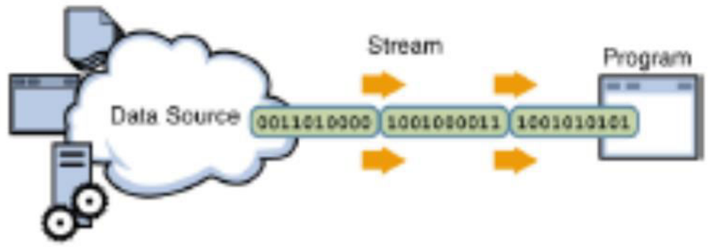
Unit-9

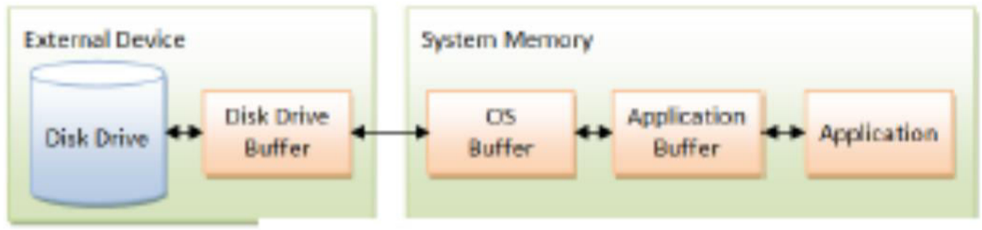
Nonblocking I/O

1	An Example Client	14	Channels
2	An Example Server	15	SocketChannel
3	Buffers	16	ServerSocketChannel
4	Creating Buffers	17	The Channels Class
5	Filling and Draining	18	Asynchronous Channels (Java 7)
6	Bulk Methods	19	Socket Options (Java 7)
7	Data Conversion	20	Readiness Selection
8	View Buffers		The Selector Class
9	Compacting Buffers	14	Channels
10	Duplicating Buffers	15	SocketChannel
11	Slicing Buffers	16	ServerSocketChannel
12	Marking and Resetting	17	The SelectionKey Class
13	Object Methods		

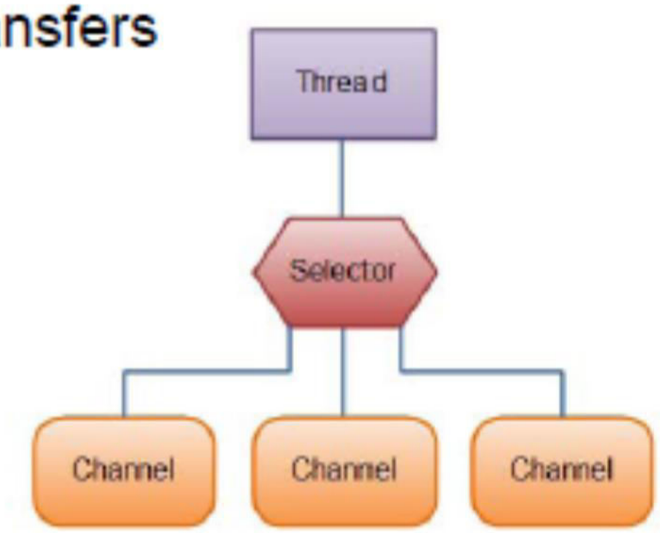
Unit-9

Java I/O





- Two typical I/O models
  - Stream-oriented I/O
    - Movement of single bytes, one at a time
    - Byte streams and character streams
    - Simple
  - Block-oriented I/O
    - Dealing with data in blocks, especially for bulk data transfers
      - A low-level data transfer mechanism
    - Channels and buffers
    - Faster



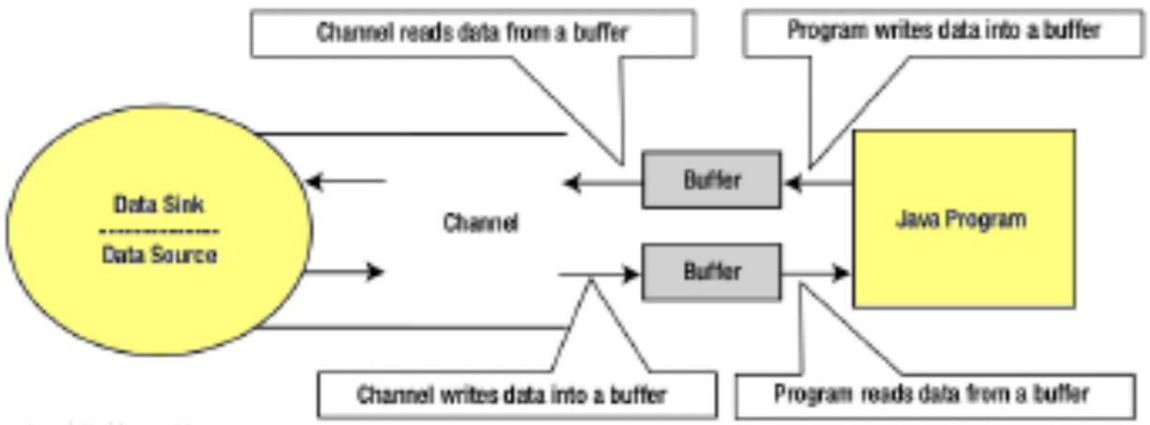
- Java I/O
  - Original I/O package: java.io.*
  - New I/O package: NIO (JDK 1.4+)
    - Channels and Buffers
      - Data is always read from a channel into a buffer, or written from a buffer to a channel
    - **Non-blocking I/O**
      - After asking a channel to read data into a buffer, a thread can do something else while the channel reads data into the buffer
    - Selector: an object that can monitor multiple channels for events
      - A thread can monitor multiple channels for data

