AP Computer Science A: Array Creation and Access

CodeHS

Think Back To Our Trivia Class

In Unit 5, we looked at creating a trivia class with 5 questions:

```
private String q1 = "What country hosted the 2016 Summer Olympics?";

private String q2 ="Which NBA team plays its home games at Madison Square Gardens?";

private String q3 = "Ankara is the capital of which country?";

 private String q4 ="In nautical folklore, which ship is condemned to sail the seas
       for all eternity?";

private String q5 ="Which car manufacturer's name means \"to hear\"?";
```

What If We Wanted 100 Questions?

If we wanted 100 questions, would we have to create 100 different variables?

```
private String q1 = "What country hosted the 2016 Summer Olympics?";

private String q2 = "Which NBA team plays its home games at Madison Square Gardens?";

private String q3 = "Ankara is the capital of which country?";

 private String q4 = "In nautical folklore, which ship is condemned to sail the seas
       for all eternity?";

private String q5 = "Which car manufacturer's name means \"to hear\"?";

. . .

private String q100 = "Did I really just create 100 questions?";
```

Introducing Arrays

An **array** is an object that can store many values of the same type in a single variable.

CodeHS

What is an array?

A simple way to think about an array is that an array is just a **list**. It may be a list of `ints` or a list of `Strings` or a list of anything you want! Here, it is a list of exam scores.

```
int[] scores
```

80	92	91	68	88

What is an array?

Definition: Arrays store a fixed number of elements of the same type in a single variable.

```
int[] scores
```

80	92	91	68	88

What is an array?

Definition: Arrays store a **fixed** number of elements of the **same type** in a single variable. Once an array has been created, the size cannot change.

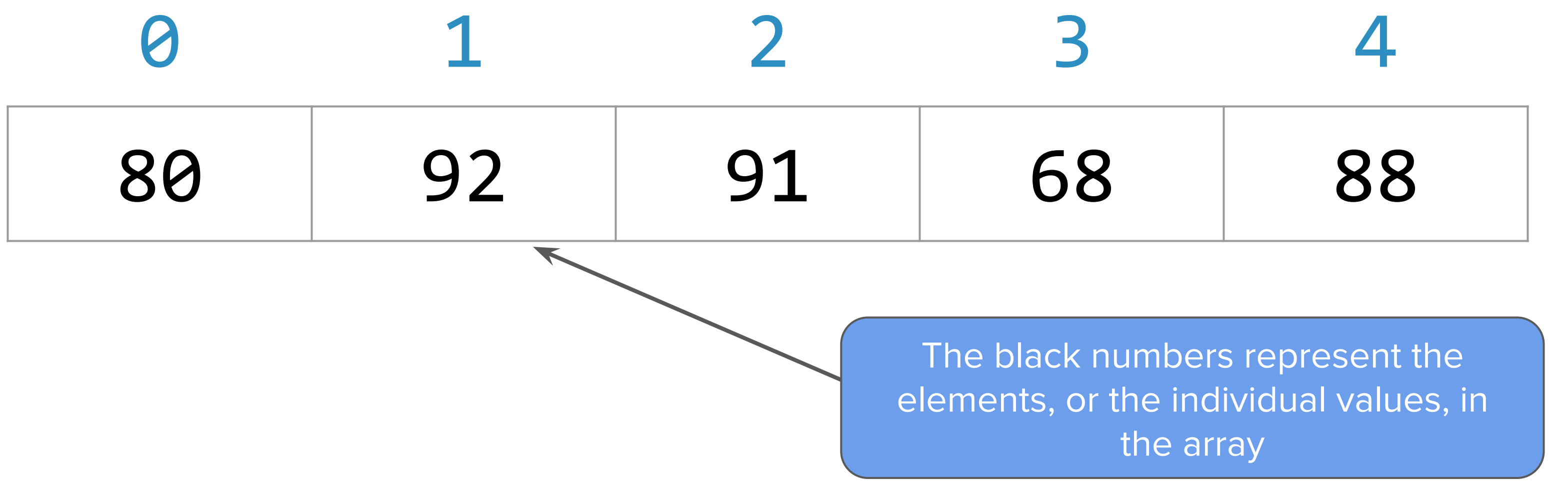
```
int[] scores
```

0	1	2	3	4
80	92	91	68	88

What is an array?

