Sign-In Here

Jr. BME Program

JAVA PROGRAMMING

Dean Zepeda

Agenda

01

Intro!

- Who I am
- What I do

02

Why Java?

- APCSA
- Data Structures
- Translate to other languages

03

Objectives

- Syntax & Structure
- Functions & Commands
- Multi-function Code
- CHALLENGE!

Java Syntax

- Very Structured
  - Variable types must be defined
    - Int, string, double, etc...
  - Uses a compiler
  - Whitespace does NOT matter
    - Denotation with brackets etc...
  - Need a SEMICOLON (;) after each line
  - Sub-functions are denoted by periods
- Can get “rambly”
- Similar to C (carries over in languages)

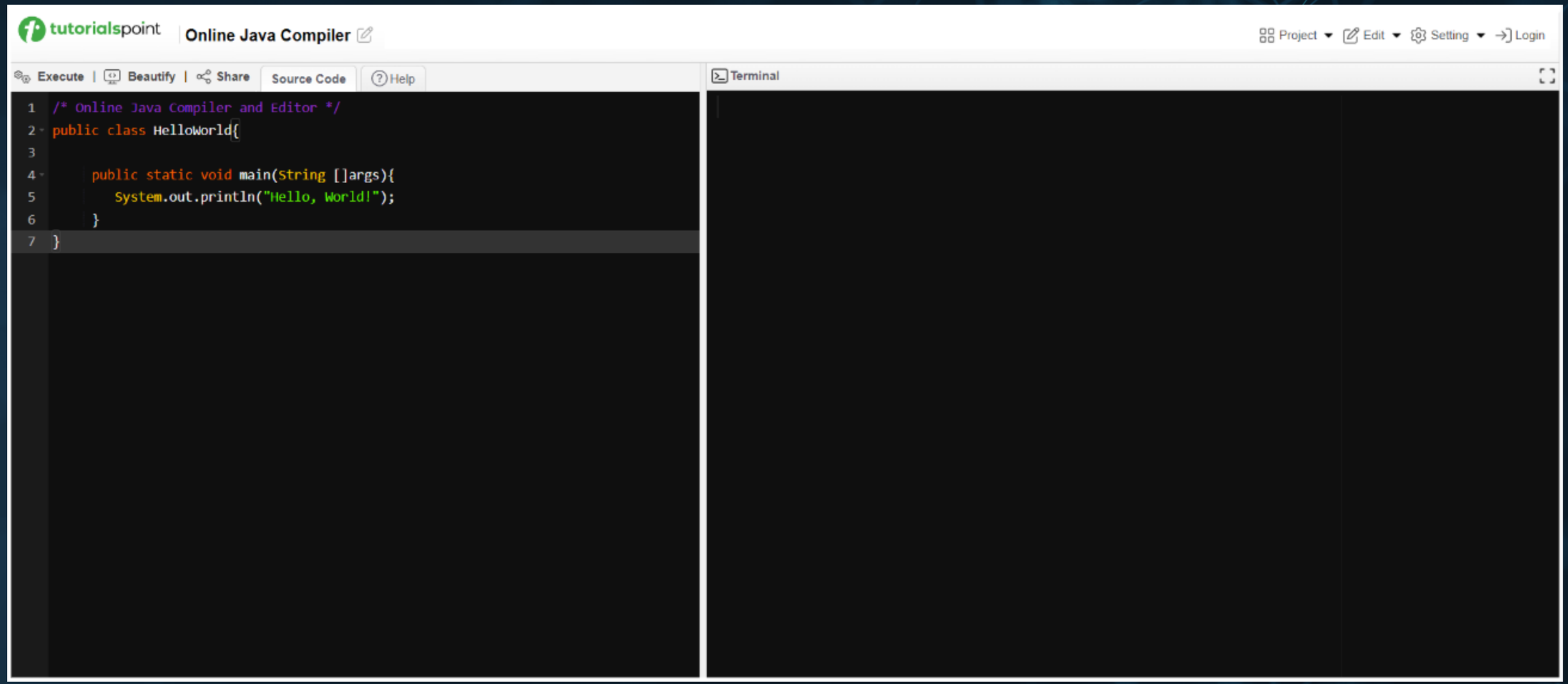
About Our Project

- Go to **tutorialspoint.com/compile_java_online.php**
- Alternative Options
  - Google: “tutorialspoint coding ground” → search “java” → click “online java compiler”
  - https://onecompiler.com/java