Unit-9

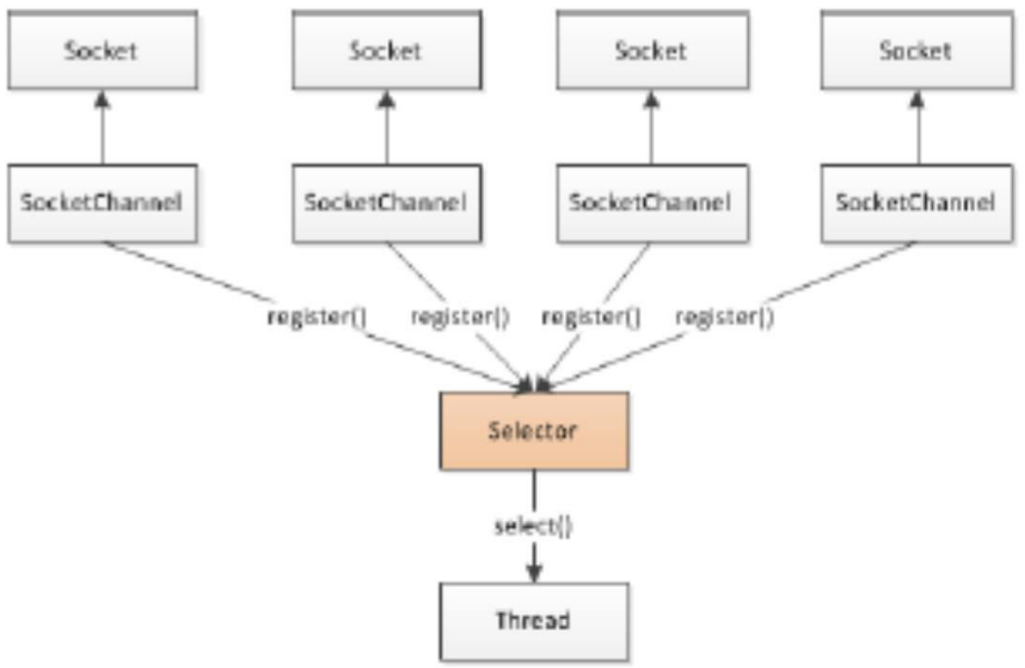
java.nio Package

http://docs.oracle.com/javase/8/docs/api/java/nio/package-summary.html

- Central abstractions of the NIO APIs
  - **Buffers**: containers for for a fixed amount of data of a specific primitive type
    - ByteBuffer (MappedByteBuffer), CharBuffer
    - ShortBuffer, IntBuffer, LongBuffer
    - FloatBuffer, DoubleBuffer
  - **Channels**: represent connections to entities capable of performing I/O operations
    - FileChannel, DatagramChannel, SocketChannel, ServerSocketChannel
  - **Selectors** and selection keys, which together with selectable channels: define a multiplexed, non-blocking I/O facility
  - **Charsets** and their associated decoders and encoders: translate between bytes and Unicode characters



Interaction between a channel, buffers, a Java program, a data source, and a data sink



Unit-9

NIO

In this Section, we are going to take a look at...

- Java NIO structure
- JNIO Buffers
- JNIO scatter and gather
- JNIO transfer
- JNIO select
- JNIO Socket

Unit-9

NIO

Java NIO Building Blocks

- Channels and Buffers
- Data always travels from a channel to a Buffer or from a Buffer to a channel

Unit-9

NIO

Transfer is Asynchronous

- Prepare data in Buffer
- Setup a channel
- Ask NIO to transfer the data
- Go on computing and check later that the data was transferred

Unit-9

NIO

Transfer is Asynchronous

- Prepare empty Buffer
- Setup a channel
- Ask NIO to transfer the data from the channel to the Buffer
- Go on computing and check later that the data was transferred and use the data happily

