Week 1 Front End Overview, Command Line, Programming Basics

Welcome!

In this course, you will gain the tools to become a competent programmer. Luckily, TTS makes this fun and challenging!

What are we getting into?

> - HTML, CSS, and Bootstrap
> - Frontend Fundamentals
> - Navigating the Command Line and multiple IDE's
> - Version Control with Gitub
> - Advanced JavaScript and Angular
> - Object Oriented Programming with Java
> - Web Apps with Modern Frameworks
> - More to come later!

The road ahead

To start, let's learn about the fundamentals of programming. In week 1, we will be introduced to computer programming and front end development.

In case you were feeling overwhelmed, the prerequisite for this class is basic computer skills. So, everyone in this class is ready to ace this course.

We will learn pair programming, how to build software as a team, and how to realize a project from conception to production through a final group project.

Week 1

During week 1, we will learn the basics in the world of programming. We'll dive right in with HTML, CSS, Bootstrap, and Javascript while also discussing modern roles, responsibilities and project management.

Keys To Success

Great programmers learn how to connect with other great programmers! So, in this class, we will learn to utilize version control through Git and GitHub.

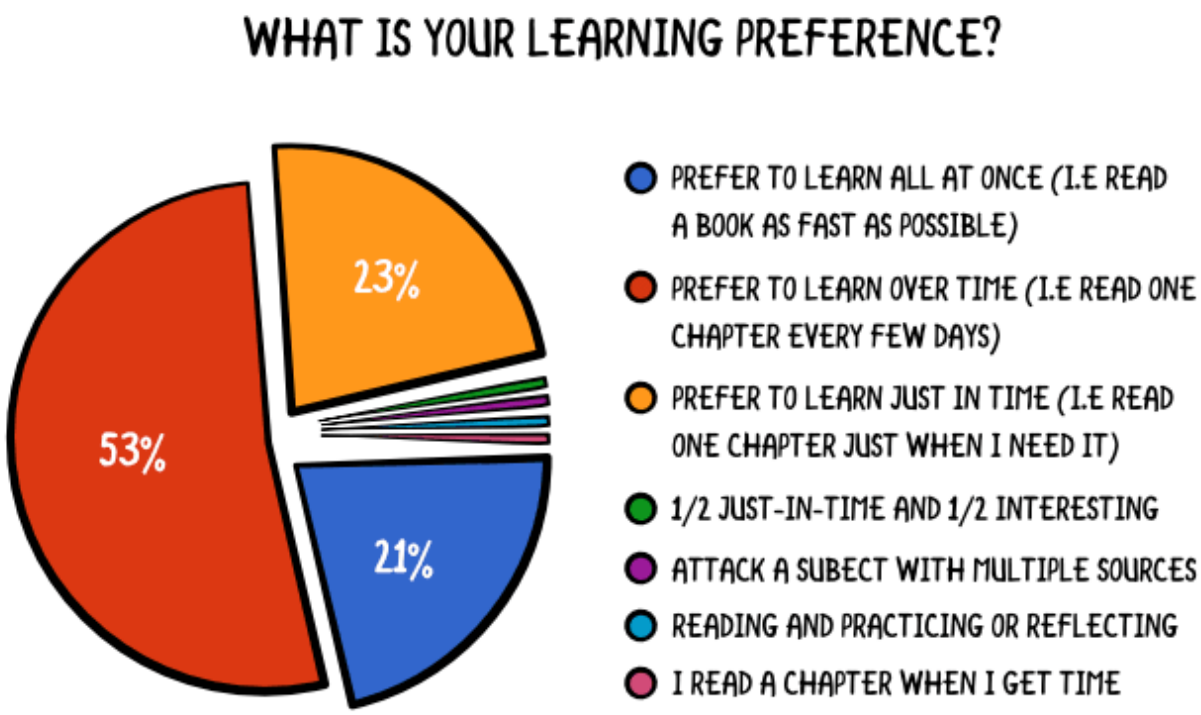
Remember, programming isn't common sense! To become an excellent programmer, you need to ask questions and not be afraid to make mistakes.

When learning to program, we usually need to go back and remember...how to learn.

Most importantly, remember to be patient and allow your strong fundamentals to rule your programming career!

You won't learn everything about the world of coding from Tech Talent South. (Well, you'll never know everything about coding) But, if you master the fundamentals here, you'll be ready for whatever software engineering challenge comes your way.

Take time to reflect on how you learn best and how you can focus on the material inside and outside of class!