Definition: Arrays store a fixed number of elements of the same type in a single variable. Once an array has been created, the size cannot change.

```
int[] scores
```



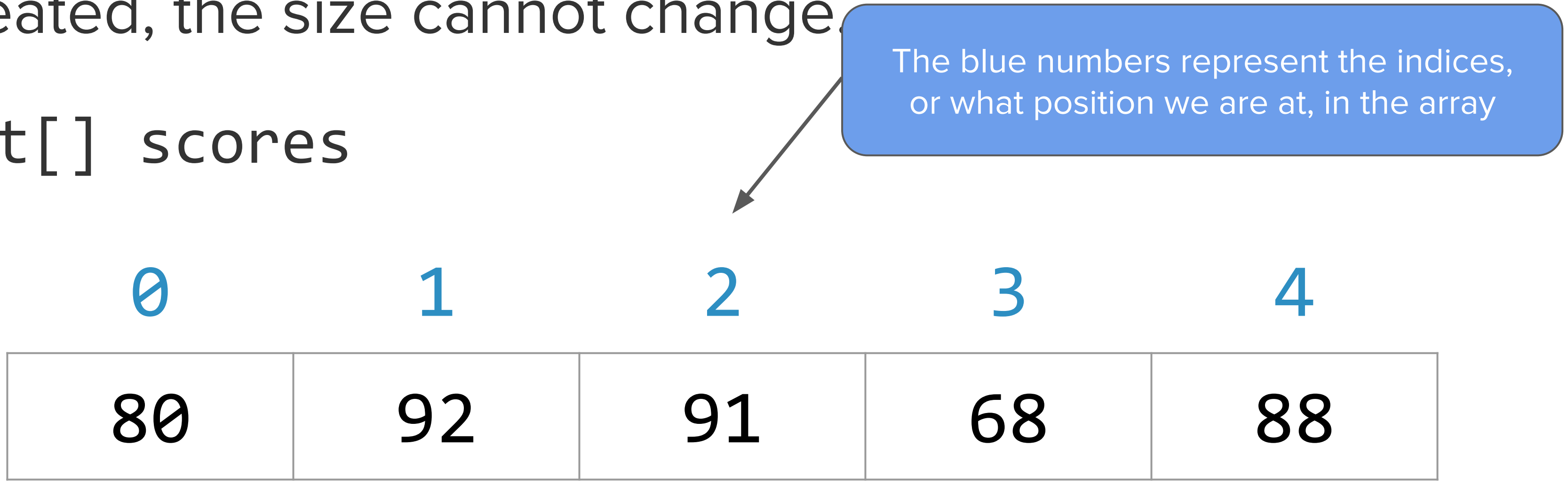
0	1	2	3	4
80	92	91	68	88

The black numbers represent the elements, or the individual values, in the array

What is an array?

Definition: Arrays store a fixed number of elements of the same type in a single variable. Once an array has been created, the size cannot change.

```
int[] scores
```


The blue numbers represent the indices, or what position we are at, in the array
0
1
2
3
4
80
92
91
68
88

Making an Array with Default Values

How to create an array:

```
int[] scores = new int[5];
```

0	1	2	3	4
0	0	0	0	0

Making an Array with Default Values

How to create an array:

Add empty square brackets [ ] after the type on the left side

```
int[] scores = new int[5];
```

0	1	2	3	4
0	0	0	0	0

Making an Array with Default Values

How to create an array:

Add empty square brackets [ ] after the type on the left side

```
int[] scores = new int[5];
```

Add square brackets [ ] with the number of elements in the array on the right side

0	1	2	3	4
0	0	0	0	0

Making an Array with Default Values

```
int[] scores = new int[5];
```

When you create an array in this manner, Java assigns default values to each member.

Type	Default Value
`int`	`0`
`double`	`0.0`
`boolean`	`false`
`Objects`	`null`

Making an Array with Initial Values

How to create an array with initializer list:

```
// Make a new int array and set the values
int[] scores = {80, 92, 91, 68, 88};
```

0	1	2	3	4
80	92	91	68	88

Making an Array with Initial Values

How to create an array with initializer list:

```
// Make a new int array and set the values

int[] scores = {80, 92, 91, 68, 88};
```

Initial values go inside curly brackets, separated by commas.

