MODULE 1: INTRODUCTION TO PROGRAMMING

# Command-Line Programs

TECH ELEVATOR
A Stride Company


Woohoo! It's Friday!

# Previously, on Tech Elevator: Java

**Variable** is a storage container paired with a symbolic name or identifier. A variable has **value** and **type.**

An **expression** is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

A **statement** forms a complete unit of execution.

# Previously, on Tech Elevator: Java

**Variable** is a storage container paired with a symbolic name or identifier. A variable has **value** and **type.**

An **expression** is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

A **statement** forms a complete unit of execution.

A **method** is a named block of code. It can take multiple values and **return back a single value**.

Access Modifier  Return Type  Descriptive Name  Parameters

# Previously, on Tech Elevator: Java

**Variable** is a storage container paired with a symbolic name or identifier. A variable has **value** and **type.**

An **expression** is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

A **statement** forms a complete unit of execution.

A **method** is a named block of code. It can take multiple values and **return back a single value**.

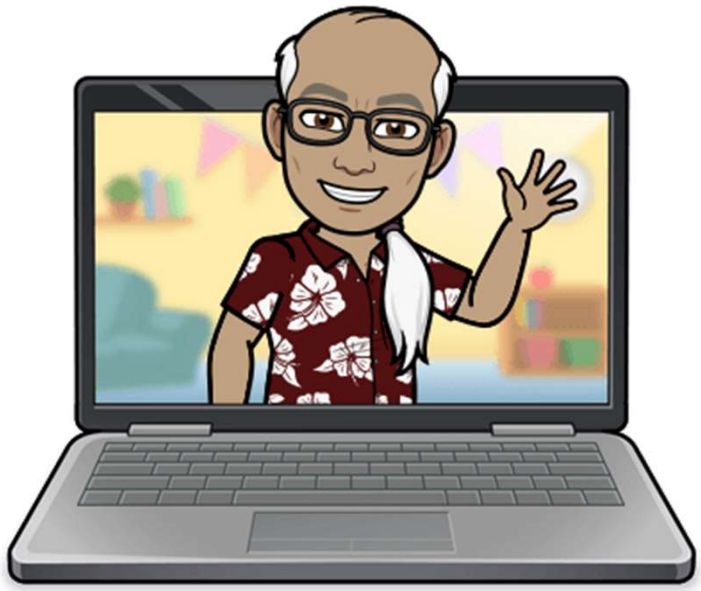
A **boolean expression** is an expression that produces a boolean value (true or false) when evaluated

# Previously, on Tech Elevator: Java

- An **if** statement executes **different code blocks** based on a **condition**

```
if (condition)
{
    <if code>
} else {
    <else code>
}
```

# LET’S CODE!



# Previously, on Tech Elevator: Java

- An **if** statement executes **different code blocks** based on a **condition**
- An **array** is a data structure that is a collection of variables of the **same type**

```
String[ ] animals = new String[6];
```

Element Type

Variable Name

Number of elements

# Previously, on Tech Elevator: Java

An **if** statement executes **different code blocks** based on a **condition**

An **array** is a data structure that is a collection of variables of the **same type**

A variable's **scope** defines where in the program that the variable exists (i.e. can be referenced). When code execution reaches a point where a variable is no longer referenceable, the variable is said to be *out of scope*.

# Previously, on Tech Elevator: Java

An **if** statement executes **different code blocks** based on a **condition**

An **array** is a data structure that is a collection of variables of the **same type**

A variable's **scope** defines where in the program that the variable exists (i.e. can be referenced). When code execution reaches a point where a variable is no longer referenceable, the variable is said to be *out of scope*.