Front End Overview

As a student learning web development, there are some technologies that are very important to you. For web, it is important to know HTML, CSS, Bootstrap, and Javascript. Let's go through a quick overview of these technologies together!

To follow along with the code examples today, just open up any plain text editor or code editor and type in the code.

Remember: Typing code seems like a waste of time when you could just copy and paste. But, time and time again, programmers prove that actually typing the code out helps with remembering syntax and concepts!

Let's start...from the beginning - with a definition

What exactly is computer programming?

Ever since the invention of Charles Babbage's difference engine (which executed tasks by changing the gears that executed the calculation) in 1822, computers have required a way of instructing them to perform a specific task.

A **programming language** is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks.

Computer languages were first composed of a series of steps to wire a particular program and these turned into a series of steps keyed into the computer and then executed. Later, these languages acquired advanced features such as branching and object orientation.

Computer programmming is relatively new!

The computer languages of the last fifty years have come in two stages, the first major languages and the second major languages, which are in use today.

Let's zoom in to the current way we code the web! We will overview the core technologies of front end programming.

What is HTML?

HTML is the standard markup language for web pages. It is the structure or "bones" of the web page!

Every web page is actually a HTML file. Each HTML file is just a plain-text file, but with a .html file extension instead of .txt, and is made up of many HTML tags as well as the content for a web page.

A web site will often contain many html files that link to each other.

HTML **tags** are the hidden keywords within a web page that define how your web browser must format and display the content.

Most tags must have two parts, an opening and a closing part. For example, is the opening tag and is the closing tag. Note that the closing tag has the same text as the opening tag, but has an additional forward-slash (/) character. You could think of this as the "end" or "close" character.

Tag Attributes

Attributes allow you to customize a tag, and are defined within the opening tag, for example:

```
  <img src="image1.jpg"> or <p align="center"> ... </p>
```

Below is a basic html document, containing all the essential tags. You can copy the code below, paste it into your editor, and save as firstPage.html to start your own web page.

```
<html>
 <head>
  <title>Page Title goes here</title>
 </head>
 <body>

  web page content goes here!

 </body>
</html>
```

CSS

CSS stands for Cascading Style Sheets, and is the preferred way for setting the look and feel of a website.

CSS describes how HTML elements are to be displayed

The style sheets define the color, size and position of text and other HTML tags, while the HTML files define the content and how it is organised. Separating them allows you to change the style of a site without having to rewrite your entire web site.

The **cascading** means that a style applied to a parent element will also apply to all children elements within the parent. For example, setting the color of **body** text will mean all headings and paragraphs within the body will also be the same color.

External CSS file

Like HTML files, CSS files are also plain text, and usually have a .css file extension. An example of a CSS file name style.css can be seen below.

```
body {
  background-color: beige;
  color: #000080;
}
h1 {
  color: red;
}
```

JavaScript

JavaScript can update and change both HTML and CSS. JS can calculate, manipulate and validate data. It is what gives a page life—the interactive elements and animation that engage a user.

JS is commonly used in websites to perform functions that the HTML cannot do. It can be used for validating forms, detecting browsers, adding dynamic functionality, and more!

If you've ever used a search box on a home page, checked a live basketball score on a news site, or watched a video, it has likely been produced by JavaScript.

Javascript Example:

```
<html>
 <head>
  <script type="text/javascript">
function firstFunction() { alert('top text'); }
function secondFunction() { alert('bottom text'); }
  </script>
 </head>
<body>
 <p><a href="#" onClick="firstFunction();">Top Text</a></p>
 <p><a href="javascript:secondFunction();">Bottom Text</a></p>
 </body>
</html>
```

If your brain hurts when looking at this example, that's a good thing! Do your best to understand what is going on, and TTS will explain later.

We are going to learn **Java** later in this course, so let us go ahead and answer the question:

What's the difference between JavaScript and Java?

Java - a general purpose object-oriented programming language, which runs on JVM. It can allow you to create compiled programs. Java supports Write Once, Run Anywhere. It is a standalone language.

In the early 90s, Java, which originally went by the name Oak and then Green, was created by a team led by James Gosling for Sun Microsystems, a company now owned by Oracle.

he fundamentals of Java came from a programming language called C++. Although C++ is a powerful language, it is complex in its syntax and inadequate for some of Java's requirements. Java built on and improved the ideas of C++ to provide a programming language that was powerful and simple to use.

Because Java was originally targeting mobile devices that would be exchanging data over networks, it was built to include a high level of security. Java is probably the most secure programming language to date.

Programs need to work regardless of the machines they're being executed on. Java was written to be a portable and cross-platform language that doesn't care about the operating system, hardware, or devices that it's running on.

