Java Foundations

One-Dimensional Arrays

Objectives

- This lesson covers the following objectives:
  - Create and initialize one-dimensional arrays
  - Modify an array element
  - Traverse a one-dimensional array by using a for loop
  - Identify the cause of an `ArrayIndexOutOfBoundsException`

Can a Variable Hold More Than One Value?

- So far we have used many types of variables, but each variable stores one value at a time:
  - one int or one String or one double
- Here's an example of a String variable, rockBand, that can hold any value – Joe, Paul, Ed, Rob:
  - Since there are only 4 possible values, it isn't too difficult to change the variable's value manually

```
String rockBand = "Joe";
String rockBand = "Paul";
String rockBand = "Ed";
String rockBand = "Rob";
```

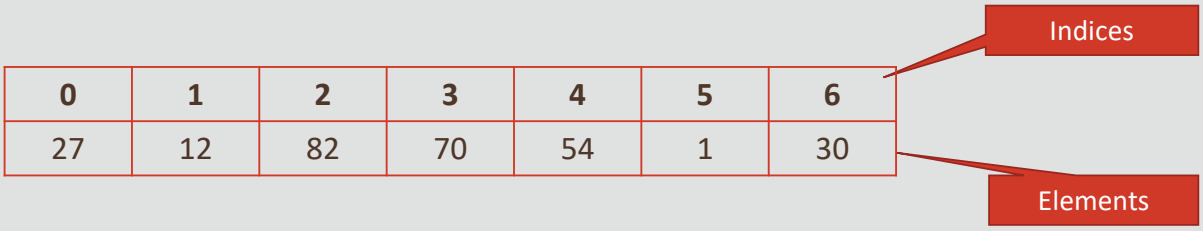
Number of Variables Required

- But there are times when you'll need to hold more than one value in a variable
- What if you wanted to set aside a variable for each one of the RockBand songs? (That would be 300 variables for each song!)
- However, it can be time-consuming and tedious to create hundreds of variables

```
String rockBandSong1 = "Rainy day";
String rockBandSong2 = "Forever";
String rockBandSong3 = "Something about you";
String rockBandSong4 = "Love you always";
……
```

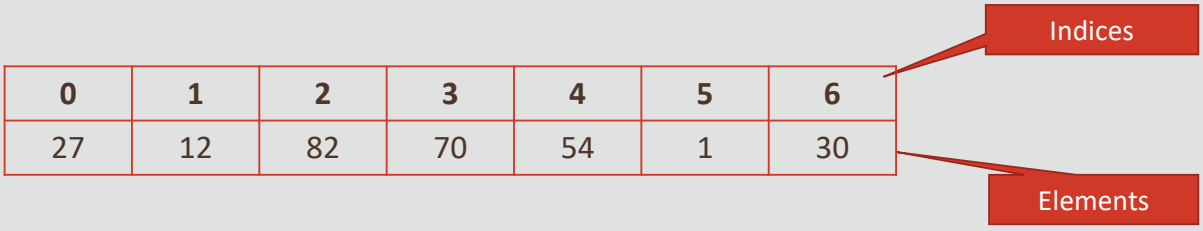
Arrays Can Provide a Solution

- In Java, an array is an indexed container that holds a set of values of a single type
- Arrays allow you to create a single identifier to organize many items of the same date type



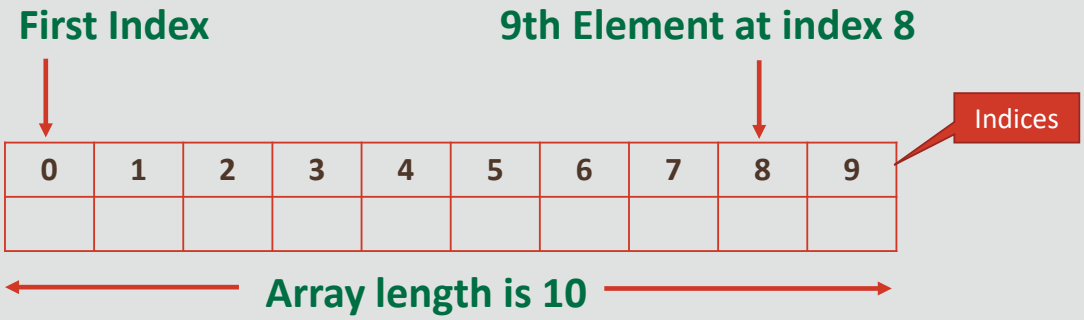
Arrays Can Provide a Solution

- Each item in an array is called an element
- Arrays make storing and accessing a large number of values simple and easy