A **for loop** allows you to reference each element in an array and executing the same code for each.

```
for(int i = 0; i < animals.length; i++) {
    ...
}
```

# While and Do While

```
while (condition) {
        loop to execute
}
```

```
do {
        loop to execute
} while (condition)
```

# That’s a lot of ifs….

```
if(name.equals("Henry")) {
        System.out.println("Edwards");
    } else if(name.equals("Mimi")) {
        System.out.println("Malone");
    } else if(name.equals("Max")) {
        System.out.println("Anderson");
    } else if(name.equals("Eddie")) {
        System.out.println("Angel");
    } else if(name.equals("Ceasar")) {
        System.out.println("Reid");
    }
```

# Switch

```
switch(name) {
    case “Henry”:
        System.out.println(“Edwards”);
        break;
    case “Mimi”:
        System.out.println(“Malone”);
        break;
    default:
        System.out.println(“I don’t know ” + name);
        break;
}
```

# Command-Line Programs.

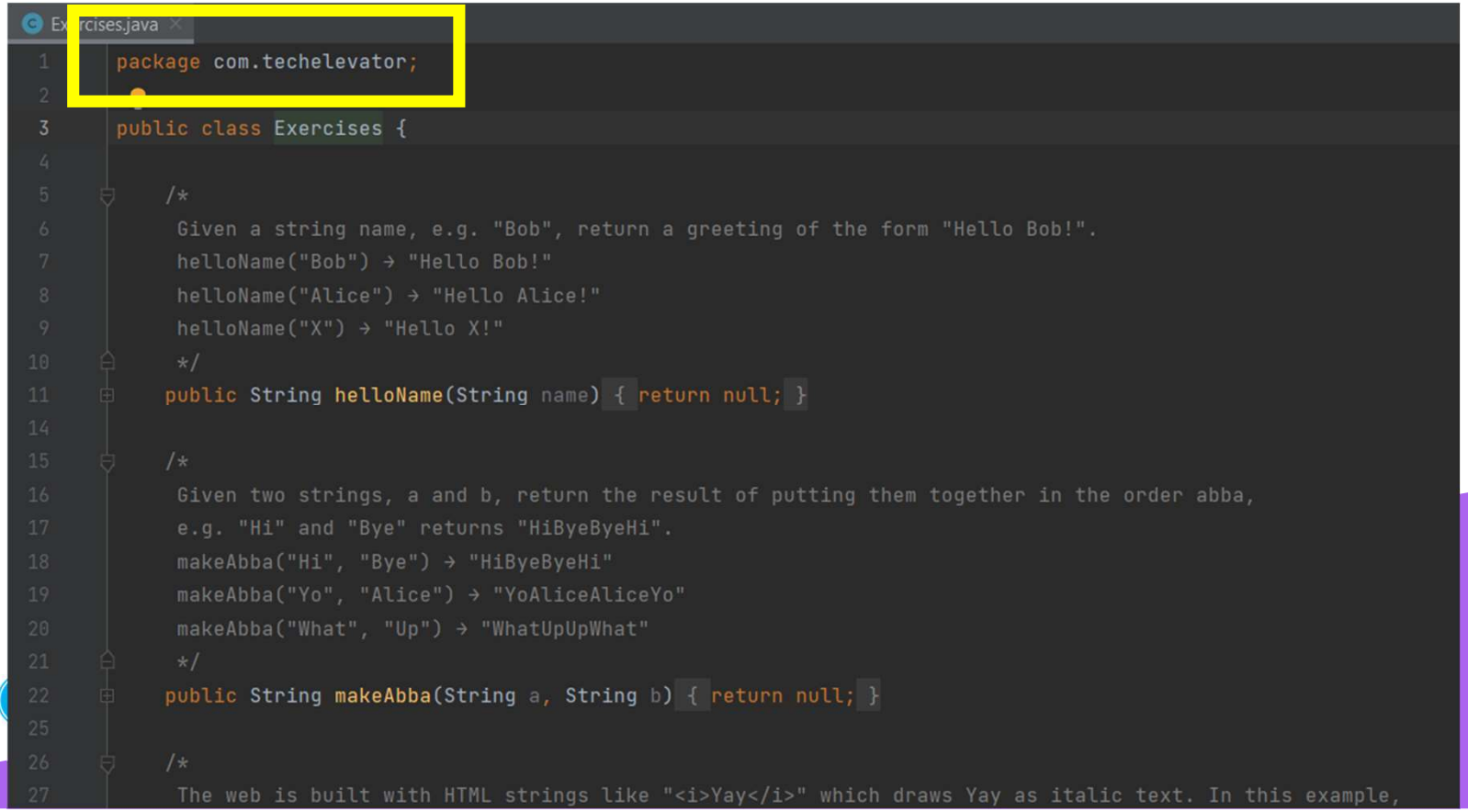
- How do we give information to the user?
- System.out.print();
- System.out.println();

- System.out refers to the standard output.

