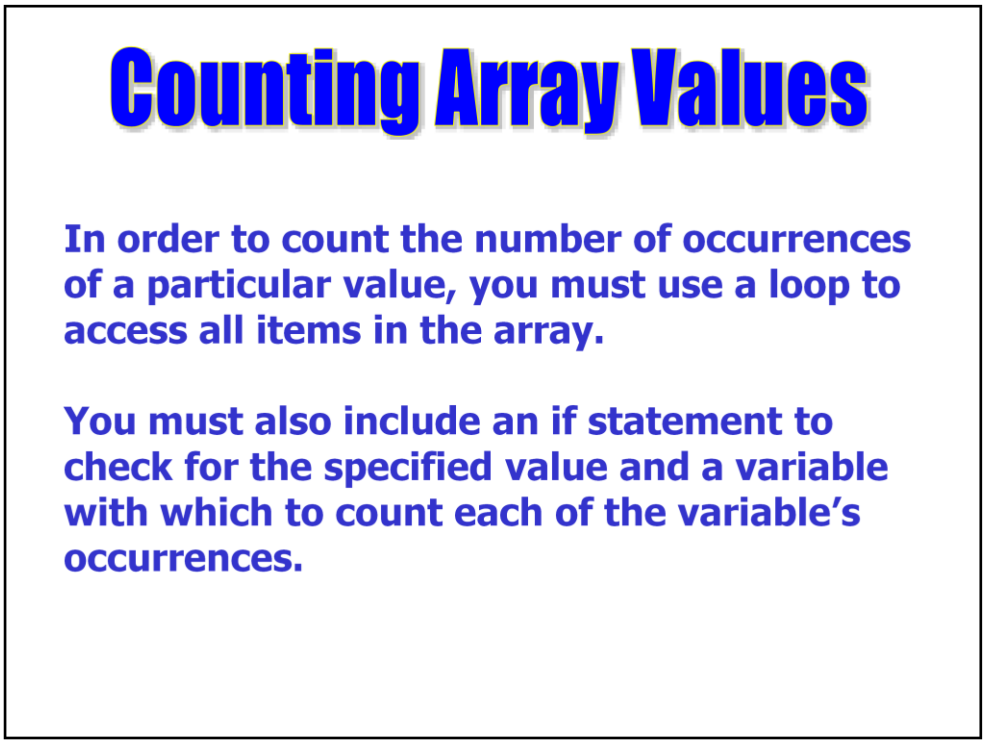
open
arrayloopone.java
arraylooptwo.java

Complete the code



Counting Array Values

© A+ Computer Science - www.apluscompsci.com

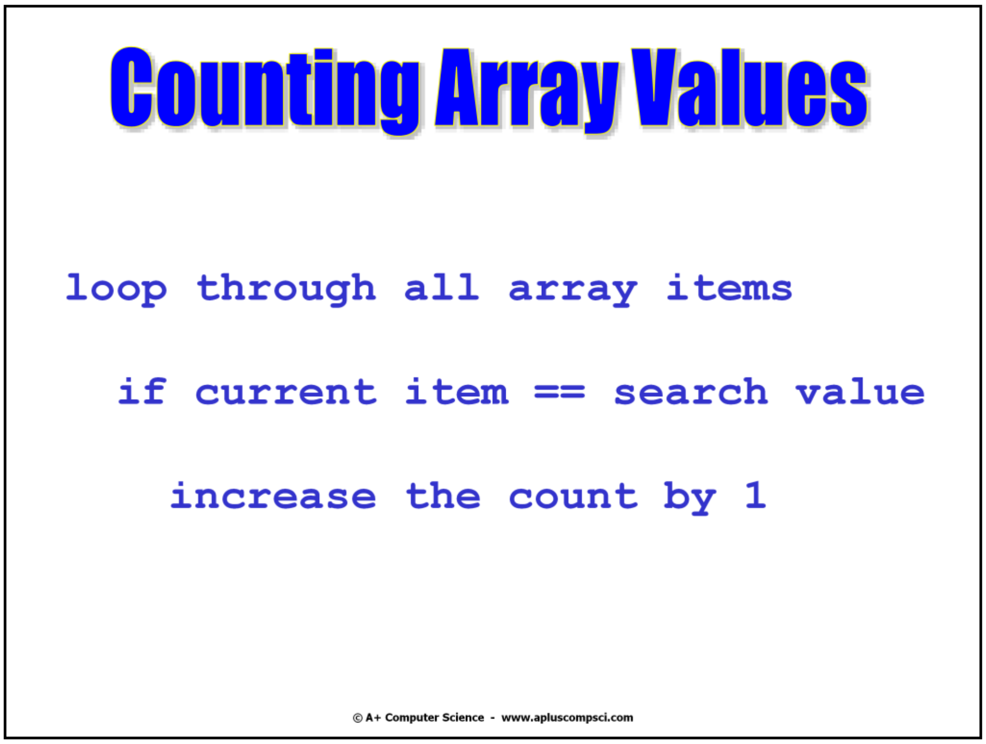


Counting Array Values

In order to count the number of occurrences of a particular value, you must use a loop to access all items in the array.

You must also include an if statement to check for the specified value and a variable with which to count each of the variable's occurrences.

Counting the number of occurrences of a particular item requires using a loop and a variable.
The loop must iterate over all items in the list and the if statement must check each item.
The variable will be used to count how many of a particular type exist.

