A Dry Sense of Humor Dripping with Sarcasm

(3 PAGES – ~15 MINUTE READ)

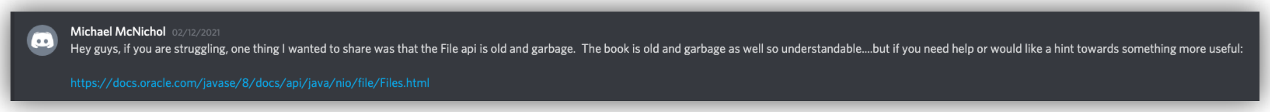
A few weeks ago I made a cutting comment on the quality of the book and this course….it was a total jerk move of me to say something callously without explaining why.

I figured I'd throw this together to provide a more *fair* or *complete* argument of why I made that statement. That being said, you didn't take this course to hear *my thoughts* but to get taught by someone who's job is to teach this material. This has nothing to do with passing the class and everything to do with what I've experienced in an industry that has money bursting at the seams whether you are terrible or amazing at software development.

Rest assured I've brought these points up to this professor, the co-chairs, and even the Board of Trustees. It's not that I feel people (or these people specifically) don't care nor that this is an issue with CoD specifically as it occurs most places. It is just that most people wake up with 30 things they know they need to do, 10 things they want to do and 1 or 2 things that are on fire. By the end of the day, most are lucky if they complete 3 of anything in those categories. We live in a world where systems that vie for our attention and time in addition to an opinion of how spend it…these systems are very effective at getting what they want.

In combatting these very effective systems, finding time to transition from *knowing* what things should be versus *doing* what it should be are worth their weight in gold. I will take a job alone on the principle that the leaders understand this and it's what helped me transition from ok money to google money in a matter of years.

So here's the comment:



Michael McNichol 02/12/2021
Hey guys, if you are struggling, one thing I wanted to share was that the File api is old and garbage. The book is old and garbage as well so understandable....but if you need help or would like a hint towards something more useful:

https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html

Hey guys, if you are struggling, one thing I wanted to share was that the File api is old and garbage. The book is old and garbage as well so understandable....but if you need help or would like a hint towards something more useful:

https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html

First the java.io.File API

Is the File API old and garbage...the answer is it depends. If you are building a simple File object and are just teaching or learning a simple concept, then no of course not. It's the salt of the earth and created the world we live and love today. Because it's definitely old and has been with us since 1996…the birth of Java. So as old as Java is, that's as old as the java.io.File API is. Just like Java is still useful, so is the File API, no argument there.

But a File is a File is a File...has that really changed?

In most ways, you are right, no files haven't really changed. Most operating systems leverage everything as if it were a file. We load their data into memory and write it to a hard-drive as we change and update things but they are all just streams of bits that we write to output devices as files most often in chunks of 16, 32, and 64 bytes.

So you got me there, I concede that point.

It's not what you do, it's how you do it.

What has changed though is how we work with files. The original **java.io.File** API is used in a ton of older projects, frameworks, and libraries. It's old but it's not going anywhere and likely will never be deprecated. I'd go as far as to say we'd likely see Java the language itself be replaced before we see the File API replaced. A scary thought...I know.

Don't worry though, the bear is still devouring Perl, PHP, Fortran, and Cobol….we've got time. /s

The way we use files is about as human as it gets….we want more, we want it all, and we want it now. We have transitioned in the past ten years to leveraging big data, neural nets, data lakes, and massive amounts of information being fed into CNNs, gradient descent algorithms, and simple linear regression or Bayesian models to improve decision making for systems in an automated way. Do these all use files? No, but they use streams of data and plenty of them are files.

Hadoop was a massive impact to the big data world. This was based off of Googles MapReduce whitepaper….you know the thing that helped them scrape the entire internet and index it and is now one of the largest advertising revenue generators on the planet. Hadoop was based on that paper and is working on….surprise surprise….file streams and MapReduce. It's a first class citizen in the world of Java. I built a fraud detection system using this that was purely automated working with data scientists that took a job from four days of processing run-time to eight hours. Just a bunch of processing crunching files. Fraud in the insurance space is 100's of millions of dollars in revenue potentially lost. If you can efficiently crunch things in this space, you will be set for life.

So what is Java doing about being more efficient with Files?

More like, what did they do about it ten years ago. As of 2011, they came out with a new File API (**java.nio.File**). It does everything that **java.io.File** does but better. The simple nail in the coffin is the fact this uses Buffering where **java.io.File** doesn't.

Quote me now…if this professor doesn't at some point go on some rant about buffering and the performance improvements and how stupid we must be to not leverage it when everything in the world uses a buffer (aka cache)….hard drives, cpu's, your phone, web-pages….if he doesn't lose his marbles over us doing something like this I'll eat my hat.

java.nio.File supports symlink traversal, file attributes, metadata support, ACLs, and more. These may not seem interesting now but when you start dealing with security controls, infinite symlink recursion (which is what caused me enough frustration to make that comment in Discord), and enterprise development…it's what signals how much you can offer a team as a valuable member.

Imagine being a mechanic and knowing what an engine is but not understanding how it works. There will be some people who argue you aren't a mechanic then when held up to the light.

