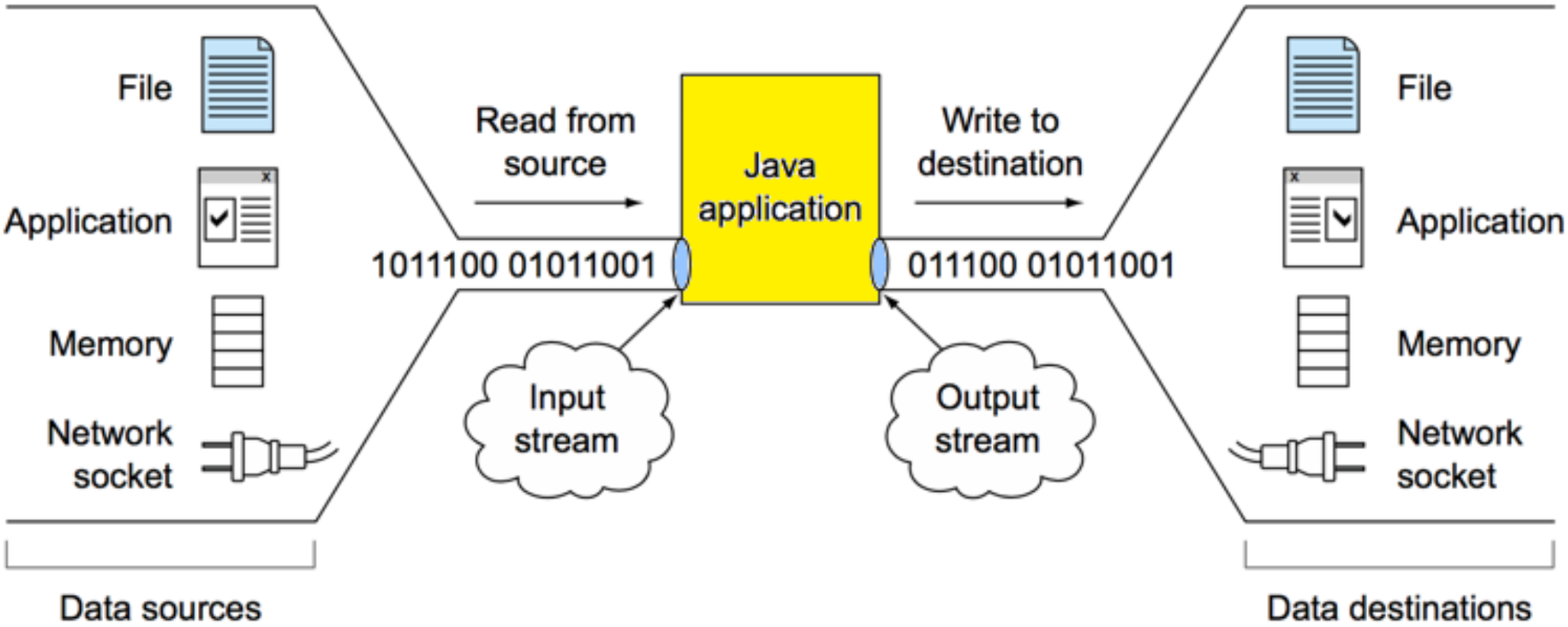
Java I/O Streams, Readers & Writers

An introduction to reading and writing bytes or text

I/O Streams

- A *stream* is a sequence of data.
- The stream can be either:
  - Input Stream: used for reading data from a source.
  - Output Stream: used for writing data to a destination.
- Java I/O stream abstracts a data source or data destination.