JavaScript - It is a Scripting (lightweight) or a browser language, used to make web pages interactive. We can insert a dynamic test into HTML through JavaScript.

JavaScript is a full-fledged programming language interpreter embedded inside your web browser. You can do anything in JavaScript that a regular language like Java allows. These include:

- Declare variables
- Store and retrieve values
- Define and invoke functions
- Define your own classes

- Load and use external modules
- Write event handlers that respond to user and other events
- And...much more

Bootstrap

Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

Bootstrap is very responsive if utilized properly.

We will be learning with Bootstrap 4 as it is the newest version of Bootstrap. Bootstrap is developed by Twitter and very well respected in the development community.

Bootstrap can be boiled down to three main files:

- bootstrap.css - a CSS framework
- bootstrap.js - a JavaScript/jQuery framework
- glyphicons - a font (an icon font set)

Additionally, Bootstrap requires **jQuery** to function. **jQuery** is an extremely popular and widely used JavaScript library, that both simplifies and adds cross browser compatibility to JavaScript.

Now that we have overviewed front end development technologies, let's jump into some programming fundamentals.

Let's write our first computer program, called FirstProgram!

Important Note for Students

Some syntax with your first program (and future programs) will **not** make sense. That's okay, syntax isn't the most important thing in the beginning of your programming career. Try your best to understand what the program is doing without worrying too much about the details. We will learn all of the syntax in time!

FirstProgram

```
class FirstProgram {
 public static void main(String args[]){
     System.out.println("I wrote this myself, and I'm wicked smart");
 }
}
```

In this coding block, we created a class called 'FirstProgram' which is capitalized because it is a *class*. We capitalized the p in 'Program' because this is camel case and easier to read.

tip - It is good practice to not allow spaces between words in class names and variable names.

High and Low Level Languages

Why do we choose certain computer programming languages over others?

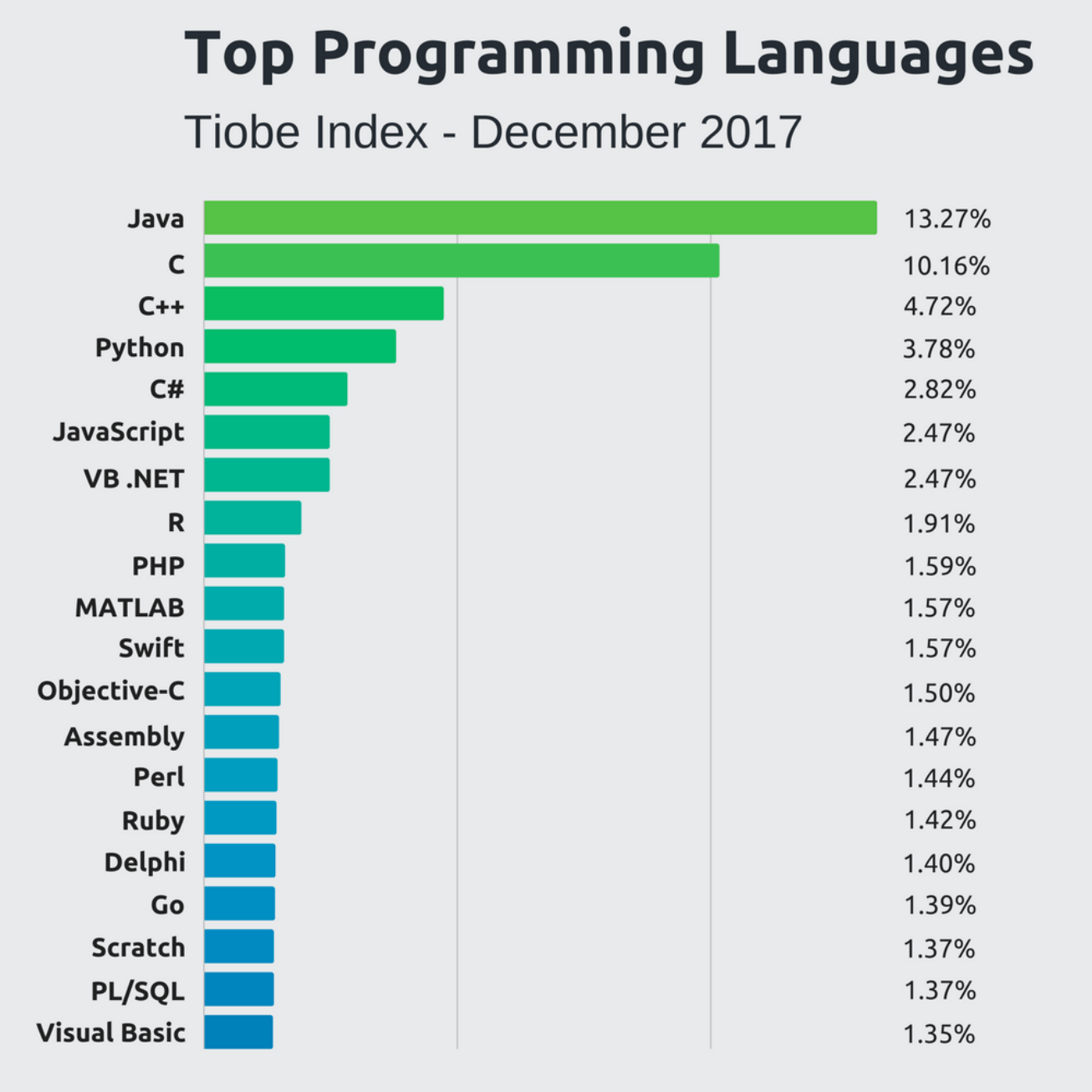
High level language - a programming language which has strong abstraction from the computer. The language is written in a simple, easy to understand format, and is independent of computer architecture.

