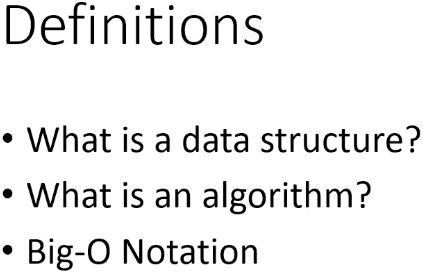
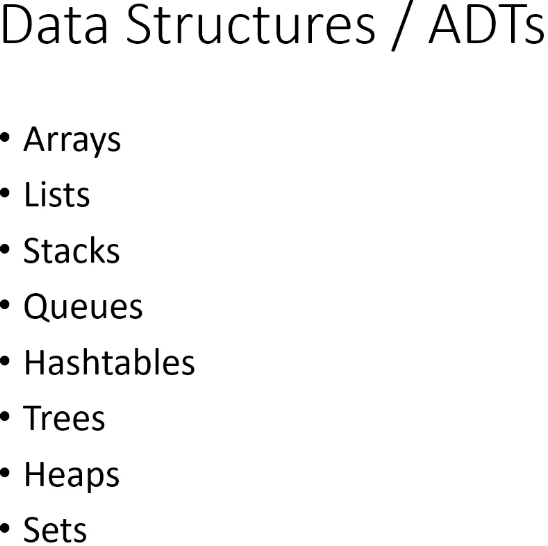
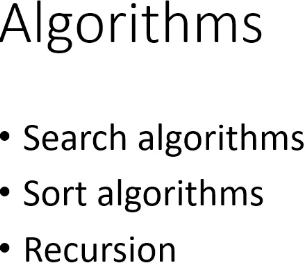
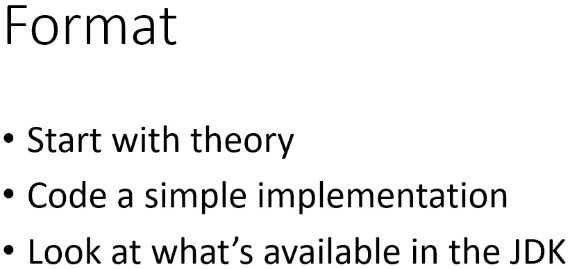
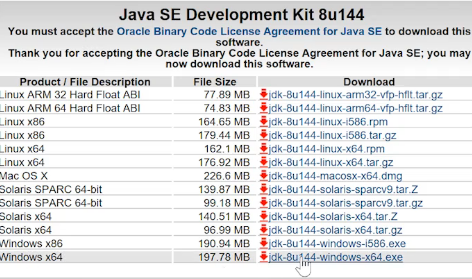
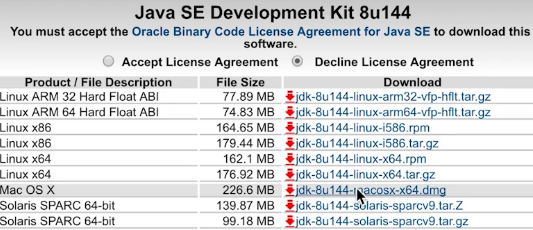
**Introduction from Tim Buchalka**  
\* Hello and welcome to the Data Structures and Algorithms Using Java course.  
\* I’ve taken over the course and I’m going to be releasing new content moving forward in this course as well as updating it.

