# Java IO

- Streams
- IO classes hierarchy
- Example

## IO Streams

An I/O Stream represents an input source or an output destination. A stream can represent many different kinds of sources and destinations, including disk files, devices, other programs, and memory arrays.

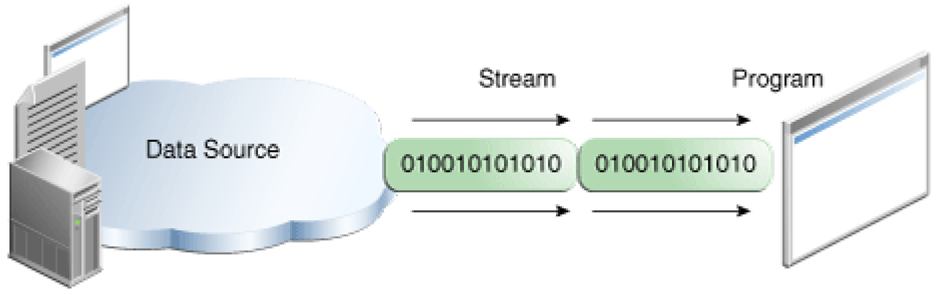
Streams support many different kinds of data, including simple bytes, primitive data types, localized characters, and objects. Some streams simply pass on data; others manipulate and transform the data in useful ways.

No matter how they work internally, all streams present the same simple model to programs that use them:

A stream is a sequence of data.

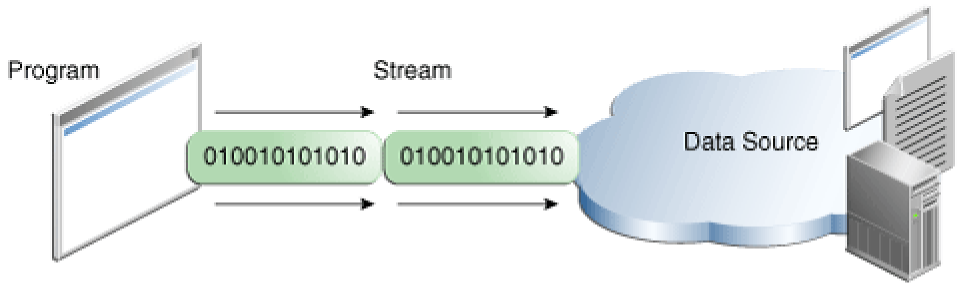
**Input stream**

A program uses an input stream to read data from a source, one item at a time.



**Output stream**

A program uses an output stream to write data to a destination, one item at time.



**java.io.InputStream**

```
public abstract class InputStream implements Closeable {
    public abstract int read() throws IOException;

    public int read(byte b[]) throws IOException {}
    public int read(byte b[], int off, int len) throws IOException {...}
    public long skip(long n) throws IOException {...}
    public int available() throws IOException {...}
    public void close() throws IOException {...}
    public synchronized void mark(int readlimit) {...}
    public synchronized void reset() throws IOException {...}
    public boolean markSupported() {...}
}
```