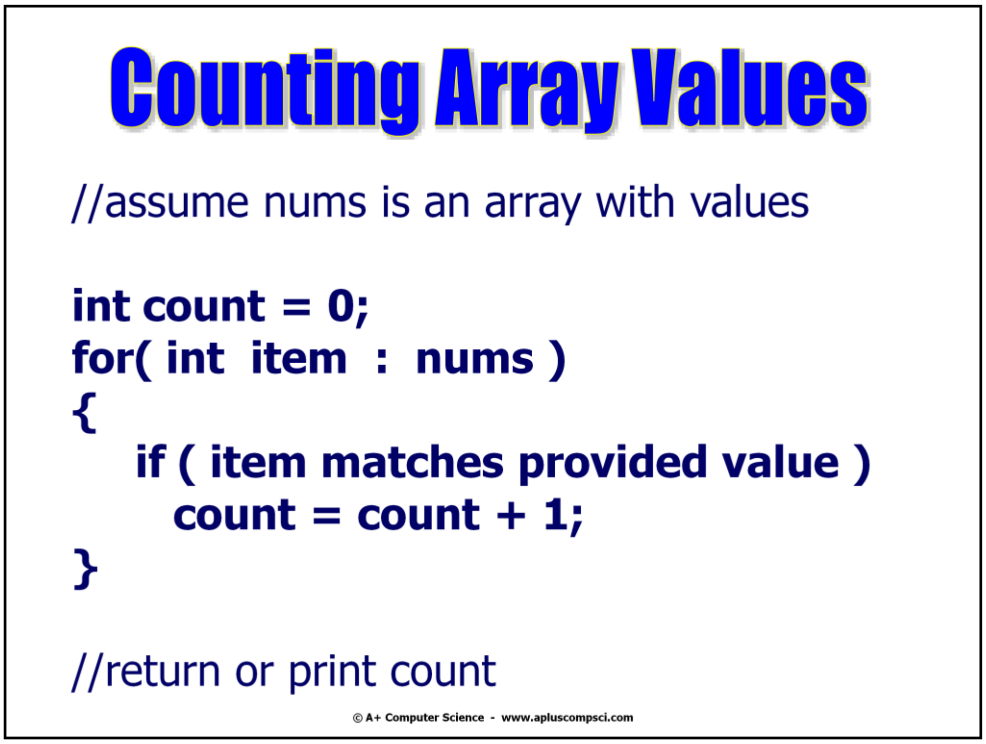


Counting Array Values

```
loop through all array items
  if current item == search value
    increase the count by 1
```

© A+ Computer Science - www.apluscompsci.com

Counting the number of occurrences of a particular item requires using a loop and a variable.
The loop must iterate over all items in the list and the if statement must check each item.
The variable will be used to count how many of a particular type exist.



Counting Array Values

```
//assume nums is an array with values

int count = 0;
for( int  item  :  nums )
{
    if ( item matches provided value )
      count = count + 1;
}

//return or print count
```

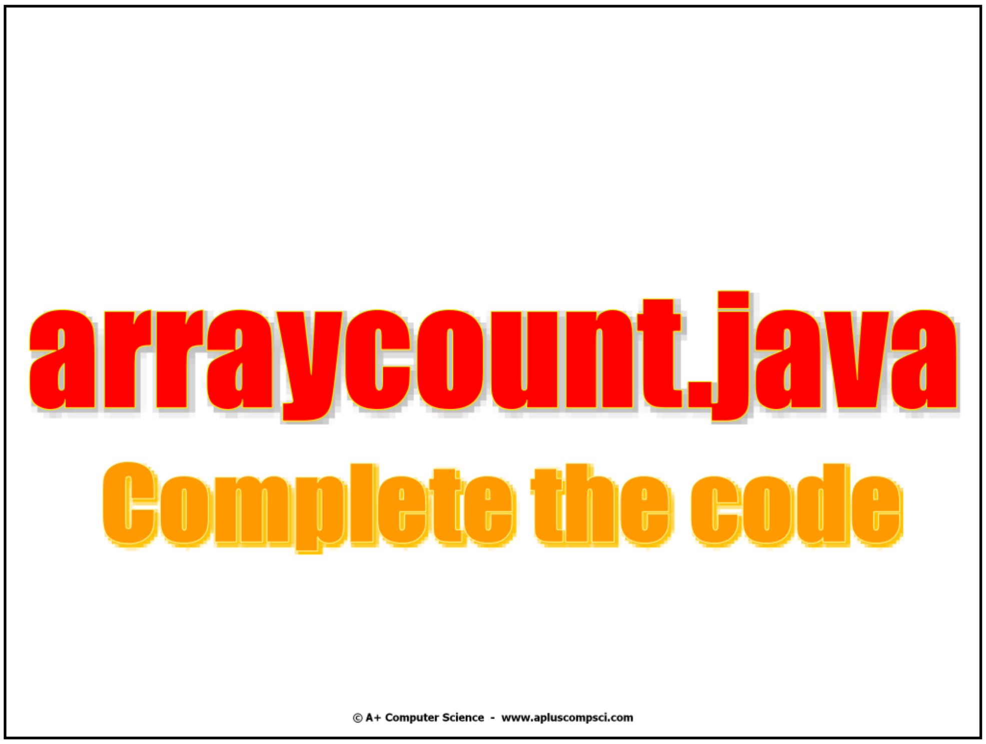
The for each loop is a great tool to use when accessing array values if a spot/index variable is not needed.

The for each loop above accesses all values in nums.

Each time the loop iterates, the next value from nums is pasted into item.

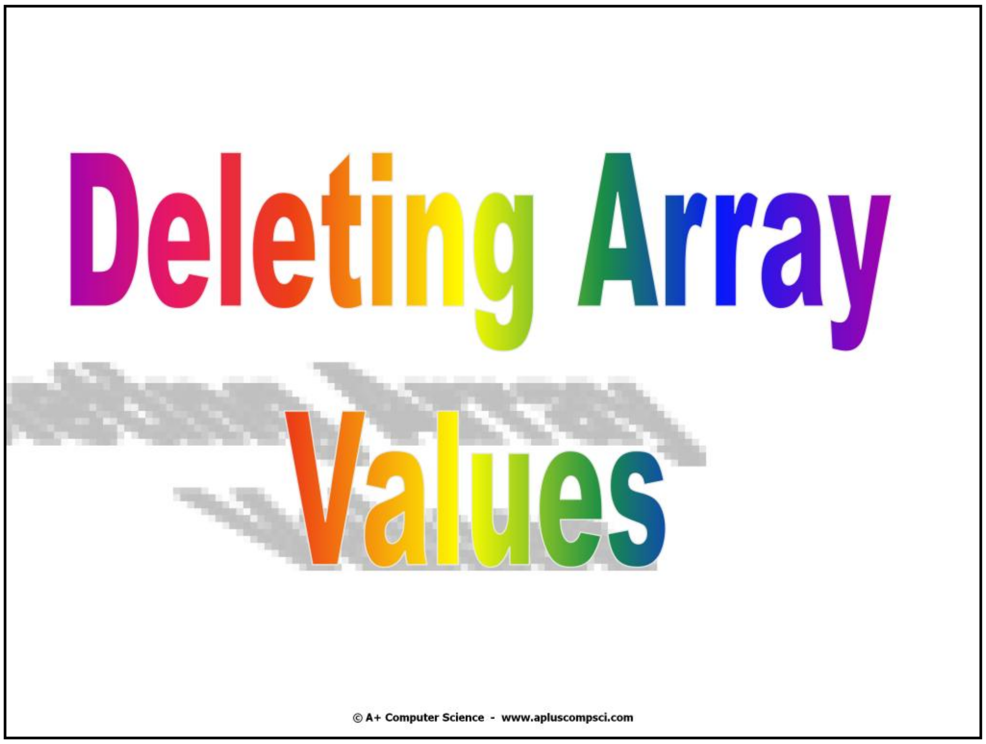
The for each loop will iterate as long as the structure it is connected to has values.

```
int[] nums = {1,2,3,4,5,6,7};
for(int  item  :  nums)
{
     out.print(item + " ");
}
//outs 1 2 3 4 5 6 7
```

