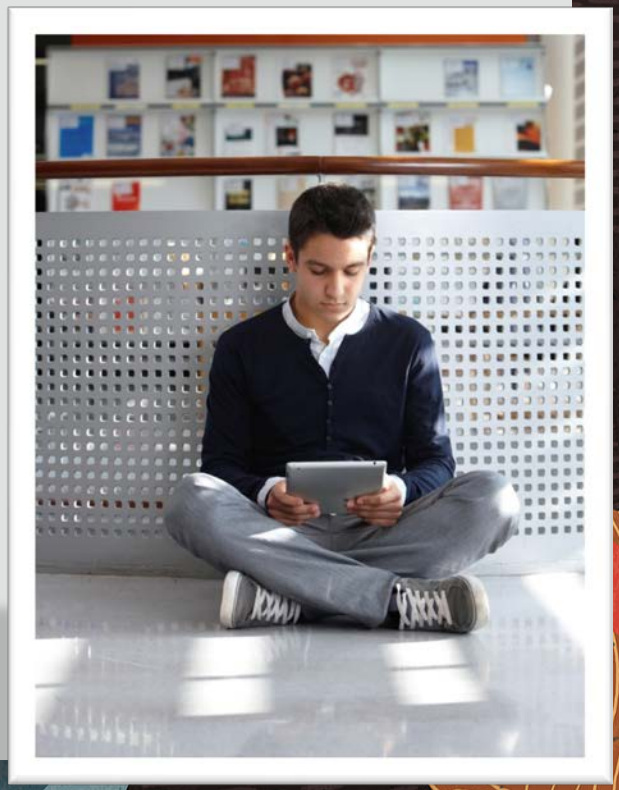

ORACLE
Academy

Oracle Academy
Java for AP Computer Science A

10-3

Arrays



ORACLE
Academy

Objectives

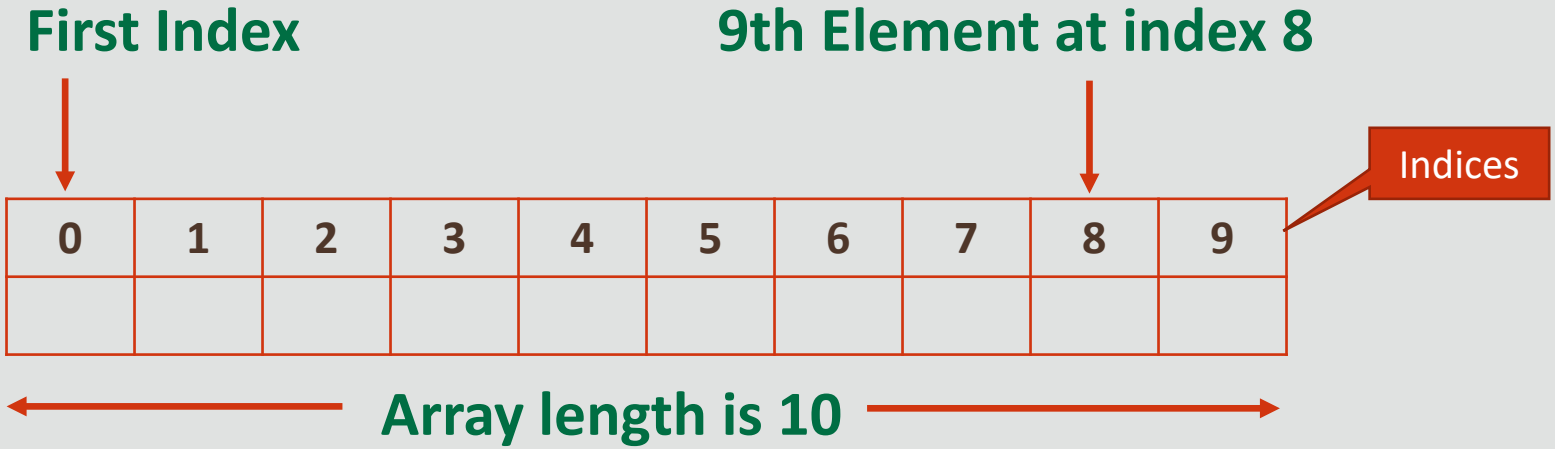
- This lesson covers the following objectives:
  - Review:
    - Write a single-dimensional array in a Java program using primitive data types
  - Review:
    - Write a single-dimensional array in a Java program using reference (Object) types
  - Write a 2-dimensional array in a Java program using primitive data types
  - Write a 2-dimensional array in a Java program using reference (Object) types
  - Declare an array, initialize an array, and traverse the array

Overview

- This lesson covers the following topics:
  - Describe array initialization
  - Distinguish between the String method length() and an array's length value
  - Rewrite a Java program to store integers into an array, perform a mathematical calculation, and display the result
  - Use alternative array declaration syntax

