# **Access Modifiers in Java**

[**https://youtu.be/T632kAJ\_9VA**](https://youtu.be/T632kAJ_9VA)

# **Unified Modelling Language Diagrams**

**https://youtu.be/UI6lqHOVHic**

**Wrapper Classes**

**https://youtu.be/4MiEznM8y8Q**

**Exception Handling: try and catch**

**https://youtu.be/1XAfapkBQjk**

**Catching Multiple Exceptions in Java**

**https://youtu.be/VSGft4ppr3U**

# **Exception Handling**

**https://youtu.be/adTDlH0lhaA**

# **Java Custom Exceptions**

**https://youtu.be/OIozDnGYqIU**

**SOLID Design Principles**

# **Collections**

**Extra Resources**

Usually, our curriculum shies away from YouTube videos that feature an actual whiteboard, but this explanation is really top-notch. Pay particular attention to Pavan's discussion of the shortcomings of arrays (the third segment and beyond in the video) to understand why collections are essential:

<https://youtu.be/hKhlkx_6HeI>

**ArrayList Basics**

Extra Resources

We highly recommend watching this video. John goes over the differences between Arrays and ArrayLists and what the various declarations and constructors do. Remember, you can always watch videos like this one at a higher speed, then slow things down if something confuses you. Alternatively, you can try to follow along in your own IDE and see what happens when you recreate his examples. It's up to you and the time you have available.

https://youtu.be/NbYgm0r7u6o

## LinkedList and Memory

While an Array stores its data as one unit at one position in your computer's memory, a LinkedList stores its information by linking each data point in the list to at least one other data point in the same list. In this way, they're not bound to each other in one place in memory. While we could spend a few dozen paragraphs explaining the ins and outs of the LinkedList, it would probably be better to watch someone work with the two directly. Follow along with the examples in John's video here, then continue reading below:

https://youtu.be/5dscMs2hnDI

# **Java Iterator**

An **Iterator** is an object that can be used to loop through collections, like ArrayList and HashSet. It is called an "iterator" because "iterating" is the technical term for looping–just like your for or while loops, but with some extra abilities that can help us achieve our goals more easily. Different types of collections have different implementations of the Iterator Class, but, as far as we're concerned, they will all behave the same way by giving us access to each element in our Collection one at a time.

To get started, how about an extended analogy? Watch the following video to get a basic understanding of what an iterator is, then follow through with the rest of the lesson:

https://youtu.be/wqD4fOiGep4

## Extra Resources

If you'd like to see an example of an iterator running on a LinkedList, feel free to watch this video here. Feel free to start at the 1:50 mark, if you want to get right to the example:

https://youtu.be/Is-nu-6jrUY

**Set Interface and the HashSet**

Therefore, for an explanation of HashSet, complete with a little algorithm chat and comparisons to the ArrayLists we've already seen, watch this video here. Pro-tip: Don't forget that you can adjust the speed of YouTube videos if you find that a presenter is talking too slowly (or too quickly) for you. This video may be best watched at 1.5x speed, depending on your comfort with Collections as a topic:

https://youtu.be/WPcKwA5WF7s

## **Getting Specific: HashMap**

Let's let a fellow learner explain how HashMap works with examples. Given that Alex here used to struggle with the concept, he does a really good job of putting it in the appropriate context for new learners. For those of you still struggling with the kinds of underlines that Eclipse throws at you while you're trying to craft your code, pay attention to how he works with those, too:

https://youtu.be/70qy6\_gw1Hc

Extra Resources

If you'd like another video that reviews similar material as the above, feel free to watch John's interpretation of Map and HashMaps here:

https://youtu.be/H62Jfv1DJlU

## Creating Threads in Java

There are two ways to create a thread in Java:

* By extending the **Thread** class
* By implementing the **Runnable** interface

Here's a friendly face to walk us through creating some Java threads and a conversation about how they behave. There's even a few methods thrown in there at the end:

https://youtu.be/r\_MbozD32eo

Extra Resources

If you'd like to know more about how multithreading works on a theoretical level, feel free to check out this blog here.

For another walkthrough and a different perspective on creating threads, feel free to check out this video here:

https://youtu.be/D\_8Tuq8PQgw

**Creating a Thread by Extending the Thread Class**

**For more information, check out the first few minutes of the video below. The discussions of priority and daemon threads are beyond what we're trying to do right now, so feel free to stop watching around the 3:20 mark unless you really want to see more:**

**https://youtu.be/a\_LBuCx1KTE**

# **Testing in Java: JUnit**

**https://youtu.be/vZm0lHciFsQ**

# **Understanding Search Algorithms**

**Computers store vast amounts of data, which is nice, but the true strength of having all of that data is being able to quickly find what you need.**

**https://youtu.be/DHLCXXX1OtE**

## **Binary Search**

**https://youtu.be/xrMppTpoqdw**

## **Selection Sort**

**https://youtu.be/g-PGLbMth\_g**

## **Insertion Sort**

**https://youtu.be/JU767SDMDvA**

## **Merge Sort**

**https://youtu.be/4VqmGXwpLqc**

**Bubble Sort Algorithm**

**https://youtu.be/nmhjrI-aW5o**

## **Quick Sort**

**https://youtu.be/XE4VP\_8Y0BU**

# **Generics in Java**

**https://youtu.be/jUcAyZ5OUm0**

**https://youtu.be/K1iu1kXkVoA**

# **What is Recursion?**

**https://youtu.be/V2S\_8E\_ubBY**

**Functional Interfaces and Lambda Expressions**

**https://youtu.be/4HC\_WyBSDGA**

**https://youtu.be/Gs8ZPKCFlTc**

**Data and Databases Demystified: Essentials**

**https://youtu.be/Tk1t3WKK-ZY**

**https://youtu.be/W2Z7fbCLSTw**

**Database Management System (DBMS)**

**https://youtu.be/f2M-dodlISQ**

**Relational Databases (RDBMS)**

**https://youtu.be/Q45sr5p\_NmQ**

# **Method References**

**https://youtu.be/UIJLegcCFyw**

# **Java Streams**

**https://youtu.be/FWoYpM-E3EQ**

## **Extra Resources**

If you'd like to see some more examples of Stream sources, check out [this blog here](https://www.swtestacademy.com/java-streams-sources/).

**https://youtu.be/tklkyVa7KZo**

**https://youtu.be/G\_bxZgtRlK4**

**https://youtu.be/Q93JsQ8vcwY**

**Requirements Analysis**

**https://youtu.be/fc-5HJPBZMQ**

**Data Models**

**Extra Resources:We highly recommend this video for a walkthrough of the various models:**

**https://youtu.be/OwQoj3GJfNY**

**Introduction to SQL**

**https://youtu.be/nWyyDHhTxYU**

**https://youtu.be/OfM5lC-7R4Y**

# **Keys in SQL**

**Extra Resources: Primary keys can also be explained in terms of "Functional Dependencies." Here's a quick video explaining how that definition works. (Don't forget that you can watch videos at 1.5x speed, if that makes the experience more useful to you.)**

**https://youtu.be/X3c1phdyjbU**