0	1	2	3	4
80	92	91	68	88

Examples of Making Arrays

Examples of default value arrays:

```
//String list with 10 null elements

String[] str = new String[10];


//double list with 100 zeros

double[] nums = new double[100];
```

Examples of Making Arrays

Examples of initial value arrays:

```
//4 element String list

String[] greetings = {"Hello", "Hola", "Bonjour", "Ni hao"};


//List with 5 student items

Student[] class = {julian, larisa, amada, mikka, jay};
```

Making an Array of Any Type

How to create an array with default values:

```
Type[] variableName = new Type[numElements];
```

How to create an array with initial values:

```
Type[] variableName = { initial values list };
```

Getting Value at an Index

```
int[] scores = {80, 92, 91, 68, 88};
```

```
int idaScore = scores[2]; //idaScore now gets 91
```

0	1	2	3	4
80	92	91	68	88

Getting Value at an Index

```
int[] scores = {80, 92, 91, 68, 88};
```

Array variable name

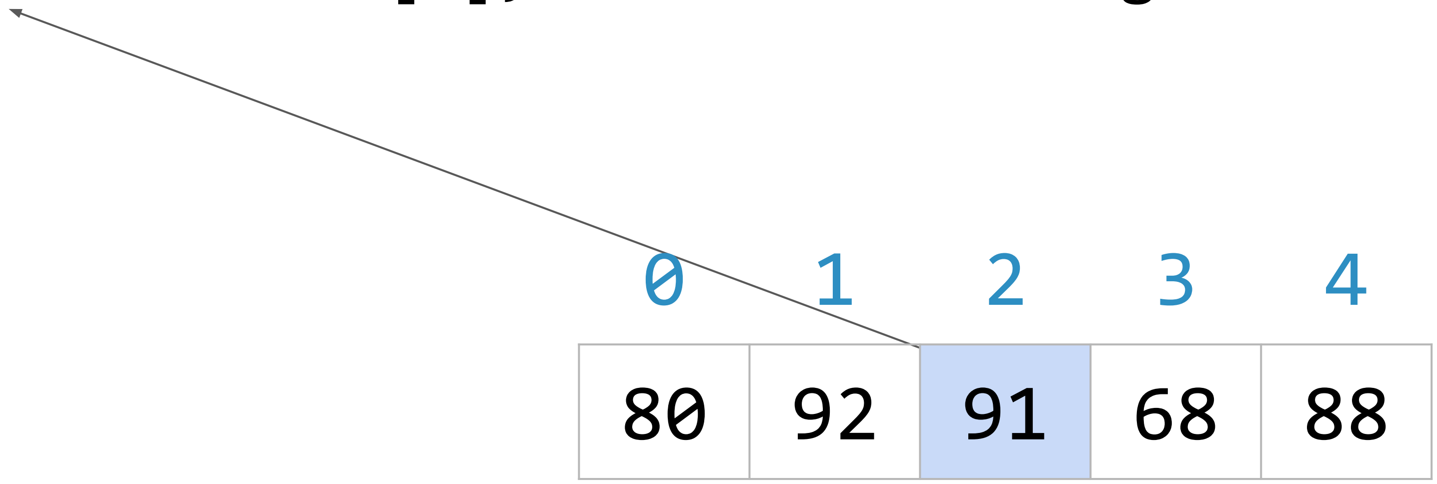
```
int idaScore = scores[2]; //idaScore now gets 91
```

