Week 1 – Topics

- Tools Installation & Setup
    - Java
    - Eclipse
    - Git & GitHub

- What is Programming?

- Java Intro
    - Basic Programs
    - Variables & Data Types
    - Operations

- Command Line Interface (CLI)

- Source Control
    - Git / GitHub

What is Programming?

What is Programming?

The definition of Computer Programming in the simplest form is the process of preparing a program to complete tasks on a computer

Programming is writing instructions to tell a computer how to move, manipulate and display data in an automated fashion

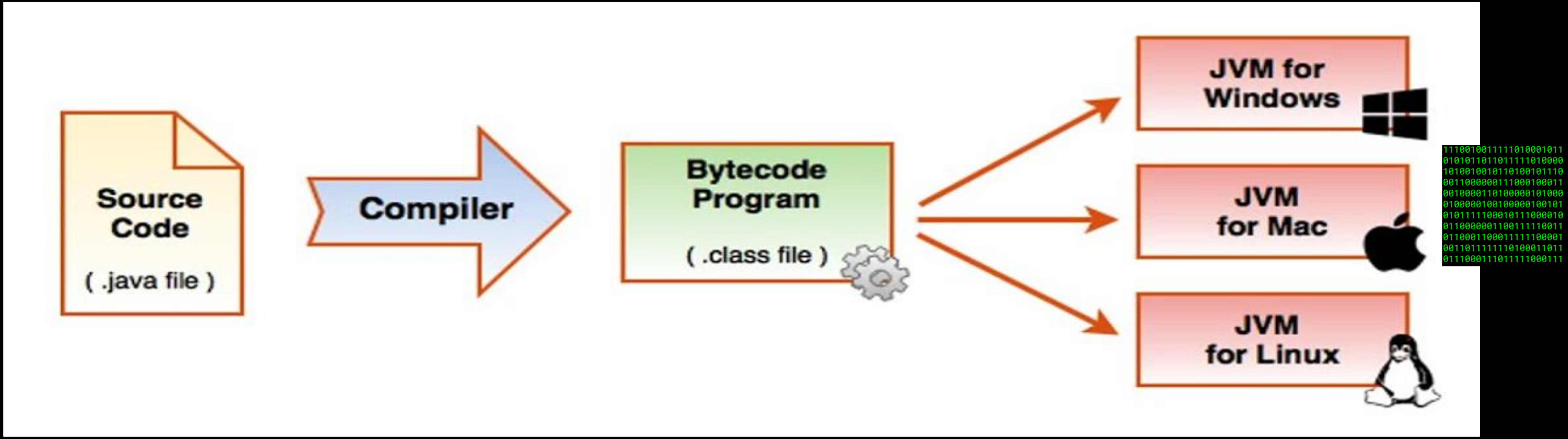
The goal is to break down a problem into smaller pieces to instruct a computer how to do each task:

- The computer would need to have a place to store & manipulate its **data** (**variables**)
- The computer would need to know **what** we want it to do (**tasks, methods**)
- The computer would have to be taught **how** to accomplish those tasks (**operations**)