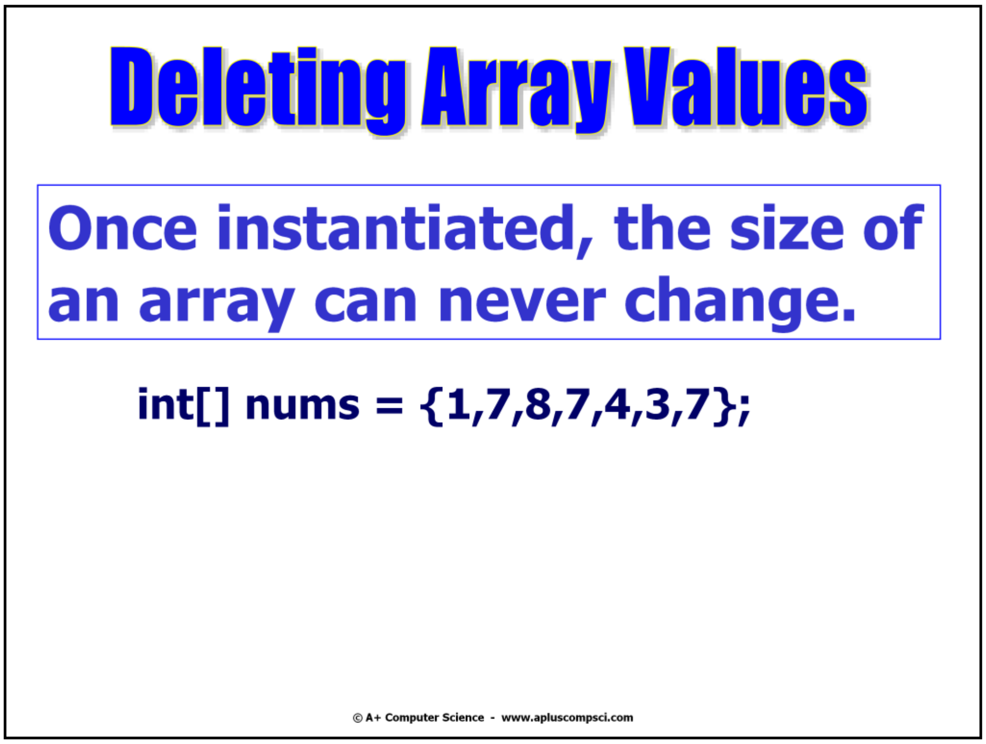


arraycount.java

Complete the code



Deleting Array Values



Deleting Array Values

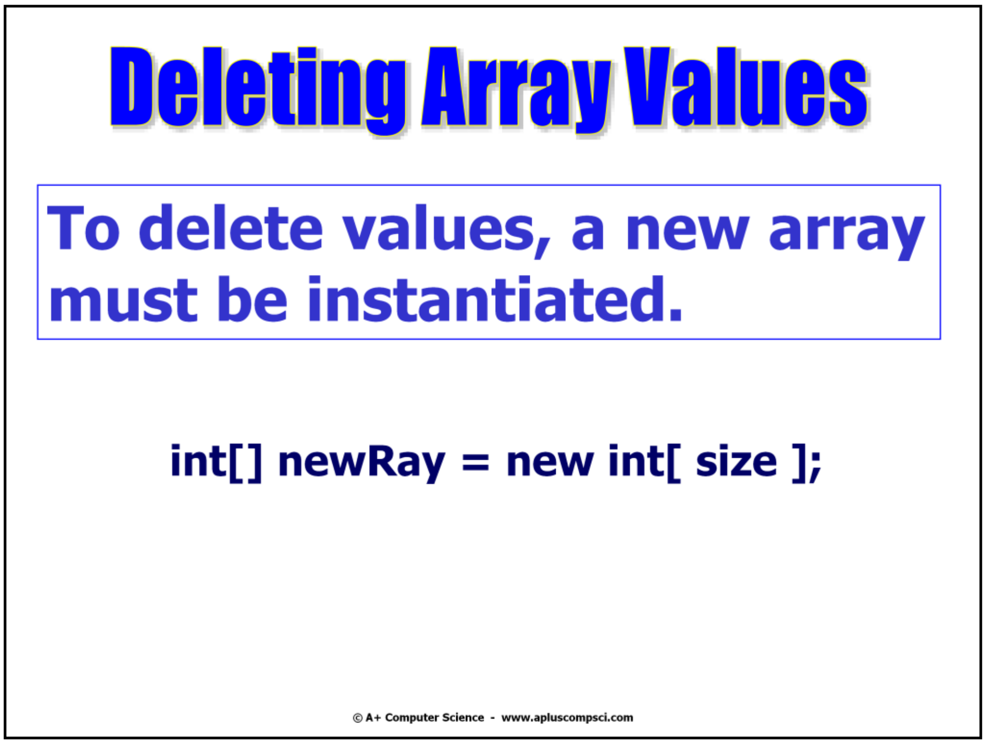
Once instantiated, the size of an array can never change.

```
int[] nums = {1,7,8,7,4,3,7};
```

Once an array Object has been created, the size of that array can never change.