0	1	2	3	4
80	92	91	68	88

Getting Value at an Index

```
int[] scores = {80, 92, 91, 68, 88};
```

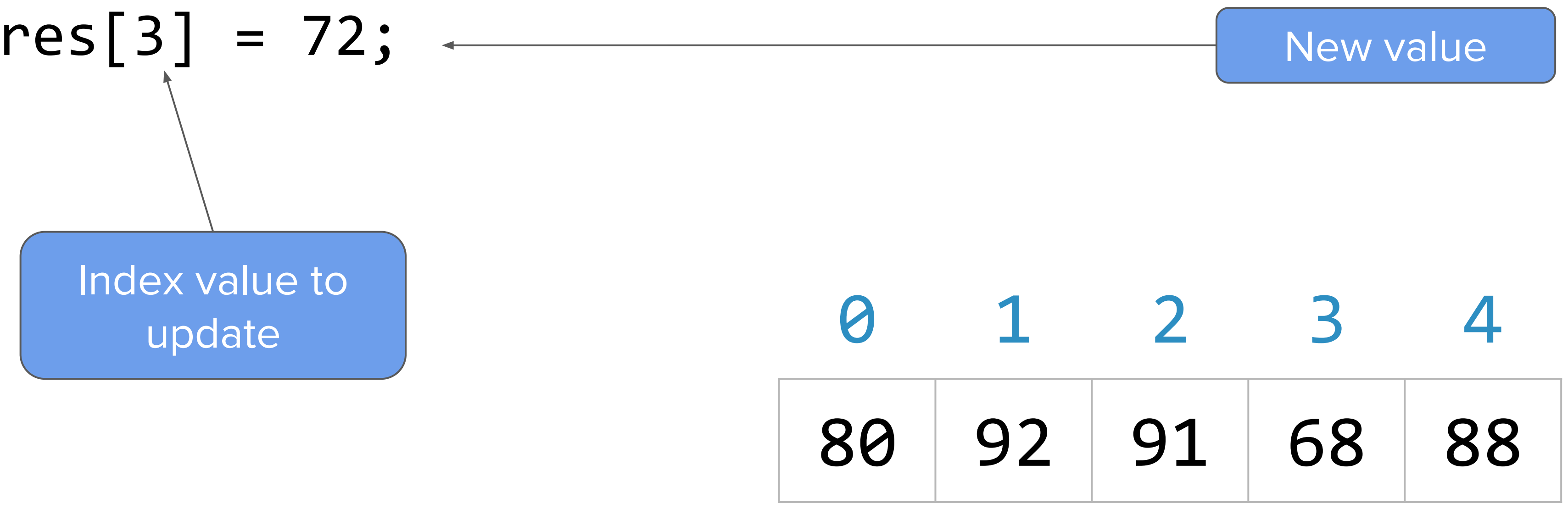
```
int idaScore = scores[2]; //idaScore now gets 91
```



Setting a Value at an Index

```
int[] scores = {80, 92, 91, 68, 88};
```

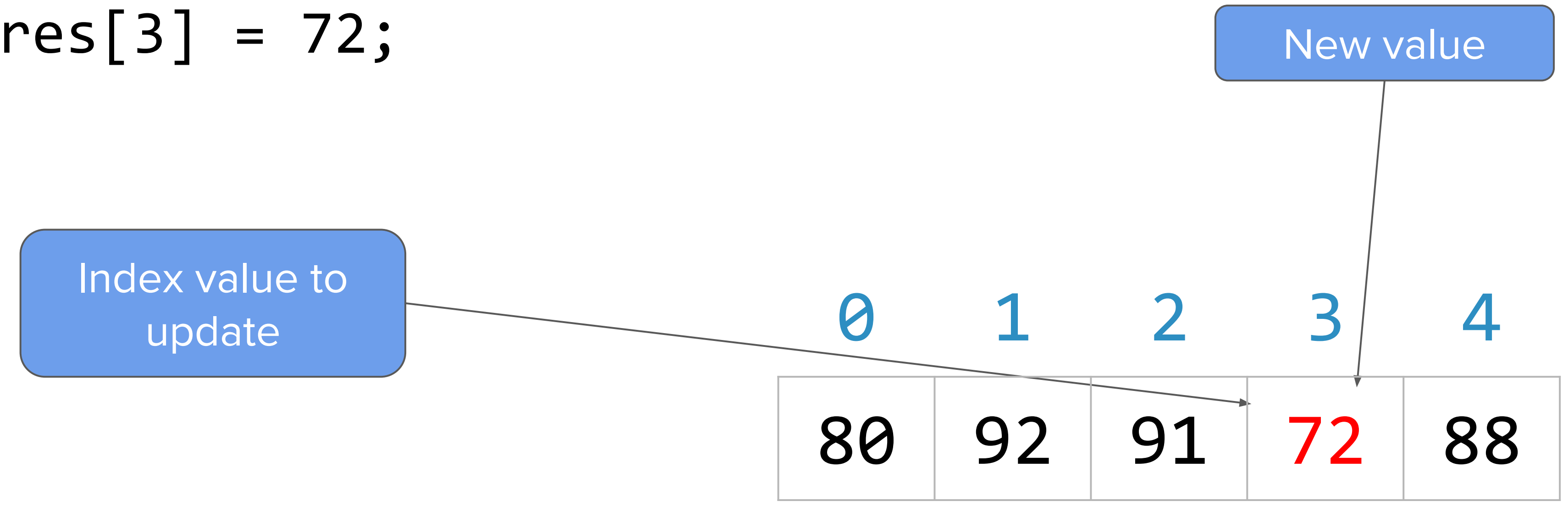
```
scores[3] = 72;
```



Setting a Value at an Index

```
int[] scores = {80, 92, 91, 68, 88};
```

```
scores[3] = 72;
```



Arrays Start at Index 0!