Arrays Are Accessed by Their Index

- You can access each element in an array by its numerical index
- The index of the first element is 0
- A 10-element array has 0 to 9 indices



Array Data Types

- Arrays can be of any data type, but all elements have to share the same type, such as:
  - Primitive:
    - Example: Array of int types

27	12	82	70	54	1	30

  - Predefined objects:
    - Example: Array of Strings

Sun	Mon	Tue	Wed	Thu	Fri	Sat

Array Data Types

- Arrays can be of any data type, but all elements have to share the same type, such as:
  - Programmer-defined objects:
    - (such as instances of a class that you create)
    - Example: Array of objects of the Student class

Student1	Student2	Student3	Student4	Student5

Declaring an Array

- Arrays, like all variables, must be declared prior to use
- You can declare an array by using the following syntax:

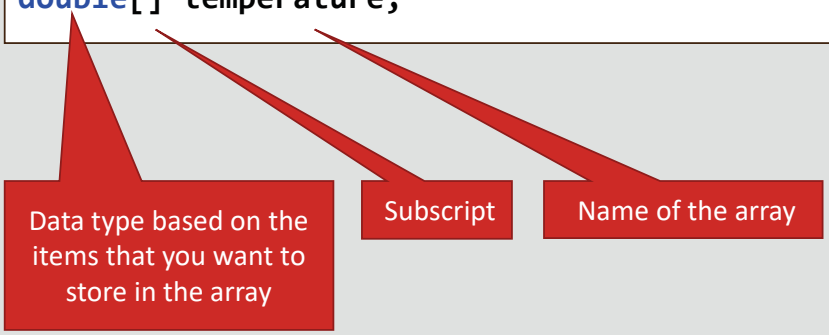
```
type[] arrayIdentifier;
```

- Notice the bracket notation [ ] after the data type

Declaring an Array of Temperature Values

- Suppose you want to store different temperature readings in an array
- You can declare an array as follows:

```
double[] temperature;
```



Declaring an Array: Two Methods

- You can declare an array in two ways:

```
1. int[] prime;
2. int prime[];
```

- Both syntaxes are equivalent
- The first format generally is more readable and should be used

Is Declaring an Array Sufficient?

- Declaring an array isn't enough to begin using it in your program
- Before you use an array, you need to tell Java to create space in memory for the elements that it will hold

Is Declaring an Array Sufficient?

- Use the following syntax:

```
data_type[] variable_name = new data_type[size];
variable_name[index] = value; //repeat for each element
```

- The size value determines the number of items that your array can hold
- Arrays can't grow beyond this size

Creating an Array

- For example, if you want to create an array to hold 100 integers, you could do the following:

```
int[] myIntArray;
myIntArray = new int[100];
```

- Alternatively, you could perform these two lines in one step:

```
int[] myIntArray = new int[100];
```

What Do the Code Snippets Do?

1

```
int[] ages = new int[3];
ages[0] = 19;
ages[1] = 42;
ages[2] = 92;
```

2

```
String[] names = new String[3];
names[0] =  "Mary";
names[1] =  "Bob";
names[2] =  "Carlos";
```

What About Declaring and Initializing an Array in a Single Step?

- You can also declare and initialize the array in a single step with known values:

```
type[] arrayIdentifier = {comma-separated list of values};
```

- For example, declare arrays of types String and int:

```
String[] names = {"Mary", "Bob", "Carlos"};
int[] ages = {25, 27, 48};
```

Declaration and initialization in one step

What About Declaring and Initializing an Array in a Single Step?

