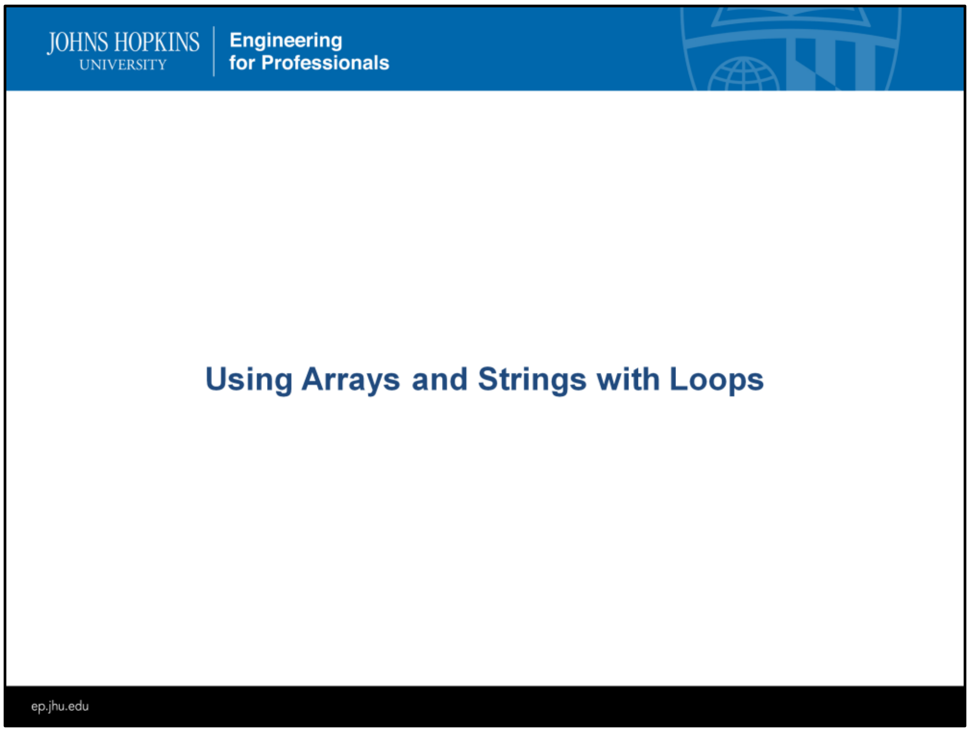
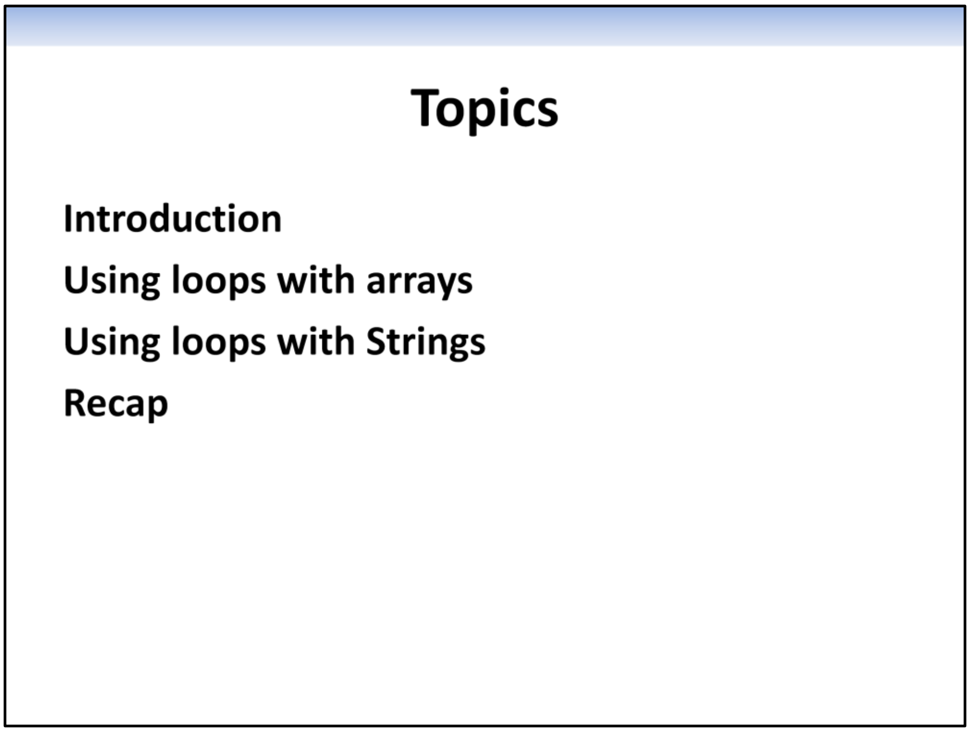

JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

Using Arrays and Strings with Loops

ep.jhu.edu

This is a short presentation about using arrays and Strings with loops.

