Introduction to Java:
Basic 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 variables are and how to declare them in Java
- Stu dents will understand core programming logic and how to implement them
- Students will apply the concepts together to create a more complex program that performs a given task

Materials

- Online Java Compiler

Basic Vocab

- **Variables**
  - Symbolic name or reference for a value stored in memory that can change over time as the program runs.
- **Conditional**
  - A statement in programming that lets the code make decisions.
- **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

Programming is like giving instructions to a computer so that it can perform tasks. In Java, we use different tools to write these instructions, such as variables, conditional statements, and loops. These are fundamental building blocks that will help you create programs to solve problems or automate tasks.

Begin by having your class open their online Java compilers. Everything today will take place within the “main method” which is what will be called when you run the program. Run the program by clicking the run button and observe the result on the right side. The print statement should have been executed and “Try programiz.pro” is printed.

Basic Variable Types:

- **Int**: Represents all integers
  Example: -2, -1, 0, 1, 2
- **Boolean**: Holds a True or False value
- **Char**: A single letter or character
  Example: ‘a’, ‘,’
- **String**: A combination of zero or more characters

Variable Declaration:

Variables in Java are declared with the following format:

Type name = value;

For example:

int x = 5;

This assigns the integer variable x with the value of 5.

Have your class create a new variable, assign it a value, and print that value out! The code should look something like this:

int myNumber = 10; // Declare and assign a value System.out.println(myNumber); // Print the value

Try setting a string to your int x and observe what happens. Java tells you they have “incompatible types” and won’t compile the code. You can either change the variable type or fix the value you are trying to assign. Play around with the variable types and get familiar with declaring them!

If Conditionals:

Execute a block of code IF something holds true. If conditionals are declared with the following syntax:

if (conditional)

If Conditionals:

Execute a block of code IF something holds true. If conditionals are declared with the following syntax:

if (conditional)

For example:

Because the value of x is 5, and 5 > 3 evaluates to true, the print statement inside is executed.

int x = 5; if (x > 3) { System.out.println("inside if"); }

While Loops:

Execute a block of code WHILE the conditional holds true. While loops are declared with the following syntax:

while (conditional)

For example:

Count is initialized with a value of 0. The while loop runs as long as count < 10. As a result, in each iteration of the loop, we print the current value of count and increment it by 1 until it is no longer less than 10

int x = 0; while (x < 10) { System.out.println(x); x = x+1; }

Class Activity

Option 1:

Have the class try to print out a triangle of stars with the height specified by you! A resulting triangle of height 4 should look something like this:

*

**

***

****

Solution:

```
int height = 4;
int row = 0;
int star = 0;
while (row <= height){
while (star < row){
System.out.print("*");
star = star + 1;
}
star = 0;
row = row + 1;
System.out.println();
}
```

Walking through the solution would be a great activity as well! You can demonstrate the logic of the while loop and how it terminates only once the conditional holds false. Tracking the values in each variable of each iteration is also very important when figuring out how the code works or debugging.

First we initialized three variables:

- height holds the unchanging height of the triangle we want to build, four in this example solution
  - Observe how changing its value is all that is necessary to change the triangle
  - Good variable usage should bring about varying results with minimal changes to the rest of the code!
- row holds the value representing which level of the triangle we are on
- star holds the value representing which star we are on within the same row

In our outer while loop, our conditional is row <= height. This allows the code within to execute while we are still on a row of the triangle we are trying to build. Now that we have a way to access each “row” of the triangle, the code within is what we want each row to do: print the right number of stars!

The key observation that allows us to solve the problem, is the row we are on is equivalent to the number of stars we want to print. As a result, this allows us to create our inner while loop conditional, setting it to star < row.Until we have printed enough stars to equal the row we are on, we want to keep printing them. Within the while loop, we simply have a print statement. Note that this one is missing the “ln” at the end of print, all this does is it doesn’t jump to the next line after printing, allowing us to print consecutive stars on the same line. Finally to avoid being stuck in an infinite while loop, we must change the variable(s) of the while loop. Each time we print a star, we add one to our star variable!

Now back to the outer while loop, after running the code we wanted to execute in each iteration (the inner while loop), we must update the variables we used and those of our conditional! First we reset star to 0, as for our next row, we would like to start from 0 stars again. Then we add 1 to row to move on to the next one. And finally, we execute a println to move our output to the next line, ready for the next row of stars!

Option 2:

Write a program that prints out the months corresponding to a number. For example, if x is set to 10, we should print out “October”.

This would be a good option to introduce error checking. How would a value of 100 be handled? How about a -5 or a string?

Solution:

```
int month = 5;
if (month == 1){
System.out.println("January");
} else if (month == 2){
System.out.println("February");
} else {
System.out.println("Invalid month input");
}
```

This activity serves as excellent practice for getting used to if-else statements! Essentially, you can check if the value of the month variable is equal to the numeric representation of each month of the year. For example, 1 would correlate to January. Using if else branches, you should run all the other months, beyond just February which is displayed here. And finally, the else

System.out.println("Invalid month input");
}

This activity serves as excellent practice for getting used to if-else statements! Essentially, you can check if the value of the month variable is equal to the numeric representation of each month of the year. For example, 1 would correlate to January. Using if else branches, you should run all the other months, beyond just February which is displayed here. And finally, the else at the very end acts as error checking. We know if the code doesn't enter any of the if/else if statements, it isn't a valid month input, and so we print an appropriate error message

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
    - If and while loops are the core of all logic
      - There are other ways to simplify their code but the logic behind it remains the same
    - Understanding the end conditions of loops is essential
  - Practical Applications
    - Game design
    - Basic tasks (calculator, filtering, etc)
    - Almost Anything!