In case of Java, the programming language, sits on top of a Java Virtual Machine or JVM. The JVM is the one interacting with the hardware, and Operating System. So as an end developer, you are only interested in writing the code.

Low level language

A low-level programming language is a programming language that provides little or no abstraction from a computer's instruction set architecture—commands or functions in the language map closely to processor instructions

According to stackify.com, here are the top languages in 2017. Developers work with high level lanaguages so that they can work faster and solve high level problems.



JAVA

One of the high level languages we will be focusing on a lot in the future, **Java**, is shown below

Compared to C, Java is less syntax heavy, but still performs well!

```java
import java.util.Scanner;
import java.net.URL;

public class Main {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(
            new URL("http://www.google.com").openStream()
        );
        while (sc.hasNext())
            System.out.println(sc.nextLine());
    }
}
```

Remember, computers are dumb repetition machines!

Humans are what make computers smart! Using programming languages, we give them a set of instructions (code) needed to complete a task. Where computers beat humans is in speed. They work **very fast**! However, being fast doesn't make you smart; being accurate does. Computers get their accuracy from the the code we provide them.

- Computers can only do a few things
- Computers work well with values and calculations
- Computers can do these calculations **really fast**, making them appear smart

In the end, your computer performs a lot of operations on 0s and 1s and does this in an instant.

Before we learn the basics of programming, let's learn about navigating the command line

The **command line** or **terminal** is an interface in which you can type and execute text based commands.

It can be much faster to complete some tasks using a Terminal than with graphical applications and menus. Another benefit of using terminal or command line is having access to many more commands and scripts.

A common terminal task of installing an application can be achieved within a single command, compared to navigating through the Software Centre or Synaptic Manager.

Lab Activity

With the help of your instructor, tackle this Java coding problem: https://codingbat.com/prob/p171896

The Command Line

We're going to review some of the command line prompts to make sure you're all good and comfortable navigating through your command line, as that we'll be moving around the terminal a lot as this class goes on.

Review the following commands (MAKE NOTE THIS IS GOING TO BE DIFFERENT IN WINDOWS)

- `cd directory_name` - change directory
- `cd ..` - go one level up in the directory structure
- `ls` (`dir` in Windows) - list contents of current directory

- `pwd` - print current working directory
- `mkdir directory_name` - make a new directory
- `touch file_name` (`type NUL >> filename` in windows) - create a new file
- `mv from_path to_path` (`move` in windows) - move file or directory, also used to rename
- `cp from_path to_path` (`copy` in windows) - copy file or directory
- `rm file_name` (`del` in windows) - deletes a file an with some flags can delete directories as well. `rmdir` is available as well for directory removal

Lab Activity

- **cd** or **cd ~** to get to our root
- **ls**
- **mkdir LionKing** (can't have spaces or it will make 2 directories)
- **cd LionKing**
- **mkdir goodLion badLion**
- **ls**
  - (open up our GUI to see what is actually going on)
- **cd goodLion**
- **touch simba.html**
- **touch mufasa.html nala.jpg**
- **pwd**
- **up arrow, down arrow** (cycle through previous commands)
- **mv nala.jpg scar.jpg**
- **mv scar.jpg ../badLion**
- **rm *.html**
- **cd**
- **rm -r LionKing**

Now that we have some understanding of navigating via command line, let's go over some programming fundamentals!