Lets get started with discussing the major topics to be presented.



Topics

Introduction

Using loops with arrays

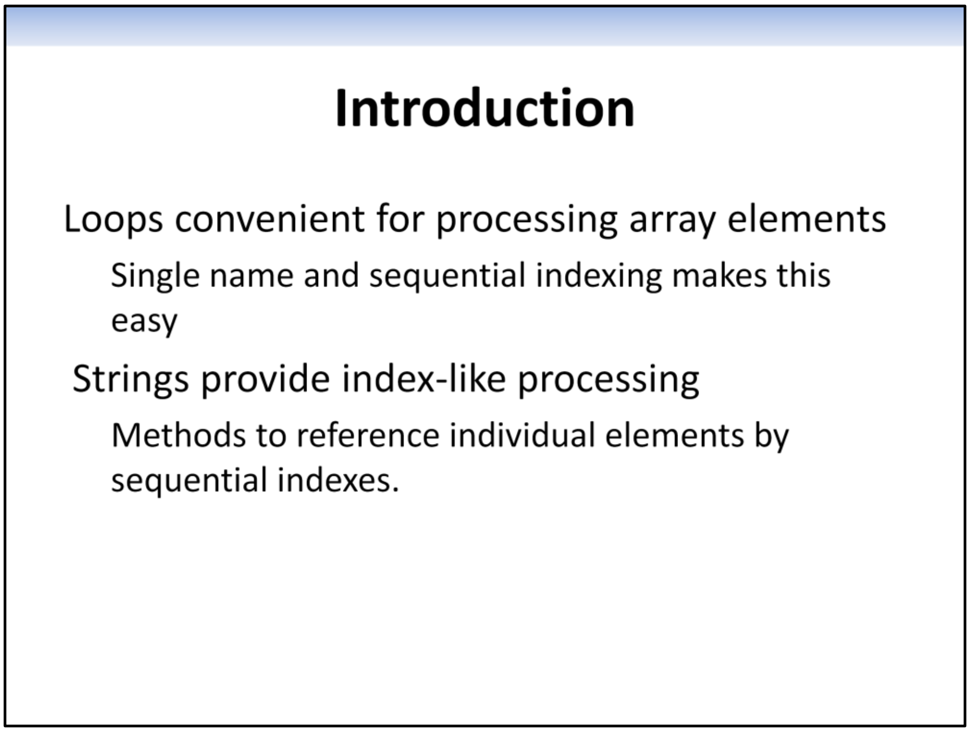
Using loops with Strings

Recap

The major topics of this presentation are

- An introduction on using arrays and Strings with loops.
- Using loops with arrays
- Using loops with Strings
- And ending with a recap of the major points of this presentation.

Lets start with the introduction on using loops with arrays and Strings.