Unit-9

NIO

Monitor Multiple Channels Using Selectors

- Using selectors a single thread may control several channels
- Whenever one is finished with the transfer the thread can attend to the result
- When there are more the thread can attend one after the other
- When there is none the thread can go on computing

Unit-9

NIO

Channel

Unit-9

NIO

Java NIO Channel

- Read and write
- Channels always read from Buffer or write to a Buffer

Unit-9

NIO

Type of Channels

- Java NIO defines the following channels
- FileChannel
- DatagramChannel
- SocketChannel
- ServerSocketChannel

Unit-9

NIO

How to Use Channels

- Create a channel
- Read from the channel to the buffer

```
public static void main(String[] args) throws IOException {
    RandomAccessFile sampleFile = new RandomAccessFile( name: "sample.txt",  mode: "rw");
    FileChannel channel = sampleFile.getChannel();
    ByteBuffer buf = ByteBuffer.allocate(10);
    int nrBytes = channel.read(buf);
    while (nrBytes != -1) {
        System.out.println("Read " + nrBytes);
        buf.flip();
        while (buf.hasRemaining()) {
            System.out.print((char) buf.get());
        }
        buf.clear();
        nrBytes = channel.read(buf);
    }
    sampleFile.close();
}
```

Unit-9

NIO

Buffer

Unit-9

NIO

Java NIO Buffer

- Buffers store the data to be read by a channel or to store the data that comes from a channel
- Buffer has position, limit and capacity

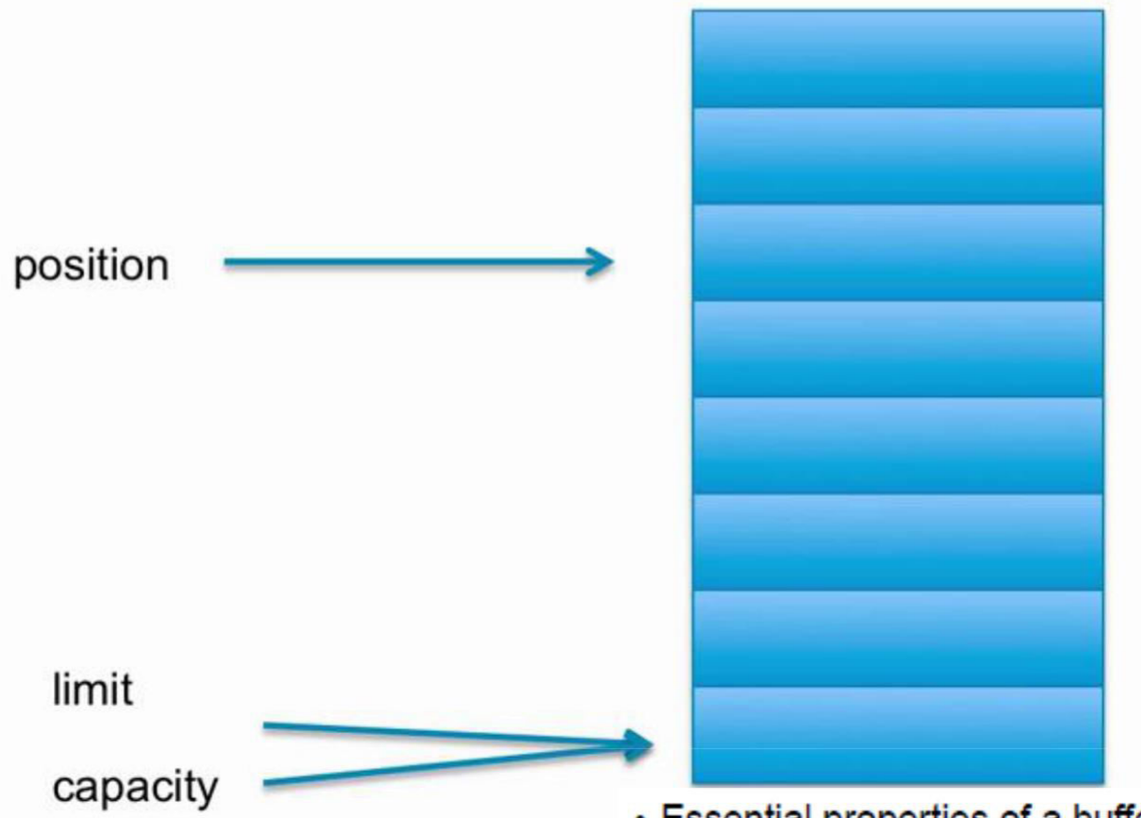
- Essential properties of a buffer
  - **Capacity**: the number of elements it contains
    - Specified when the Buffer is constructed and cannot be changed (similar to an array)
    - Never changes
  - **Limit**: specifies the current occupancy (valid data in the range of 0 to limit-1)
    - Never greater than its capacity
  - **Position**: the index of the next element to be read or written
    - Never greater than its limit