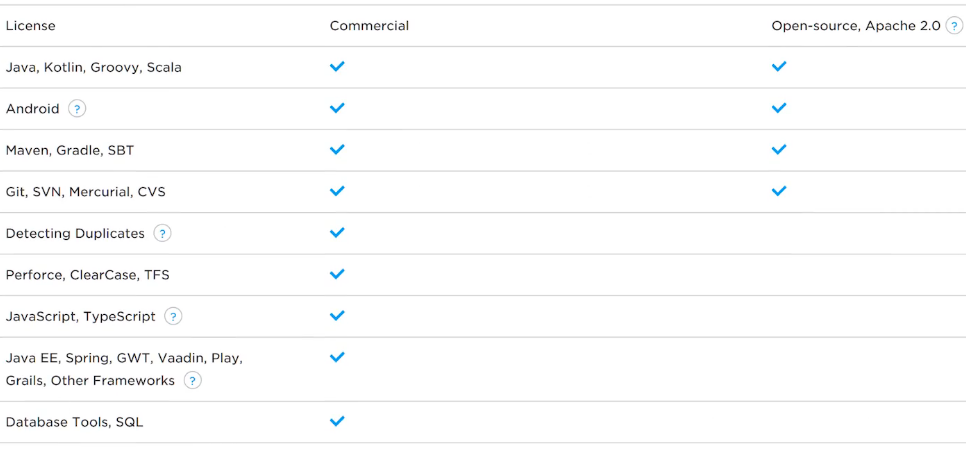
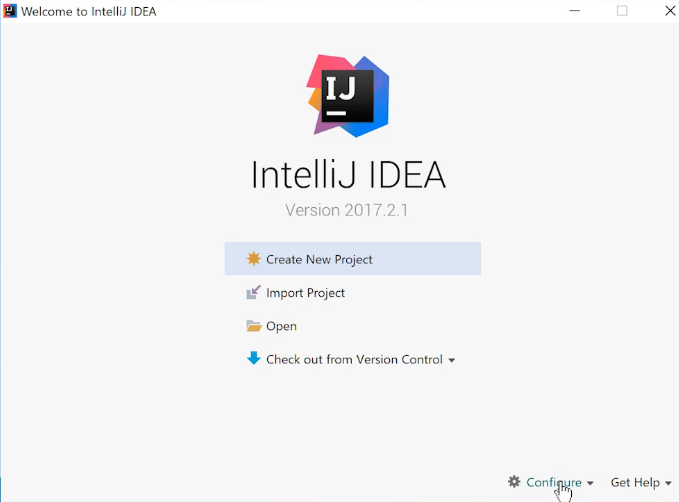
**Introduction to the Course**  
\* As a software developer, it’s important that you understand the options available to you in terms of the data structures you can use in your applications and the algorithms you can use to sort and search data.  
\* Big-O Notation is a way of comparing how different algorithms perform in an objective manner that doesn’t rely on hardware.  


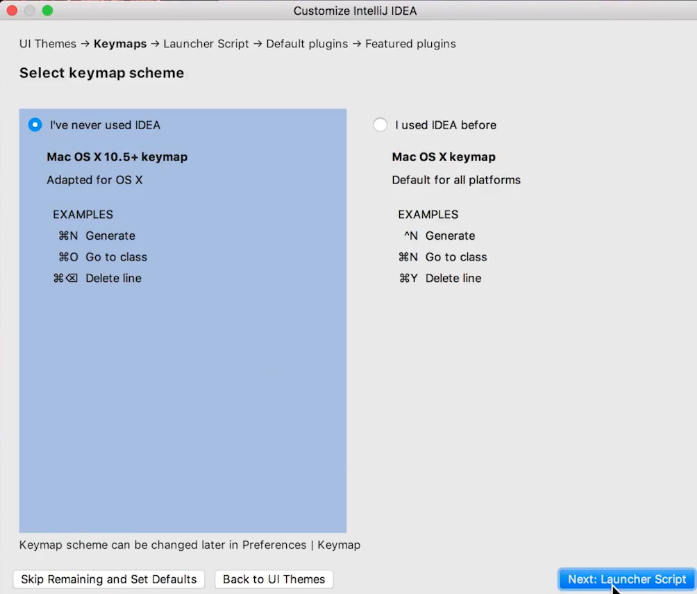
  
\* **Some of those are Abstract Data Types**.  
\* We’re going to focus mainly on Binary Search Trees in the tree section.  
  
\* We’re going to cover Linear Search and Binary Search.  
\* Many sort algorithms, for example: bubble sort, merge stort, quick sort, radix sort, insertion sort and quite a number of others.  
  
