Introduction to Java 2.0:
Intermediate Programming Concepts

STEELS Standards

- 3.5.6-8.H
- 3.5.6-8.I
- 3.5.6-8.U
- 3.5.6-8.BB
- 3.5.6-8.CC
- 3.5.6-8.EE

Objectives

- Students will understand what arrays are and how to access elements within them
- Students will understand another form of control flow: for loops
- Students will apply the concepts together to create a more complex program that performs a given task

Materials

- Online Java Compiler

Basic Vocab

- **Arrays**
  - A data structure that stores a fixed-size sequence of elements of the same type.
- **Syntax**
  - The set of rules that defines the structure and format of code in a programming language. It specifies how symbols, keywords, and operators should be used to write valid code statements that the compiler or interpreter can understand.
- **Loops**
  - A programming construct that repeats a block of code multiple times.
- **Compiler**
  - A program that translates code written in a high-level programming language (like C++, Java, or Rust) into machine code that the computer's processor can execute directly.
- **Program**
  - A set of instructions written in a specific programming language that tells a computer how to perform a particular task or set of tasks.

Introduction

In this module, we'll continue expanding on our knowledge of Java by exploring TWO key concepts that are essential for creating more complex programs: arrays AND for loops. These tools will allow us to handle multiple pieces of data at once and execute repeated actions with control.

Arrays let us store and work with multiple values of the same type in a single variable.

For Loops give us a convenient way to repeat code a specified number of times.

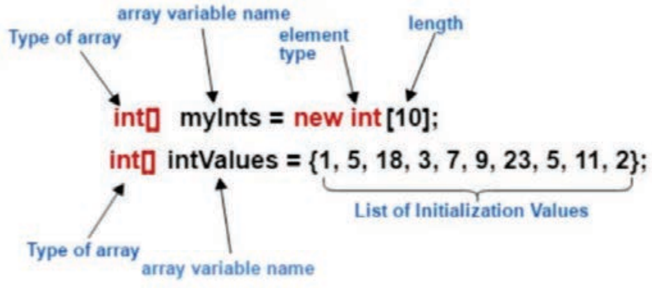
Ask the class how arrays and for loops may relate to the concepts learned in the first module. Notice that the arrays must hold values of the same variable type, and a for loop seems to have the same functionality as a while loop.

Arrays:

Arrays allow us to store a series of values of the same type in a single variable. Each element in an array can be accessed by its index, counting elements starting at 0.

Declaration:

Both of the arrays below hold ints. An array of strings would be declared String[] name = new String[10] myInts is a variable containing an empty array of size 10. intValues is an array of numbers (1, 5, 18, 3, 7, 9, 23, 5, 11, 2)



```java
int[] myInts = new int[10];
int[] intValues = {1, 5, 18, 3, 7, 9, 23, 5, 11, 2};
```

Declaration & Initialization of an Array in Java

Declaration & Initialization of an Array in Java

Each of the elements has its own unique index, counting from 0. For example, in the above example, the number 3 is held in index 3, and 7 in 4. To access the element in the array, we use the following syntax:" name[index]"

Try instantiating your own array of numbers, and print out different elements of it through the index. An example is shown below:

Printing out index two of the array results in 6.

```
int[] temp = {1, 4, 6, 2}; System.out.println(temp[2]);
```

You can also update values of an array. Simply specify which index you want modified and pass in the desired value with the syntax: "temp[index] = new value;". Using the same array from above, try changing a value!

Instead of 6, 10 is printed instead as we have changed the value prior to printing

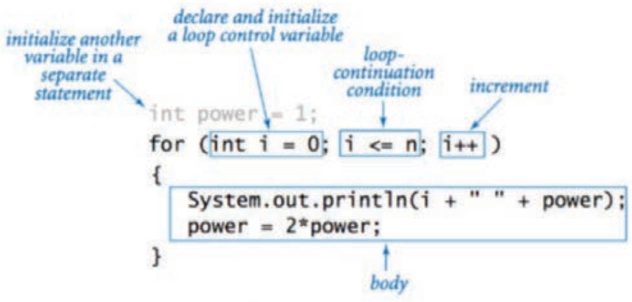
```
int[] temp = {1, 4, 6, 2}; temp[2] = 10; System.out.println(temp[2]);
```

For Loops:

A for loop helps us repeat actions for a specific number of times, and it's great when you know how many times you want to repeat something. Often used together with arrays, it allows us to traverse the entire array in sequence.

Declaration:

A new variable, in this case "i" is declared and initialized. In each iteration of the for loop, i is incremented by 1 as decreed by "i++". Note that this incrementation can be changed to any other statement, "i = i - 2" for example. For as long as the loop continuation condition holds true, the for loop will keep running with each new iteration having i at a different value.



```
int power = 1;
for (int i = 0; i <= n; i++)
{
   System.out.println(i + " " + power);
   power = 2*power;
}
```

initialize another variable in a separate statement
declare and initialize a loop control variable
loop-continuation condition
increment
body

Have the class attempt to use a for loop to print every element of their array from early. Think about when the for loop should end.

Hint: Arrays can't go out of bounds. You want it to loop through exactly every element of the array and not one more! You can do this manually if you know the length of the array, or use "name.length" , a built-in java method.

Solution:

```
int[] temp = {1, 2, 3, 4, 5, 6}; for (int i = 0; i < temp.length; i++){ System.out.println(temp[i]); }
```

Class Activity

Option 1:

Create an array of 15 grades using any values you desire. Calculate the average of the class! For an extra challenge, apply a curve by adding 10 to every, or just some grades before recalculating the average.