Unit-9

NIO

Position, Limit, and Capacity

- Write mode



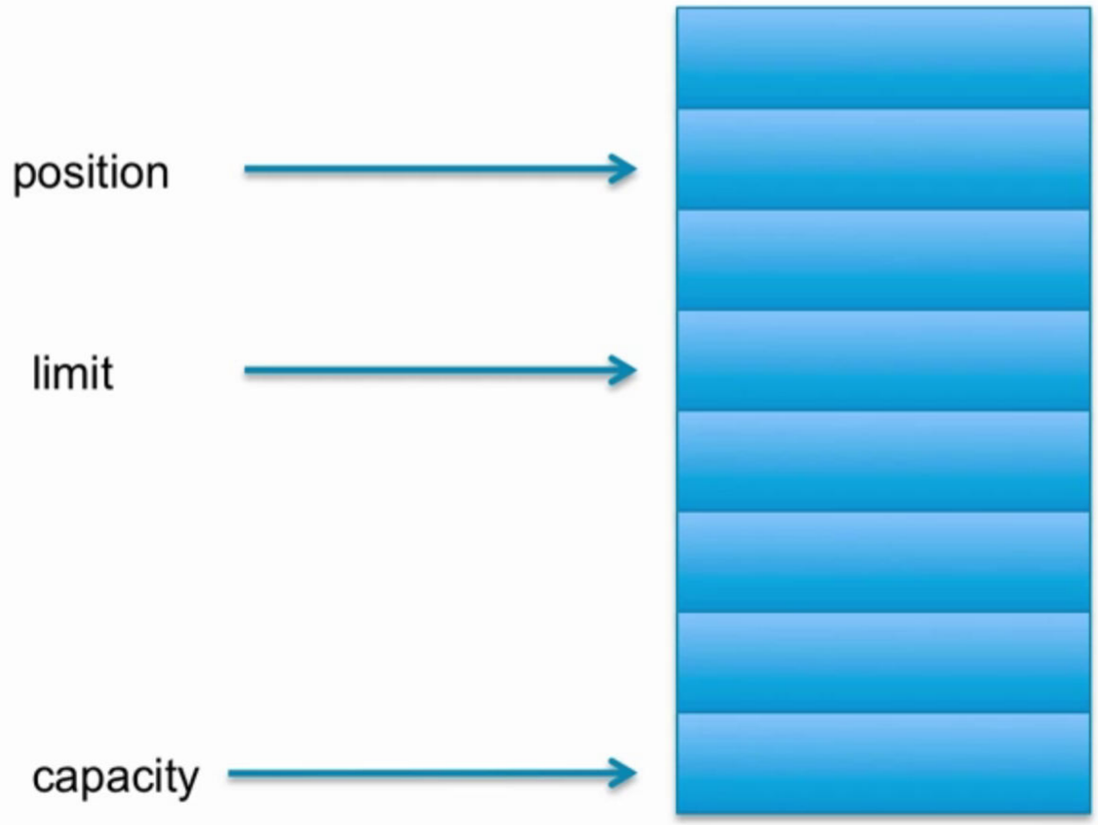
- Essential properties of a buffer
  - **Capacity**: the number of elements it contains
    - Specified when the Buffer is constructed and cannot be changed (similar to an array)
    - Never changes
  - **Limit**: specifies the current occupancy (valid data in the range of 0 to limit-1)
    - Never greater than its capacity
  - **Position**: the index of the next element to be read or written
    - Never greater than its limit

Unit-9

NIO

Position, Limit, and Capacity

- Read mode



Unit-9

NIO

Type of Buffers That Can Be Used

- ByteBuffer
- CharBuffer
- DoubleBuffer
- FloatBuffer
- IntBuffer
- LongBuffer
- ShortBuffer

Unit-9

NIO

How to Use Buffers

- Allocate Buffer

```
public static void main(String[] args) throws IOException {
    RandomAccessFile sampleFile = new RandomAccessFile( name: "sample.txt",  mode: "rw");
    FileChannel channel = sampleFile.getChannel();
    ByteBuffer buf = ByteBuffer.allocate(10);
    int nrBytes = channel.read(buf);
    while (nrBytes != -1) {
        System.out.println("Read " + nrBytes);
        buf.flip();
        while (buf.hasRemaining()) {
            System.out.print((char) buf.get());
        }
        buf.clear();
        nrBytes = channel.read(buf);
    }
    sampleFile.close();
}
```

Unit-9

NIO

How to Use Buffers

- Allocate Buffer
- Write data to the Buffer from a channel
- Flip the Buffer
- Read the data from the Buffer