\* The way that most sections will go is we’ll start with some theory - algorithm - the steps for the algorithm, data structure - what the data structure is and what its advantages and disadvantages are and then we’ll code a simple implementation of the data structure or if it’s an algorithm, we’ll code a representative implementation. And after that, we’ll look at what’s available in the JDK because this is a for Java course and in practice you’re not going to code the algorithms and the data structures because the Java team has already done that for us most of the time.  
\* There are cases where there isn’t much available in the JDK but for the data structures we’ll be looking at, there’s good support in the JDK for most of them.  
\* And when it comes to the sort algorithms, the JDK does have methods for sorting Collections.  
\* I’ve been a software developer for over 25 years and every time I take a tech course, I always watch the videos from beginning to end because usually there’s one or two little nuggets that I pick up that I didn’t already know.  
\* So I recommend that you watch the videos on order.

**JDK8 for Windows**  
\* Let’s install the JDK - Java Development Kit which contains the software to be able to run programs.  
\* (oracle is the name of the company that own the Java product)  
**=> java.oracle.com => Technologies: Java SE => Downloads => Click on the icon => Choose 32 or 64 bit version.**  
=> Find out if you’re running 32 or 64 bit processor.  
  
\* Ignore the part to the right (x64-based processor) because it is possible confusingly to have a 64-bit processor but to be running a 32-bit version of the Windows operating system.  
  
**=> Run it and install it.**  
\* It also wants to install the JRE - Java Runtime Edition. You need to have both JDK and JRE.

**JDK8 for MAC**  
\* (oracle is the name of the company that own the Java product)  
**=> java.oracle.com => Technologies: Java SE => Downloads => Click on the icon => Choose Mac OS X**  
  
**=> Open it => Double click the pop-up to start installation => Continue => Install**

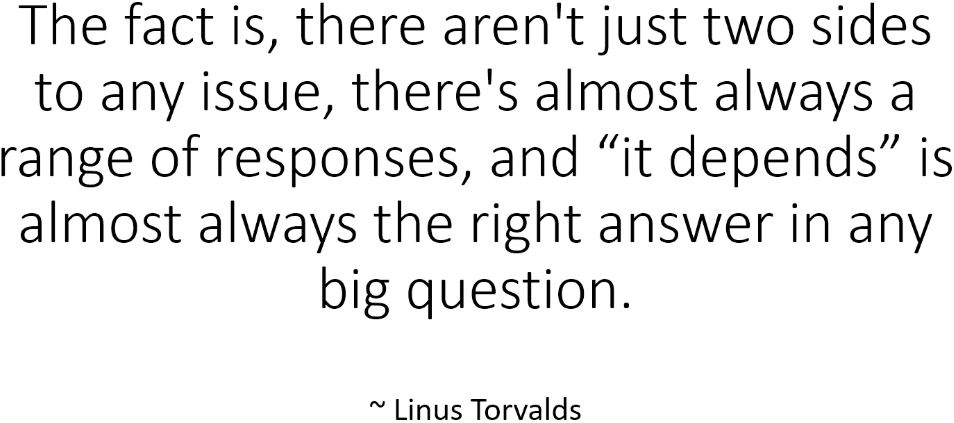
**JDK8 for Linux**  
\* We’re going to install it on Ubuntu 16.04.  
**=> Terminal => sudo apt-get update** (to update the package index)  
**=> sudo add-apt-repository ppa:webupd8team/java**  
**=> sudo apt-get update**  
**=> sudo apt-get install oracle-java8-installer => y**  
**=> sudo update-alternatives --config java**  
**=> java -version or javac or javac -version to confirm that it’s installed correctly**