- Notice that this method doesn't specify size
- It's assigned a size based on the number of elements between the braces (**{ }**)

```
String[] names = {"Mary", "Bob", "Carlos"};
int[] ages = {25, 27, 48};
```

Declaration and initialization in one step

Accessing Array Elements

- Arrays are sequential structures, meaning that items are stored one after another in an array
- You can access an individual element of an array by using a bracket notation
- For example, here's how you get values from the ages array:

```
int[] ages = {25, 27, 48};
int myAge = ages[0];
int yourAge = ages[1];
System.out.println("My age is " + ages[0]);
```

How Do You Set the Value of an Array Element?

- You can set values to the array's elements like this:

```
String[] names = {"Mary", "Bob", "Carlos"};
names[0] = "Gary";
names[1] = "Rob";
```

- After you set the values to the elements at indices 0 and 1, the names array looks like this:

0	**1**	2
Gary	**Rob**	Carlos
names[0]	names[1]	names[2]

Exercise 1

- Can you identify the three components of an array declaration for each of these arrays of primitive data types?
  - Data Type
  - Name
  - Size

```
int[] myArray;

myArray = new int[20];

char[] sentence = new char[100];

double[] teamPoints = new double[5];
```

Default Initialization of Arrays

- When arrays are declared but not yet initialized, the elements are given the default value associated with the data type
- Here's an example:

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

Default values for the elements of this array

Index:	0	1	2	3	4
Values:	0	0	0	0	0

How Do You Access the Length of an Array?

- So far, you created an array with a certain number of elements
- After creation, you can't change the length of an array. They can't grow beyond this size
- You can access the size of any array by using the array's length property

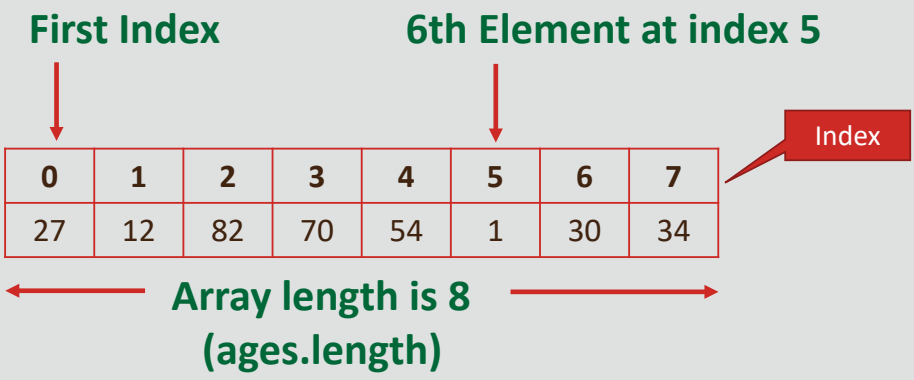
```
int primes[] = {2, 3, 5, 7, 11, 13, 17};
System.out.println("Array length: " + primes.length);

//prints 7
```

Array Indices and Length

- For example, the following code snippet displays the size of the ages array:

```
int ages[] = {27, 12, 82, 70, 54, 1, 30, 34};
System.out.println(ages.length); //prints 8
```



Exercise 2

- Import and open the `ArrayEx` project
- Examine `ArrayEx1.java`
- Modify the program to implement …
  - Declare a one-dimensional array named score of type int that can hold 9 values
  - Declare and initialize a one-dimensional byte array named values of size 10 so that all entries contain 1
  - Uncomment the two lines that are commented out and then resolve the syntax errors

Traversing an Array

- To iterate through, or traverse, an array means to process through each element of the array by index number
- You can access each element of an array to …
  - Print the elements
  - Search for an element
  - Initialize the elements of an array with the same value

Using a for Loop to Traverse Arrays

- You can use a for loop to traverse arrays
- The for loop lets you easily iterate over a known range
- You can visit every array element by using the length property of the array in the iteration condition