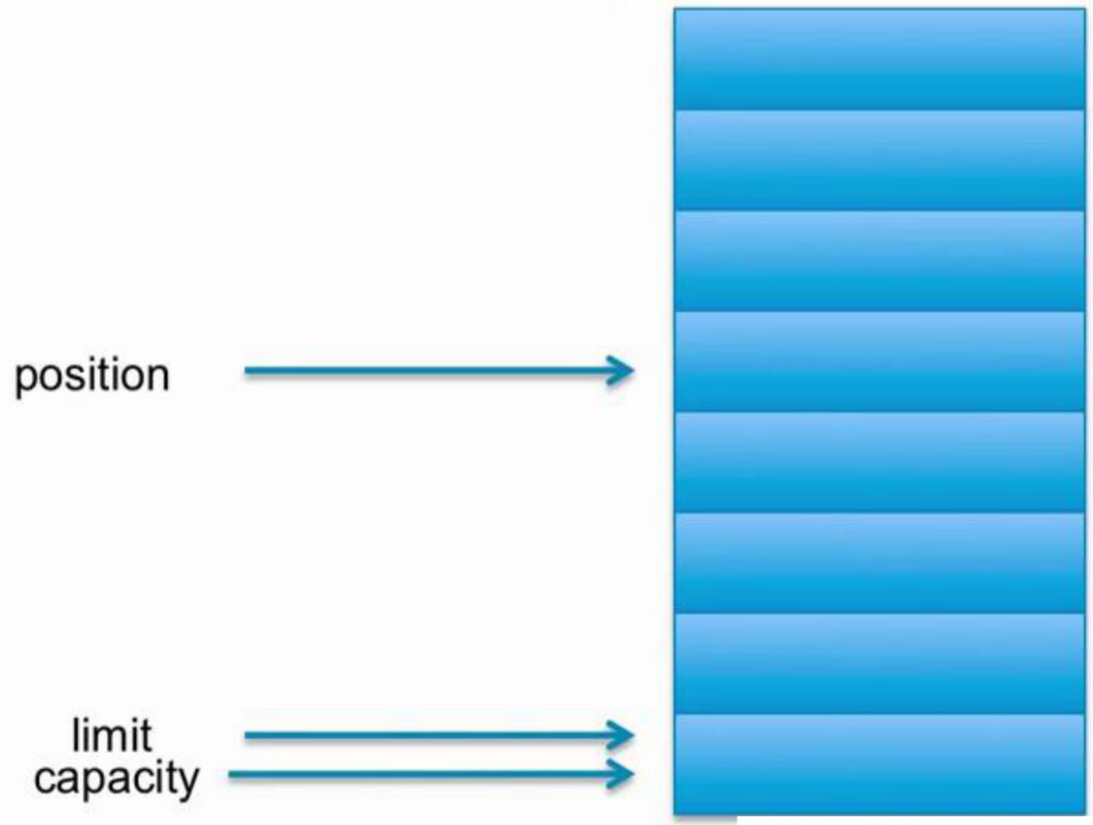
```
public static void main(String[] args) throws IOException {
    RandomAccessFile sampleFile = new RandomAccessFile( name: "sample.txt",  mode: "rw");
    FileChannel channel = sampleFile.getChannel();
    ByteBuffer buf = ByteBuffer.allocate(10);
    int nrBytes = channel.read(buf);
    while (nrBytes != -1) {
        System.out.println("Read " + nrBytes);
        buf.flip();
        while (buf.hasRemaining()) {
            System.out.print((char) buf.get());
        }
        buf.clear();
        nrBytes = channel.read(buf);
    }
    sampleFile.close();
}
```

