**6.00: GETTING STARTED WITH ARRAYS**

* In programming, assigning a new value to a variable overwrites the old value in memory
  + EX:

String str = "make a doctor's appointment";  
System.out.println("The initial value of str is: " + str);  
str = "edit your college essay";  
str = "buy an exam review book for AP Computer Science A";  
str = "rent a movie";  
str = "clean up your room";  
System.out.println("The new value of str is: " + str);

* Lists can help to circumvent this in the real world. In programming they are called arrays

**6.01: ONE-DIMENSIONAL ARRAYS**

* The solution to only being able to store one value at a time in a variable is a **data structure** called an **array**.
  + **Data structure** – a collection of information that is referred to by one name such as arrays and array lists
  + **Array** – a data structure containing a single type of data which is accessed by index positions
* Java has a variety of different data structures for processing lists:
  + *Arrays*
  + **Linked lists** – a data structure that can hold an arbitrary number of objects in which each object points to the next link
  + **Trees** – a data structure hierarchy that grows from a single root node
  + **Array lists** – a data structure which stores objects and allows methods to add, delete, and retrieve indexed items
* Characteristics:
  + Arrays are structured, organized, and consecutively ordered.
  + Items can be added, deleted, modified, or reorganized.
  + Everything is in an easily identifiable place in memory.
  + All items in the list can be referred to by the same name.
  + Entries in one list can be reassigned to another list.

Strings in Arrays

* Arrays can be declared in two statements or one

**String [] names;**  
**names = new String[10];**

or

**String [] names = new String[10];**

* Values can be assigned to arrays by direct assignment within the program

**dwarfNames[0] = "Sleepy";**  
**dwarfNames[1] = "Sneezy";**

* Arrays can be initialized by direct assignment during array declaration

**String []dwarfNames= {"Sleepy, …, ”Bashful", "Grumpy"};**

* Users may also assign values to arrays

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

* It is very common to read large amounts of data into an array from a text file

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

Numeric Arrays

* The syntax for declaring and initializing numeric arrays is the same as for String arrays

**int [] intValues;**  
**intValues = new int[10];**

or

**int maxIndex = 100;**  
**double [] doubleValues = new double[maxIndex];**

**6.02: FORMATTING OUTPUT**

* Formatting output in Java has not always been easy, but Java solved this problem by adding a method commonly used in the C programming language to the PrintStream class: “**printf()**”
* The key feature of printf() is the inclusion of a formatting code as part of the method's argument

System.out.printf("City: **%-15s%n**", cityName);  
System.out.printf("Zip Code: **%10d%n**", zipCode);  
System.out.printf("pi = **%10.4f%n**",pi);

* These three examples indicate the typical syntax of a printf() statement:
  1. a String literal to be printed (e,g., "City: " , "Zip Code: ", and "pi: ")
  2. a format specifier (%-15s%n , %10d%n, %10.4f%n)
  3. a variable to be printed (e.g., cityName, zipCode, pi)
* A format specifier is a placeholder for a variable and consists of 4 parts
  1. The leading percent sign (%)
  2. An optional flag or specifier (+, -, ,)
  3. Field width (optional)
  4. A converter (s, d, or f)
* If only one variable will be printed on a line, the format specifier must include %n to force a linefeed

**06.03: THE FOR-EACH LOOP**

* There is a more effective version of the for loop that can be used with indexed data collections, like arrays
  + Specializes in iterating through data structures.
* The **for-each** loop

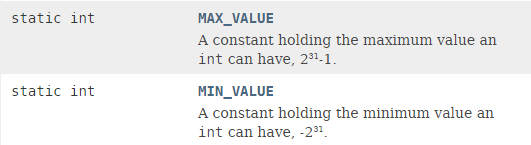
for(String person : names)  
{  
     System.out.println(person);  
}

* Index always starts at zero and increments by one
* All elements in the array will be processed.
* Just remember to read the colon (:) as "in" when you say the statement aloud.
  + The loop will traverse the names array from beginning to end and assign each element one-at-a-time

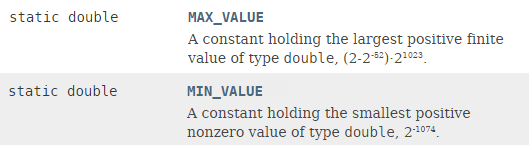
|  |  |
| --- | --- |
| Pros | Cons |
| No index variable is needed. | When not iterating over an array (or other "iterable" collection). |
| No condition needs to be stated. | When an index is needed for some purpose other than referencing the next element. |
| The index variable does not need to be incremented. | When an index is used to reference the elements of a collection in a particular order or a particular selection from among those elements. |
| It works on any data structure, regardless of the length or size. | When values are to be assigned to the elements of a collection. |
| An ArrayIndexOutOfBoundsException will never be thrown. |  |

**6.04: REVIEW**

* Learned so far:
* printing **String** literals
* writing arithmetic statements
* using variables
* creating menus and requesting input
* changing the flow of control with decision statements
* using relational and logical operators
* managing the flow of control with loop statements
* generating random numbers
* reading and writing text files
* using arrays
* formatting output



Smallest possible integer; constant



Smallest possible floating point number; constant