

Strings and Arrays

The objectives of this chapter are:

- ▣ To discuss the String class and some of its methods
- ▣ To discuss the creation and use of Arrays

The String Class

- ❑ Although we haven't yet discussed classes and object, we will discuss the String class.
- ❑ String objects are handled specially by the compiler.
 - ❑ String is the only class which has "implicit" instantiation.
- ❑ The String class is defined in the java.lang package.
- ❑ Strings are immutable. The value of a String object can never be changed.
 - ❑ For mutable Strings, use the StringBuffer class.

Creating String Objects

- Normally, objects in Java are created with the *new* keyword.

```
String name;  
    name = new String("Craig");
```

- However, String objects can be created "implicitly":

```
String name;  
    name = "Craig";
```

- Strings can also be created using the + operator. The + operator, when applied to Strings means concatenation.

```
int age = 21;  
String message = "Craig wishes he was " + age + " years old";
```

Commonly used String methods

- ❑ The String class has many methods. The most commonly used are:
 - ❑ `length()` - returns the number of characters in the String
 - ❑ `charAt()` - returns the character at the specified index
 - ❑ `equals()` - returns true if two strings have equal contents
 - ❑ `compareTo()` - returns 0 if equal, Less than zero – if the invoking String is "less than" the other, Greater than zero - if the invoking String is "greater than" the other.
 - ❑ `indexOf()` - returns the index of specified String or character
 - ❑ `substring()` - returns a portion of the String's text
 - ❑ `toUpperCase()`, `toLowerCase()` - converts the String to upper or lower case characters

String Examples

```
String name = "Craig";  
String name2 = "Craig";  
  
if (name.equals(name2))  
    System.out.println("The names are the same");
```

```
String name = "Craig Schock";  
int lastNameIndex = name.indexOf("Schock");
```

```
String grade = "B+";  
double gpa = 0.0;  
  
if (grade.charAt(0) == 'B')  
    gpa = 3.0;  
  
if (grade.charAt(1) == '+')  
    gpa = gpa + 0.3;
```

Testing Strings for Equality

- Important note: The `==` operator cannot be used to test String objects for equality
 - Variables of type String are references to objects (ie. memory addresses)
 - Comparing two String objects using `==` actually compares their memory addresses. Two separate String objects may contain the equivalent text, but reside at different memory locations.
- Use the `equals` method to test for equality.

The StringBuffer Class

- ❑ StringBuffer objects are similar to String objects
 - ❑ Strings are immutable
 - ❑ StringBuffers are mutable
- ❑ The StringBuffer class defines methods for modifying the String value
 - ❑ insert()
 - ❑ append()
 - ❑ setLength()
- ❑ To clear a StringBuffer, set it's length to 0

```
StringBuffer nameBuffer = new StringBuffer("Joe");  
[...]  
nameBuffer.setLength(0); // clear StringBuffer
```

StringBuffer Example

```
StringBuffer sql = new StringBuffer();  
  
    sql.setLength(0);  
    sql.append("Select * from Employee");  
    sql.append(" where Employee_ID = " + employeeId);  
    sql.append(" and Employee_name = '" + employeeName + "'");
```


Arrays in Java

- ❑ Java supports arrays
- ❑ An array is a collection of elements where each element is the same type.
 - ❑ Element type can be primitive or Object
 - ❑ Each element is a single value
 - ❑ The length of the array is set when it is created. It cannot change.
- ❑ Individual array elements are accessed via an index.
 - ❑ Array index numbering starts at 0.
- ❑ ***Note: Some references claim that arrays in Java are Objects. THIS IS NOT TRUE.***
 - ❑ Arrays do exhibit some behaviour which is similar to objects, but they are not themselves, objects.