Unit-9

NIO

Flipping the Buffer

- Switches from write-to-buffer mode to read-from-buffer



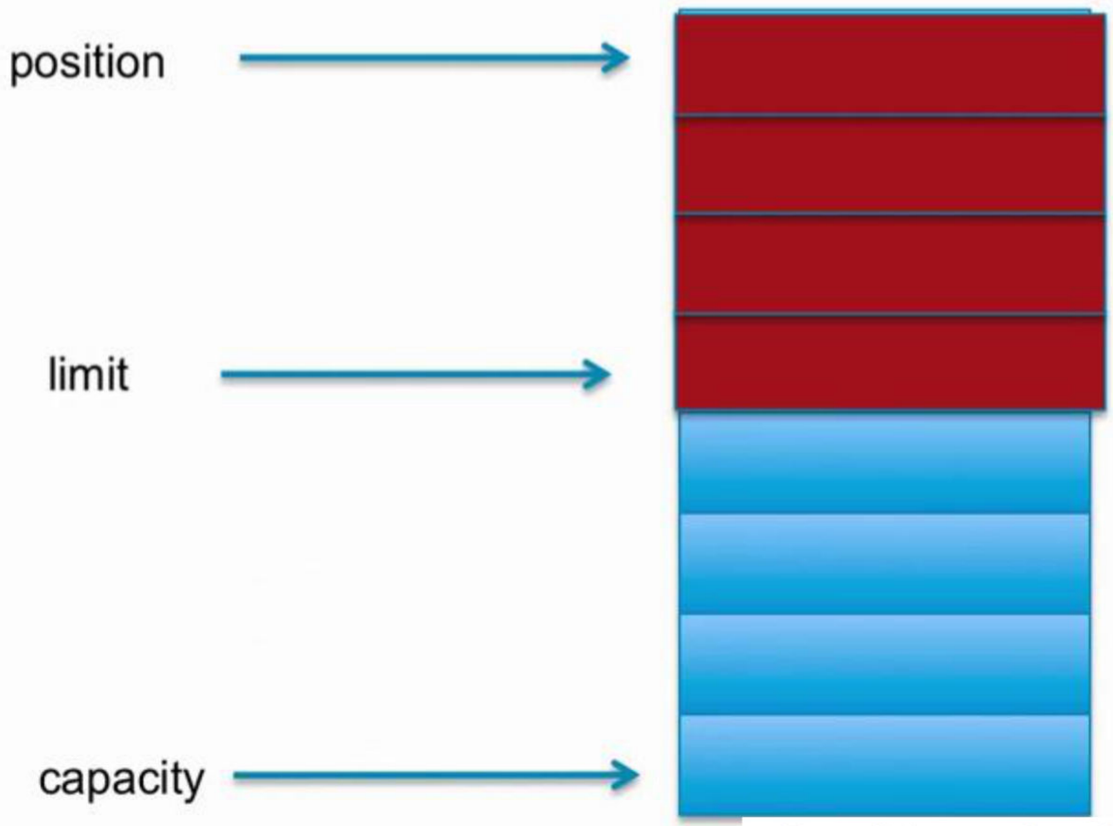
- Essential properties of a buffer
  - **Capacity**: the number of elements it contains
    - Specified when the Buffer is constructed and cannot be changed (similar to an array)
    - Never changes
  - **Limit**: specifies the current occupancy (valid data in the range of 0 to limit-1)
    - Never greater than its capacity
  - **Position**: the index of the next element to be read or written
    - Never greater than its limit

Unit-9

NIO

Flipping the Buffer

- Switches from write-to-buffer mode to read-from-buffer



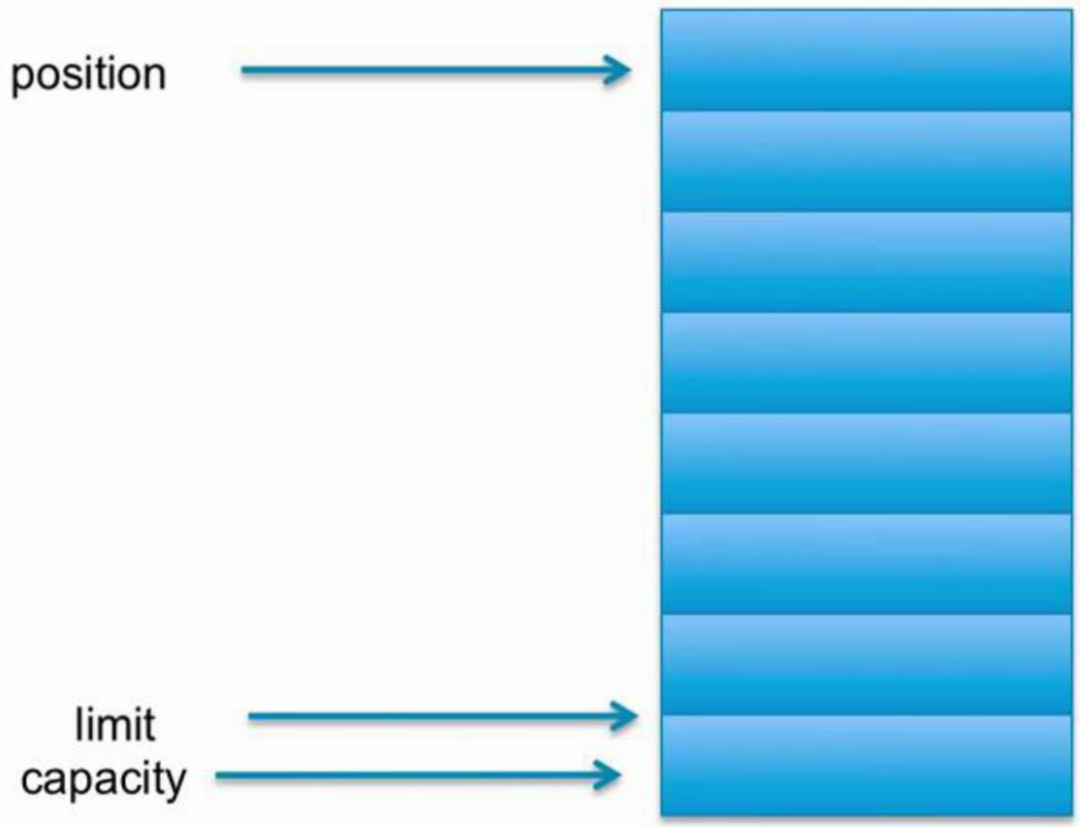
- Essential properties of a buffer
  - **Capacity**: the number of elements it contains
    - Specified when the Buffer is constructed and cannot be changed (similar to an array)
    - Never changes
  - **Limit**: specifies the current occupancy (valid data in the range of 0 to limit-1)
    - Never greater than its capacity
  - **Position**: the index of the next element to be read or written
    - Never greater than its limit