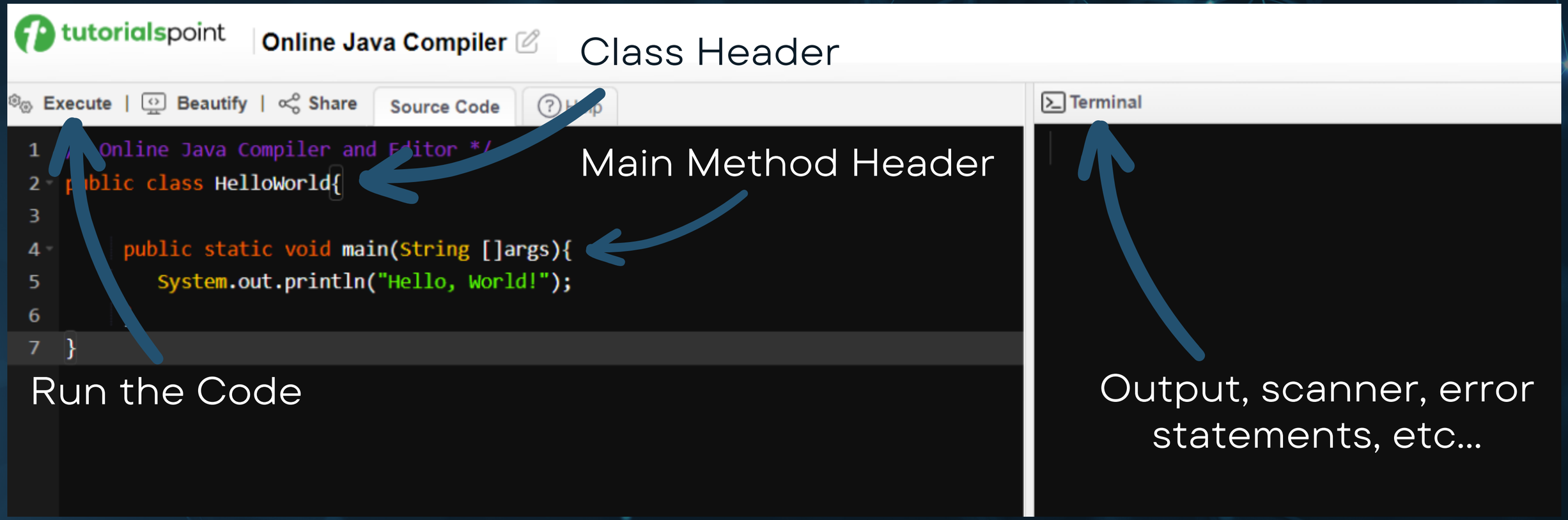
This is called an IDE, integrative development environment. Since it is online, it is more limited but can run simple programs and test code functions!

Open the Online IDE

It should look like this!



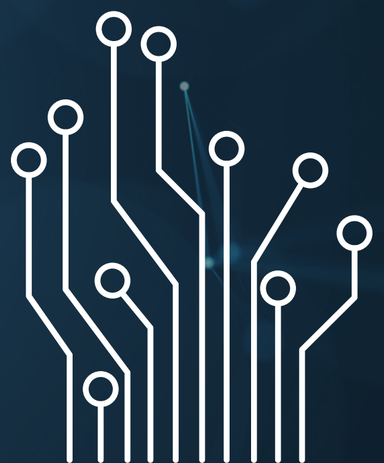
What am I looking at?



How to run Code

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

- Class Heading: **public class NAME {}**
- Within this class, create the main function that the code will run from
  - **public static void main (String[] args) {}**
- When you press “Execute,” anything in the **main(String{} args)** brackets will run



Try it Out!

Some Basic Syntax

- Denote Comments with double slash
  - // This will start a comment
  - */ This is a multi-line comment */
- Printing to system terminal
  - **System.out.println(BLAH)**
    - Use apostrophes if BLAH is a string
    - Don’t use apostrophes if BLAH is a variable

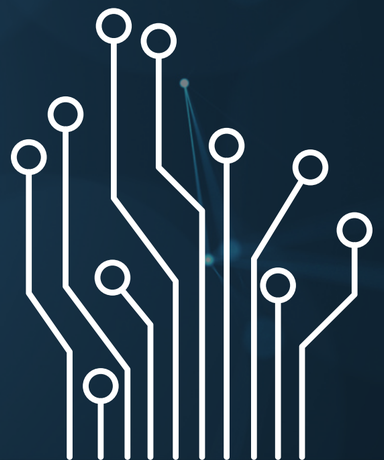
```java
public class HelloWorld{

    public static void main(String []args){
        System.out.println("blah blah"); //this is a comment
        /*
        this is a multi-line comment
        i am so cool cuz i can comment multiple lines
        unlike the double slash
        */
    }
}
```