Introduction

Loops convenient for processing array elements

- Single name and sequential indexing makes this easy

Strings provide index-like processing

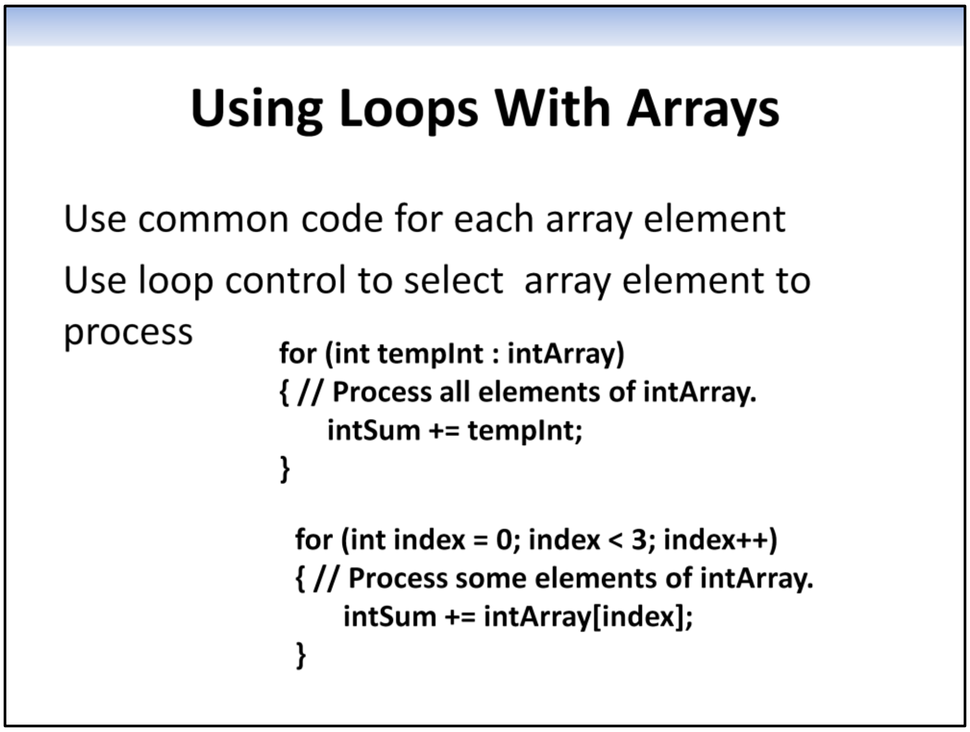
- Methods to reference individual elements by sequential indexes.

Arrays and loops and to a lesser extent Strings and loops are natural teams for processing similar data. This is because arrays (and the characters in Strings) are collections of identical typed elements and loops are designed to process multiple instances of similar logic.

For example, if you have an array of integers and you want to do similar processing on each integer element, loops provide a convenient way to provide this processing on each element of the array, no matter how many elements there are in the array.

Strings also provide way to process its collection of characters in a sequential way using indexes manipulated by loop constructs.

Lets look in a bit more detail about using loops with arrays.



Using Loops With Arrays

Use common code for each array element

Use loop control to select array element to process

```
for (int tempInt : intArray)
{ // Process all elements of intArray.
     intSum += tempInt;
}

  for (int index = 0; index < 3; index++)
  { // Process some elements of intArray.
       intSum += intArray[index];
  }
```

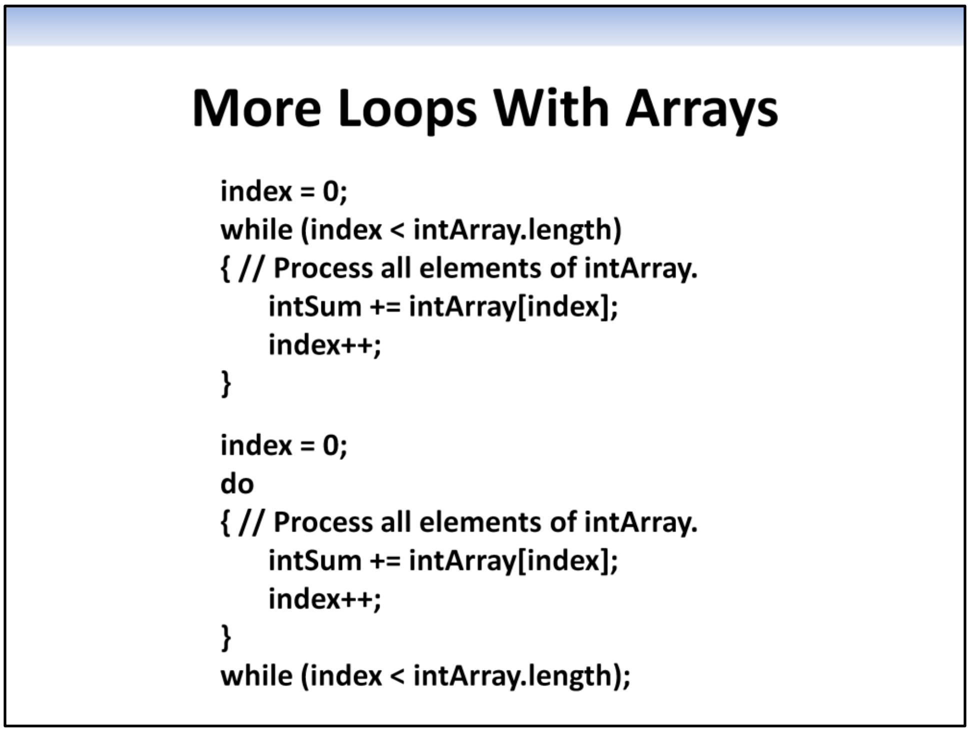
As already mentioned, loops are a natural working with arrays since they can provide similar processing for each individual element of an array in a compact format. It is also very flexible since you can easily very the number of array elements process by manipulating the loop controlling variables.