Arrays start at index 0, not 1!

This is just like what we saw with Strings. In fact a String is nothing more than an Array of characters.

This is a source of a lot of bugs, so be careful!

0	1	2	3	4
80	92	91	68	88

Getting the Array Length

```
int[] scores = {80, 92, 91, 68, 88};

int length = scores.length; // gets value 5
```

Note: There are 5 elements, but the first index is 0 and the last index is 4!

0	1	2	3	4
80	92	91	68	88

Getting the Array Length

```
int[] scores = {80, 92, 91, 68, 88};

int length = scores.length; // gets value 5
```

No ( ) for arrays!

Note: There are 5 elements, but the first index is 0 and the last index is 4!

0	1	2	3	4
80	92	91	68	88

Last Index of Array

If you want to get the last index of the array it is always at:

```
int lastIndex = array.length - 1;
```

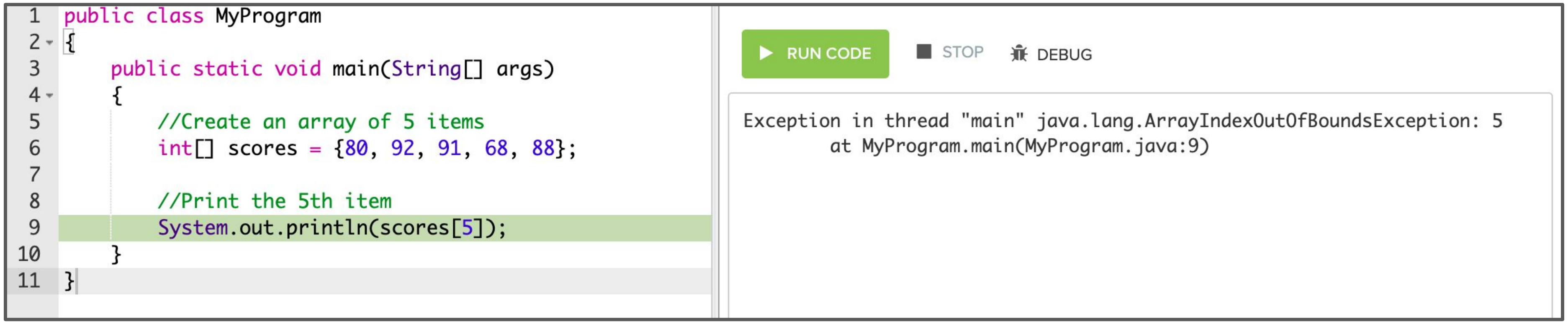
This is because arrays are 0 - indexed!

0	1	2	3	4
80	92	91	68	88

ArrayIndexOutOfBoundsException

If we try to access a value of our array outside the index value list, we get an ArrayIndexOutOfBoundsException.

```
int[] scores = {80, 92, 91, 68, 88};
```



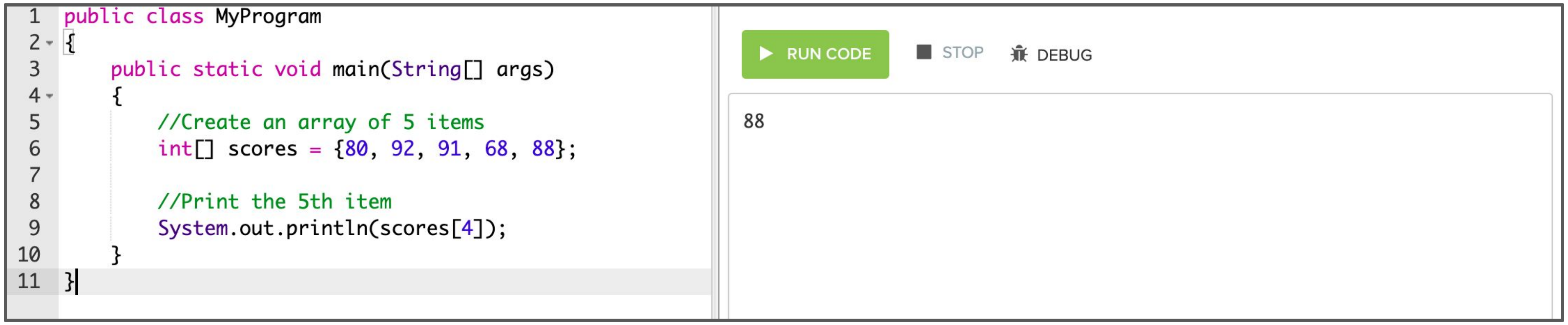
```java
public class MyProgram
{
    public static void main(String[] args)
    {
        //Create an array of 5 items
        int[] scores = {80, 92, 91, 68, 88};

        //Print the 5th item
        System.out.println(scores[5]);
    }
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at MyProgram.main(MyProgram.java:9)
```