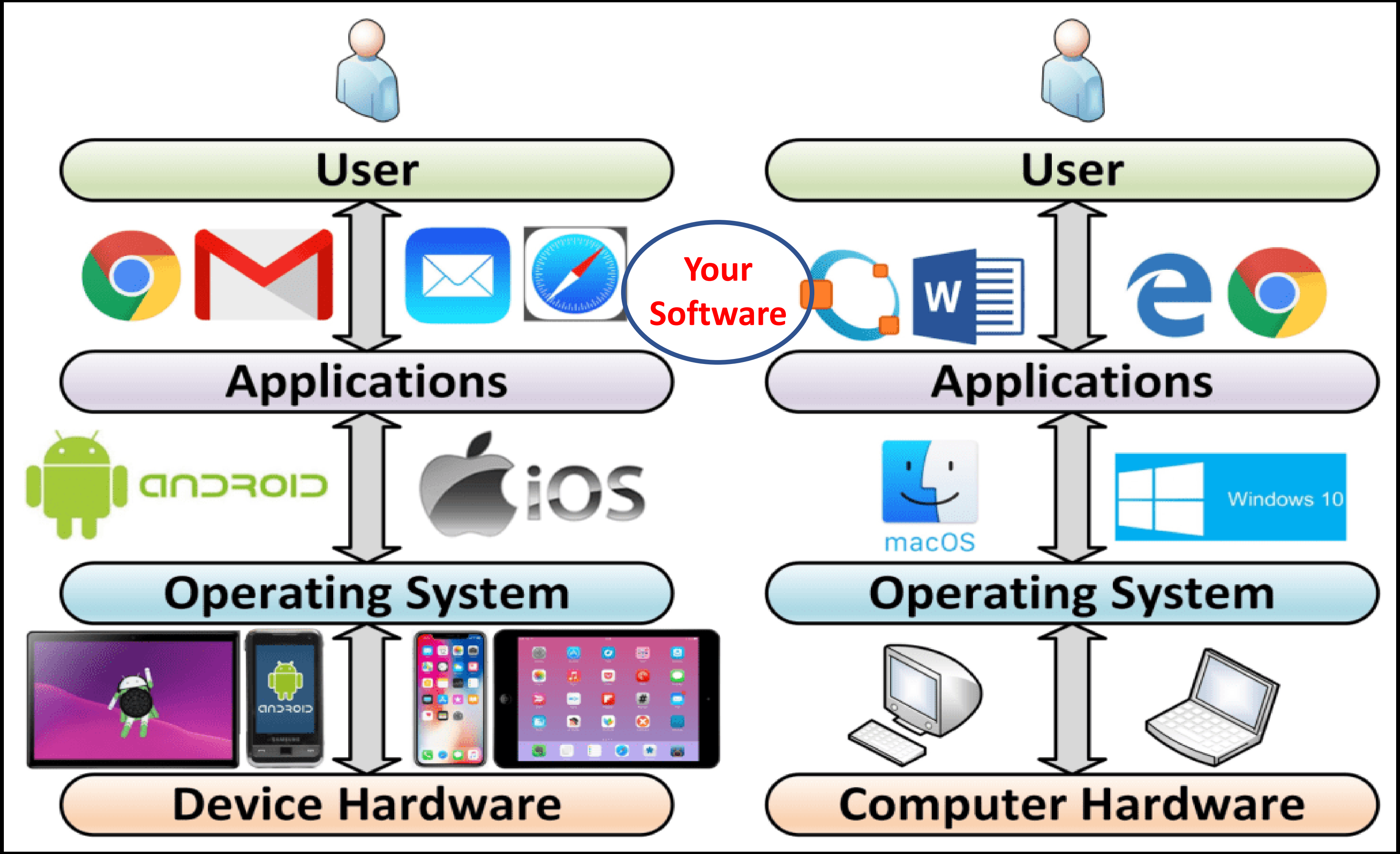
Content Link in LMS

Why (and how) do we write programs?