You use the loop control to select the array element to process in the current iteration. This is usually done by having an index calculated by the loop controls and used to determine which array element to process in the current iteration.

Lets look at some examples.

Here we are going to process the entire intArray using the short-cut for loop. At each iteration, the next element of the intArray is loaded into the tempInt variable and its value is added to the accumulator intSum. It does not matter if the intArray has zero elements or a million elements, the code stays the same. Nice.

The next example uses the traditional for loop controls but the general idea is the same as the previous example. In this case, we are only processing the first 3 elements of the intArray. To process all the elements of intArray in this matter, we could replace the 3 in the loop heading with a **intArray.length** which would allow indexes to increment up to the last valid index. We will see this technique soon.



More Loops With Arrays

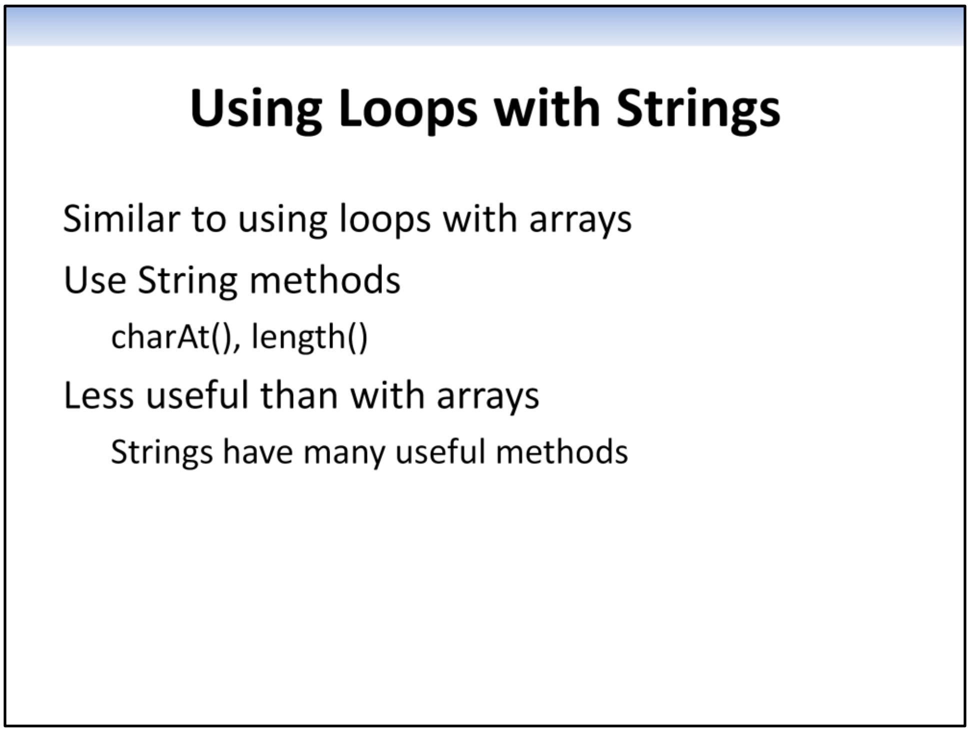
```
index = 0;
while (index < intArray.length)
{ // Process all elements of intArray.
     intSum += intArray[index];
     index++;
}

index = 0;
do
{ // Process all elements of intArray.
     intSum += intArray[index];
     index++;
}
while (index < intArray.length);
```

Lets look at a couple more examples of loops and arrays.

Here we are processing the entire intArray but using a while instead of a for loop. The idea is the same, by controlling the loop variables this code will process integer loops of any size and the code stays the same.

The example here uses a do-while loop to process the entire intArray array. Note for this to work the zero element of the intArray must exist since , as with all do-while loops, the loop control is not encountered until after the loop is executed once.

Lets move on to discussing loops with Strings.