Variables and Data Types

```java
/* Online Java Compiler and Editor */
public class HelloWorld{

    public static void main(String []args){
        int num1 = 69; //integer variable "num1" has value 69
        double num2 = 3.14; //double variable "num2" has value 3.14
        float num3 = 4.20; //float variable "num3" has value 4.20
        char char1 = 'E'; //character variable "char1" has value 'E'
        boolean isSmart = true; //boolean variable "isSmart" has value true
        String hello = "hello"; //string variable "hello" has value "hello"
        //note how string must be capitalized and the value is in quotation marks
    }
}
```



- A variable **stores information**
  - defined with a data type before they can be used (initialization):
- Primitive data types (predefined by Java)
  - int (integer)
  - double (64 bit decimal number)
  - float (32 bit decimal number)
  - char (character)
  - boolean (true/false)
- Non-primitive data types (call methods)
- String (words) *special case
  - **NAME = value;**
    - int num = 69;
- Think dishware and food

Using Variables

```java
public class HelloWorld{

    public static void main(String []args){
        int num1 = 69;
        System.out.println(num1); //prints out the value of num1, which is 69
        int num2 = 420;
        int ans1 = num1 + num2; //adds variables num1 and num2, assigns answer to variable ans
        System.out.println(ans1); //prints out the ans variable
        System.out.println(num1 + num2); //prints out the value of num1 + num2, but doesn't save
        double ans2 = num2 / num1; //divides num2 by num1, saves answer as double b/c decimal
        System.out.println(ans2);
        int ans3 = num2 / num1; //divides num2 by num1, forces answer to be an integer
        System.out.println(ans3);
    }
}
```

```
69
489
489
6.0
6
328509.0
```

- Once variables are initialized, they can be called on simply by using their name (no need to put the type beforehand)
- Variables can be printed: **System.out.println(VARIABLE);**
- Variables can be used to do operations like math. If you want to store data, the answers must be set to a variable
- Math operators:
  - Add: +
  - Subtract: -
  - Multiply: *
  - Divide: /
- Other math operations will need to import the math library (next couple slides)

Keep in Mind!

```java
import java.lang.Math.*;
public class HelloWorld{

    public static void main(String []args){
        int num1 = 69;
        System.out.println(num1);
        num1 = 420; //reassign num1 to 420
        System.out.println(num1);
        num1 = num1 / 6; //divide num1 by 6 and reassign num1
        System.out.println(num1);
    }
}
```

```
69
420
70
```

- No need to **re-initialize** variables
  - Just call on their names!
- You can assign new values to variables with the equal sign
  - If **int blah** has already been initialized to **4**, it can be reassigned to a different value: **blah = 69;**
  - Variables can call on themselves: **blah = blah + 1;**

YOU TRY!

- Practice **making initializing variables** of **different** data types and doing **math operations** with them
- Raise your hand for any errors, since there can be very specific number conversion errors and whatnot
  - A volunteer will come help!
- Main Goal:
  - Write code to crunch some numbers
  - Print the math results to the terminal

STRINGS

```java
import java.lang.Math.*;
public class HelloWorld{

    public static void main(String []args){
        String word1 = "BMES";
        String word2 = " is very ";
        System.out.println(word1+word2+"cool.");
    }
}
```

```
BMES is very cool.
```

- Strings are a non-primitive data type; they are technically another library, but already included into Java (**no need to import**)
- MUST INITIALIZE WITH A **CAPITAL**
- Strings can concatenate (add) with the +: System.out.println(STR1 + STR2); which can make printing different things very nice
- Strings can take in variables, or manually add with quotations

STRINGS CONT...

```java
import java.lang.Math.*;
public class HelloWorld{

    public static void main(String []args){
        String word1 = "BMES";
        String word2 = "UCI anteaters";
        int wordLength = word1.length(); //how long it is
        System.out.println(wordLength);
        boolean hasWord = word2.contains("ant"); //has word "ant"?
        System.out.println(hasWord);
    }
}
```

```
4
true
```

- Strings have their own methods (like the Math library) that can be called on from the library:
  - **STRINGNAME.METHOD();**
- Certain methods will return certain values
  - **.length()** returns an int of how long the string is
  - **.contains()** returns a true/false if the string contains a certain sequence of characters

Scanners

```java
import java.lang.Math.*;
import java.util.Scanner;
public class HelloWorld{

    public static void main(String []args){
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter something.");
        String userInput = scan.next();
        System.out.println("You typed: "+userInput);
    }
}
```

```
Enter something.
bruh
You typed: bruh
```