Unit-9

NIO

Compacting the Buffer

- Reset the buffer to be filled again

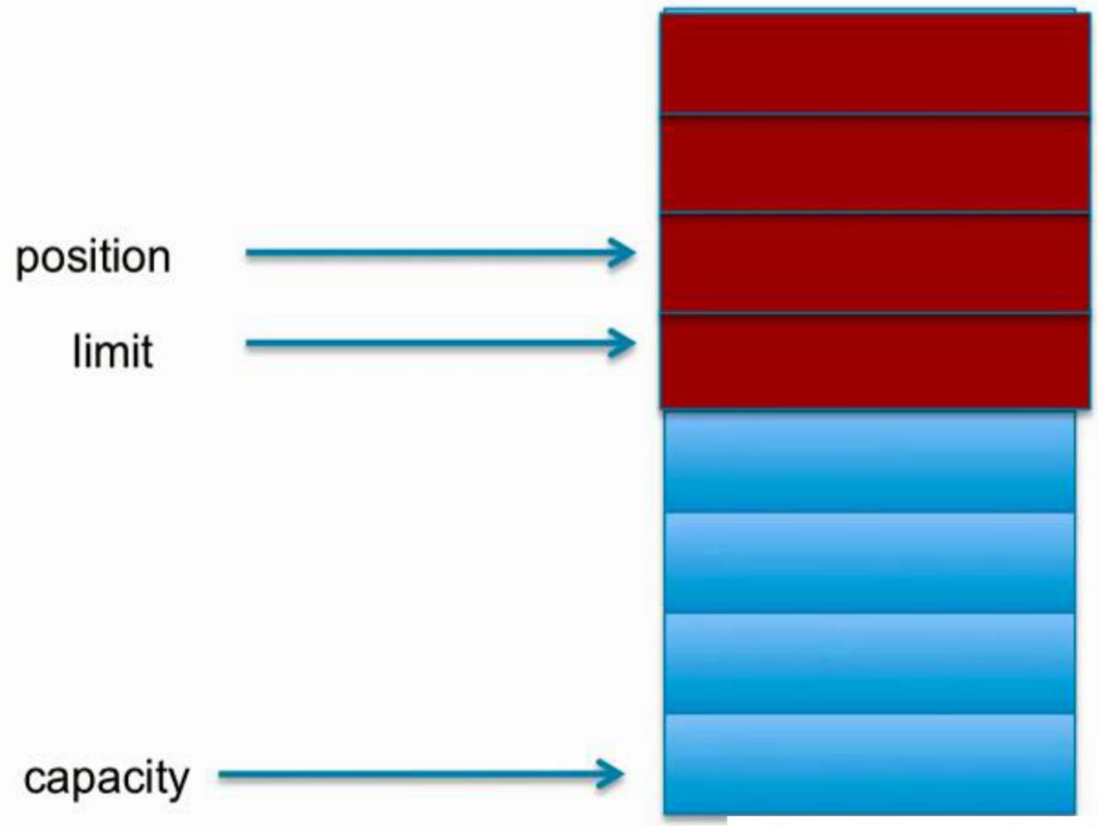


Unit-9

NIO

Compacting the Buffer

- Clear but saves unread data



- Essential properties of a buffer
  - **Capacity**: the number of elements it contains
    - Specified when the Buffer is constructed and cannot be changed (similar to an array)
    - Never changes
  - **Limit**: specifies the current occupancy (valid data in the range of 0 to limit-1)
    - Never greater than its capacity
  - **Position**: the index of the next element to be read or written
    - Never greater than its limit

Unit-9

NIO

Scatter Gather

Unit-9

NIO

Scattering and Gathering

- Write data from a channel to a Buffer array
- Write data from Buffer array to channel