The values in the array can change, but not the size.

In order to remove values from an array, a new array must be created with an appropriate size considering how many items are to be removed.
Old values must then be copied to the new array.



Deleting Array Values

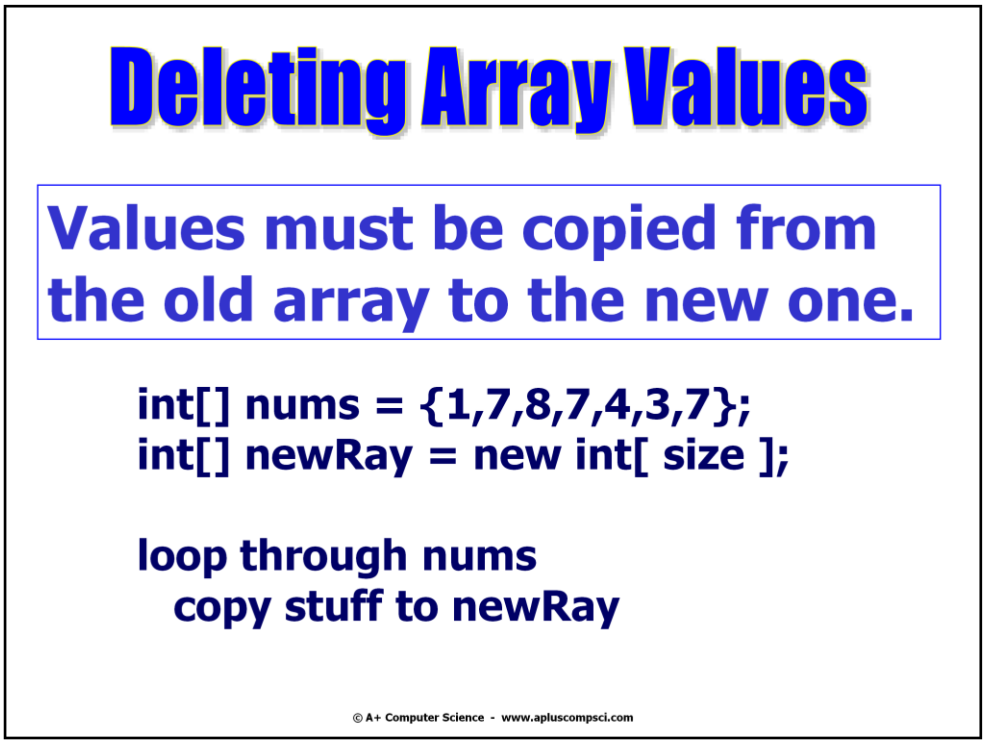
To delete values, a new array must be instantiated.

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

Once an array Object has been created, the size of that array can never change.

The values in the array can change, but not the size.

In order to remove values from an array, a new array must be created with an appropriate size considering how many items are to be removed.
Old values must then be copied to the new array.



Deleting Array Values

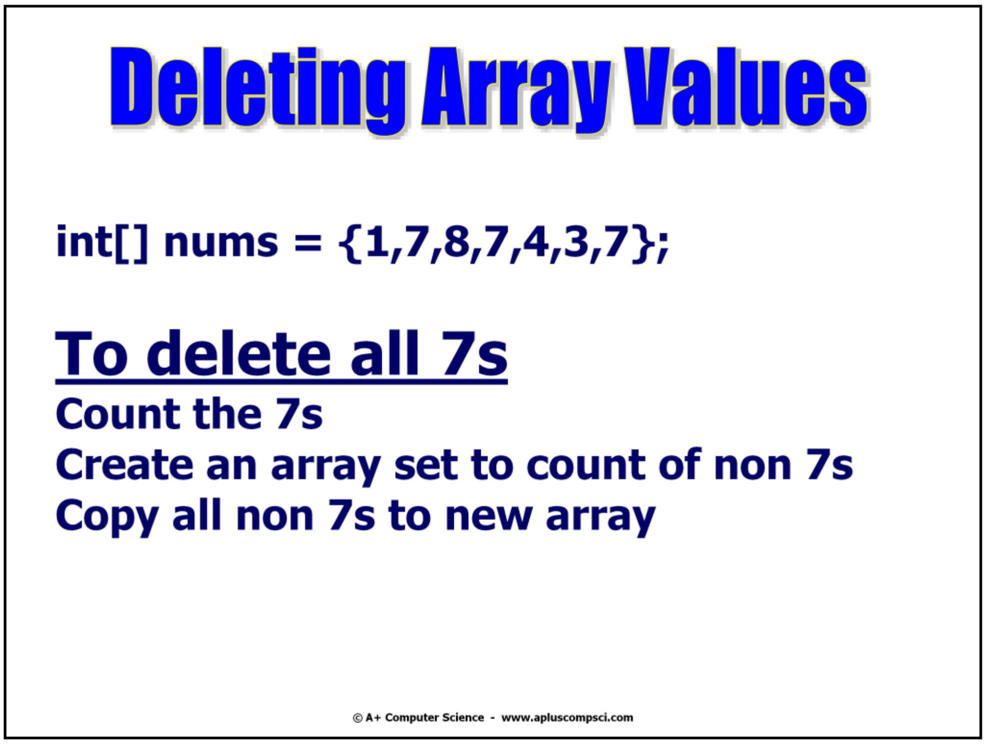
Values must be copied from the old array to the new one.

```
int[] nums = {1,7,8,7,4,3,7};
int[] newRay = new int[ size ];

loop through nums
  copy stuff to newRay
```

Once an array Object has been created, the size of that array can never change.

The values in the array can change, but not the size.

In order to remove values from an array, a new array must be created with an appropriate size considering how many items are to be removed.
Old values must then be copied to the new array.



