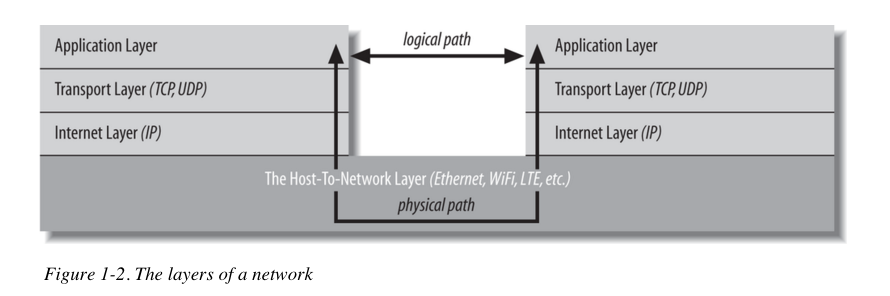
# Network Progamming



Streams are used to read/write bytes of data whereas Reader/Writer are used to read/write character array of data.

Filter Streams are chained to manipulate the data while reading or writing.

For example **java.io.DataOutputStream** will take integer value and converts it into four bytes and writes it into underlying streams. Similary there can be other methods on this stream class that takes different primitive types.

**Readers** and **Writers** are chained to input/output streams to read direct text or characters array with different encoding types like UTF-8

Streams are synchronous it reads data sequentially; java also offers non-blocking IO using **channels** and **buffers**.

**Streams:**

Java Basic **OutputStream** class imlpements **Closable**, **Flushable** interfaces.

AutoClosable interface introduced in 1.7 to support resource try block to auto close the streams.

This is an abstract class having below abstract method

It has method called **public abstract write (int b) throws IOException**

Sub classes of this calss are used to wtie data into particular media.

* FileOutputStream 🡺 writes data into file
* TelnetOutputStream🡺 writes data into network connection
* ByteArrayOutputStream🡺 writes data into expandable array

ByteArrayOutputStream does have constructor of OutputStream .it have method toByteArray [] to read the available bytes back.

**What is dispose pattern? (from java-network-programming)**

Declare the stream variable outside the **try** block but initialize it inside the **try** block. Furthermore, to avoid **NullPointerExceptions** you need to check whether the stream variable is null before closing it. Finally, you usually want to ignore or at most log any exceptions that occur while closing the stream

OutputStream out = null;

try {

out = new FileOutputStream("/tmp/data.txt"); // work with the output stream...

} catch (IOException ex) {

System.err.println(ex.getMessage());

} finally

{

if (out != null) {

try { out.close();

} catch (IOException ex) {

// ignore

}

}

}

**Q) What are the difference betweeen URL and URLConnection classes?**

URLConnection class provides access to HTTP header

URLConnection can read/write data to and from the server

URLConnection can configure parameters to send to the server

**Q) What are method we need to call before reading /writing to the server?**

There are two methods **setDoInput (boolean)** before reading data from server and **setDoOutput (boolean)** before writing data to the server.

Data is transmitted across the Internet in packets of finite size called datagrams. Each datagram contains a header and a payload. The header contains the address and port to which the packet is going, the address and port from which the packet came, a checksum to detect data corruption, and various other housekeeping information used to ensure reliable transmission.

An InputStreamReader contains an underlying input stream from which it reads raw bytes. It translates these bytes into Unicode characters according to a specified encoding. An OutputStreamWriter receives Unicode characters from a running program. It then translates those characters into bytes using a specified encoding and writes the bytes onto an underlying output stream.