Computer Programming Basics

Let's briefly go over some programming basics.

1. declaring and initializing variables

2. commenting

Since you will be focusing on Java later in this course, we will be learning the fundamentals today in Java. Of course, Java isn't the only language you can learn programming with. But, because of its widespread use and longevity, its a great one!

What is a variable?

Variable Declaration:

Languages can be strong, weak, and duck-typed

Strong Typed - choose a data type and you cannot change it

Weak Typed - You can assign variables to different types

Duck Typed - don't need a type in order to invoke an existing method on an object

To declare a variable, you must specify the data type & give the variable a unique name.

Note: just about all programming languages do not care about whitespace. But in general, we write one statement per line as follows.

```
int a,b,c;

float pi;

double d;

char a;
```

Variable Initialization:

To initialize a variable, you must assign it a valid value.

```
pi = 3.14f;

do = 20.22d;

a= 'v';
```

Commenting

Commenting is very important when programming by yourself. Commenting is even more important when you are sharing code with other people. In c based languages we start a line with two forward slashes like: //

```
// delare name variable (String)
var myName = "Jim"

var annualSalary = 90000000 // Jim's salary (Int)
```

Strings

What are Strings? A string in literal terms is a series of characters. Hey, did you say characters, isn't it a primitive data type in Java. Yes, so in technical terms, the basic Java String is basically an array of characters.

Why use Strings? One of the primary functions of modern computer science, is processing human language.

Similarly to how numbers are important to math, language symbols are important to meaning and decision making. Although it may not be visible to computer users, computers process language in the background as precisely and accurately as a calculator. Help dialogs provide instructions. Menus provide choices. And data displays show statuses, errors, and real-time changes to the language.

As a Java programmer, one of your main tools for storing and processing language is going to be the String class.

String Syntax Examples Now, let's get to some syntax,after all, we need to write this in Java code isn't it.

String is an array of characters, represented as:

String is an array of characters

```
Java
char[] arrSample = {'R', 'O', 'S', 'E'};
String strSample_1 = new String (arrSample);
```

Arrays

Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

The first element of an array starts with index zero.

In simple words, it's a programming construct which helps to replace this:

```
x0=0;
x1=1;
x2=2;
x3=3;
x4=4;
x5=5;
```

with this:

```
x[0]=0;
x[1]=1;
x[2]=2;
x[3]=3;
x[4]=4;
x[5]=5;
```

Constructing an Array

```
arrayname = new dataType[]
```

Initialize an Array

```
intArray[0]=1; // Assigns an integer value 1 to the first element 0 of the array

intArray[1]=2; // Assigns an integer value 2 to the second element 1 of the array
```

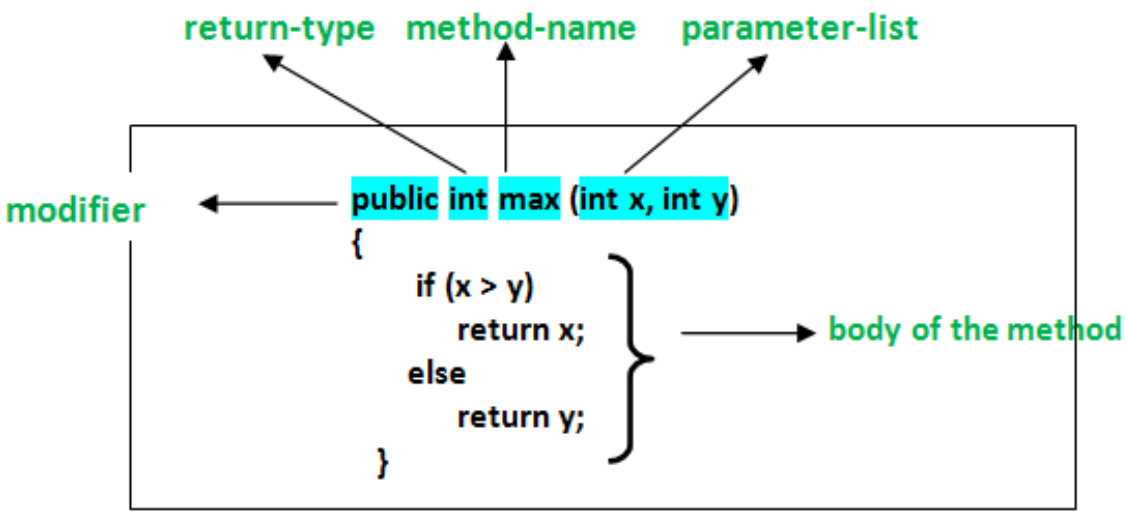
Using an array in Java:

```
class ArrayDemo{
     public static void main(String args[]){
        int array[] = new int[7];
        for (int count=0;count<7;count++){
           array[count]=count+1;
       }
       for (int count=0;count<7;count++){
           System.out.println("array["+count+"] = "+array[count]);
       }
      //System.out.println("Length of Array  =  "+array.length);
      // array[8] =10;
      }
}
```

Arrays are an integral part of programming! Great job

What is a Function or Method in Java?