Deleting Array Values

```
int[] nums = {1,7,8,7,4,3,7};
```

To delete all 7s

Count the 7s
Create an array set to count of non 7s
Copy all non 7s to new array

© A+ Computer Science - www.apluscompsci.com

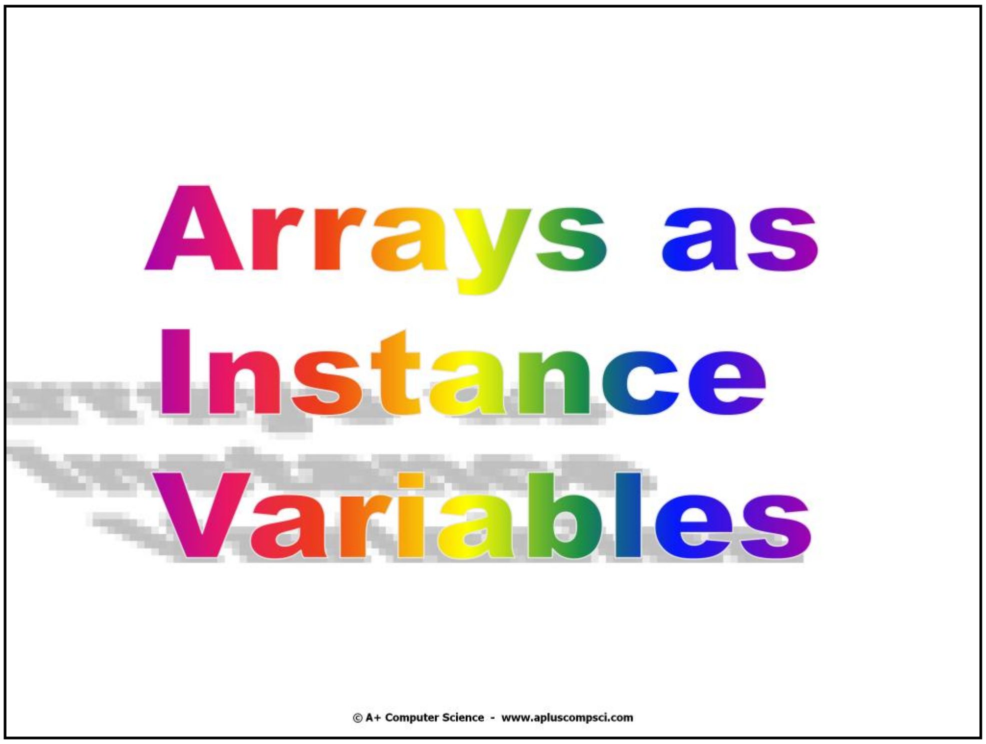
Once an array Object has been created, the size of that array can never change.

The values in the array can change, but not the size.

In order to remove values from an array, a new array must be created with an appropriate size considering how many items are to be removed.
Old values must then be copied to the new array.

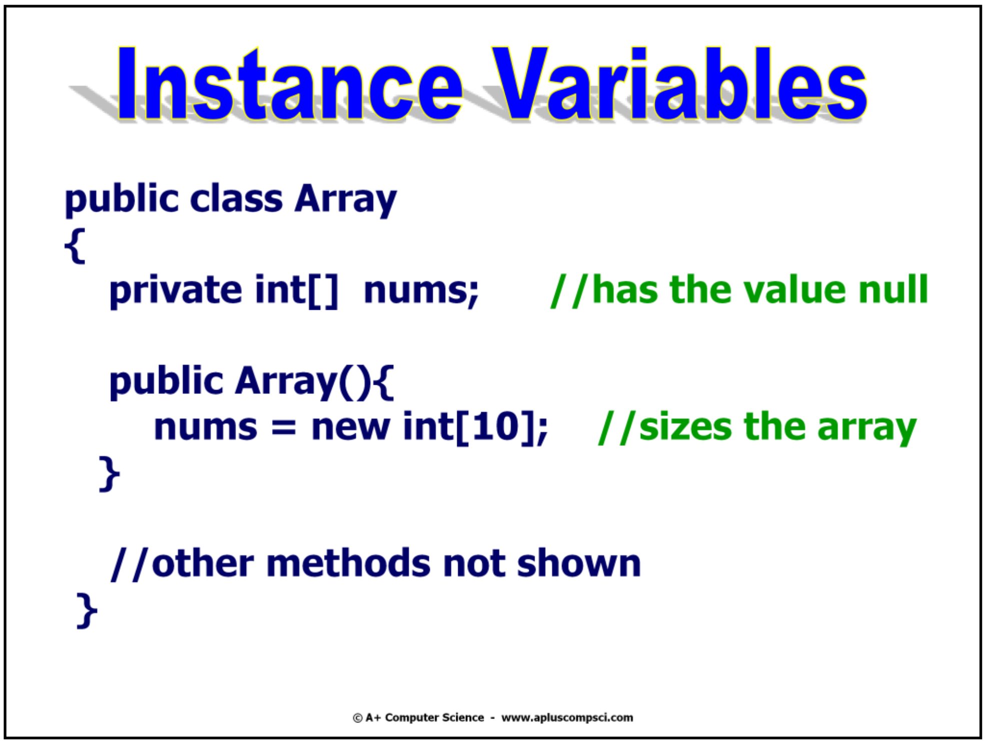
arraydelete.java

Complete the code



Arrays as Instance Variables

© A+ Computer Science - www.apluscompsci.com



Instance Variables

```
public class Array
{
   private int[]  nums;      //has the value null

   public Array(){
      nums = new int[10];    //sizes the array
  }

   //other methods not shown
 }
```

`int[]` should only appear in front of nums once.