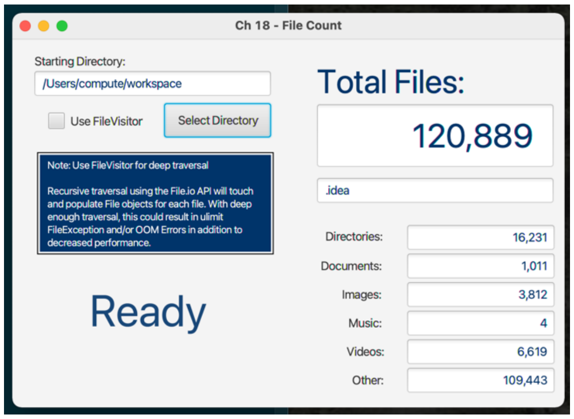
Whether you agree with that point or not, I hope we can agree that there is a level of understanding associated to learning that justifies claiming knowledge of a thing.

Java NIO is also non-blocking. The older Java IO's various streams are blocking. That means, that when a thread invokes a **read( )** or **write( )**, that thread is blocked until there is some data to read, or the data is fully written. The thread can do nothing else in the meantime. If you make the mistake in JavaFX of not leveraging **Platform.runLater( )** so as not to use the JavaFX UI thread, this means you end up with the spinning wait icon until all your work is done.

Not the end of the world but I'm going to guess people solved this by just doing shorter traversals or simply not thinking about it.

Here's the deep dive comparison from the horses mouth written in 2009.

By working around the limitations of the **java.io.File** constraints and leverage a user interface that realtime updated all files, file types, and directory counts as the application walked the file tree. Even if you jump to the root directory, you won't get hit with **`java.lang.OutOfMemoryError`** or **`OutOfMemoryError: Too many open Files`**



Ch 18 - File Count

Starting Directory: /Users/compute/workspace

Use FileVisitor

Note: Use FileVisitor for deep traversal

Recursive traversal using the File.io API will touch and populate File objects for each file. With deep enough traversal, this could result in ulimit FileException and/or OOM Errors in addition to decreased performance.

Ready

Total Files: 120,889

.idea

Directories:	16,231
Documents:	1,011
Images:	3,812
Music:	4
Videos:	6,619
Other:	109,443

Summary:

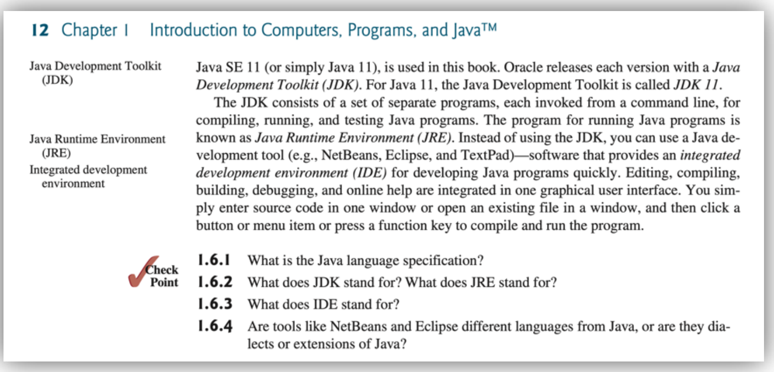
For a course on Collections in Java, I'd say understanding how a file works and simple mechanics is a pretty low bar. For a book that claims to be a ***Comprehensive*** introduction to Java that doesn't mention something that's been around at least 7 years prior to it being written (updated), I'd say that's an embarrassment. Look for yourself. **java.nio.File** really isn't discussed.

There is an update to this book (12th edition written in 2019) where they did mention **java.nio.File**….in an example……twice. Hardly worth the value it provides.

Mind you we are using the 11th edition book from 2017 that doesn't even cover JDK 11 which was made the LTS Standard several years ago.

Does the 12th version cover JDK 11….why yes it does.

College books are a business at the end of the day but that is an entirely different conversation/rant that gets me fired up.



12 Chapter 1 Introduction to Computers, Programs, and Java™

Java Development Toolkit (JDK)

Java SE 11 (or simply Java 11), is used in this book. Oracle releases each version with a *Java Development Toolkit (JDK)*. For Java 11, the Java Development Toolkit is called *JDK 11*.

Java Runtime Environment (JRE)

Integrated development environment

The JDK consists of a set of separate programs, each invoked from a command line, for compiling, running, and testing Java programs. The program for running Java programs is known as *Java Runtime Environment (JRE)*. Instead of using the JDK, you can use a Java development tool (e.g., NetBeans, Eclipse, and TextPad)—software that provides an *integrated development environment (IDE)* for developing Java programs quickly. Editing, compiling, building, debugging, and online help are integrated in one graphical user interface. You simply enter source code in one window or open an existing file in a window, and then click a button or menu item or press a function key to compile and run the program.

Check Point

1.6.1 What is the Java language specification?

1.6.2 What does JDK stand for? What does JRE stand for?

1.6.3 What does IDE stand for?

1.6.4 Are tools like NetBeans and Eclipse different languages from Java, or are they dialects or extensions of Java?

If you are looking to just pass this class, don't listen to a word that I say. You won't need it and it's a waste of your time. I'm coming at this for a completely different reason. If you want to talk more about **THAT**…well so do I. Hit me up.