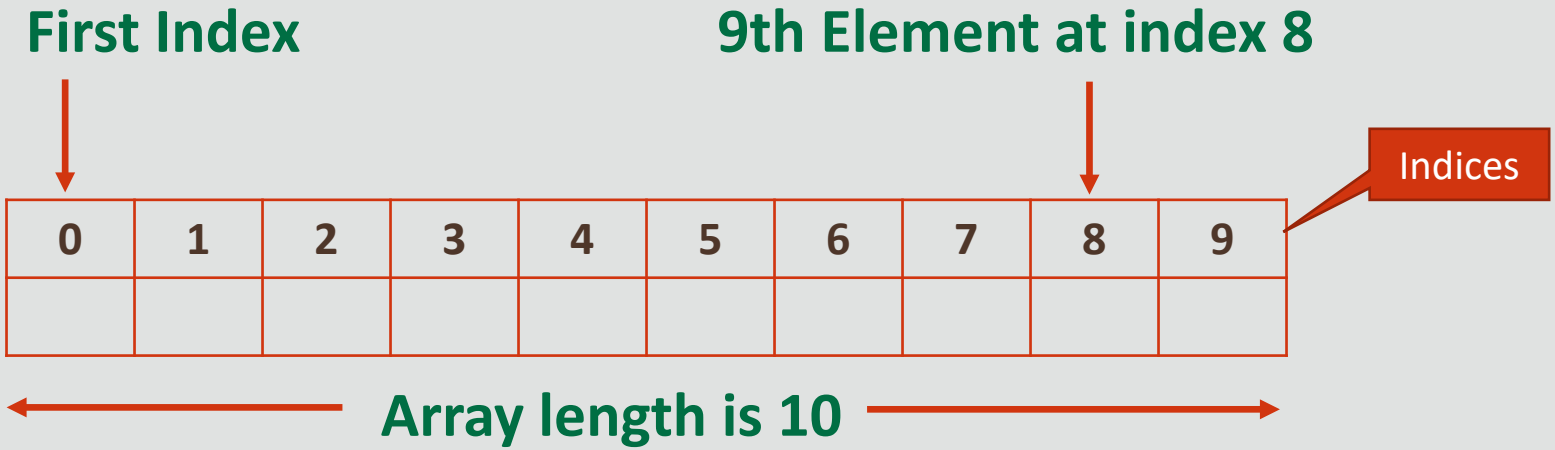
What is an Array?

- An array is a collection of values of the same data type stored in a container object
- Can be any number of values
- Length of the array is set when the array is declared
- Size is fixed once the array is declared



Array Elements

- Every value is called an element
- Each element is accessed by an index
- Index must be an integer
- Index always starts at 0



Array Data Types

- Arrays may contain any data type, such as:
  - Primitive
  - Pre-defined objects, such as Strings from the Java API
  - Programmer-Defined objects, such as instances of a class that you create

Declaring an Array

- There are three parts to declaring an array:
  - Data type
  - Variable name
  - Array size

Declaring an Array Examples

- An array declaration can be done in one or two lines
- Both examples below are equivalent array declarations

```
data_type[] variable_name;
variable_name = new data_type[size];
//declare an array using two lines of code
```

```
data_type[] variable_name = new data_type[size];
//declare an array using one line of code
```

Declaring an Array Example

- A friend brings you a bouquet of six different flowers
- You want to write a program to store each type of flower in the bouquet
- The code segments below are identical ways to declare an array of Strings called myBouquet
- The size is set to six elements so it may store each type of flower in your bouquet

```
String[] myBouquet;
myBouquet = new String[6];
```

```
String[] myBouquet = new String[6];
```

myBouquet Array Example Explained

```
String[] myBouquet;
myBouquet = new String[6];
```

```
String[] myBouquet = new String[6];
```

The size of the array is 6 elements, one for each flower.

Data type stores the types of flowers as Strings.

Array name is myBouquet.

Identify Components of an Array Declaration

- Identify the three components of an array declaration for each of these arrays of primitive data types

```
1.   int[] myArray;
     myArray = new int[20];

2.   char[] sentence = new char[100];

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

	Data Type	Name	Size
1			
2			
3			

Alternative Array Declaration Syntax

- An alternative way to declare arrays is to place the brackets used to declare an array on either side of the array identifier
- Syntax introduced:

```
int[] primeNumbers;
int[] evenNumbers;
double[][] prices;
String[] words;
Point[] coordinates;
Rectangle[][] blocks;
```

Alternate syntax:

```
int primeNumbers[];
int evenNumbers[];
double prices[][];
String words[];
Point coordinates[];
Rectangle blocks[][];
```

Alternative Array Declaration Syntax

- Either declaration will work
  - The Alternate Syntax is similar to that used in the language of C
  - The syntax used to introduce arrays actually reads much clearer
  - The first line would read "Integer array primeNumbers."
- Syntax introduced:

```
int[] primeNumbers;
int[] evenNumbers;
double[][] prices;
String[] words;
Point[] coordinates;
Rectangle[][] blocks;
```

Alternate syntax:

```
int primeNumbers[];
int evenNumbers[];
double prices[][];
String words[];
Point coordinates[];
Rectangle blocks[][];
```

Initializing an Array

- Once you declare an array, you must initialize it to set the values for specified indexes
- There are three components to initializing an array:
  - Variable name
  - Index
  - Value

Ways to Initialize an Array

- An array can be initialized in two different ways:
  - Declare the array, then initialize each element
  - Have declaration and initialization occur in the same step
- Example of declaring and initializing in two steps:

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

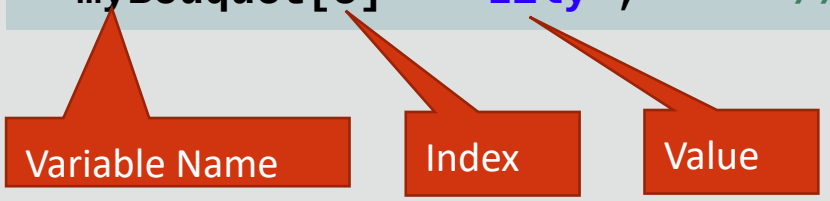
- Example of declaring and initializing in one step:

```
data_type[] variable_name = {val1, val2, …};
```

Initializing an Array Example 1

- Recall the declaration of the String array myBouquet
- Once the array is declared, the code below initializes the elements and stores the flower types
- The index of an array begins at 0, so the first element will be added to index 0

```
String[] myBouquet = new String[6];//previous declaration
 myBouquet[0] = "Rose";       //Store "Rose" as the first element
 myBouquet[1] = "Sunflower"; //Store "Sunflower" as the second
 myBouquet[2] = "Daisy";      //and so on
 myBouquet[3] = "Dandelion";
 myBouquet[4] = "Violet";
 myBouquet[5] = "Lily";       //"Lily" is the last (sixth) element
```



Initialize an Array Example 1

- The array of flower types will look like this:

Index:	0	1	2	3	4	5
Value:	Rose	Sunflower	Daisy	Dandelion	Violet	Lily

Initialize an Array Example 2

- The array myBouquet could also be declared and initialized using the second notation as follows:

```
String[] myBouquet = {"Rose", "Sunflower", "Daisy",
                      "Dandelion", "Violet", "Lily"};
```

- Notice that using this method does not specify size, but it is assigned a size based on the number of elements in the list between the {}

First Notation versus Second Notation

- It seems that the second notation is much shorter and easier to code
- Why would you need the first notation?

First Notation versus Second Notation

- Consider taking in five numbers from a user and storing them in an array called myArray
- You would need to declare the array, then initialize the elements one at a time as the user enters each number

```
int[] myArray = new int[5]; //Declare the array of size 5
MyArray[0] = 7;             //The user entered 7
MyArray[1] = 24;            //The user entered 24
MyArray[2] = 352;           //The user entered 352
MyArray[3] = 2;             //The user entered 2
MyArray[4] = 37;            //The user entered 37
```

First Notation versus Second Notation

- The second notation may be an easier way to initialize the array if you already know the contents of the array when declaring it

```
int[] myArray = {7, 24, 352, 2, 37};
```

Array Representation

- When arrays are declared but not yet initialized, the elements are given the default value associated with the data type
- For example, the default for numeric data types, such as int, is 0
- The default for Object types, such as String is "" (null)
- When the array is declared, the representation in the table is as follows

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

Index:	0	1	2	3	4
Value:	0	0	0	0	0

Updated Array Representation

- Once you begin to initialize elements, the array is updated
- The new representation in the table is as follows

```
myArray[0] = 32;
myArray[3] = 27;
```

Index:	0	1	2	3	4
Value:	32	0	0	27	0

Arrays Object Types

- Arrays are not restricted to storing primitive data types
- They can store any object type, including types you define
- For example, if a Flower class existed:
  - The flowers could be stored in the array rather than storing the flower type as a String
  - Since we know what flowers to include, the second notation can be used to initialize myBouquet of six Flowers

```
Flower[] myBouquet = {new Flower("Rose"), new Flower("Sunflower"),
                      new Flower("Daisy"), new Flower("Dandelion"),
                      new Flower("Violet"), new Flower("Lily")};
```

Accessing Array Length

- With each declaration of an array, you must define a size, or length, of the array
- The length is stored as an instance variable for that object and can be accessed using the arrayName.length notation
- This technique is useful in the following example:
  - Establish an array with a size based on the user input
  - Enter a segment of code where the user's input is no longer in scope
  - You would need to access the instance variable length for that array
  - In short, arr.length returns the length of the array, arr

Iterate Through an Array

- To iterate through, or traverse, an array means to progress through each element of the array by index number
- Iterating through an array is useful when:
  - You wish to access each element of an array in order
  - You wish to initialize the elements of an array as all the same value
- Use .length when iterating rather than the integer value entered when declaring the array
- This will assure that you do not receive an index out of bounds error

Iterate Through an Array Example

- To initialize an array of integers called allTwos, so that each element is 2, use a for loop as shown below
- The first line of code is the array declaration
- It declares an array called allTwos with a size of 10
- The for loop iterates through the indexes and for each index in the array, the value at that index is set to 2