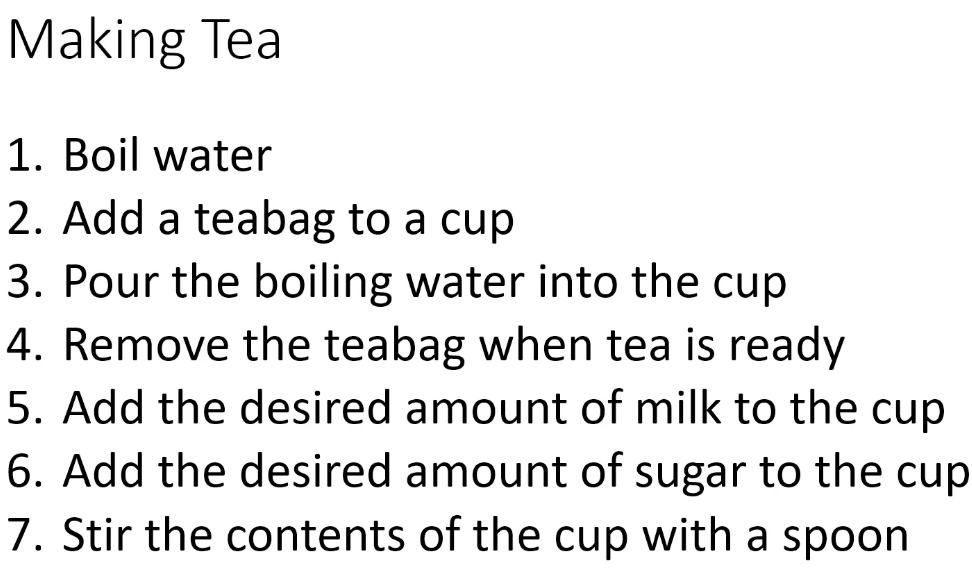
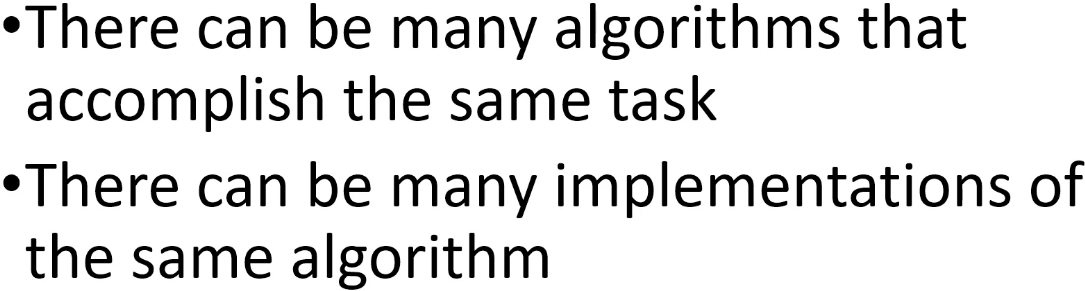
**IntelliJ for Windows**  
**=>** [**www.jetbrains.com**](http://www.jetbrains.com) **=> IDEs: IntelliJ IDEA => Download => Ultimate/Community Edition.  
=> Run the installer => 64-bit launcher => .java, .groovy, .kt (that enables those particular files to be opened up in IntelliJ)**  
\* The Ultimate Edition is the commercial product.  
\* The Community Edition is completely free.  
  
\* When you run it for the first time, it asks you about a few configuration things such as the theme.  
\* Next we need to configure the JDK.  
  
\* We need to associate the JDK with IntelliJ  
**=> Configure => Project Defaults => Project Structure => Project => New… => JDK => navigate to where you installed the JDK and select the JDK => OK**\* There are a few more things we need to configure:  
**=> Configure => Settings => Editor => General => Auto Import => tick the “Add unambiguous imports on the fly” and “Optimize importas on the fly (for current project)”**\* That’ll make your coding a little bit easier.  
**=> Appearance => Show line numbers (make sure it’s ticked, it should be by default)**