Creating Arrays

❑ Creating an array is a 2 step process

❑ It must be declared (declaration does not specify size)

declaration syntax:

`type[] arrayName;`

note the location of the []

❑ It must be created (ie. memory must be allocated for the array)

```
int[] grades;           // declaration

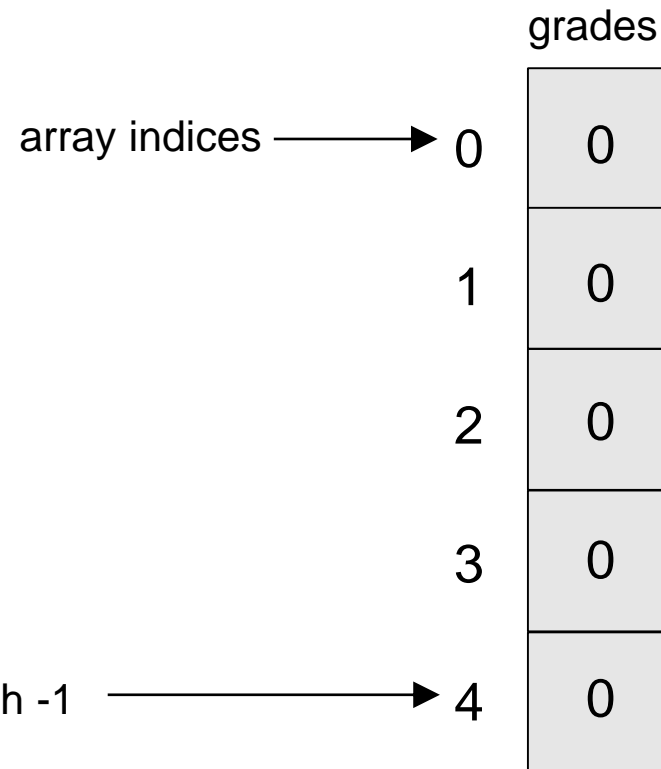
grades = new int[5];     // Create array.
                        // specify size
```

Creating Arrays

- When an array is created, all of its elements are automatically initialized
 - 0 for integral types
 - 0.0 for floating point types
 - false for boolean types
 - null for object types

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

Note: maximum array index is length -1



Initializing and Using Arrays

- ❑ Because array elements are initialized to 0, the array should be initialized with usable values before the array is used.
- ❑ This can be done with a loop
- ❑ Arrays have a length attribute which can be used for bounds checking
- ❑ Elements are accessed using an index and []

```
int[] sequence = new int[5];  
  
for (int i=0; i< sequence.length; i++)  
{  
    sequence[i] = i * 25;  
}
```

Array element being accessed. In this case, it is being assigned a value.

array length: ensures loop won't go past end of the array

Using initializer lists

- ❑ Another way of initializing lists is by using initializer lists.
- ❑ The array is automatically created
- ❑ The array size is computed from the number of items in the list.

```
type[] arrayName = {initializer_list};
```

```
int[] grades = {100, 96, 78, 86, 93};
```

```
String[] colours = { "Red", "Orange",  
                    "Yellow", "Green",  
                    "Blue", "Indigo",  
                    "Violet"};
```

Array Bounds Checking

- ❓ Whenever an array is accessed, the index is checked to ensure that it is within the bounds of the array.
- ❓ Attempts to access an array element outside the bounds of the array will cause an `ArrayIndexOutOfBoundsException` exception to be thrown.

```
int[] sequence = new int[5];  
  
sequence[0] = 50;           // ok  
sequence[1] = 60;           // ok  
sequence[-1] = 100;         //  
    Exception  
sequence[5] = 30;           //  
    Exception
```

The main() method

- ❑ You may recall that the main method takes an array of String objects as a parameter.
- ❑ This array of Strings holds the command line parameters which were passed to the java program when it was started