ArrayIndexOutOfBoundsException

If we try to access a value of our array outside the index value list, we get an ArrayIndexOutOfBoundsException.

```
int[] scores = {80, 92, 91, 68, 88};
```


```
public class MyProgram
{
    public static void main(String[] args)
    {
        //Create an array of 5 items
        int[] scores = {80, 92, 91, 68, 88};

        //Print the 5th item
        System.out.println(scores[4]);
    }
}
```

```
88
```

Array Recap

- An array is an object that can store many values of the **same type**.

- Making an array requires declaring how many values the array can hold, and what type of values the array will hold.

- The array *type* and *capacity* **cannot change** once the array has been made.

Now It’s Your Turn!

CodeHS

Concepts Learned this Lesson

Term	Definition
Array	Arrays are lists that store many values of the same type
Index	Array values are stored at a particular index and we access elements in the array by referencing this index value. Index values in Arrays start a 0.
array.length	Returns the length of the array
array[index]	Accesses an element in the array to either update or retrieve.

Standards Covered

- (LO) VAR-2.A Represent collections of related primitive or object reference data using one- dimensional (1D) array objects.
- (EK) VAR-2.A.1 The use of array objects allows multiple related items to be represented using a single variable.
- (EK) VAR-2.A.2 The size of an array is established at the time of creation and cannot be changed.
- (EK) VAR-2.A.3 Arrays can store either primitive data or object reference data.

Standards Covered

- (EK) VAR-2.A.4 When an array is created using the keyword new, all of its elements are initialized with a specific value based on the type of elements:
    - Elements of type int are initialized to 0
    - Elements of type double are initialized to 0.0
    - Elements of type boolean are initialized to false
    - Elements of a reference type are initialized to the reference value null. No objects are automatically created
- (EK) VAR-2.A.5 Initializer lists can be used to create and initialize arrays.
- (EK) VAR-2.A.6 Square brackets ([ ]) are used to access and modify an element in a 1D array using an index.
- (EK) VAR-2.A.7 The valid index values for an array are 0 through one less than the number of elements in the array, inclusive. Using an index value outside of this range will result in an ArrayIndexOutOfBoundsException being thrown.

Additional Notes: Arrays are Objects

Arrays in Java are objects, not primitives. This means when it is passed to a method you are getting the actual object, not a copy. Any change in the method, updates the original array.

```
public static void main(String[] args) {

  int[] numbers = {20, 30, 40, 50};
  changer(numbers);
  for (int num : numbers){
      System.out.println(num);
  }
}

public static void changer(int[] nums){
    nums[1] += 10;
}
```

Output:

```
20
40
40
50
```

Additional Notes: Arrays are Objects

Arrays in Java are objects, not primitives. This means when it is passed to a method you are getting the actual object, not a copy. Any change in the method, updates the original array.

```
public static void main(String[] args) {

  int[] numbers = {20, 30, 40, 50};
  changer(numbers);
  for (int num : numbers){
      System.out.println(num);
  }
}

public static void changer(int[] nums){
    nums[1] += 10;
}
```

Change made in method

Output:

```
20
40
40
50
```

Additional Notes: Arrays are Objects

Arrays in Java are objects, not primitives. This means when it is passed to a method you are getting the actual object, not a copy. Any change in the method, updates the original array.

```java
public static void main(String[] args) {

  int[] numbers = {20, 30, 40, 50};
  changer(numbers);
  for (int num : numbers){
      System.out.println(num);
  }
}

public static void changer(int[] nums){
    nums[1] += 10;
}
```

Reflected here

Change made in method

Output:

```
20
40
40
50
```