Using Loops with Strings

Similar to using loops with arrays

Use String methods

- charAt(), length()

Less useful than with arrays

- Strings have many useful methods

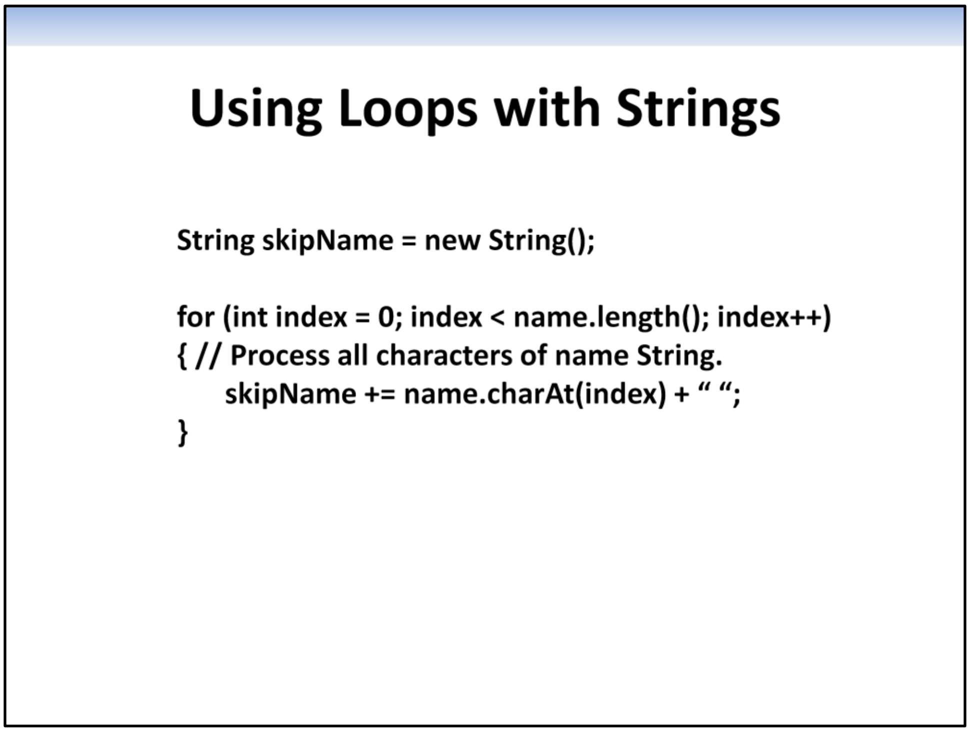
Using Strings with loops is similar to using arrays with loops. The difference is with Strings you use methods to use specific character components rather than indexes directly.

Two String methods which are useful in using Strings with arrays are **charAt()** and **length()**. **charAt()** is used to get at individual characters of a String with the parameter being the index of the character desired. This works similar to directly applying an index for example with integer arrays.

The **length()** method returns the number of characters currently contained in the String object. This works the same as the length property of arrays, it returns the number of character elements in the target String.

Loops are less useful with Strings then arrays since Strings have many useful methods to manipulate parts of strings, negating much of the need to use loops. Several of these methods are discussed in the separate presentation on String methods.

Lets look at an example of using Strings with loops.



Using Loops with Strings

```
String skipName = new String();

for (int index = 0; index < name.length(); index++)
{ // Process all characters of name String.
     skipName += name.charAt(index) + " ";
}
```

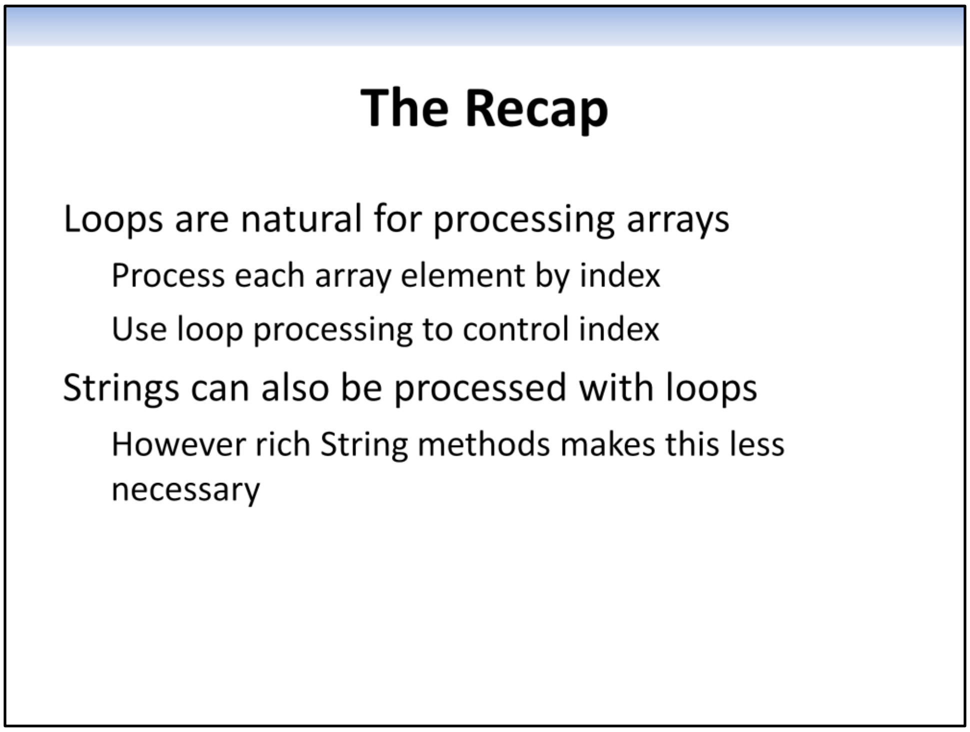
Here is an example of using Strings with loops.