**IntelliJ for MAC**  
**=>** [**www.jetbrains.com**](http://www.jetbrains.com) **=> IDEs: IntelliJ IDEA => Download => Ultimate/Community Edition.  
=> Open it => drag it to the Applications folder**   
**=> Finder => Applications => IntelliJ IDEA CE => Open**  
**=> Create Launcher Script tick it (so that if we want to use that in the future, we can do that)**  
\* We need to associate the JDK with IntelliJ  
**=> Configure => Project Defaults => Project Structure => Project => New… => JDK => navigate to where you installed the JDK and select the JDK => OK**\* There are a few more things we need to configure:  
**=> Configure => Settings => Editor => General => Auto Import => tick the “Add unambiguous imports on the fly” and “Optimize importas on the fly (for current project)”**\* That’ll make your coding a little bit easier.  
**=> Appearance => Show line numbers (make sure it’s ticked, it should be by default)**

**IntelliJ for Linux**  
**=>** [**www.jetbrains.com**](http://www.jetbrains.com) **=> IDEs: IntelliJ IDEA => Download => Ultimate/Community Edition.  
=> Open it with Archive Manager => create a folder to which you want to put IntelliJ => drag the downloaded file in that folder  
=> Terminal => navigate to that folder and to the bin folder  
=> ./idea.sh  
=> Create Launcher Script tick it (so that if we want to use that in the future, we can do that)**

**Introduction to Data Structures**  
\* A Data Structure organizes and stores data.  
\* There are many different data structures and they differ from each other in the way that they organize and store the data.  
\* Arrays order the data sequentially and they place each value in its own slot. We can get to a slot using an index.  
\* That’s different form a Tree which is a hierarchical data structure - or some would say Abstract Data Type. Trees have the notion of parents and children.  
\* Each data structure has strengths and weaknesses. They all do some things well and other things not so well.  
\* For example:   
=> Arrays are great for random access when you know the index of the item you want to access. They’re not so great when you don’t know the index because then you have to search the dataset to find what you’re looking for.

  
\* The best data structure will depend on the data you want to store, how your application will need to access the data, the operations your application will perform the most on the data.  
\* There isn’t one size fits all answer.  
\* As a developer, you’ll have to decide which data structure to use based on your application’s specific needs.  
\* And thata’s why you’re in this course - that’s why it’s important to learn about and understand the different types of data structures, what each one does well, and what each one doesn’t do so well.  
\* Armed with that knowledge, you’ll be able to choose the best data structure for your specific scenario because it’s going to depend on what you want to do with the data and how often.  
\* You’ll be able to choose the best data structure for your specific job and the code you write will perform better because of that.  
\* A data structure organizes and stores data. That’s what a data structure is.

**Introduction to Algorithms**  
\* An Algorithm describes the steps you have to perform to accomplish a specific task.  
\* Here’s a common example but using tea instead of coffee:  
  
\* **There can be more than 1 algorithm for accomplishing a task**.  
\* This tea algorithm makes some assumptions - it assumes that we’re using a teabag and that we’re making the tea directly in the cup rather than using a teapot.  
=> **It’s not unusual for an algorithm to make assumptions**.  
\* **Some of the sort algorithms make assumptions about the data they’re sorting and some don’t**.  
\* There’s 1 really important distinction that you have to understand:   
=> **An algorithm is not an implementation**. They’re not the same thing.  
=> **An algorithm describes the steps you have to perform**.  
=> **An implementation is the code you write to perform those steps**.  
\* **There can be many implementations of the same algorithm**.  
\* **In our tea algorithm we don’t specify how to boil the water, we just say you have to boil water. And so it doesn’t matter if the water is boiled in a kettle or in a saucepan or if somebody is boiling it in a pot over a campfire. As long as you boil water, you’re performing step 1**.  
\* So in the same way when we implement an algorithm, it may be possible to implement a step in many different ways.  
\* **In this course I’m going to show you 1 implementation of each algorithm that we study, but there can be, and likely are, other implementations of the algorithm**.  
  
=> **For example many algorithms for sorting and many implementations of one sorting algorithm**.  
\* As long as the code you’re writing is following the steps described by the algorithm, you have an implementation for the algorithm.  
\* Often the words algorithm and implementation are used interchangeably even though they’re different.  
\* **So that’s algorithms, they just describe the steps you have to perform to accomplish a task**.