```
int[] allTwos = new int[10];
for(int index = 0; index < allTwos.length; index++){
   allTwos[index] = 2;
}//end for
```

Note how the length of the array is accessed to keep from going out of bounds for the array's index.

When Iteration is Helpful

- In the flower example, iteration helps if you wish to print out the flower types stored in the myBouquet array

```
String[] myBouquet = {"Rose", "Sunflower", "Daisy", "Dandelion",
                      "Violet", "Lily"};
```

- Use a for loop to iterate through this array
- The initialized counter within the for loop can be used for incrementing through the indexes, as shown below
- What displays as a result of this code?

```
//remember that the index range is 0 to 5 for an array of size 6
  for(int index = 0; index < myBouquet.length; index++){
          System.out.println(myBouquet[index]);
```

for-each Loop

- Java offers a for-each loop, an alternative to using the initialized counter to iterate through an array
  - When used to access the elements of an array, the for- each loop works the same way as the for loop, but is implemented in a simpler way
  - If we replace the for loop code from our previous example with the following code, we get the same result

```
//remember that the index range is 0 to 5 for an array //of size 6
for (String myFlower : myBouquet)
{
    System.out.println(myFlower);
}//end for
```

for-each Loop

- The for-each loop accesses and (one-at-a-time) returns all elements of an array
  - Changes to the array elements cannot be made using a for - each loop
  - If we replace the for loop code from our previous example with the following code, we get the same result
  - The example below will print the length of each string in the array myBouquet

```
//remember that the index range is 0 to 5 for an array //of size 6
for (String myFlower : myBouquet)
{
    System.out.println(myFlower);
}//end for
```

for-each Loop Example

- Both implementations of the code below will display the information about the array elements

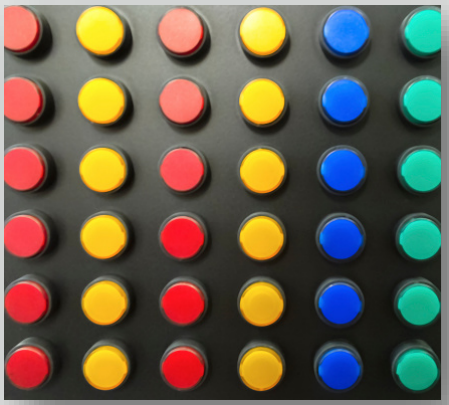
```
public class Bouquet {
   public static void main(String[] args){
      String[] myBouquet = {"Rose","Sunflower","Daisy",
                            "Dandelion","Violet", "Lily"};

      //use a for loop to iterate through the array
      for(int index = 0; index < myBouquet.length; index++){
           System.out.println(myBouquet[index]);
      }//end for
      //use a for each to iterate through the array
      for (String myFlower : myBouquet){
        System.out.println(myFlower);
      }//end for
   }//end method main
}//end class Bouquet
```

What We Know About Arrays

- What we know about arrays:
  - Arrays are an object type that can store any primitive or object type
  - Therefore, arrays can store arrays
  - The concept of storing an array of arrays is called a two-dimensional array

Two-Dimensional Arrays

- A two-dimensional array, called an "array of arrays," is an array that stores other arrays
- The number of arrays contained within the array is defined upon declaration
- The number of items in each internal array is also defined upon declaration



Two-Dimensional Array Example

- This example shows an array of two arrays with three elements in each array
- A two-dimensional array can be visualized as a table with rows and columns
- The example below has two rows and three columns

```
int[][] nums = { {14,51,16}, {12,73,87} };
```

Declaring a Two-Dimensional Array

- Components of two-dimensional arrays:
  - Data type
  - Variable name
  - Array size
- To declare a two-dimensional array, use either of the syntax examples below

```
data_type[][] variable_name;
variable_name = new data_type[size1][size2];
//declare it using two lines of code
```

```
data_type[][] variable_name = new data_type[size1][size2];
//declare it using one line of code
```

Declaring a Two-Dimensional Array Example 1

- Identify the three components in the following primitive data type examples

```
int[][] myArray;
myArray = new int[2][3];
char[][] sentence = new char[10][10];
```

Declaring a Two-Dimensional Array Example 1

- When declaring a two-dimensional array:
  - The number in the first set of brackets [2] is the number of arrays the container holds (rows)
  - The number in the second set of brackets [3] is the number of elements in each of those arrays (columns)
- Another way to declare myArray may look like this:

```
int[][] myArray = { {0,0,0}, {0,0,0} };
```

Declaring a Two-Dimensional Array Example 2

- Your friend brought you three of each type of flower, and each of the three flowers are different colors
- A single-dimensional array makes it tedious to keep track of this data
- A two-dimensional array allows you to store six arrays, one for each flower type, and have each array store the colors of each of the three flowers
- The following code declares an array that holds six arrays, each of length three:

```
String[][] myBouquet = new String[6][3];
```

Initializing a Two-Dimensional Array

- Two-dimensional arrays, just like single-dimension arrays, can be initialized using two different methods
- Method 1:
  - i is the index of the internal array (row) and j is the index of the element within that array (column) that is being initialized

```
public class TwoDTester{
  public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    int[][] nums = new int[3][2];

    for(int i = 0; i < nums.length; i++){
      for(int j = 0; j < nums[i].length; j++){
        System.out.println("Enter a value for row " + i + ", column " + j);
        nums[i][j] = in.nextInt();
      }//end for
    }//end for
  }//end main
}//end class TwoDTester
```

Initializing a Two-Dimensional Array

- This method is also called "row-major order"
  - where i =row number and j=column number
  - Each row is completed with column values before moving on to the next row:

```
public class TwoDTester{
  public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    int[][] nums = new int[3][2];

    for(int i = 0; i < nums.length; i++){
      for(int j = 0; j < nums[i].length; j++){
        System.out.println("Enter a value for row " + i + ", column " + j);
        nums[i][j] = in.nextInt();
      }//end for
    }//end for
  }//end main
}//end class TwoDTester
```

Initializing a Two-Dimensional Array

- But what if you need to enter all of the row values for each column before moving on to the next column?
- Below, the array is filled in "column-major order"
  - where i =column number and j=row number
  - Each column is completed with all row values before moving on to the next column:

```
public class TwoDTester{
  public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    int[][] nums = new int[3][2];

    for(int i = 0; i < nums[0].length; i++){
      for(int j = 0; j < nums.length; j++){
        System.out.println("Enter a value for row " + j + ", column " + i);
        nums[j][i] = in.nextInt();
      }//end for
    }//end for
  }//end main
}//end class TwoDTester
```

Initializing a Two-Dimensional Array

- Method 2: Declares and initializes all arrays and all elements within those arrays in the same line of code
- However, you must know the values that you want the array to contain to initialize the array at the same time that it is declared

```
public class TwoDTester2{
   public static void main(String[] args){
      int[][] nums = {{2,3,7},{15, 98, 2}};
   }//end main
}//end class TwoDTester2
```

Initializing a Two-Dimensional Array Example

- Method 1:

```
int[][] myArray = new int[3][2];
MyArray[0][0] = 7;
MyArray[0][1] = 24;
MyArray[1][0] = 352;
MyArray[1][1] = 2;
MyArray[2][0] = 37;
MyArray[2][1] = 65;
```

Index of internal array

Index of element within the internal array

- Method 2:

```
int[][] myArray = new int[][] {{7, 24}, {352, 2}, {37, 65}};
```

Using Second Notation to Initialize the Array

- Since we already know the content of the arrays for our bouquet, use the second notation to initialize the array
  - Recall that the order is "Rose", "Sunflower", "Daisy", "Dandelion", "Violet", then "Lily"
  - These flowers will be represented by the indexes 0, 1, 2, 3, 4, and 5

```
String[][] myBouquet = {{"Red", "Peach", "Yellow"},
                        {"Yellow", "White", "Blue"},
                        {"Green", "Blue", "Purple"},
                        {"White", "White", "White"},
                        {"Purple", "Pink", "Violet"},
                        {"Pink", "Orange", "White"}};
```

Two-Dimensional Array Representation

- The two-dimensional array is represented as follows

Index	0	1	2
0 (Rose)	Red	Peach	Yellow
1 (Sunflower)	Yellow	White	Blue
2 (Daisy)	Green	Blue	Purple
3 (Dandelion)	White	White	White
4 (Violet)	Purple	Pink	Violet
5 (Lily)	Pink	Orange	White

Visualizing a Two-Dimensional Array

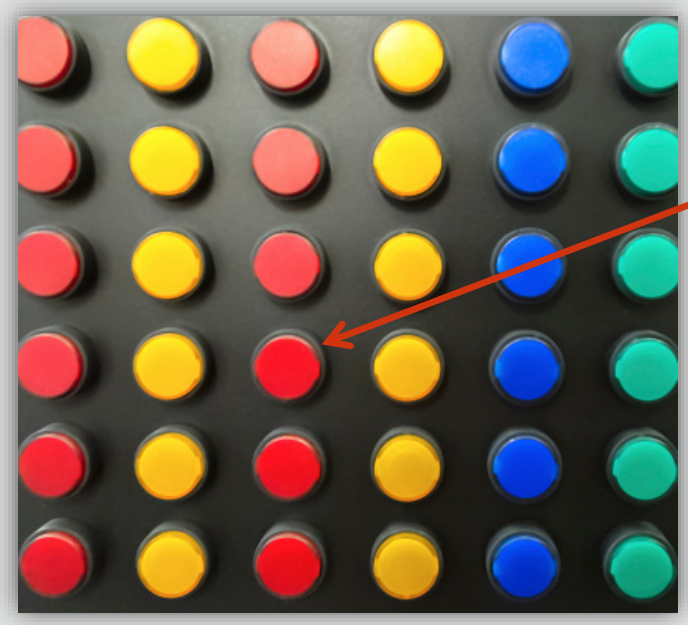
- Lining up the values during initialization helps to keep track of how data is organized in a two-dimensional array
- In the code examples below, notice how the second form is easier to visualize

