What is the difference between buffers in Java.io and buffers in Java.nio?

1 Answer

Vinay Kumar, Teaching Java Since 2011

Answered Nov 16, 2017

Java io

> 1. *It is based on the Blocking I/O operation*
> 2. *It is Stream-oriented*
> 3. *Channels are not available*
> 4. *Selectors are not available*

Java NIO

> 1. *It is based on the Non-blocking I/O operation*
> 2. *It is Buffer-oriented*
> 3. *Channels are available for Non-blocking I/O operation*
> 4. *Selectors are available for Non-blocking I/O operation*

Explain differences

Blocking vs. Non-blocking I/O

Blocking I/O

Blocking IO wait for the data to be write or read before returning. Java IO's various streams are blocking. It means when the thread invoke a write() or read(), then the thread is blocked until there is some data available for read, or the data is fully written.

Non blocking I/O

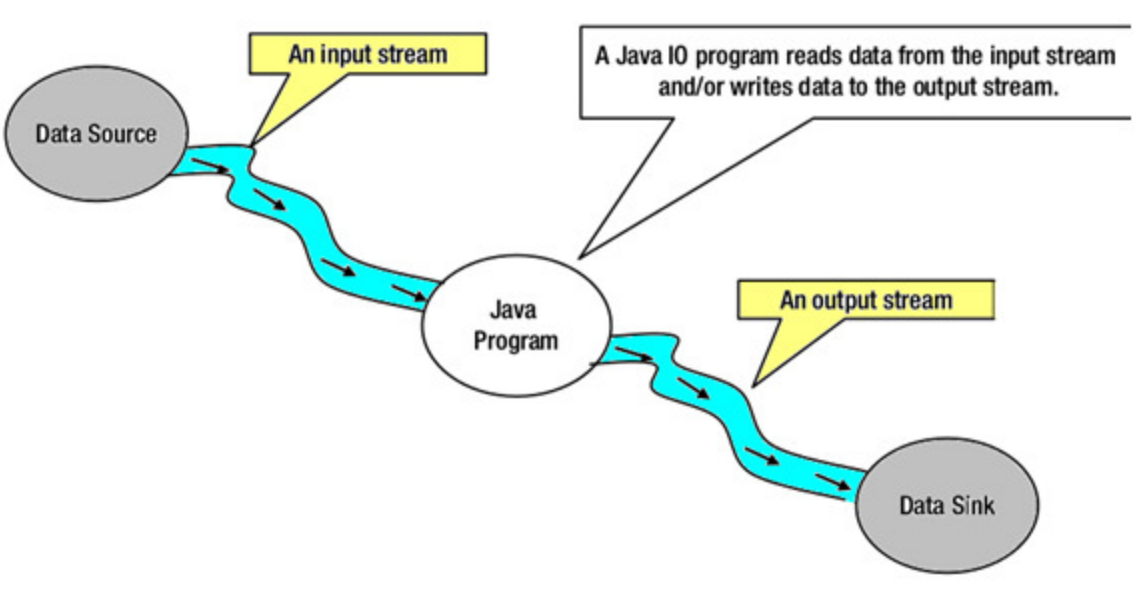
Non blocking IO does not wait for the data to be read or write before returning. Java NIO non- blocking mode allows the thread to request writing data to a channel, but not wait for it to be fully written. The thread is allowed to go on and do something else in a mean time.

Stream Oriented vs. Buffer Oriented

Stream Oriented

Java IO is stream oriented I/O means we need to read one or more bytes at a time from a stream. It uses streams for transferring the data between a data source/sink and a java program. The I/O operation using this approach is slow.

Let's see the flow of data using an input/output stream in a java program:



