Virtual Lecture Notes (Part 1)

String Array Basics

Arrays can be declared in two statements or one, as illustrated below.

```
String [] names;
names = new String[8];
```

or

```
String [] names = new String[8];
```

Although these examples use a specific value to set the size of the array, a variable could just as easily have been used. The context of each program will indicate whether to use values or variables, and whether to declare and initialize an array with one statement or two. Before anything is assigned to an array, each index position is initialized with a place holder indicating it is empty. Type the following program into the 6.01 String Arrays project.

```
public class InitializingStringArray
{
    public static void main(String [] args)
    {
        String [] names;
        names = new String[10];

        for(int n = 0; n <= 9; n++)
        {
            System.out.println("index position " + n + " = " + names[n]);
        }
    }
}
```

Run the program and observe the output. Answer the following questions and make the suggested modifications. Some changes may cause errors, so restore statements to their original form before moving on to another question.

- What value does each index position contain before information is assigned to it?
- What happens if you change 10 to 9 in the array declaration? (Change it back afterwards.)
- What happens if you change <= to just < ? (Change it back afterwards.)
- What happens if you change 0 to 1 and 9 to 10?
- What statements would you change to increase the size of the array to 100 and print the value of each position?
- What could you change to print only the odd numbered index positions?
- What is the meaning of an IndexOutOfBoundsException and how can it be fixed?

Once an array is declared and initialized, there are four ways to assign information to each index position, three of which you already know from writing assignment statements in other programs.

Direct Assignment within the Source Code

Open the StringArrayDemo1 class and examine the source code. Values can be assigned to arrays by direct assignment within the program, as follows.

```
names[0] = "Sleepy";
names[1] = "Sneezy";
```

Run the program and carefully observe the output.

The index value within the square brackets indicates to which position within the array the String literal should be assigned. However, notice that the dwarfs occupy index positions ranging from 0--6, not 1--7. Therefore, “Sleepy” is assigned to index position 0 not 1, even though he is the first dwarf. This leads to a common error in arrays (and array lists) referred to as the “1 off” error.

Only seven positions (0--6) are used in the `dwarfNames` array, but it was initialized with eight positions, since 8 is the sentinel value that indicates when to terminate the `while` loop. These are design considerations you will be faced with every time you use an array.

Experiment with the program by changing the Strings assigned to the array.

- What happens if you try to assign an eighth or ninth dwarf?
- What happens if you remove the quotes from around the names?

Direct Assignment during Array Initialization

Open the StringArrayDemo2 class and examine the source code. Arrays can be initialized by direct assignment during array declaration as follows. (Due to space limitations, not all of the dwarfs could be shown, see actual program for correct statement.)

```
String [] names = {"Sleepy,...,”Bashful","Grumpy"};
```

With this technique each value is automatically assigned to the appropriate index position, but only *after* the array has been declared. Notice that the size of the array is established by the number of values listed between the curly braces.

Run the program and observe the output carefully. Experiment with the program.

- In lesson 1, you made a list of 10 things that are important to you. Modify this program to print your list, instead of the names of the Seven Dwarfs.

User Assignment from the Keyboard

Open the StringArrayDemo3 class and examine the source code. Users may also assign values to arrays, as illustrated in the following statement.

```
dwarfNames[n] = in.next();
```

As long as the index variable **`n`** has a value within the range of the array's length, the name entered by the user will be assigned to array index position **`n`**.

Run the program and carefully observe the output. Experiment with the program.

- What happens if you change the index variable name from **`n`** to **`x`**?
- What happens if you try to assign numbers (**`ints`** or **`doubles`**) to a **`String`** array?

Read Array Data from a File

Open the StringArrayDemo4 class and examine the source code. It is very common to read large amounts of data into an array from a text file, as shown in the following example.

```
dwarfNames[index] = inFile.next();
```

This is no different from the technique used to read information from a text file with simple variables. As long as the proper classes have been imported and the loop is correctly implemented, values can be very efficiently assigned to array positions.

Run the program and carefully observe the output. Experiment with the program.

- Add some additional names to the text file with a text editor. What happens when you run the program? Can you fix the problem?
- Determine the size of the array with the length property of arrays.

Wrap Up

Declaring, initializing, and assigning values to arrays are fundamental programming skills. Be sure that you understand how to do each before attempting to write any of the programs in this module. Array index positions are numbered starting at 0, which is often the source of errors. In addition, depending on the specific circumstances, it may be necessary to use < instead of <= to establish a loop's terminating condition.