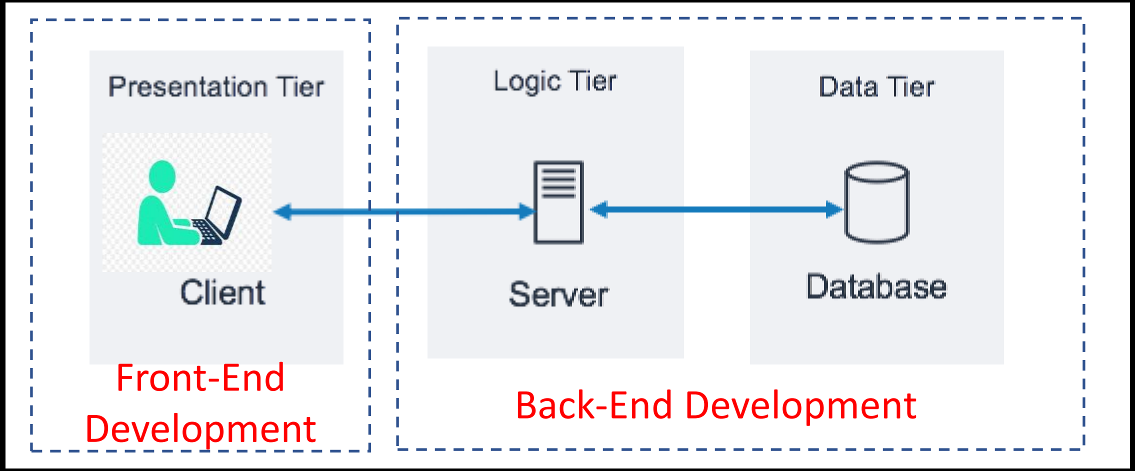
- Computers think and understand Binary (1’s and 0’s)…period!
- We use programming languages like Java to write instructions in human readable/understandable code that becomes binary executable code that is run on a computer