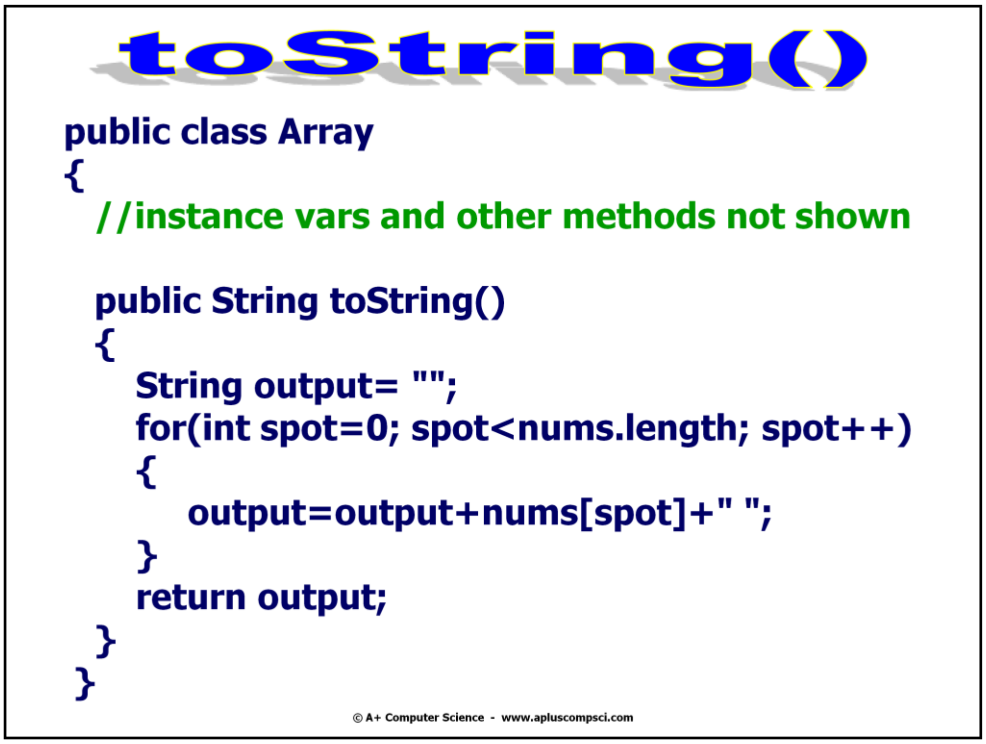
`int[]` should only appear on the left of nums when defining nums as an instance variable.

`int[]` should never appear on the left of nums in a constructor or any method.

The array must be instantiated and sized in the constructor.

arrayinstancevars.java

Complete the code

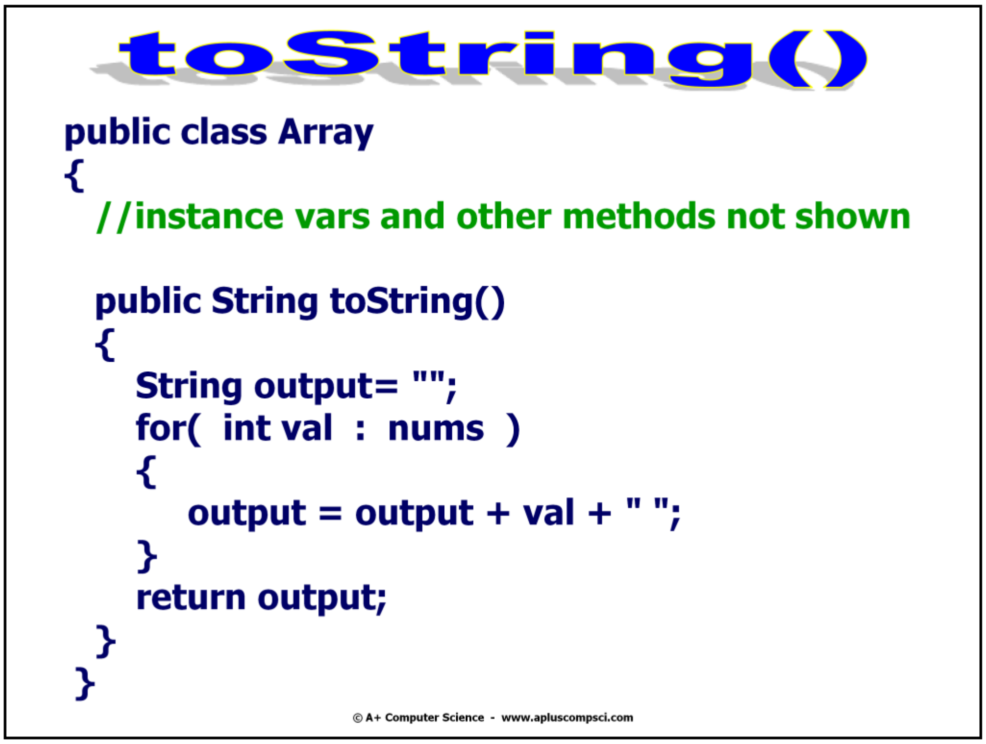


toString()

```
public class Array
{
  //instance vars and other methods not shown

  public String toString()
  {
     String output= "";
     for(int spot=0; spot<nums.length; spot++)
     {
        output=output+nums[spot]+" ";
     }
     return output;
  }
}
```

© A+ Computer Science - www.apluscompsci.com

To toString() method will use a loop to access all spots in the array. The value in each spot will be added to output and returned at the end of the toString() method.