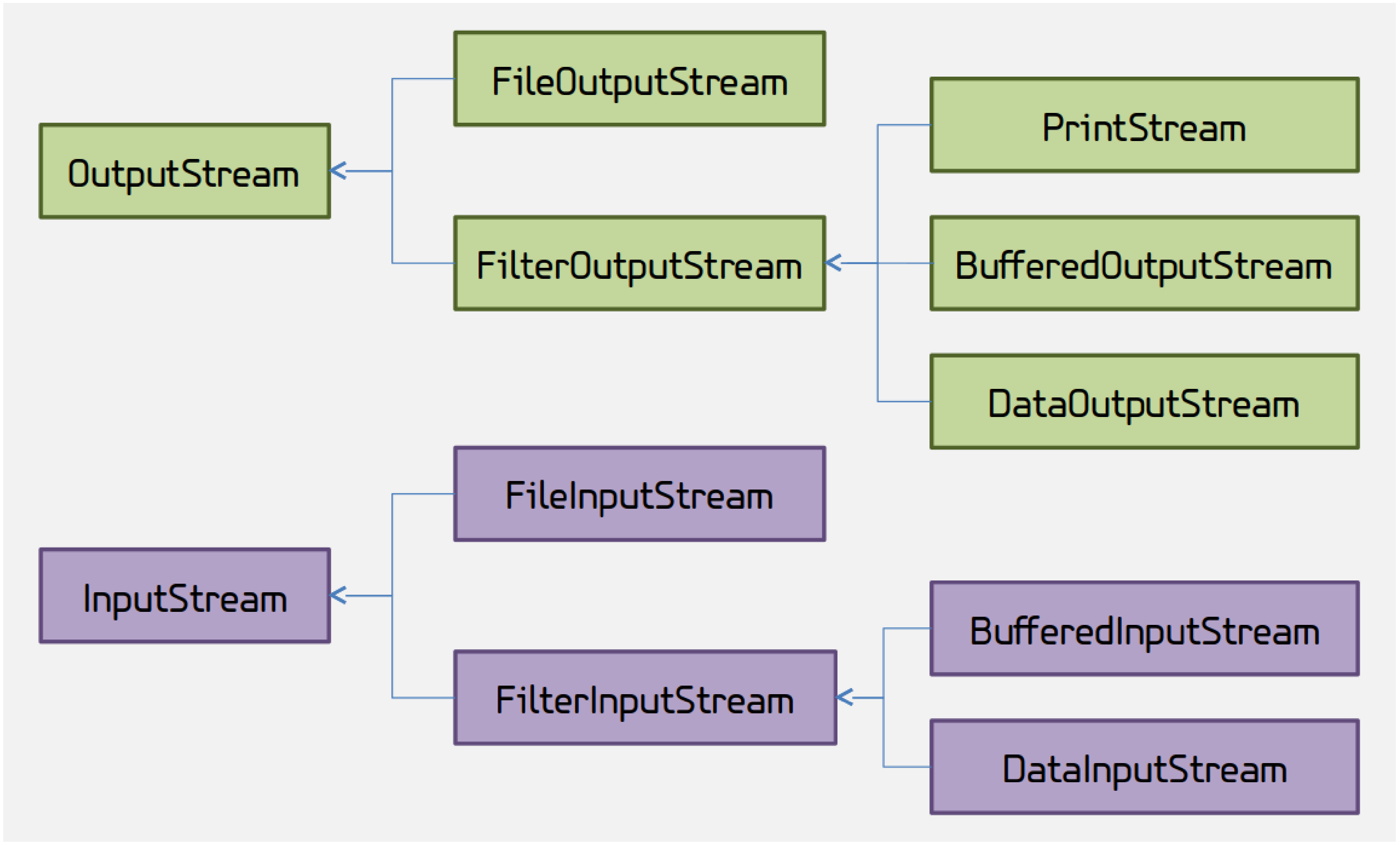
**java.io.OutputStream**

```
public abstract class OutputStream implements Closeable, Flushable {
    public abstract void write(int b) throws IOException;

    public void write(byte b[]) throws IOException {...}
    public void write(byte b[], int off, int len) throws IOException {...}
    public void flush() throws IOException {}
    public void close() throws IOException {}
}
```

## IO classes hierarchy

Each stream in java.io is a subclass of InputStream or OutputStream.



**Sources and sinks**

FileInputStream and FileOutputStream are source and sink of data.

They represent external resources (files).

If you need to work with external resource like: file, printer, byte[] in the memory…

You need source and sink for it.

Like:

- FileInputStream and FileOutputStream
- PrinterInputSream and PrinterOutputStream
- ByteArrayInputStream and ByteArrayOutputStream

**Decorators**

FilterImputStream and FilterOutputStream are decorators.

```
public class FilterInputStream extends InputStream {

    protected volatile InputStream in;

    protected FilterInputStream(InputStream in) {
        this.in = in;
    }
...
}
```

FilterInputStream contains other input stream, which it uses as its  basic source of data, possibly transforming the data along the way or providing  additional functionality.

**Some decorators**

ObjectInputStream

BufferedInputStream

ZipInputStream