```
String[][] myBouquet = {{"Red", "Peach", "Yellow"}, {"Yellow",
"White", "Blue"}, {"Green", "Blue", "Purple"}, {"White", "White",
"White"}, {"Purple", "Pink", "Violet"}, {"Pink", "Orange",
"White"}};
```

```
String[][] myBouquet = {{"Red", "Peach", "Yellow"},
                        {"Yellow", "White", "Blue"},
                        {"Green", "Blue", "Purple"},
                        {"White", "White", "White"},
                        {"Purple", "Pink", "Violet"},
                        {"Pink", "Orange", "White"}};
```

Accessing Data in a Two-Dimensional Array

- To access data in a two-dimensional array, you must know:
  - The index of the array to access
  - The index of the content in that array to access



Accessing Data in a Two-Dimensional Array Example

- For example, to access the color of the Rose in myBouquet, use the index of the Rose array (0) and the index of the first color (0)

```
//Previously initialized array
String[][] myBouquet = String[][] {
                       {"Red", "Peach", "Yellow"},
                       {"Yellow", "White", "Blue"},
                       {"Green", "Blue", "Purple"},
                       {"White", "White", "White"},
                       {"Purple", "Pink", "Violet"},
                       {"Pink", "Orange", "White"}
                       };
String rose1 = myBouquet[0][0]; //accesses first element
                                // of first array
```

Accessing Data in a Two-Dimensional Array Example

- Review the previous array initialization

```
//Previously initialized array
String[][] myBouquet = String[][] {
                {"Red", "Peach", "Yellow"},
                {"Yellow", "White", "Blue"},
                {"Green", "Blue", "Purple"},
                {"White", "White", "White"},
                {"Purple", "Pink", "Violet"},
                {"Pink", "Orange", "White"}};
```

- What will display in the console screen after the following code is executed?

```
System.out.println(myBouquet[0][1] + " Rose.";
System.out.println(myBouquet[5][2] + " Lilly.";
```

Accessing Data in a Two-Dimensional Array Example Solution

- What will display in the console screen after the following code is executed?

```
System.out.println(myBouquet[0][1] + " Rose.";
System.out.println(myBouquet[5][2] + " Lilly.";
```

- Solution:
  - Peach Rose
  - White Lilly

Accessing the Length of Two-Dimensional Arrays

- Length is an instance variable defined by the size of each declared array
- Two-dimensional arrays have two different lengths:
  - Length of the outer array
  - Length of the internal arrays

Length of Outer and Inner Arrays

- The length of the outer array, which is the length that describes the number of arrays contained (rows), is accessed like a typical array

```
int numArrays = arrayName.length;
```

- The length of the internal arrays, which is the length that describes the number of elements each array contains (columns), is accessed using the following notation
- Brackets [ ] tell the program that it is accessing the length of the internal arrays, and row says which array

```
int numElementsInEach = arrayName[row].length;
```

Length of Outer and Inner Arrays Example

- Review this code

```
String[] one = new String[7];
int[][] two = new int[5][3];
double[][] three = new double[3][2];
People[] four = new People[5];
```

- Which of the following is not true?
- Can you identify which one has improper syntax for accessing the length?

```
one.length == 7;
two.length == 3;
three.length == 3;
two[0].length == 3;
three[0].length == 2;
four[0].length == 5;
```

Arrays Practice Exercises

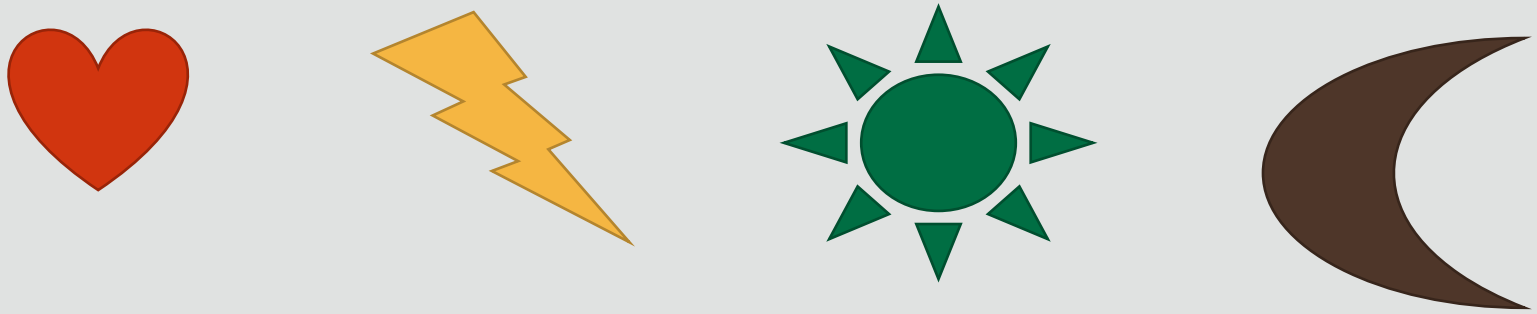
- Complete the following exercises from Section 10 of the Learning Path for this course:
  - **ArraysPractice1.pdf**
  - **ArraysPractice2.pdf**
  - **ArraysPractice3.pdf**
  - **ArraysPractice4.pdf**
  - **MatrixMath.pdf**

Length of Outer and Inner Arrays Example Solution

- Solution:
  - `two.length == 3;` **is FALSE**
  - `four[0].length == 5;` **is not valid syntax**

Two-Dimensional Arrays Object Types

- Just like single-dimensional arrays, two-dimensional arrays can store any type of object



Two-Dimensional Arrays Object Types - Example

- How can we organize the students in a classroom into three groups, with five students per group as an array?
- Answer:
  - Declare a two-dimensional array that holds three arrays, one for each group

Two-Dimensional Arrays Object Types - Example

- Each array stores five students
- Define a class called `Student`
- The class variable name can be used to create an instance of Student as shown here:

```
public class Student {
    String name;
    public Student(String n){
        name = n;
    }
    public String toString(){
        return name;
    } //override the Object method toString()
}
```

Two-Dimensional Arrays Object Types - Example

- Create a main method tester that will declare and initialize the array that stores the group of students:

```
Student[][] groups = new Student[3][5];
groups[0][0] = new Student("Anna");
groups[0][1] = new Student("Bob");
groups[0][2] = new Student("Carrie");
groups[0][3] = new Student("Don");
groups[0][4] = new Student("Emily");
groups[1][0] = new Student("Fred");
groups[1][1] = new Student("Gary");
groups[1][2] = new Student("Hazel");
groups[1][3] = new Student("Inez");
groups[1][4] = new Student("John");
groups[2][0] = new Student("Kim");
groups[2][1] = new Student("Linda");
groups[2][2] = new Student("Mary");
groups[2][3] = new Student("Nancy");
groups[2][4] = new Student("Opal");
```

Two-Dimensional Arrays Object Types - Example

- Add the code that will output the names of each `Student` using the overridden `toString()` method from the `Student` class and a **nested `for` loop:**

```
System.out.println("Your Students:");
      for(int i = 0; i < groups.length; i++){
          System.out.print("Group "+ i + ": ");
        for(int j = 0; j < groups[i].length; j++){
          System.out.print(groups[i][j] + " ");
        }
        System.out.println();
      }
```

Nested for Loops

- A nested for loop:
  - Is a for loop inside a for loop
  - Is can be used to iterate through two-dimensional arrays using the outer for loop counter as the index for the arrays (rows) and the inner for loop counter as the index for the elements of each array (columns)
- Consider the following two-dimensional array declaration:

```
Student[][] groups = new Student[3][5];
```

Nested for Loops

- To iterate through the array and initialize each student as "Temp", use nested for loops as shown below

```
//i keeps track of the rows
for(int i = 0; i < groups.length; i++){
   //j keeps track of the columns
   for(int j = 0; j < groups[i].length; j++){
      groups[i][j] = new Student("Temp");
   }//end for
}//end for
```

Nested for Loops

- We can iterate through the array as shown in the previous slide (in row order)

OR

- As shown below (in column order)

```
//i keeps track of the rows
for(int i = 0; i < groups.length; i++){
   //j keeps track of the columns
   for(int j = 0; j < groups[i].length; j++){
      groups[i][j] = new Student("Temp");
   }//end for
}//end for
```

Complete Tasks Using Nested for Loops

- How could we complete all three tasks?
  - Iterate through the existing array
  - Create a new student for each provided name
  - Place each student into one of the three groups

Complete Tasks Using Nested for Loops