I/O Streams

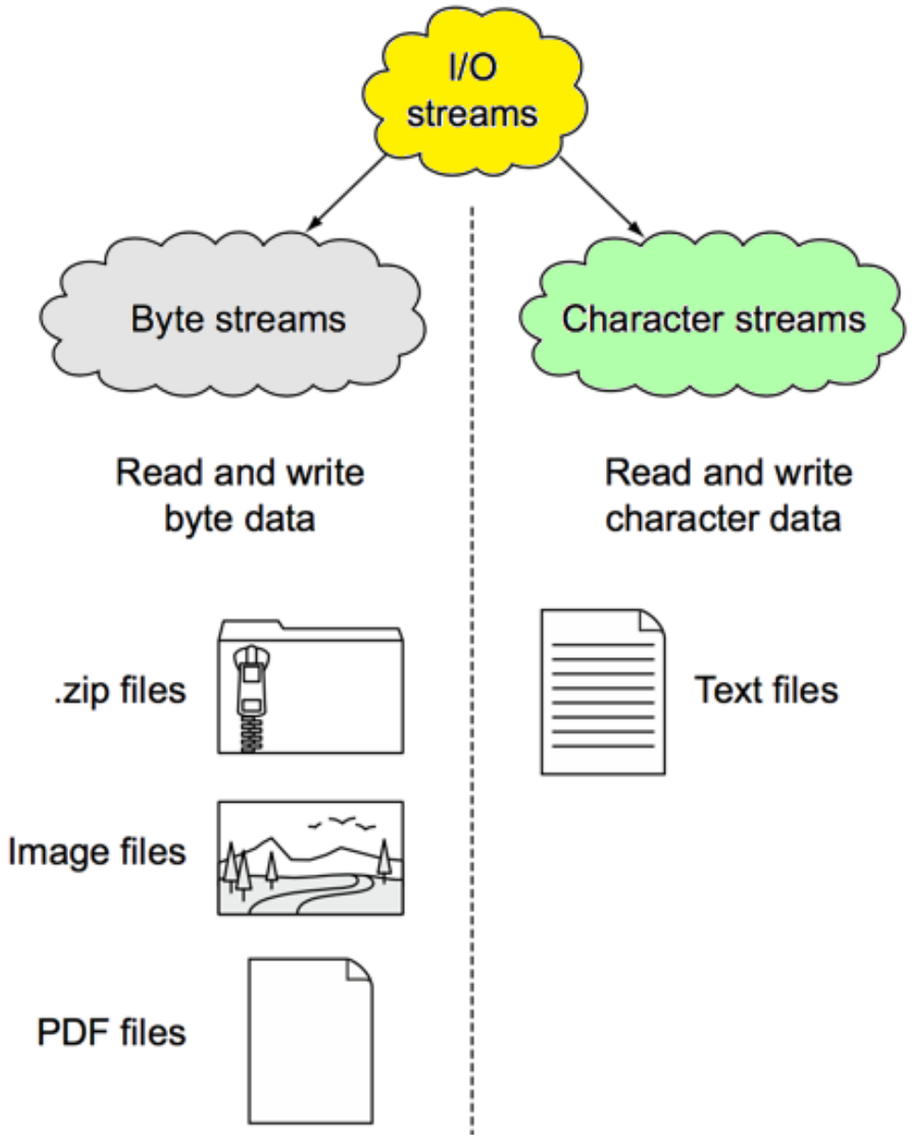

File
Application
Memory
Network socket
Data sources
Read from source
1011100 01011001
Input stream
Java application
Output stream
011100 01011001
Write to destination
File
Application
Memory
Network socket
Data destinations

Types of Streams

- The data can be either a byte data or character data.