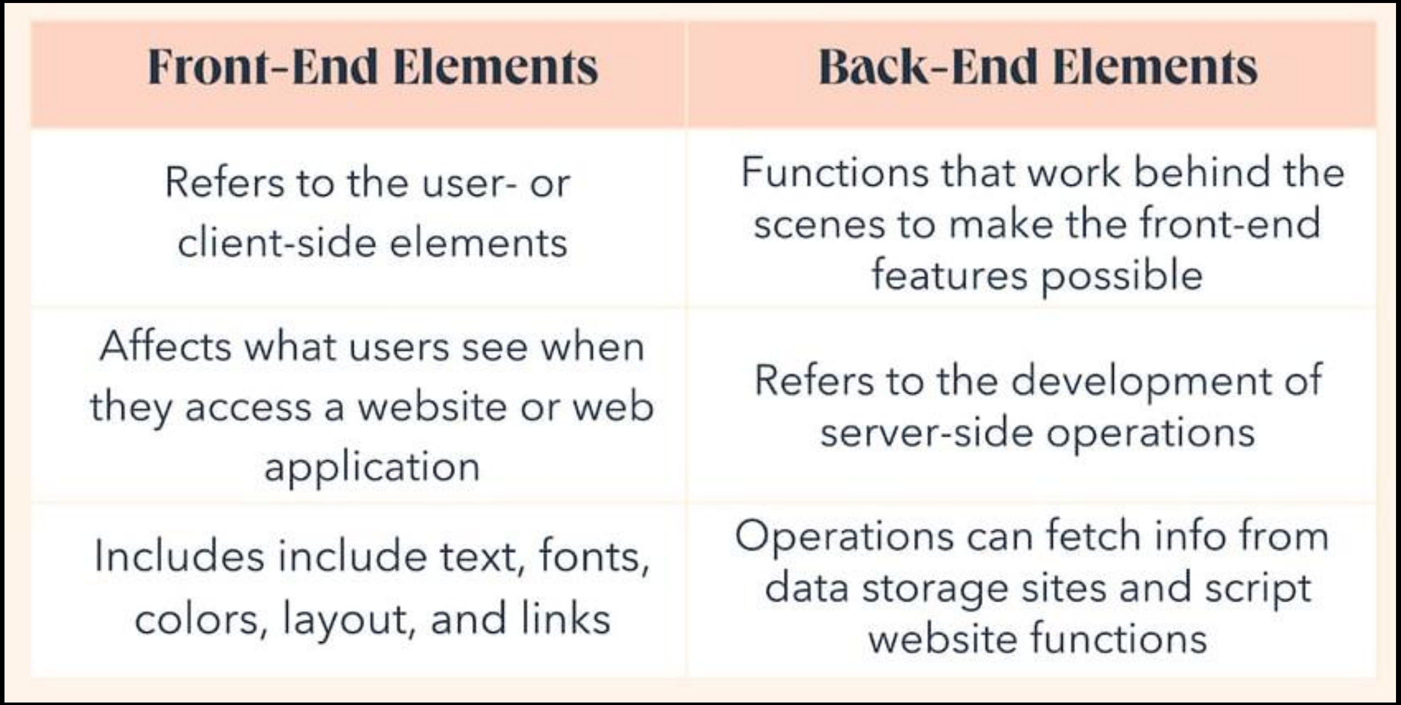


The Technology Layer Cake: Users, Applications (Software), Operating System, Hardware



Front-End vs Back-End Development





Front-End Elements	Back-End Elements
Refers to the user- or client-side elements	Functions that work behind the scenes to make the front-end features possible
Affects what users see when they access a website or web application	Refers to the development of server-side operations
Includes include text, fonts, colors, layout, and links	Operations can fetch info from data storage sites and script website functions

Steps to Programming

- Decide how to solve a task or request (Design) -- remember to break the task into smaller pieces to be solved

- Write the code to solve the task by first writing code to solve each smaller piece of the solution

- Compile and/or Run the program that has been written.

- Test and Debug the program
  - Test: Make sure the code works as designed.
  - Debug: Track Down and Fix any flaws in the code that you wrote.

- Document exactly what has been coded and how it works.

Content Link in LMS

Java Introduction

Overview of Java

Java is the computer programming language that we are going to be using throughout this course.

- **Java** was created at Sun Microsystems, Inc. with James Gosling as the lead architect, first released in 1995, and brought something new to the programming world. The Java implementation minimizes dependencies, and is platform independent and portable.

- The Java compiler (*javac*) compiles Java source code, and converts that code into *bytecode*, which is executed by the **JVM** (Java Virtual Machine). The **JVM** transforms the *bytecode* into machine-readable code.

- The two-step compilation process is how platform independence leading to portability is accomplished -- one of Java's greatest features.
  - **platform independence**: A **Java** program that is compiled on one machine can be executed on any other machine, regardless of the Operating System, as long as the target machine has a **JVM** installed.
  - **portability:** a program (set of code) will run identically on different platforms, with no changes such as recompilation or rewriting source code.

Common Java Acronyms:
- **JDK** -- the development platform -- **Java Developer's Kit** -- what you will install on your system.
- **JVM** -- for runtime execution -- **Java Virtual Machine** -- provides a place for Java byte code to be executed.
- **JRE** -- a software environment, which includes a set of tools used for Java application development, also referred to as the implementation of the JVM -- **Java Runtime Environment**

Java Tools / Environment Set-up

Installation of Java:

We will be installing the current **LTS** of Java. Java has a number of **Long Term Support** versions of the language. The videos here speak of installing Java 8 (v1.8). Since Java 8 was released, there have only been two additional LTS releases of Java, including Java 11, and Java 17. At this point, you are welcome to install any of these LTS releases.

In this course, we will be using the **JDK**, which is the Java Developers Kit. It is important to download the **JDK** of the Java Version that you choose.

Download and install the Java JDK version of your choice, keeping in mind the LTS versions.

Installation of Eclipse:

We will also be installing an **Integrated Development Environment** or **IDE**. The IDE that we use in this course is **Eclipse (for Java Developers)**. An **IDE** is a sophisticated text editor that has built-in knowledge of one or more programming languages. This is the location where we will be writing our code, in particular our Java code.

After you download and install **Eclipse (for Java Developers)** following the Promineo Tech installation instructions, open the application. The first time that you open Eclipse, you will see a "Welcome" page. Feel free to read the information that is located there. For our purposes, we are going to exit out of that Welcome page, and you should see the standard Eclipse layout.