A method is a collection of statements that perform some specific task and return the result to the caller. A method can perform some specific task without returning anything. Methods allow us to reuse the code without retyping the code. In Java, every method must be part of some class which is different from languages like C, C++, and Python. Methods are time savers and help us to reuse the code without retyping the code



Method signature: It consists of the method name and a parameter list (number of parameters, type of the parameters and order of the parameters). The return type and exceptions are not considered as part of it.

Method Signature of above function:

```
max(int x, int y)
```

How to name a Method?: A method name is typically a single word that should be a verb in lowercase or multi-word, that begins with a verb in lowercase followed by adjective, noun..... After the first word, first letter of each word should be capitalized. For example, findSum, computeMax, setX and getX

Generally, A method has a unique name within the class in which it is defined but sometime a method might have the same name as other method names within the same class as method overloading is allowed in Java.

Calling a method

The method needs to be called for using its functionality. There can be three situations when a method is called: A method returns to the code that invoked it when:

1. It completes all the statements in the method
2. It reaches a return statement
3. Throws an exception

```
// Program to illustrate methodsin java
import java.io.*;

class Addition {

  int sum = 0;

  public int addTwoInt(int a, int b){

    // adding two integer value.
    sum = a + b;

    //returning summation of two values.
    return sum;
  }

}

class GFG {
  public static void main (String[] args) {

    // creating an instance of Addition class
    Addition add = new Addition();

    // calling addTwoInt() method to add two integer using instance created
    // in above step.
    int s = add.addTwoInt(1,2);
    System.out.println("Sum of two integer values :"+ s);

  }
}
```

output

```
Sum of two integer values : 3
```

```
// Java program to illustrate different ways of calling a method
```

```java
import java.io.*;

class Test
{
  public static int i = 0;
  // constructor of class which counts
  //the number of the objects of the class.
  Test()
  {
    i++;

  }
  // static method is used to access static members of the class
  // and for getting total no of objects
  // of the same class created so far
  public static int get()
  {
    // statements to be executed....
    return i;
  }

  // Instance method calling object directly
  // that is created inside another class 'GFG'.
  // Can also be called by object directly created in the same class
  // and from another method defined in the same class
  // and return integer value as return type is int.
  public int m1()
  {
    System.out.println("Inside the method m1 by object of GFG class");

    // calling m2() method within the same class.
    this.m2();

    // statements to be executed if any
    return 1;
  }

  // It doesn't return anything as
  // return type is 'void'.
  public void m2()
  {

    System.out.println("In method m2 came from method m1");
  }
}
```