```
String[] studentNames;
//Assume studentNames is initialized with 15 names

Student[][] groups = new Student[3][5];//Declare your array
   int x = 0;
   for(int i = 0; i < groups.length; i++){
      for(int j = 0; j < groups[i].length; j++){
         String name = studentNames[x];
         groups[i][j] = new Student(name);
         x++;
      }//End inner for loop
  }//End outer for loop
```

Arrays Exercise – Find the Mode

- Complete the **FindMode** Project from Section 10 of the Learning Path for the course

Command Line Arguments

- An array has always been a part of the main method
- The String array args is always a parameter for the main method

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

Command Line Arguments: Adding Extra Arguments

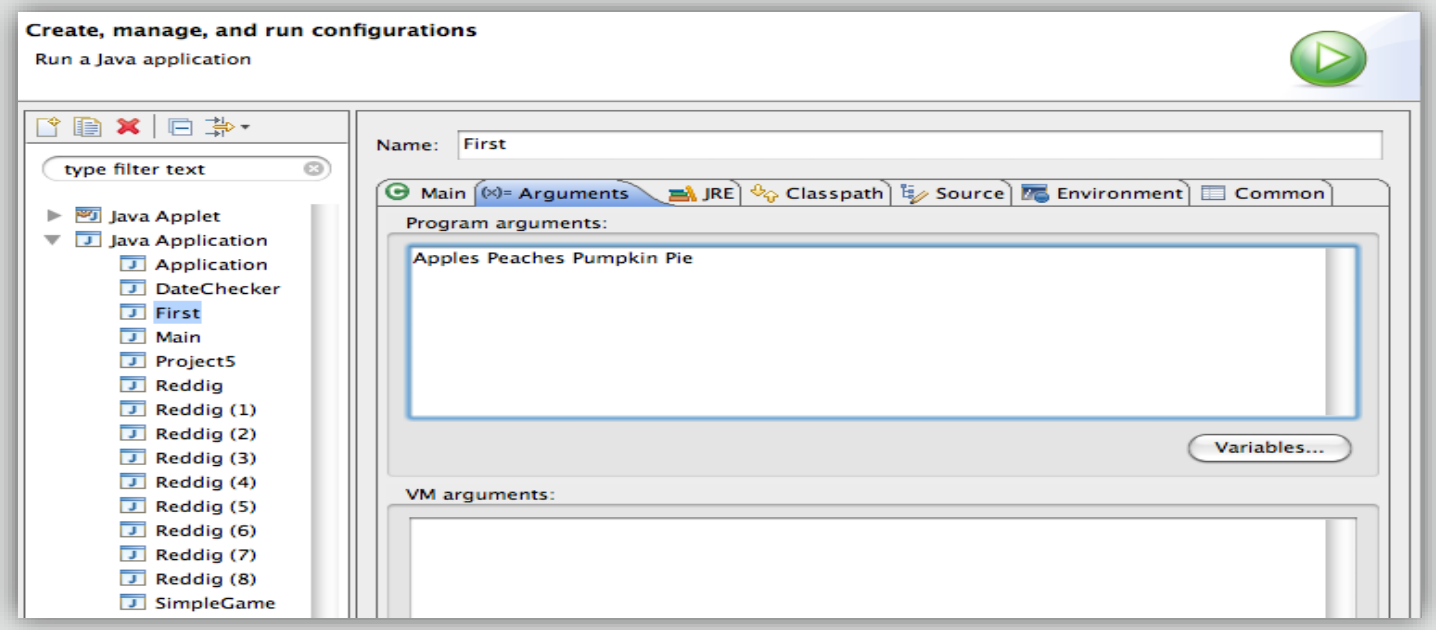
- If you run a Java program from a command line environment, you can type in extra arguments as follows:

```
java Test apples peaches pumpkin pie
```

  - The keyword java tells the command line environment to use the JVM to run the program Test
  - The array args gets populated with the Strings: apples, peaches, pumpkin, and pie
  - Using Eclipse, we avoid the command line environment However, we can use command line arguments in Eclipse

Using Command Line Arguments in Eclipse

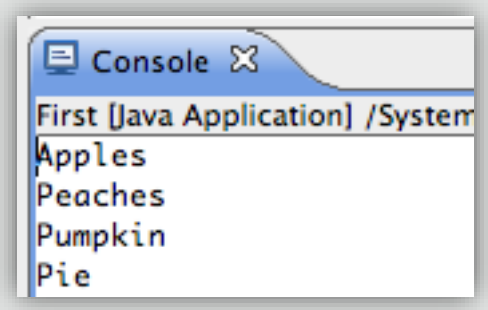
- Access the arguments tab for your program by going to the Run menu and choosing Open Run Dialog
- Click the Arguments tab and type in the Strings: Apples Peaches Pumpkin Pie



Using Command Line Arguments in Eclipse

- Click Run for the program
- All of the Strings that were typed in the Arguments tab will print to the console

```
public class TestArgs {
   public static void main(String[] args){
      for(int i=0;i<args.length;i++){
         System.out.println(args[i]);
      }//end for
   }//end main
}//end class TestArgs
```



Terminology

- Key terms used in this lesson included:
  - Algorithm
  - Array
  - Array of arrays
  - Command-line argument
  - Index
  - Iterate
  - Nested for loop
  - Single-dimensional array
  - Traverse
  - Two-dimensional array

Summary

- In this lesson, you should have learned how to:
  - Write a single-dimensional array in a Java program using primitive data types
  - Write a single-dimensional array in a Java program using reference (Object) types
  - Write a 2-dimensional array in a Java program using primitive data types
  - Write a 2-dimensional array in a Java program using reference (Object) types
  - Declare an array, initialize an array, and traverse the array

Summary

- In this lesson, you should have learned how to:
  - Describe array initialization
  - Distinguish between the String method length() and an array's length value
  - Rewrite a Java program to store integers into an array, perform a mathematical calculation, and display the result
  - Use alternative array declaration syntax


ORACLE
Academy