Java NIO consist of the following core components:

* Channels
* Buffers
* Selectors

Java NIO has more classes and components than these, but the Channel, Buffer and Selector forms the core of the API, in my opinion. The rest of the components, like Pipe and FileLock are merely utility classes to be used in conjunction with the three core components. Therefore, I'll focus on these three components in this NIO overview. The other components are explained in their own texts elsewhere in this tutorial. See the menu at the top corner of this page.

**Channels and Buffers**

Typically, all IO in NIO starts with a Channel. A Channel is a bit like a stream. From the Channel data can be read into a Buffer. Data can also be written from a Buffer into a Channel. Here is an illustration of that:

|  |
| --- |
| Java NIO: Channels and Buffers |
| **Java NIO: Channels read data into Buffers, and Buffers write data into Channels** |

There are several Channel and Buffer types. Here is a list of the primary Channel implementations in Java NIO:

* FileChannel
* DatagramChannel
* SocketChannel
* ServerSocketChannel

As you can see, these channels cover UDP + TCP network IO, and file IO.

There are a few interesting interfaces accompanying these classes too, but I'll keep them out of this Java NIO overview for simplicity's sake. They'll be explained where relevant, in other texts of this Java NIO tutorial.

Here is a list of the core Buffer implementations in Java NIO:

* ByteBuffer
* CharBuffer
* DoubleBuffer
* FloatBuffer
* IntBuffer
* LongBuffer
* ShortBuffer

These Buffer's cover the basic data types that you can send via IO: byte, short, int, long, float, double and characters.

Java NIO also has a MappedByteBuffer which is used in conjunction with memory mapped files. I'll leave this Buffer out of this overview though.

**Selectors**

A Selector allows a single thread to handle multiple Channel's. This is handy if your application has many connections (Channels) open, but only has low traffic on each connection. For instance, in a chat server.

Here is an illustration of a thread using a Selector to handle 3 Channel's:

|  |
| --- |
| Java NIO: Selectors |
| **Java NIO: A Thread uses a Selector to handle 3 Channel's** |

To use a Selector you register the Channel's with it. Then you call it's select() method. This method will block until there is an event ready for one of the registered channels. Once the method returns, the thread can then process these events. Examples of events are incoming connection, data received etc.