In different programming languages, we have to communicate with multiple different devices (like hard-ware, computer over network, console, web type communication).

In all these scenarios, there are **different end points (terminals).** Mechanism of saving data on hardware is totally different than that of console.

For each end-point or terminal, we need to design……

Instead of learning multiple ways, Java provides **generic way, abstract layer (streams).**

**Stream -> Channel / Path / Pipe between java code and end point.**

Stream **establishes channel/ path** between these two and reads/ writes data.

If route of stream is towards file, then reading/ writing of data will be done with file.

We are **considering data source** (where the data is coming from), we are not considering different devices.

**Stream -> Set of libraries, API’s, classes**. It is built-in Java Code. And these are available in a package called **java.io.** In this package, there are many stream classes ( to read…..).

We can write data byte by byte. We can **also write complete objects**. Other end point will read data from there. We can read/ write data in form of objects on end point.

* **Classification of Streams:**
* **With Respect To Data:**

1. **Byte Streams**
2. **Char Streams**

|  |  |
| --- | --- |
| **Byte Streams** | **Char Streams** |
| Launched in JDK 1.0 (early version). | Launched by JDK 1.1 |
| Byte Streams will read data byte by byte | Char streams will read and write data char by char. |
| It is based on ASCII (8 bits). | Char Streams are based on Unicode (16 bits). |

In backend, ASCII is bits or bytes. In ASCII, size of each character is 1 byte and in Unicode, size of each character is 2 bytes.

**Q: What is the difference between both? What is the benefit?**

* Char Streams were launched later.
* ASCII: American Standard Code for Information Interchange. It doesn’t provides support to characters which are worldwide like it doesn’t support Euro.
* Unicode: It provides support to chars which are used worldwide. Unicode provides us the feature of **internationalization**. You can create your program and deploy in any version without any hesitation/ care of any symbol.
* **With Respect To Functionality:**

1. **Low Level Streams**
2. **High Level Streams**

* **Low level Streams** directly communicate with your end points.
* **High level Streams** do not directly communicate with your end point. It actually communicates with your low level Stream to provide some extra functionality.

**Q: What this extra functionality means?**

**A:** We see videos on you-tube. If screen is slow, we see a circle moving. Movie plays. Circle moves. What is the story of moving circle**? Buffering**.

Media Player is a software which moves images in a special way. 25 images pass in front of me per second.

Media -> loads data

When buffer becomes full, it carries those frames and give them to media player.

Media player stores that buffer, plays and again stores.

Buffer provides extra functionality to low level.

We will also study **Cypher** which is used in **encryption/ decryption**.

All of these streams give functionality to low-level streams.

**When buffer becomes full, it sends to end point.**

End

Point

Java

Program

Low Level

High Level