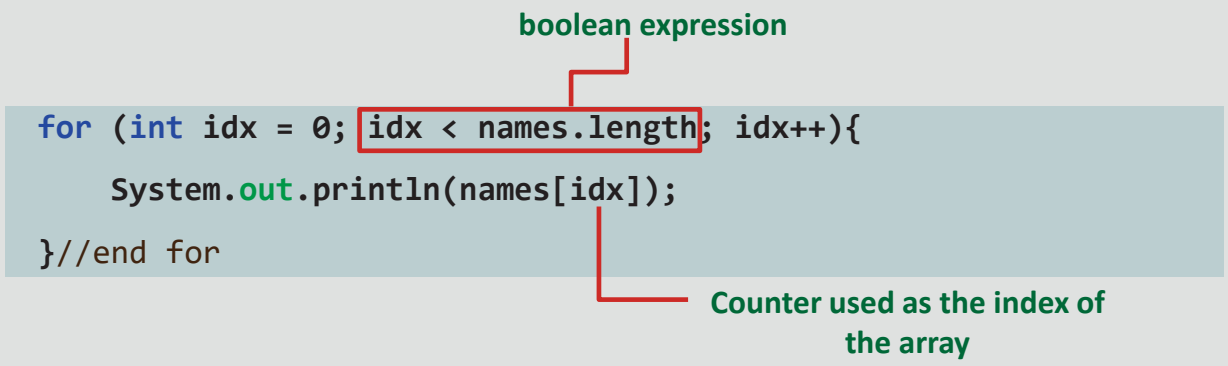
```
int[] array = { -20, 19, 1, 5, -1, 27, 19, 5 } ;
int min = array[0]; // initialize the current minimum
for (int index=0; index < array.length; index++ )
  if (array[index] < min)
      min = array[index] ;
System.out.println("The minimum of this array is: " + min);
```

How Do You Print the Values of a names Array?

- Consider an array of Strings, names:

```
String names[] = new String["Tom", "David", "Mike"];
```

- Traverse the names array by using the for loop:



Using a for-each Loop to Traverse an Array

- You can use a for-each loop, an alternative to using the for loop, to iterate through an array
- The for-each loop …
  - Works the same way as the for loop, but it's implemented in a simpler way
  - Is also called an enhanced for loop

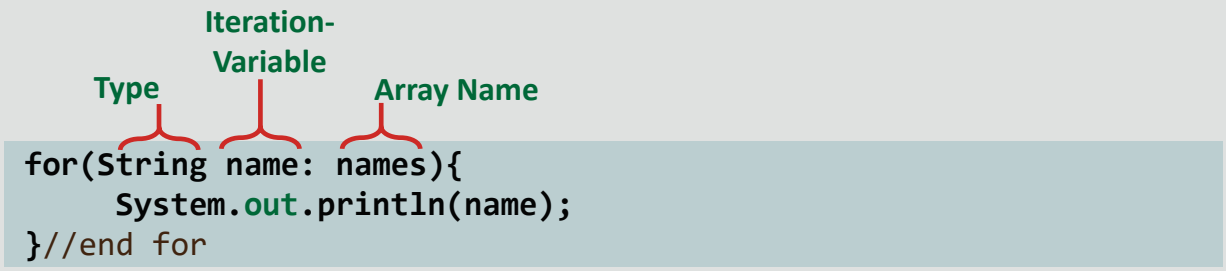
Using a for-each Loop to Traverse an Array

- Syntax:

```
for (<type> <iteration variable> : <array name>) {
   <code_block to be performed for each array element>
}//end for
```

How Do You Print the Values of a names Array by Using a for-each Loop?

- Here's an example of traversing the names array by using a for-each loop:




```
for(String name: names){
     System.out.println(name);
}//end for
```

How Do You Print the Values of a names Array by Using a for-each Loop?

- For each iteration of the loop, the next element in the array is retrieved and stored in an iteration-variable
- The type must be the same as the elements stored in the collection