Unit-9

NIO

Transfer

Unit-9

NIO

Transfer from Channel to Channel

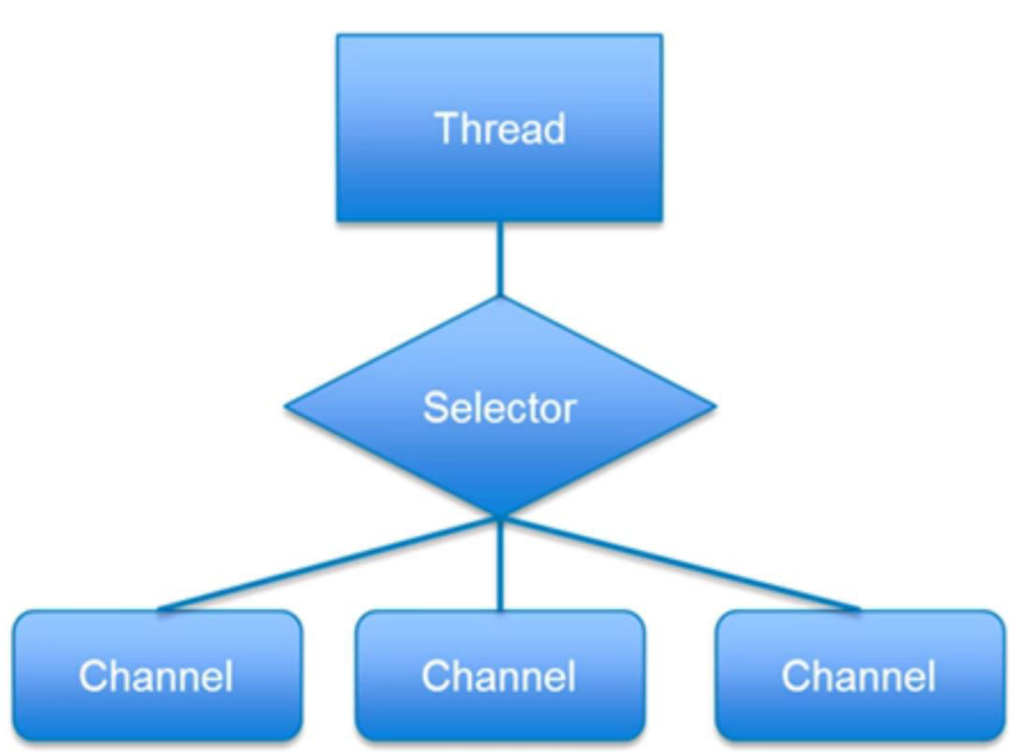
- Transfer data directly between channels
- There is no need to code Buffer handling if we just read from one and write to the other

Unit-9

NIO

Selectors

- Selectors are effective when the application has to handle many low volume connections



- Register a channel to a selector
- Call select() to wait for an event on any of the channels registered

Unit-9 List all supported socket options for different types of network channels

```
import java.io.*;
import java.net.*;
import java.nio.channels.*;

public class OptionSupport {
  public static void main(String[] args) throws IOException {
    printOptions(SocketChannel.open());
    printOptions(ServerSocketChannel.open());
    printOptions(AsynchronousSocketChannel.open());
    printOptions(AsynchronousServerSocketChannel.open());
    printOptions(DatagramChannel.open());
  }

  private static void printOptions(NetworkChannel channel) throws
IOException {
    System.out.println(channel.getClass().getSimpleName() + " supports:");
    for (SocketOption<?> option : channel.supportedOptions()) {
      System.out.println(option.name() + ": " +
channel.getOption(option));
    }
    System.out.println();
    channel.close();
  }
}
```

Unit-9 Write a program to implement the concept on Data Conversion

```
import java.nio.BufferUnderflowException;
import java.nio.ByteBuffer;
public class DataConversionTest {
    public static void main ( String[] args ) {
        int capacity= 8;
        try {
            ByteBuffer bb = ByteBuffer.allocate(capacity);
            bb.asIntBuffer().put(10).put(20);
            bb.rewind();
            // print the ByteBuffer
            System.out.println("Original ByteBuffer: ");
            for (int i = 1; i <= capacity / 4; i++) {
                System.out.println(bb.getInt() + "");
            }
            bb.rewind();
            int value = bb.getInt();
            System.out.println("\n\n Byte Value: " + value);
            int value1 = bb.getInt();
            System.out.println("Next Byte Value: " + value1);
            int value2 = bb.getInt();
            // continue..
        }catch(BufferUnderflowException ex){
                System.out.println("\n There r fewer than" + "four bytes remaining in this
                buffer"); System.out.println("Exception Thrown: " + ex);
        }
    }
}
```

Unit-9

Summary

Java New I/O (NIO): buffers, channels, selectors and non-blocking I/O

11.1 An Example Client
- ChargenClient (Example 11-1)

11.2 An Example Server
- ChargenServer (Example 11-2)

11.3 Buffers
- ByteBuffer, Direct ByteBuffer, View
- IntgenServer (Example 11-3), IntgenClient (Example 11-4)
- *EchoServer (Example 11-5)*, NonblockingSingleFileHTTPServer (Example 11-6)

11.4 Channels
- SocketChannel, ServerSocketChannel, and DatagramChannel
- OptionSupport (Example 11-7)

11.5 Readiness Selection
- Selector, SelectionKey