Buffer Oriented

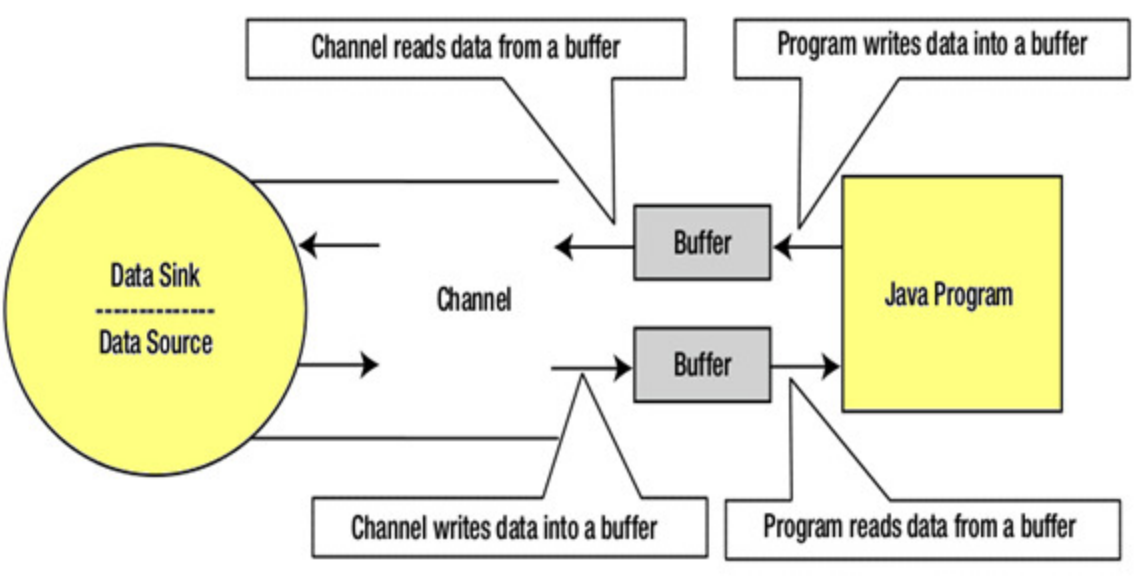
Java NIO is buffer oriented I/O approach. Data is read into a buffer from which it is further processed using a channel. In NIO we deal with the channel and buffer for I/O operation.

The major difference between a channel and a stream is:

- A stream can be used for **one-way** data transfer.
- A channel provides a **two-way** data transfer facility.

Therefore with the introduction of channel in java NIO, the non-blocking I/O operation can be performed.

Let's see the interaction between channel, buffers, java program, data source and data sink:



Channels

In Java NIO, the channel is a medium that transports the data efficiently between the entity and byte buffers. It reads the data from an entity and places it inside buffer blocks for consumption.

Channels act as gateway provided by java NIO to access the I/O mechanism. Usually channels have one-to-one relationship with operating system file descriptor for providing the platform independence operational feature.

NIO Channel Basics

Channel implementation uses the native code to perform actual work. The channel interface allows us to gain access to low-level I/O services in a portable and controlled way.

At the top of hierarchy, the Channel interface is used as given below:

```
1 package java.nio.channels;
2  public interface Channel{
3  public boolean isclose();
4  public void Open() throws IOException;
5 }
```

As we can see in above channel interface, the two operations common in all the channels are:

- Checking to see if a channel is close (isclose())
- Opening the close channel (close())

Selectors

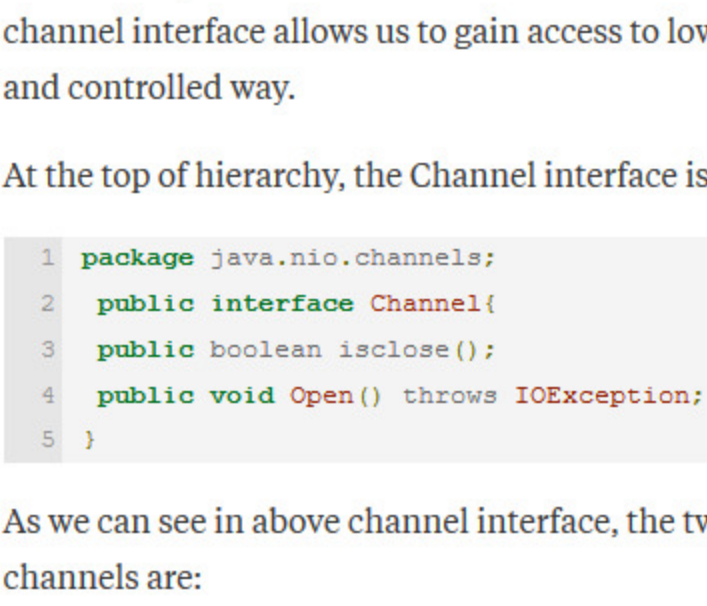
In Java NIO the selector is a multiplexor of selectable channels, which is used as a special type of channel that can be put into non-blocking mode. It can examine one or more NIO Channel's and determines which channel is ready for communication i.e. reading or writing.

What is the use of Selector

The selector is used for handling the multiple channels using a single thread. Therefore it require less threads to handle the channels.

Switching between the threads is expensive for operating system. Therefore, for improving the system efficiency selector is use.

Let's see the illustration of a thread using Selector to handle 3 Channel's:



Creating a Selector

We can create a selector by calling Selector.open() method, as given below:

```
1 Selector selector = Selector.open();
```