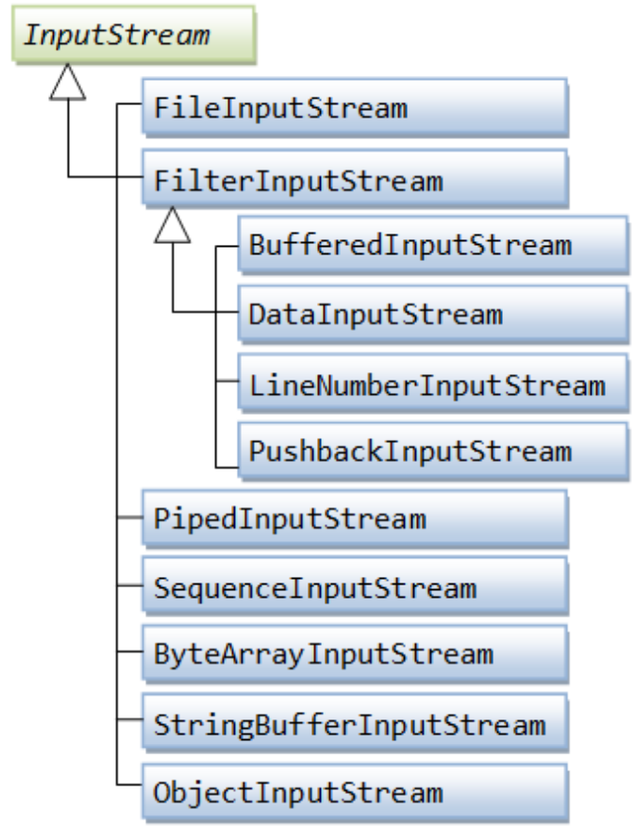
- *Character streams* represent *readers* and *writers* of characters and text data.

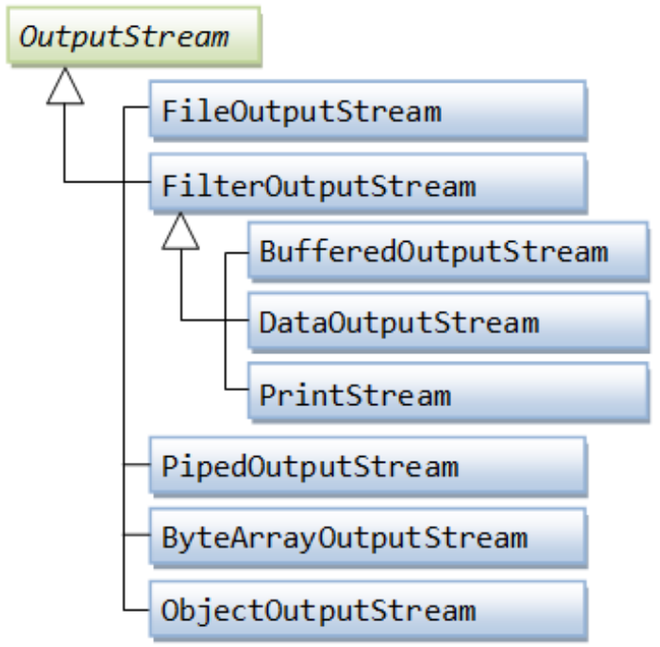
- *Byte streams* handle all types of data. This includes Text or PDF files, archive files, images, and so on.



I/O Streams

- The base I/O stream classes:
  - `java.io.InputStream`
    - is the superclass of all input streams
    - is used for reading byte based data, one byte at a time.
  - `java.io.OutputStream`
    - is the superclass of all output streams
    - is connected to a destination like file, network connection, or pipe.