Java Tools / Environment Set-up

- Installation of Java
- Installation of Eclipse

Java Project Set-up

- Create a new Java Project
- Create a new Java Package
- Create a new Java Class → Basic Java first program

Content Link in LMS

Variables, Data Types, & Operations

Variables / Data Types

- Since programming is defined as moving, manipulating, and displaying data, we need a way to know what data we are working with. We need a way to **assign names to data**. To do this, we use something called **variables**.
- **Variables** can refer to different types of data. **Variables** are placeholders to hold our data. For example, alphanumeric/textual data or numeric data deals more with values and math.
- **Java** is a statically typed programming language, which means that any variable must first be declared before it can be used.
- To **declare a variable**, you must determine the variable's **data type** and **name**.

There are 8 primitive data types in Java:

```
byte    -  8-bit integer, Range = -127 to 128    (1 = 01, 2 = 10, 3 = 11)
short   - 16-bit integer, Range = -32,768 to 32,767
int     - 32-bit integer, Range = -2^31 to 2^31-1
long    - 64-bit integer, Range = -2^63 to 2^63-1
float   - 32-bit floating point with a decimal
double  - 64-bit floating point with a decimal
boolean - true/false
char    - Single 16-bit character (e.g., ‘a’, ‘B’, ‘c’, ‘G’)
```

- Additionally, there are predefined Object data types. Here is an example of one of those:
  String - textual data - "This is a string"

Content Link in LMS

Operations

- Knowing that there are different types of data is great, but what is data good for if we don't use it in some way? *For example, point of sales systems (the software used when we purchase something from a store, restaurant, etc) have to add up line items and then apply a tax to them.*
- That means we have to perform actions on data (addition and multiplication in this case). In programming, these actions are called **operations**.
- An **operation** consists of one or more pieces of data, known as **operands**, and an **operator**, and performs a calculation or action on the operands thus resulting in a new value. One operator we are already familiar with is the **assignment operator** (the equals sign or =), which assigns the data on the right-hand side to the **variable** name/identifier on it's left.

Content Link in LMS

Command Line Interface

Command Line Interface (CLI) vs Graphical User Interface (GUI)

A **Graphical User Interface or GUI** is the visual representation of data that users interact with via mouse and keyboard. If we open a window or an application and there are buttons and text boxes, that is a **GUI**. **Graphical User Interfaces** or **GUI**s are great for displaying data in an extremely user-friendly manner, however, they are not always the quickest, most effective way to work with computers.

Command Line Interfaces or **CLI**s are text-based tools that allow us to interact with a computer and data. Rather than visual elements, everything on a **CLI** is represented via plain text. Instead of clicking on different buttons to perform some sort of action, users type in text commands. This removes the need for a mouse and reduces the time spent to perform certain operations.

Why do we care? Many programs used in software development only come as a **Command Line Interface** or **CLI** tool and have no **Graphical User Interface** or **GUI** version. There are many different **CLI**s, but the main two are **Command Prompt** (Windows) and **Terminal** (Mac and Linux).

Note: We will discuss some basic commands to navigate the file system via **CLI**, but before we do, it is important to realize that what we see in the **CLI** is the **same thing** we see in the **GUI**. If we are in a **folder** (also known as a **directory**) in our **GUI File Explorer** and in our **Command Prompt** (Windows), or our **GUI Finder** and in our **Terminal** (Mac), we will see the same files in each, the **GUI** will display the files graphically, and the **CLI** will display the files textually.

Content Links in LMS

Source Control

Git and GitHub

Git and GitHub

- **git** is a "distributed version control system designed to handle everything from small to very large projects with speed and efficiency". *(git, Retrieved on 10/5/2022 from https://git-scm.com).*

- **git** is used in many software development environments, and provides a number of benefits for any software developer. It provides a backup of code that you are writing on your local system. In addition, when working on a project with a team, it provides a way to ensure everyone has the same, up-to-date code.

- **git** is a Source Control tool used to enable teams to collaborate on projects at the same time. If users make changes to the same file that would overwrite one another, **git** provides merge conflict messages that enable users to manually adjust the multiple changes so that no one's code is overwritten. **git** also keeps track of all the changes made to a project by using save-points called commits.