This example is similar to the earlier arrays with loops examples. The loop control determines how many character elements of the String are to be process and the code is the same no matter how many characters are contained in the string.

The loop control is the same as earlier examples except the String method **length()** is used to stop processing at the end of the String.

The **charAt()** method is used to retrieve the next character to be processed. The index value used in this method is similar to directly using an index in the loop with array examples.

Time to wrap this presentation up.



The Recap

Loops are natural for processing arrays

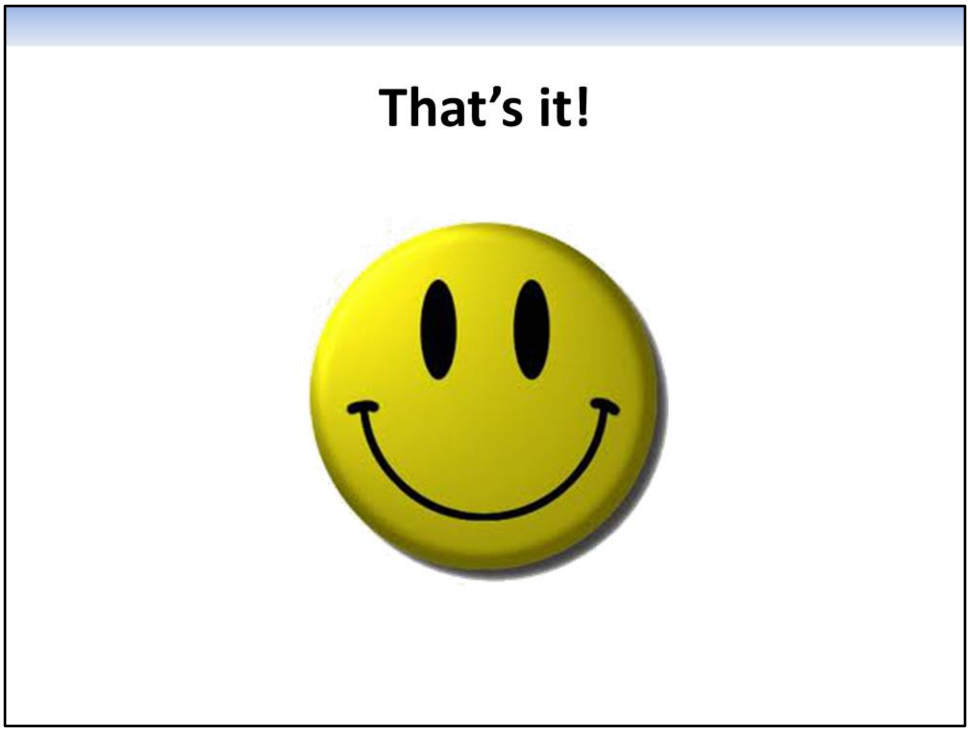
- Process each array element by index
- Use loop processing to control index

Strings can also be processed with loops

- However rich String methods makes this less necessary

OK here is the recap of this presentation.

- Loops are natural for processing arrays
    - Use index values to select array element to be processed
    - Use array loop controls to determine the next element to process
- Strings can also be processed with loops
    - However rich String methods makes using loops much less necessary than with other types of arrays.



That’s it!

And that’s it!