InputStream Example

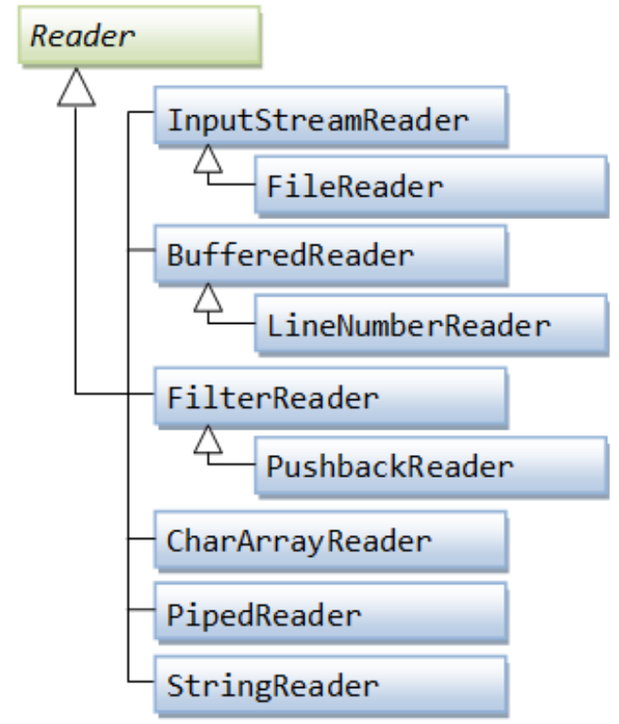
- `FileInputStream` can be used for reading the contents of a file as a stream of bytes. Being a subclass of it, it can be used as an `InputStream`.

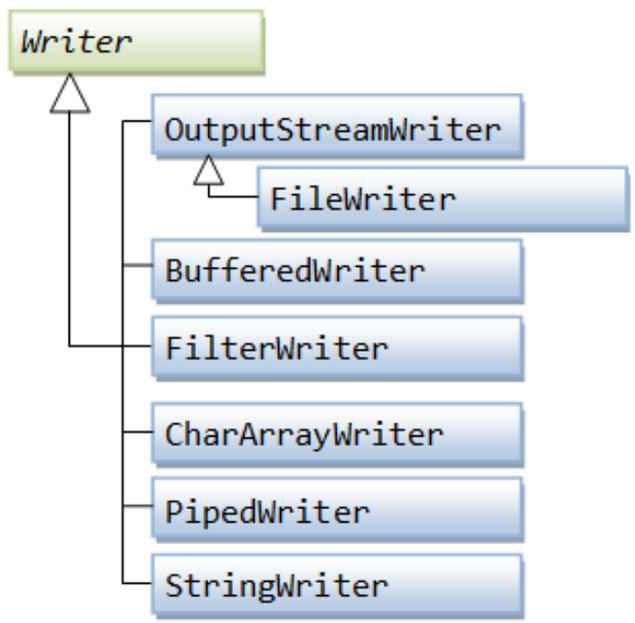
```
String filePath = "/tmp/input.txt";    // or "c:\\tmp\\input.txt"

InputStream input = new FileInputStream(filePath);

int data = input.read();
while (data != -1) {
  doSomethingWithData(data);
  data = input.read();
}
input.close();
```

It reads the file as a stream of bytes. `read()` returns an `int` which contains the byte value of the byte read.

Readers & Writers

- `InputStream` and `OutputStream` are byte based.
- The alternatives for text/character based operations are:
- java.io.Reader
  - is the base of all readers in Java I/O.
  - is intended for reading text data.
- java.io.Writer
  - is the base of all writers in Java I/O.
  - is used for writing text data.





Reader Example

- `FileReader` can be used for reading the contents of a file as a stream of characters. It behaves like an Reader, as being a subclass of it, so it can be used as an Reader.

```
Reader reader = new FileReader(filePath);

// read() returns an integer that is the value of the char read
int data = reader.read();

while (data != -1) {
    System.out.print((char) data);
    data = reader.read();
}
reader.close();
```

It reads the file as a stream of characters.
`read()` reads a single character
and returns it as an `int`.
Casting it to char will show
the character.

Mixing together

- You can mix a `Reader` with an `InputStream` using an `InputStreamReader`. Use case: the need to read characters from a source represented as an `InputStream`.

```
Reader reader = new InputStreamReader(inputStream);
```

- A `Writer` can be mixed with an `OutputStream` using an `OutputStreamWriter`. Use case: the need to write characters to a destination represented by an `OutputStream`.

```
Writer writer = new OutputStreamReader(outputStream);
```

Further Reading

- OCP Java SE 7 - Programmer II Certif Guide (Manning, 2015)

- http://tutorials.jenkov.com/java-io/index.html