- **GitHub** is a web-based git **GUI** that enables us to visually manage our repositories and push our code to the cloud (**GitHub** servers).

Content Link in LMS

- Free Book on Git
  https://git-scm.com/book/en/v2

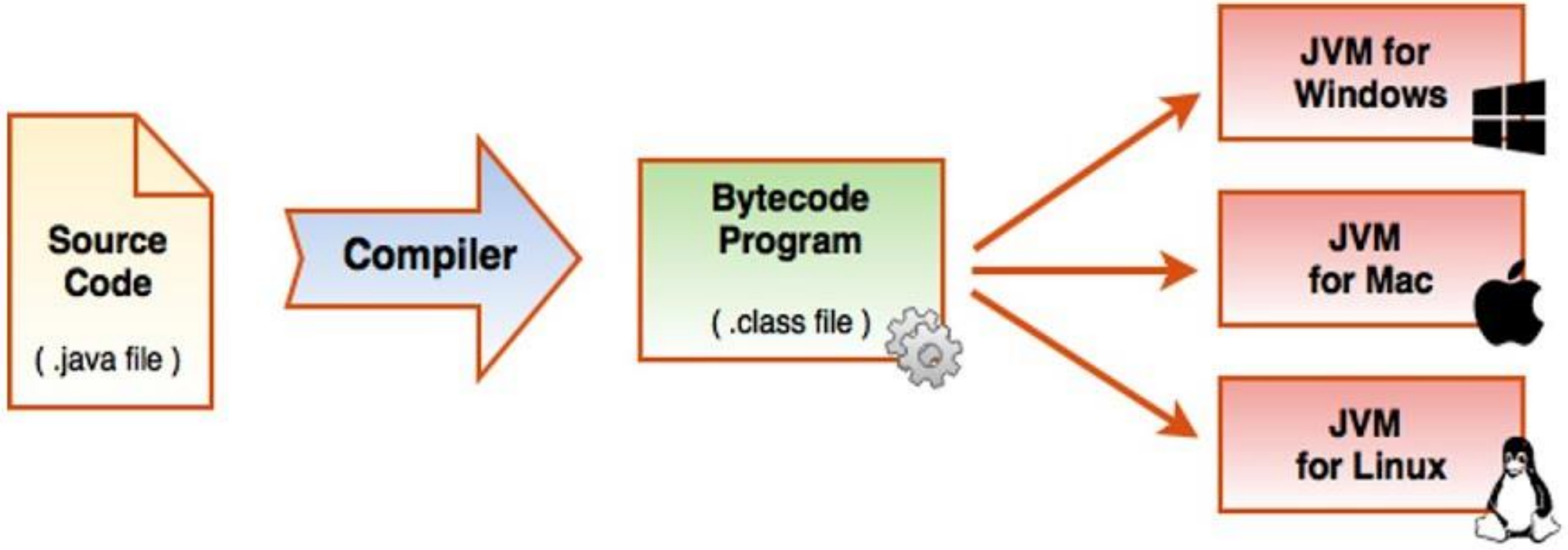
Starting a New Project Repository in Git / GitHub

- Starting a New Project Repo in Git / GitHub
- Basic git Workflow
- Uploading your Assignments to GitHub
- Free Book on Git
  https://git-scm.com/book/en/v2