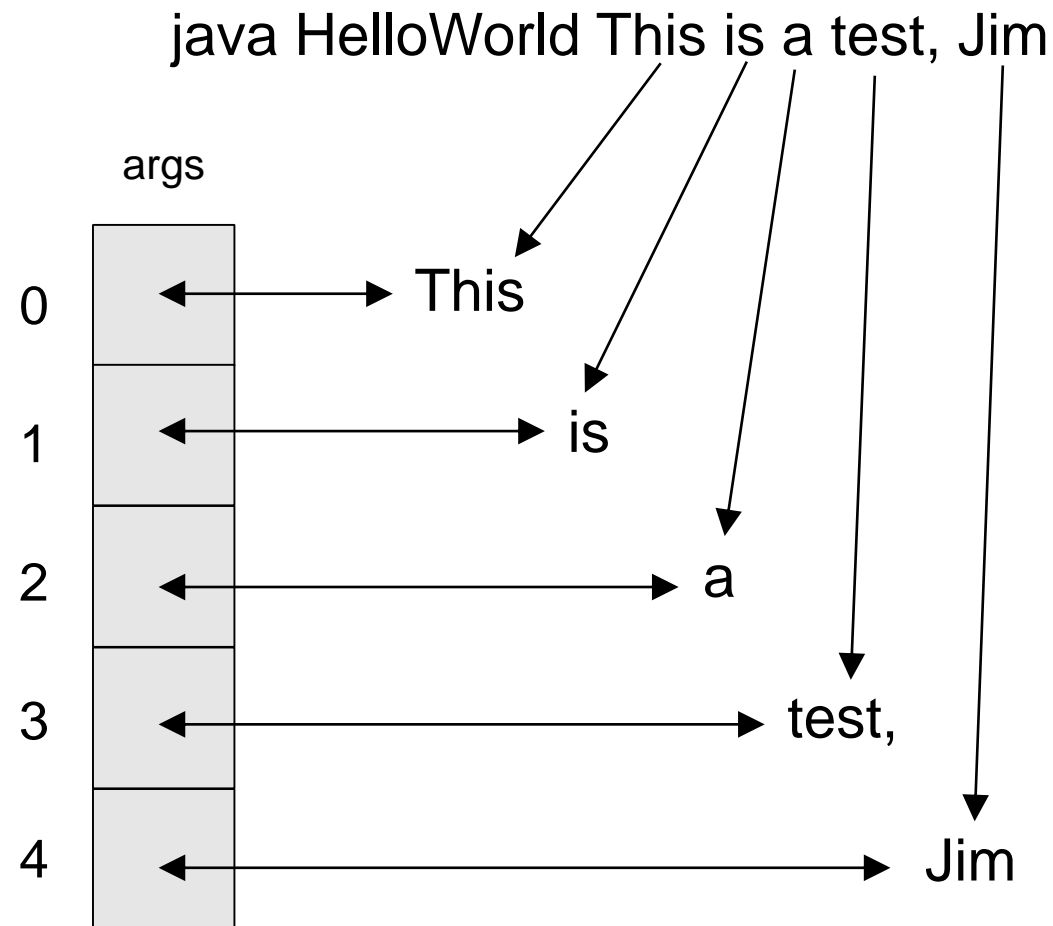
```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

Array holding command line parameters



Command line parameters

name of class containing the main() method



Multi-dimensional Arrays

- Arrays with multiple dimensions can also be created.

declaration syntax:

```
type[][] arrayName;
```

each [] indicates another dimension

- They are created and initialized in the same way as single dimensioned arrays.

```
int[][] grades = new int[20][5];  
for(int i = 0; i < 20; i++)  
    for(int j = 0; j < 5; j++)  
        grades[i][j] = 100;
```

```
String[][] colours = {{"Red", "Green", "Blue"},  
                      {"Cyan", "Magenta", "Yellow"},  
                      {"Russet", "Mauve", "Orange"}};
```

Review

- ❑ Is String a fundamental data type in Java?
- ❑ How is the String class treated specially?
- ❑ Name some commonly used methods of the String class and describe their function.
- ❑ What is a StringBuffer?
- ❑ What is an array?
- ❑ What are the steps needed to create and use an array?
- ❑ How are arrays initialized?
- ❑ How does bounds checking work in Java?
- ❑ What are the parameters to the method called "main"?