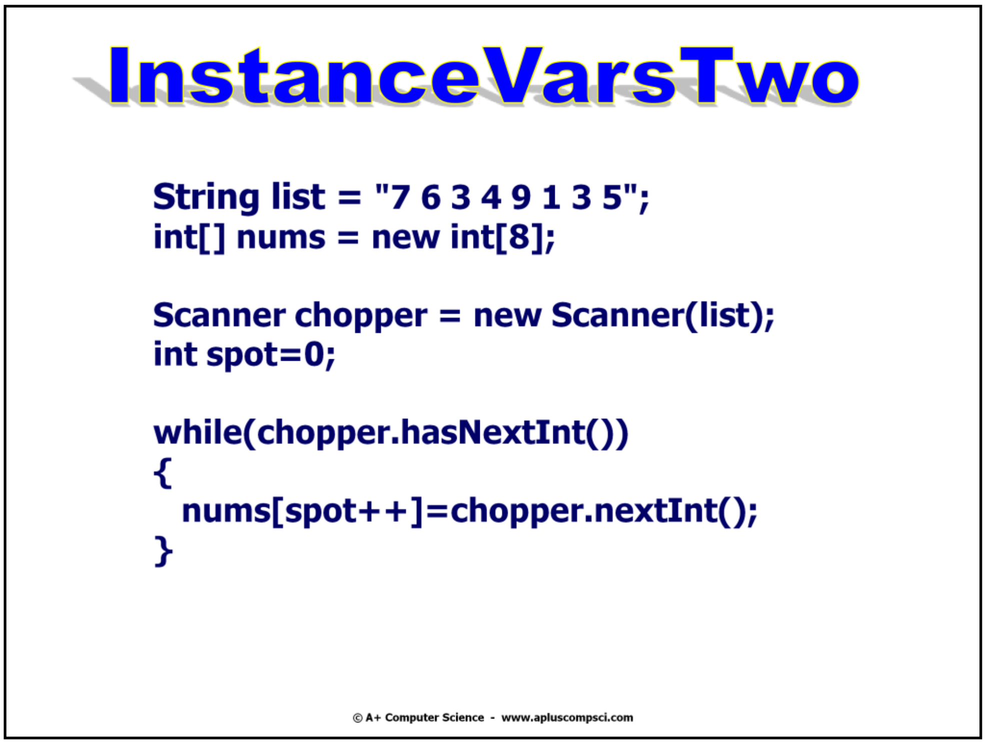


toString()

```
public class Array
{
  //instance vars and other methods not shown

  public String toString()
  {
     String output= "";
     for(  int val  :  nums  )
     {
         output = output + val + " ";
     }
     return output;
  }
}
```

To toString() method will use a loop to access all spots in the array. The value in each spot will be added to output and returned at the end of the toString() method.

arrayinstancevarstwo.java



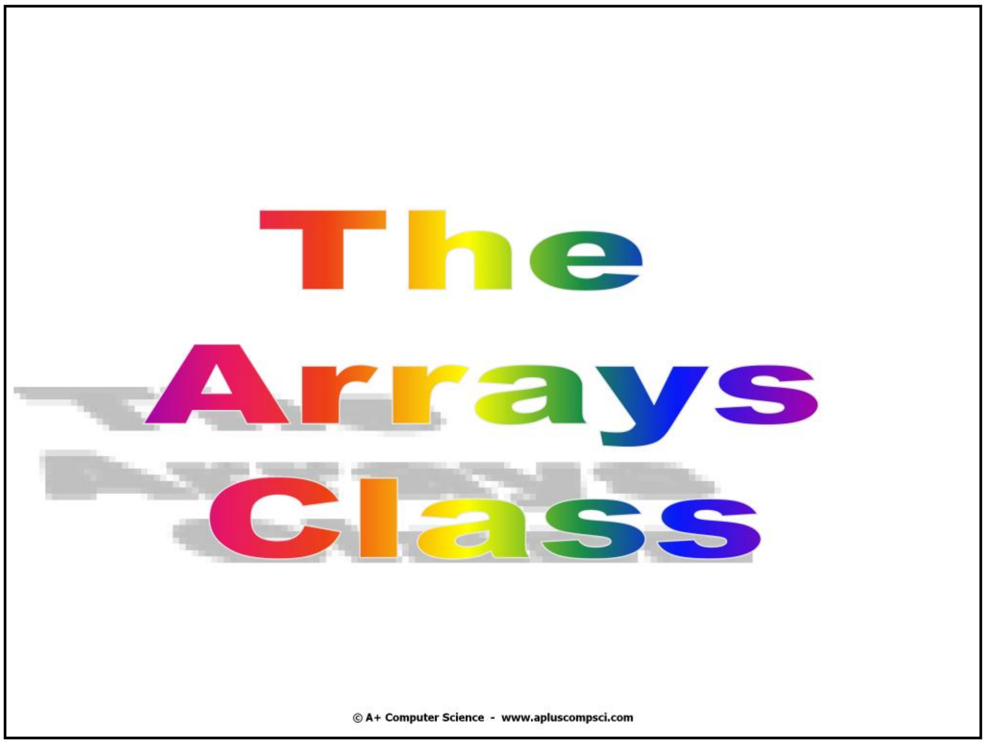
InstanceVarsTwo

```
String list = "7 6 3 4 9 1 3 5";
int[] nums = new int[8];

Scanner chopper = new Scanner(list);
int spot=0;

while(chopper.hasNextInt())
{
  nums[spot++]=chopper.nextInt();
}
```

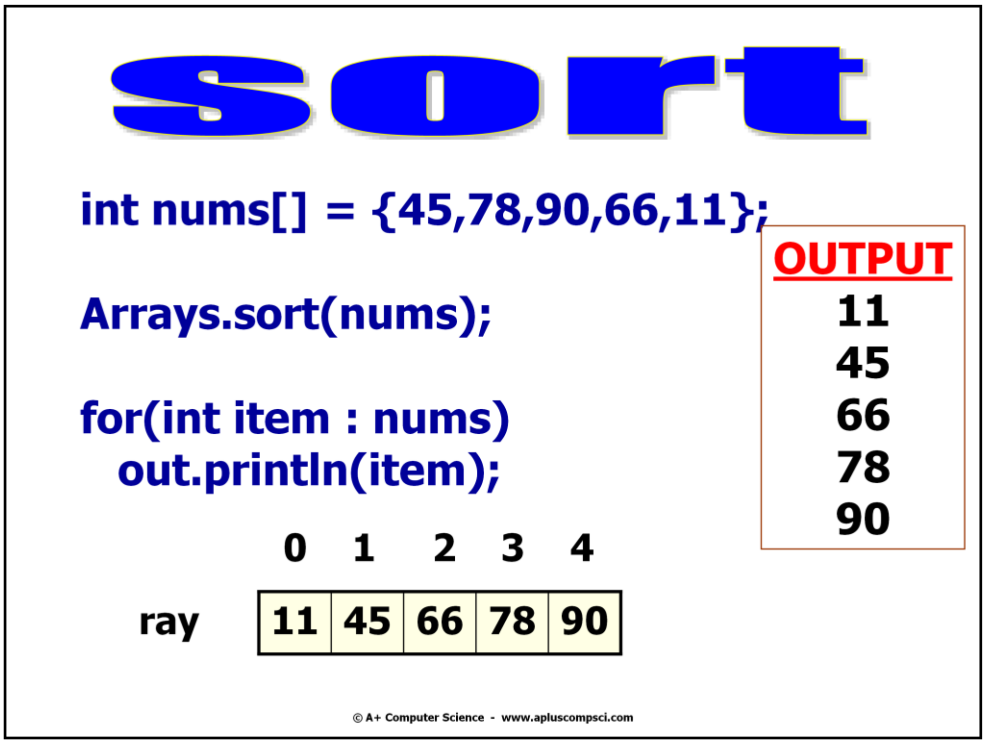
© A+ Computer Science - www.apluscompsci.com

The idea of chopping up a String with an unknown number of values is very important.