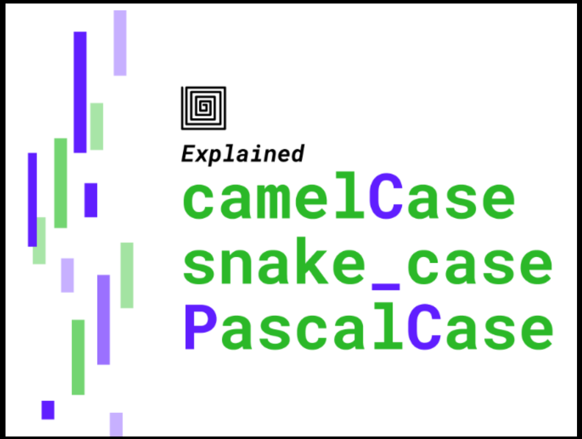
Content Link in LMS

Appendix

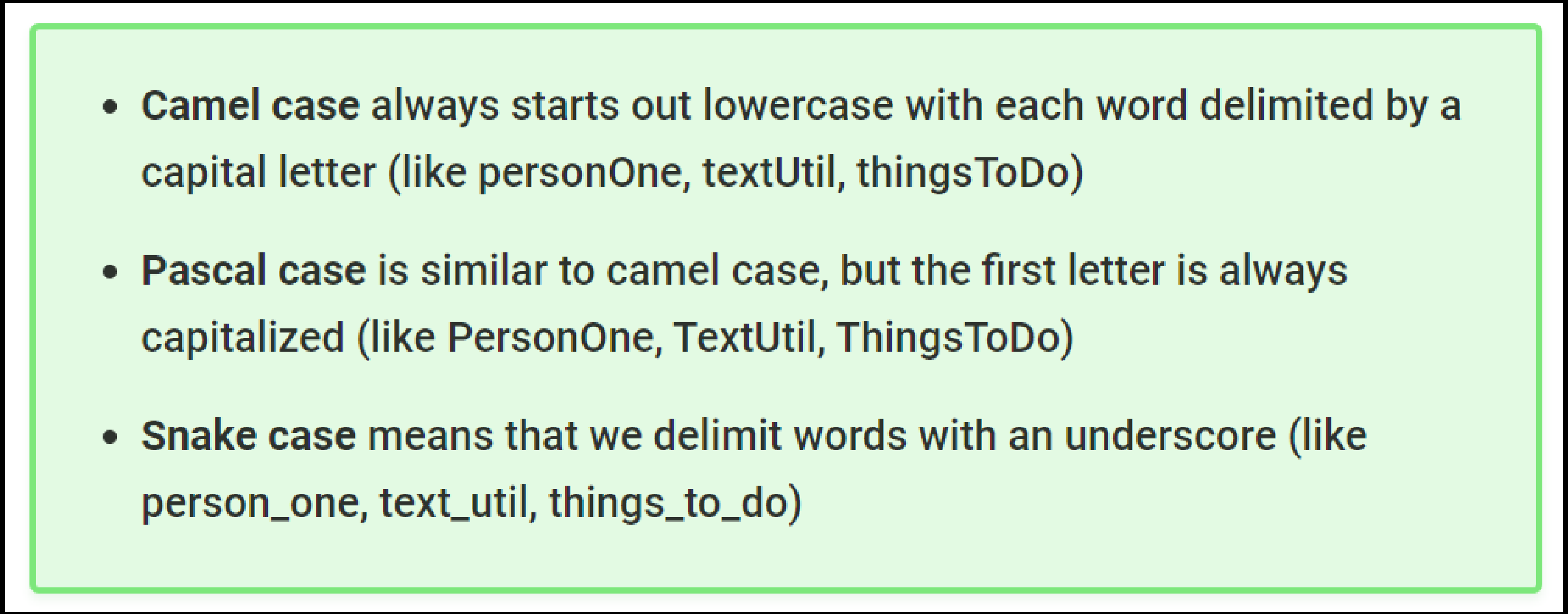
Java's BYTECODE & JVM working



Java Naming Conventions / Standards



In Java:
- Class names use PascalCase
- Variable names & Method names use camelCase
- We generally don't use snake_case in Java



- **Camel case** always starts out lowercase with each word delimited by a capital letter (like personOne, textUtil, thingsToDo)
- **Pascal case** is similar to camel case, but the first letter is always capitalized (like PersonOne, TextUtil, ThingsToDo)
- **Snake case** means that we delimit words with an underscore (like person_one, text_util, things_to_do)