for-each Loop vs. for Loop

- for-each loop

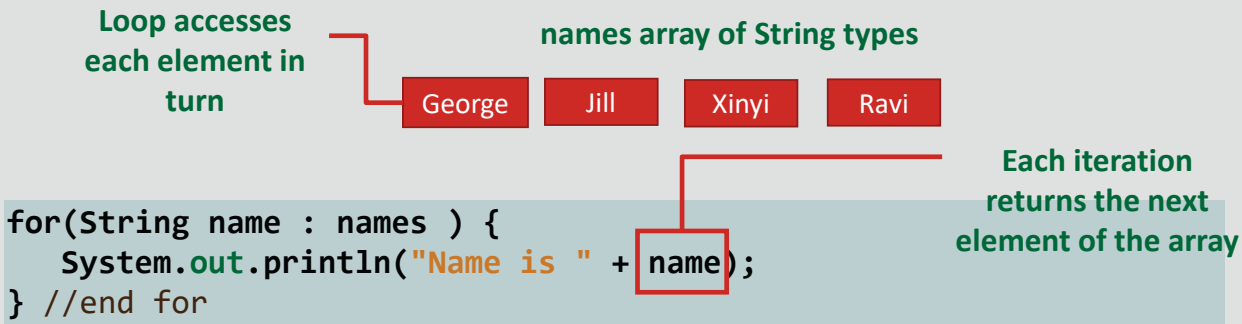
```
for(String name: names){
    System.out.println(name);
}//end for
```

- for loop

```
for (int idx = 0; idx < names.length; idx++){
    System.out.println(names[idx]);
}//end for
```

- The output of both loops is the same

Processing a String Array




```
for(String name : names ) {
   System.out.println("Name is " + name);
} //end for
```

- Output:

```
Name is George
Name is Jill
Name is Xinyi
Name is Ravi
```

Putting It All Together

- Let's look at an example where you need to …
  - Enter the scores of 10 students by using a Scanner object
  - Display the scores that you entered
  - Calculate the average of the scores that you entered

Let's Compute the Average Score

```
public class StudentScores {
   public static void main(String args[]) {
      double scores[] = new double[10];
      double sum = 0.0, avg = 0.0;
      Scanner keyboard = new Scanner(System.in);

      System.out.println("Enter scores of 10 students");
      for(int i = 0; i < scores.length; i++) {
         scores[i] = keyboard.nextInt();
      }//end for
      System.out.println("Display the  scores of 10 students");
      for(int i = 0; i < scores.length; i++) {
         System.out.println(scores[i]);
      }//end for
      for(int i = 0; i < scores.length; i++) {
         sum = sum + scores[i];
         avg = sum / scores.length;
      }//end for
      System.out.println("The average score of the class: " + avg);
    }//end method main
}//end class StudentScores
```

Exercise 3

- Import and open the `ArrayEx` project
- Examine `ComputeAvg.java`
- Modify the program to implement …
  - In a certain class, there are five tests, each worth 100 points
  - Input five test scores from the console
  - Store the test scores in an array
  - Calculate the student's average scores

What is an ArrayIndexOutOfBoundsException?

- As you already know, an array has a fixed size
- The index must be in a range interval [0, n-1], where n is the size of the array
- If an index is either negative or greater than or equal to the size of the array, then the array index is out of bounds
- If an array index is out of bounds, the JVM throws an ArrayIndexOutOfBoundsException
- This is called automatic bounds checking

What Happens When This Exception Occurs?

- The ArrayIndexOutOfBoundsException is thrown only at run time
- The Java compiler doesn't check for this exception when a program is being compiled
- The program is terminated if this exception isn't handled

How Do You Identify the ArrayIndexOutOfBoundsException?

```
public static void main(String[] args) {
    int primes[] = {2, 3, 5, 7, 11, 13, 17};
    System.out.println("Array length: " + primes.length);
    primes[10] = 20; //

    System.out.println("The first few prime numbers are:");
    for (int i : primes) {
        System.out.println(i);
    }//end for
}//end method main
```

The index of the array is 0-6, and it's trying to access an element at index 10

- Output:

```
Array length: 7
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 10
        at arraysdemo.ArraysDemo.main(ArraysDemo.java:21)
Java Result: 1
```

Exercise 4

- Import and open the `ArrayEx` project
- Examine `ArrayEx2.java`
- Perform the following:
  - Run the program and observe the error
  - Modify the program to resolve the error
  - Using a for-each loop, display all browsers that are stored in the array

Summary

- In this lesson, you should have learned how to:
  - Create and initialize one-dimensional arrays
  - Modify an array element
  - Traverse a one-dimensional array by using a for loop
  - Identify the cause of an `ArrayIndexOutOfBoundsException`