A while loop is needed to chop up a String with an unknown number of values.



The Arrays Class



sort

```
int nums[] = {45,78,90,66,11};

Arrays.sort(nums);

for(int item : nums)
   out.println(item);
```

OUTPUT

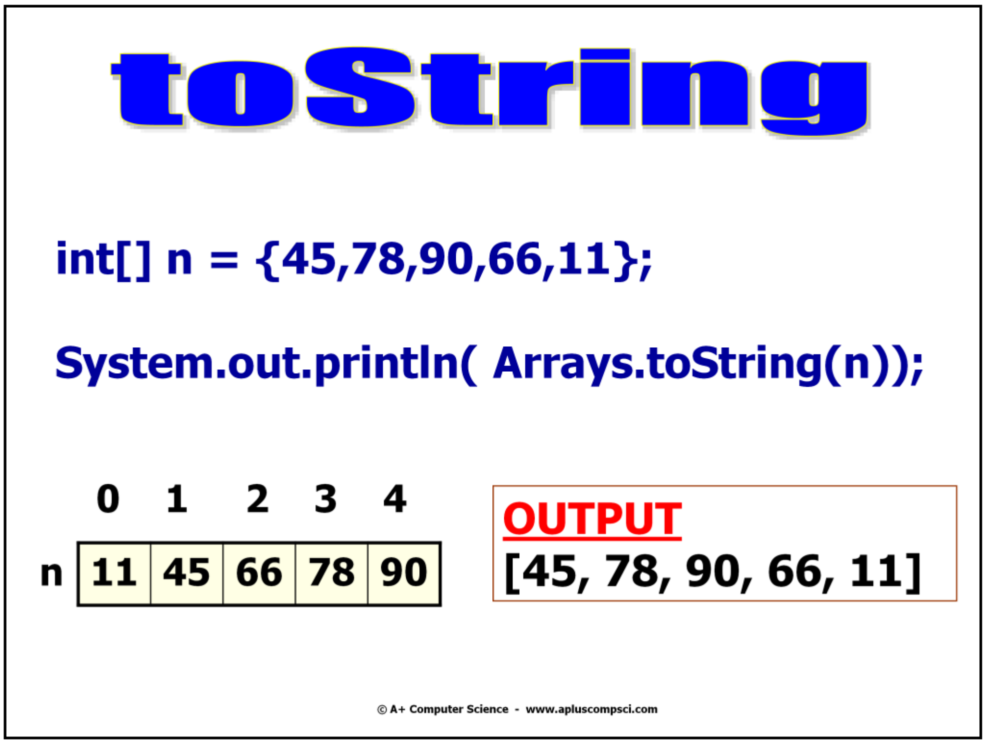
```
11
45
66
78
90
```

	0	1	2	3	4
ray	11	45	66	78	90

The built in Java Arrays.sort() method will naturally order all values in the array.

The values in the array will be in ascending order after the call to sort().

Arrays.sort() uses a quick sort to sort primitives and a merge sort to sort references.



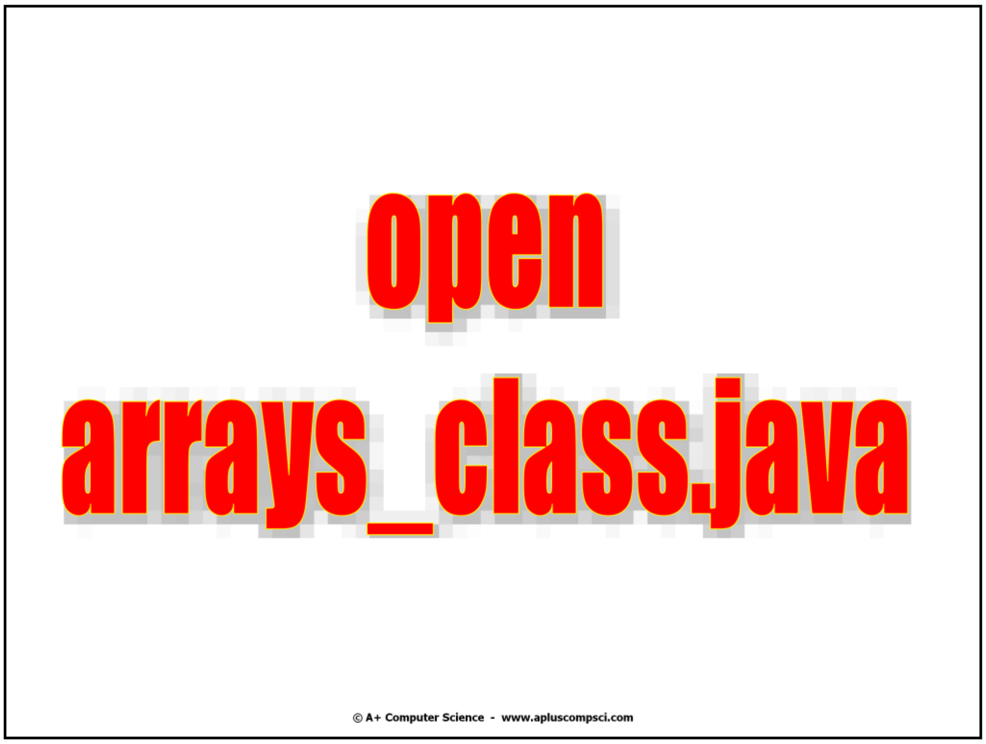
toString

```
int[] n = {45,78,90,66,11};

System.out.println( Arrays.toString(n));
```

0 1 2 3 4
n 11 45 66 78 90

OUTPUT
```
[45, 78, 90, 66, 11]
```

toString will print out the array just like the toString for ArrayList.



open
arrays_class.java

Start work
On the labs