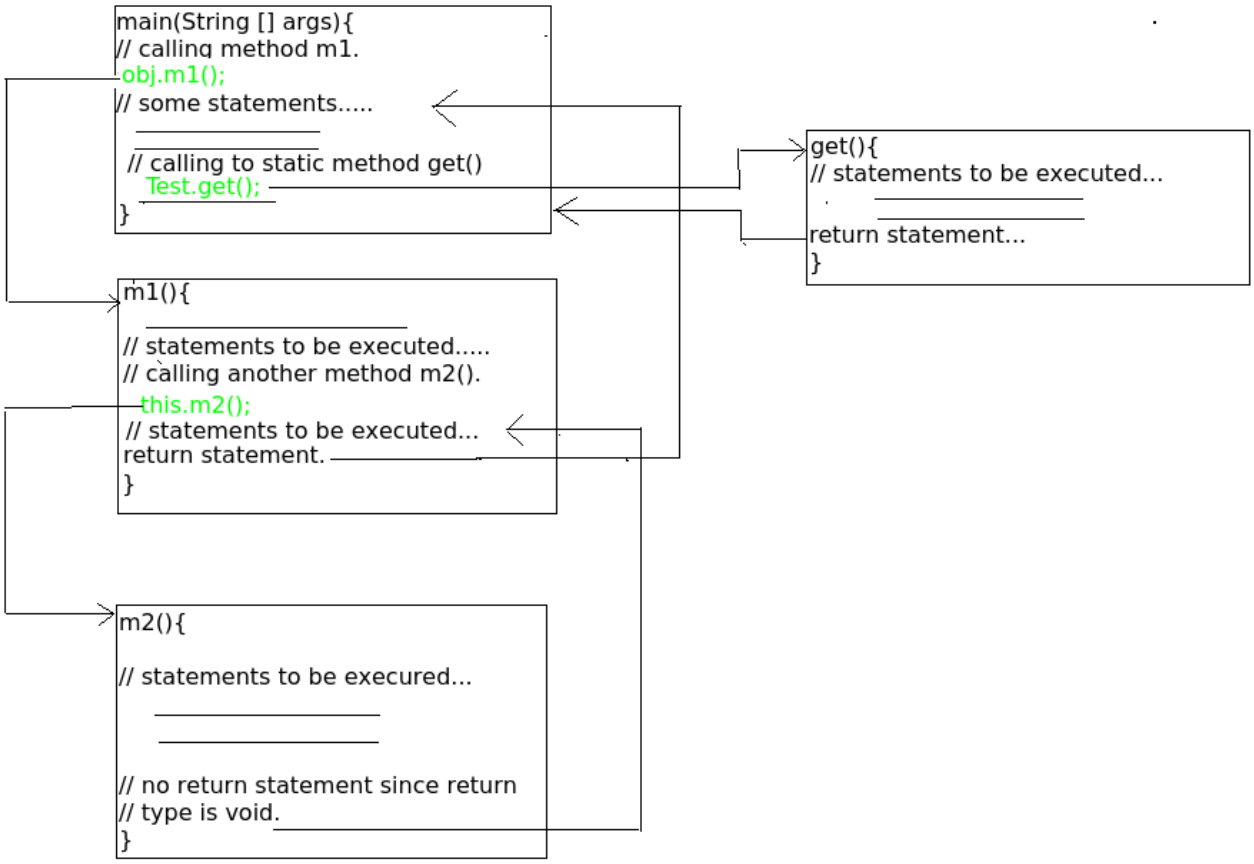
```
class GFG
{
  public static void main(String[] args)
  {
    // Creating an instance of the class
    Test obj = new Test();

    // Calling the m1() method by the object created in above step.
    int i = obj.m1();
    System.out.println("Control returned after method m1 :" + i);

    // Call m2() method
    // obj.m2();
    int no_of_objects = Test.get();

    System.out.print("No of instances created till now : ");
    System.out.println(no_of_objects);

  }
}
```



Data Operators

There are mainly 4 different types of operators, which are listed below:

Arithmetic Operator

Perform arithmetic operations such as addition, subtraction, multiplication, division and modulus. Unary Operator: Unary operators are used to increment or decrement a particular value. For example: ++ stands for increment, – – stands for decrement. Relational Operator: It defines some kind of relation between two entities. For example: <, >, <=, >=, !=, ==. Logical Operator: Logical operators are typically used with boolean (logical) values.

Control statements // to do picture

Control statements are the statements that define the flow of your program. There are 3 types of control statements in Java: Selection, iteration and jump statements.

Selection Statements

Selection statements allow you to control the flow of the program during run time on the basis of the outcome of an expression or state of a variable. For example: you want to eat pizza, but then where can you get that pizza in best price. You can select between various popular options like Domino's, Pizza Hut or any other outlet. So here you are following a selection process from the various options available.

Now these statements can be further classified into the following:

If-else Statements Switch Statements

Example:

```
public class Compare {
     int a=10,
     int b=5;

if(a>b)
      {  // if condition
     System.out.println(" A is greater than B");
      }
else
      {     // else condition
      System.out.println(" B is greater");
      }
}
```