- The scanner allows us to take in user input with our code
- Must import (just like the Math library):
  - **import java.util.Scanner;**
- Initialize before using:
  - **Scanner NAME = new Scanner(System.in);**
- Calling **.next();** will take in the string entered into the terminal

Scanners + Math

```java
import java.lang.Math.*;
import java.util.Scanner;
public class HelloWorld{

    public static void main(String []args){
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a number.");
        double userInput = scan.nextDouble(); //look for next double input
        double ans1 = Math.sqrt(userInput); //square root function
        System.out.println("Square root of "+userInput+" equals "+ans1);
    }
}
```

```
Enter a number.
69
Square root of 69.0 equals 8.306623862918075
```

- Calling **.nextInt()** will take in the next integer, **.nextDouble()** the next double, etc
- Inputting anything else will result in an ERROR
- These can be used for math operations dictated by the user
- BE CAREFUL OF VARIABLE TYPES!!!!

YOU TRY!

- Using the scanner and math operations/methods, try and program an linear equation solver (y = mx + b). Make it so that you have to input the variables m, x, and b, and then the code spits out the value of y into the terminal.
  - **Hint:** Order of operations matters, use parentheses to help
  - Can use any data type, but using double helps account for decimals

- **Bonus:** Try and make the code give input prompts (i.e. “Please input the variable m”, etc.) and output nicely (i.e. “Your answer is: ”) using strings and concatenation

- **Challenge:** If you finish early, try doing it for different types of equations
  - Quadratic: ax^2 + bx + c = 0
  - 2x2 Determinant: determinant = ad - bc
  - Magnetic Force: F = q*v*B*sin(θ)

Answer!

```java
import java.lang.Math.*;
import java.util.Scanner;
public class HelloWorld{

    public static void main(String []args){
        Scanner scan = new Scanner(System.in); //initialize scanner
        System.out.println("Enter the slope (m):"); //prompt slope
        double mVal = scan.nextDouble(); //look for next double input to be slope
        System.out.println("Enter the x value:"); //prompt x value
        double xVal = scan.nextDouble(); //look for next double input to be x value
        System.out.println("Enter the y-intercept (b):"); //prompt y intercept
        double bVal = scan.nextDouble(); //look for next double input to be y intercept
        double yVal = (mVal * xVal) + bVal; //solve for y value
        System.out.println("The y value is: "+yVal); //print out answer
    }
}
```

```
Enter the slope (m):
69
Enter the x value:
21
Enter the y-intercept (b):
420
The y value is: 1869.0
```

BREAK TIME!

Conditionals

lots of words ahead

- Conditionals (if/else statements) are used to compare values and make decisions based off them
- This allows code to do certain things based on given values

- Example:
  - You eat a chili pepper
  - **If** it’s not to spicy → eat another
  - **Else if** it’s spicy but manageable → consume something to cool down
    - ***If*** you’re craving something sweet → eat ice cream
    - ***Else*** → drink milk
  - **Else** → get help

Conditionals Cont...

- Conditionals use booleans to make decisions (conditions have to be answered with a true or false)
- If true, do what's specified. Else, move on
- Operators:
  - For equivalency, use double equal signs: **==**
  - For size comparisons, use: **>, <, >=, <=**
  - AND operator: **&&**. OR operator: **|**, NOT operator: **!** or **!=**
- Use parentheses to specify comparisons
  - **(variable1 >= variable2) && (variable1 != 0)**
- Syntax:
  - **if (CONDITIONAL) {DO CODE} //main case**
  - **else if (CONDITIONAL) {DO CODE} //secondary case, can add as many more else if as needed**
  - **else {DO CODE} //note how this does not have a conditional, can only have 1 else per chain**
-

Conditionals Example

```java
import java.lang.Math.*;
import java.util.Scanner;
public class HelloWorld{
    public static void main(String []args){
        Scanner scan = new Scanner(System.in); //initialize scanner
        System.out.println("Enter an integer:"); //prompt question
        int userInput = scan.nextInt(); //looks for next integer
        if (userInput == 69) { //if the input was 69
            System.out.println("Super funny number.");
        }
        else if (userInput == 420 | userInput == 21) { //if input was 21 or 420
            System.out.println("Funny number.");
        }
        else { //any other number
            System.out.println("Not a funny number.");
        }
    }
}
```

Terminal

```
Enter an integer:
69
Super funny number.
```

Terminal

```
Enter an integer:
21
Funny number.
```

Terminal

```
Enter an integer:
2023
Not a funny number.
```

YOU TRY!