Solution

```
int[] grades = {85, 92, 78, 64, 89, 73, 95, 88, 76, 84, 91, 67, 80, 93, 70};
int total = 0;
for (int i = 0; i < grades.length; i++){
total += grades[i];
}
System.out.println("Average grade before curve:");
System.out.println(total / grades.length);

//curving the class by 5 points
for (int i = 0; i < grades.length; i++){
grades[i] = grades[i] + 5;
}

int curvedTotal = 0;
for (int i = 0; i < grades.length; i++){
curvedTotal += grades[i];
}

System.out.println("Average grade after curve:");
System.out.println(curvedTotal / grades.length);
```

We begin by initializing an array of realistic values for grades. Note that students may populate their arrays differently resulting in varying values for averages. There is no one right answer! We then also initialize the variable total to hold the summation of all the grades. Using a for loop, we loop through the entire array of grades and add them to our total variable. Finally, we simply print the average which we calculate by dividing the total sum by the length.

For the challenge activity, first we must loop through the array and modify the values of each element. We set each element to itself + 5 through each iteration, applying the curve. Then much like before, we loop through it again from the beginning and accumulate the total sum to find the average.

Option 2:

For the challenge activity, first we must loop through the array and modify the values of each element. We set each element to itself + 5 through each iteration, applying the curve. Then much like before, we loop through it again from the beginning and accumulate the total sum to find the average.

Option 2:

Find the minimum and maximum of an array.

Solution:

```
int[] grades = {85, 92, 78, 64, 89, 73, 95, 88, 76, 84, 91, 67, 80, 93, 70};
int min = 10000;
int max = 0;
for (int i = 0; i < grades.length; i++){
if (grades[i] < min){
min = grades[i];
} else if (grades[i] > max){
max = grades[i];
}
}
System.out.println("Min of array: ");
System.out.println(min);
System.out.println("Max of array: ");
System.out.println(max);
```

We begin by initializing an array of any values we desire. Next we assign an arbitrarily high value for minimum (Keep in mind this value must be larger than any value in the array you created. Think about why! Or you can simply set it equal to the first element of the array with int min = grades[0]. Talk about why this would work as well) We do a similar thing for max, assigning a value lower than any within the array, or simply the value of the first element.

Then we traverse the array using a for loop to access every element of the array. Using our if statements, we check if the current value is smaller than min. If it is, we update min. The else if branch then checks if it is larger than max, updating max if so. A common mistake made here is simply having an else arm. Without the conditional specified, max would not be updated correctly. Finally we can print and verify that our program works as desired!

Summary

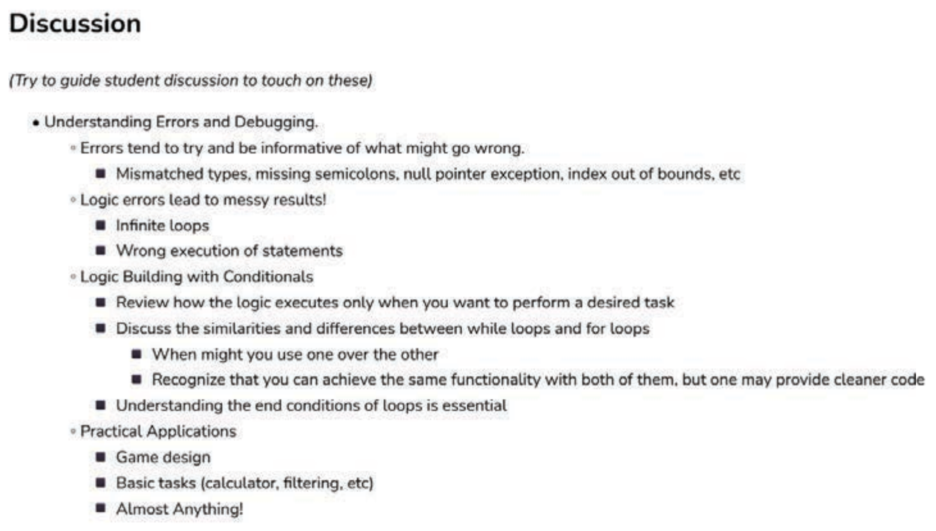
Bring the class back together and have them share their programs (as applicable based on classroom activity). Discuss the strategies they employed and the challenges they faced during the activity.

Have a few volunteers take a deeper dive into their program, walking their peers through the code line by line and verbally explaining their thought process as they coded their solution. Analyze the many ways to do the same task and weigh the pros and cons of each.

Discuss some errors they ran into and how they went about solving them. What were the most common ones? Did everyone solve them in the same way? How did they figure out the solution?

Discuss the versatility of programming languages such as java. If you have a problem or task, you can code a solution for it! This goes far beyond classroom assignments, many of the world's largest problems in a variety of fields can be addressed or supplemented with programming solutions!

In summary, familiarity with core programming concepts develop a strategic problem solving way of thinking that can be used to help automate and improve upon many small tasks, or play a role in solving important problems.



Discussion

(Try to guide student discussion to touch on these)

- Understanding Errors and Debugging.
  - Errors tend to try and be informative of what might go wrong.
    - Mismatched types, missing semicolons, null pointer exception, index out of bounds, etc
  - Logic errors lead to messy results!
    - Infinite loops
    - Wrong execution of statements
  - Logic Building with Conditionals
    - Review how the logic executes only when you want to perform a desired task
    - Discuss the similarities and differences between while loops and for loops
      - When might you use one over the other
      - Recognize that you can achieve the same functionality with both of them, but one may provide cleaner code
    - Understanding the end conditions of loops is essential
  - Practical Applications
    - Game design
    - Basic tasks (calculator, filtering, etc)
    - Almost Anything!