# Command-Line Programs.

- How do we get information from the user?
  - Scanner object to read the input
  - Scanner inputReader = new Scanner(System.in);
  - String userInput = inputReader.nextLine();

# Packages


Exercises.java
1 package com.techelevator;
2
3 public class Exercises {
4
5 /*
6 Given a string name, e.g. "Bob", return a greeting of the form "Hello Bob!".
7 helloName("Bob") → "Hello Bob!"
8 helloName("Alice") → "Hello Alice!"
9 helloName("X") → "Hello X!"
10 */
11 public String helloName(String name) { return null; }
14
15 /*
16 Given two strings, a and b, return the result of putting them together in the order abba,
17 e.g. "Hi" and "Bye" returns "HiByeByeHi".
18 makeAbba("Hi", "Bye") → "HiByeByeHi"
19 makeAbba("Yo", "Alice") → "YoAliceAliceYo"
20 makeAbba("What", "Up") → "WhatUpUpWhat"
21 */
22 public String makeAbba(String a, String b) { return null; }
25
26 /*
27 The web is built with HTML strings like "Yay" which draws Yay as italic text. In this example,

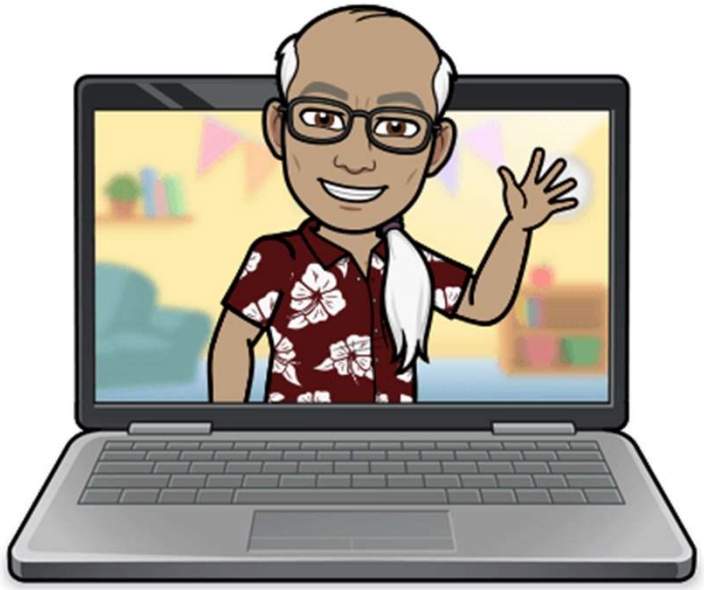
# Import Package

```java
package com.techelevator;

import java.util.Scanner;

public class Loops {

    public static void main(String[] args) {
        int[] nums = {5,10,15,20};
        for (int i = 0; i < nums.length; i++) {
            System.out.print(nums[i]);
            if(i != nums.length - 1)
            {
                System.out.print(" -- ");
            }
        }
        System.out.println();
        // try with while loop
        int i = 0;
        while (i < nums.length){
            System.out.print(nums[i]);
            if(i != nums.length - 1)
            {
                System.out.print(" -- ");
```

# Parsing the string data

- Integer.parseInt(String s);
- Boolean.parseBoolean(String s);
- Long.parseLong(String s);
- Double.parseDouble(String s);

# LET’S CODE!



# Wait, split what?

- String myName = “Henry Edwards”;
- String[] nameParts = myName.split(“ “);

# String Equality

```
String s1 = new String("HELLO");
String s1 = new String("HELLO");
System.out.println(s1 == s2);
```


```
String s1 = new String("HELLO");
String s1 = new String("HELLO");
System.out.println(s1.equals(s2));
```

# WHAT QUESTIONS DO YOU HAVE?

Reading for tonight:

**Introduction to Strings**