- Using the code from the **last activity** (the one where you programmed the equation solver for y = mx + b) and use conditionals to analyze the y value
- Have the code print out if the y value is positive, negative, or zero
  - **Hint:** use an if, else if, and else statement (though if/else if/else if can also work)
- If your y value was defined as a double, typecast it to an integer to make comparisons (comparing doubles can get wonky)
  - Typecasting forces your new number to become the new type
  - Do so by initializing a new variable to with the new type to the old variable with the parenthesis of the new type in front: **(NEWTYPE) OLDVARIABLE**
  - **yVal = 69.69;**
  - **Int newVal = (int) yVal;**

Answer!

```java
import java.lang.Math.*;
import java.util.Scanner;
public class HelloWorld{
    public static void main(String []args){
        //pretend the code for the y=mx+b is up here
        double yVal = 69.69; //pretend this was the answer
        int newY = (int) yVal; //typecast to integer for comparison
        if (newY > 0) { //if greater than 0, positive
            System.out.println("Positive value.");
        }
        else if (newY < 0) { //if less than 0, negative
            System.out.println("Negative value.");
        }
        else { //must be 0 by default, but can also use else if (newY == 0)
            System.out.println("Zero value.");
        }
    }
}
```

```
Positive value.
```

Methods

- Methods are essentially functions, where you can call them, plug in values, and get an output that can be used for computation
- Remember **.pow()** and **.contains()**? Those are all methods. The Math and String classes use the dots to call the methods they have access to
  - like folders within a cabinet drawer
- We can create our own methods within a program that are custom
- Methods can get more complex and specific when classes are involved, so we will just concern ourselves with methods in a single class

```
public class Main { //for methods within the static void main
    public static void main(String[] args) {
        //code here
    }
    public static TYPE NAME(PARAMETERS) {
        //code here
    }
}
```

SYNTAX

Methods Syntax cont...

- Methods can take in parameters (the inputs) within the parenthesis. Parameter variables are specific to the method (more on this later)
- Methods can return values (the outputs).
  - If the method spits out a value, it must return a variable that is the same type as the TYPE in the header
  - If it doesn't return anything, the type is "void"
- For technical reasons, methods must have the "static" in the header since they are within the same class as the **public static void main(String[] args)**
- Methods are called in the main code with their name followed by the parameters

```
public static TYPE NAME(PARAMETER) {
 //code here
 return VARIABLE;
}

public static void NAME(PARAMETER) {
 //code here
}
```

Method Example

```java
import java.lang.Math.*;
import java.util.Scanner;
public class HelloWorld{
    public static void main(String []args){
        //pretend the code for the y=mx+b is up here
        double yVal = 69.69; //pretend this was the answer
        int newY = (int) yVal; //typecast to integer for comparison
        if (newY > 0) { //if greater than 0, positive
            System.out.println("Positive value.");
        }
        else if (newY < 0) { //if less than 0, negative
            System.out.println("Negative value.");
        }
        else { //must be 0 by default, but can also use else if (newY == 0)
            System.out.println("Zero value.");
        }
    }
}
```

```
Positive value.
```

NOTE: Global Variables and Methods

- Global variables are variables that are initialized within the class, but outside the **public static void main(String[] args)**
- Global variables can be accessed by any method within the class
- Void methods are useful for when you want to manipulate these global variables, since they don't need to output anything
  - However, you can do this just by calling the variable in the main method or any other one
  - This has better applications in projects that utilize multiple classes and you don't want other classes to affect the global variables within the class (advanced)
- Static methods/variables belong to their respective classes; cannot be altered or accessed from outside the class

NOTE: Example

```java
import java.lang.Math.*;
import java.util.Scanner;
public class HelloWorld{
    static int global1 = 2; //these are global variables

    public static void main(String []args){
        System.out.println(global1); //prints global variable
        global1 = 3; //directly changes global variable
        System.out.println(global1);
        changeGlobal(5); //uses method to change global variable
        System.out.println(global1);
    }
    public static void changeGlobal(int newNum) {
        global1 = newNum; //sets global variable to input
    }
}
```

```
2
3
5
```

CHOICE

Challenge or Learn Something Else?

Challenge Questions

Easy Challenge

- Create a program that takes in user inputs of triangle side lengths and/or angles and outputs a bunch of cool triangle calculations
  - Area
  - Perimeter
  - Trigonometric values
  - You can find more cool triangle stuff by Googling “cool triangle equations”
- Use the Scanner class and methods to create your program

Hard Challenge

- Create a program that asks a user what type of equation they want to solve, then asks for the appropriate inputs based on which equation the user indicated
- The program should use conditionals to filter through the equation options, then call individual methods for each equation
- Use concatenation techniques to make things look nice in the terminal