Communicating outside of the JVM using resources

Resources can be files, network connections, database connections, streams, or sockets.

We use these resources to interact with file systems, networks and databases, to exchange information.

Exception Handling

When you're dealing with external resources, exception handling becomes much more important.

What really happens when you open a file from a Java application?

When you instantiate one of Java's file access classes, Java will delegate to the operating system to open a file from the OS File System.

This then performs some or all of the following steps.

- First, it checks if the file exists.
- If the file exists, it next checks if the user or process has the proper permissions for the type of access being requested.
- If this is true, then file metadata is retrieved, and a file descriptor is allocated. This descriptor is a handle to the opened file.

What really happens when you open a file from a Java application?

- An entry is made in the file table or file control block table of the operating system, to track the opened file.
- The file may be locked.
- The file may be buffered by the OS, meaning memory is allocated, to cache all or part of the file contents, to optimize read and write operations.

Closing an open File Resource

Many of the methods in Java make opening a file look just like instantiating another object.

You don't have to call open on a file, so it's easy to forget that you've really opened a resource.

Closing the file handle will free up the memory used to store any file related data, and allow other processes to access the file.

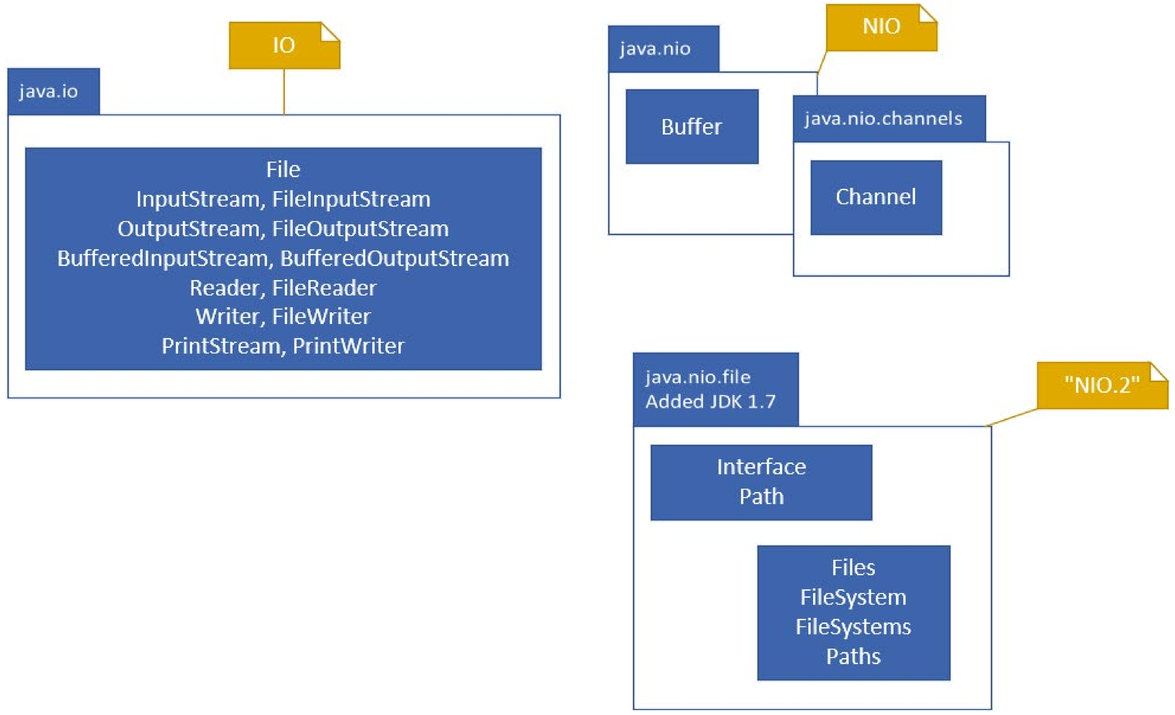
Fortunately, most of the Java classes you'll use to interact with files, also implement an AutoCloseable interface, which makes closing resources seamless.

IO, NIO, NIO.2

Java has what seems like a very confusing and large series of classes, in many packages, to support input/output.

I'll be covering the most commonly used types, but first let's talk about Java's somewhat confusing terminology which includes IO, NIO and NIO.2.

First, IO was a term for input/output, and java.io is a package that contains the original set of types, which support reading and writing data from external resources.



IO, NIO, NIO.2

NIO was introduced as Non-blocking IO, with the java.nio package in Java 1.4, as well as a few other related packages. The communication with resources is facilitated through specific types of Channels, and the data stored in containers called Buffers, when exchanged.

NIO.2 stands for New IO, and is a term that came into being with Java 1.7, emphasizing significant improvements to the java.nio package, most importantly the java.nio.file package and its types.