Iteration Statements

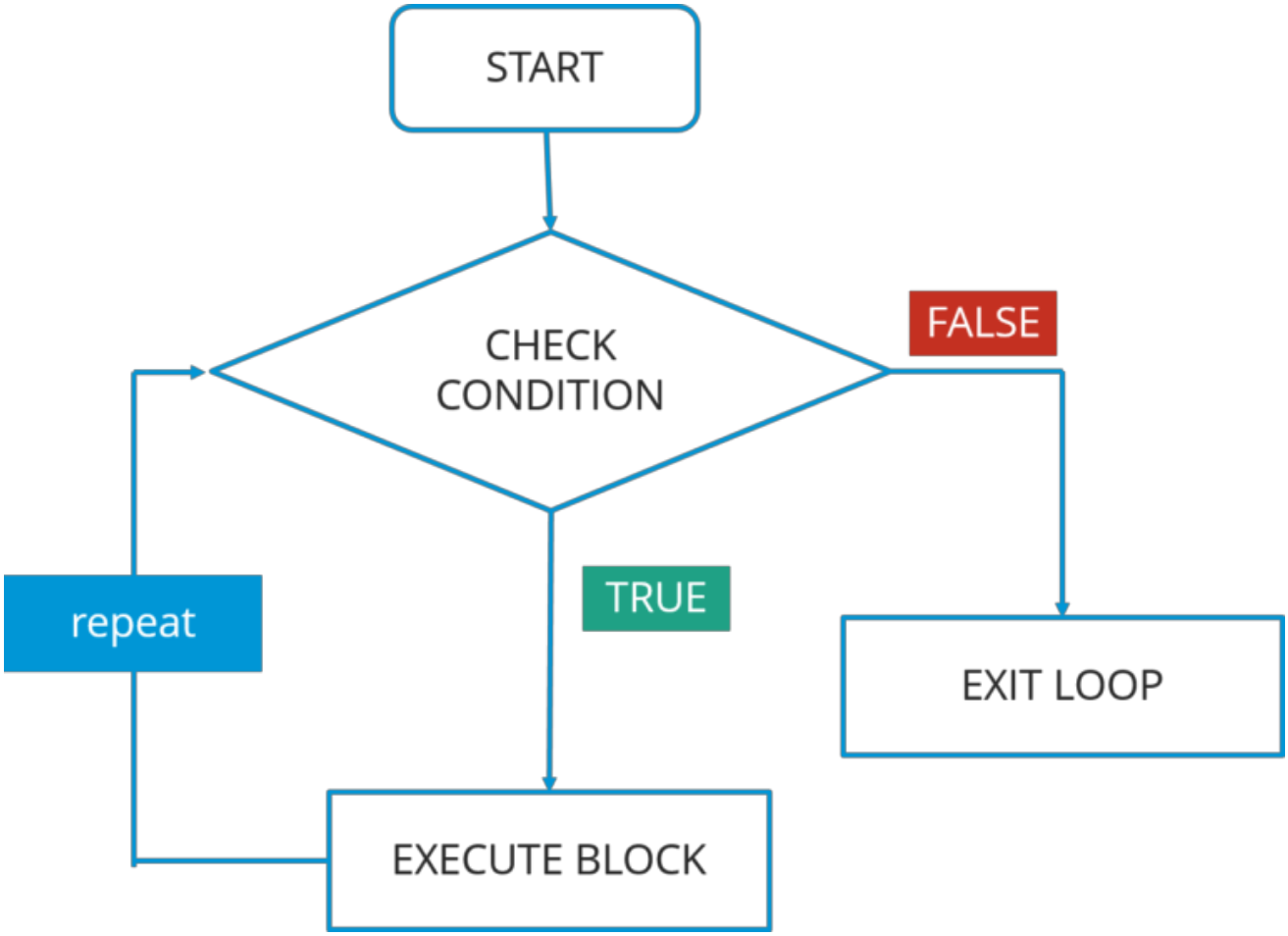
There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages −

In Java, these statements are commonly called as loops, as they are used to iterate through small pieces of code. Iteration statements provide the following types of loop to handle looping requirements.

While statement: Repeat a group of statements while a given condition is true. It tests the condition before executing the loop body.



```
public class WhileExample {
      public static void main(String args[]) {
         int a=5;
   while(a<10)   //while condition
         {
         System.out.println("value of a" +a);
         a++;
   System.out.println("
");
         }
    }
}
```

Recap: Here is a brief recap of today's class:

We have skimmed over the front end technologies we will be learning in this course. We also briefly covered the command line, so that we could navigate our computers better in the future. Lastly, we went over some essential programming fundamentals. We will go over programming fundamentals more and more in the next couple weeks.

Thanks for being here, and pat yourself on the back. You're on your way to being an amazing programmer! Programming fundamentals are s challenging to tackle at first. Here at TTS, we won't leave you on your own.

References:

1. https://www.edureka.co/blog/java-tutorial/
2. https://www.tutorialspoint.com/java/java_loop_control.htm
3. https://www.geeksforgeeks.org/methods-in-java/
4. http://www.simplehtmlguide.com/whatishtml.php
5. https://cs.brown.edu/~adf/programming_languages.html
6. https://www.thoughtco.com/what-is-java-2034117
7. https://www.makeuseof.com/tag/what-is-javascript/