Input Output

SOftEng
http://softeng.polito.it

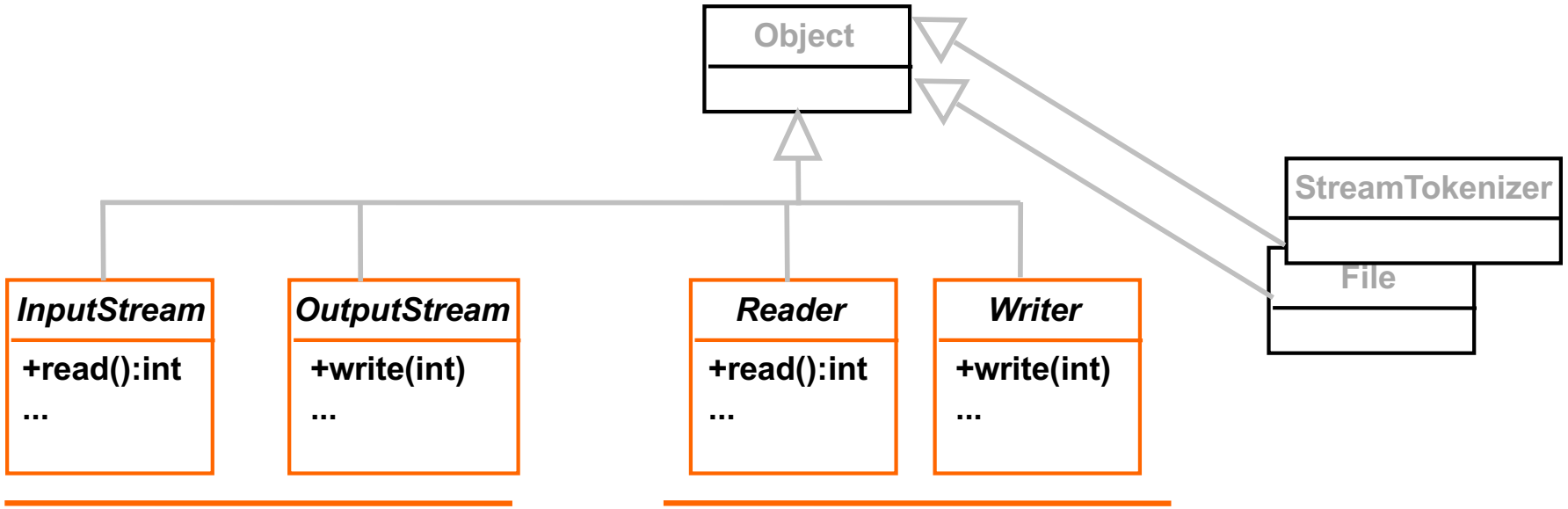
Stream

- All I/O operations rely on the abstraction of stream (flow of elements)
- A stream can be linked to:
  - A file on the disk
  - Standard input, output, error
  - A network connection
  - A data-flow from/to whichever hardware device
- I/O operations work in the same way with all kinds of stream

Stream

- Package java.io
- Reader / Writer
  - Stream of chars (Unicode chars – 16 bit)
  - All characters
- InputStream / OutputStream
  - Stream of bytes (8 bit)
  - Binary data, sounds, images
- All related exceptions are subclasses of IOException

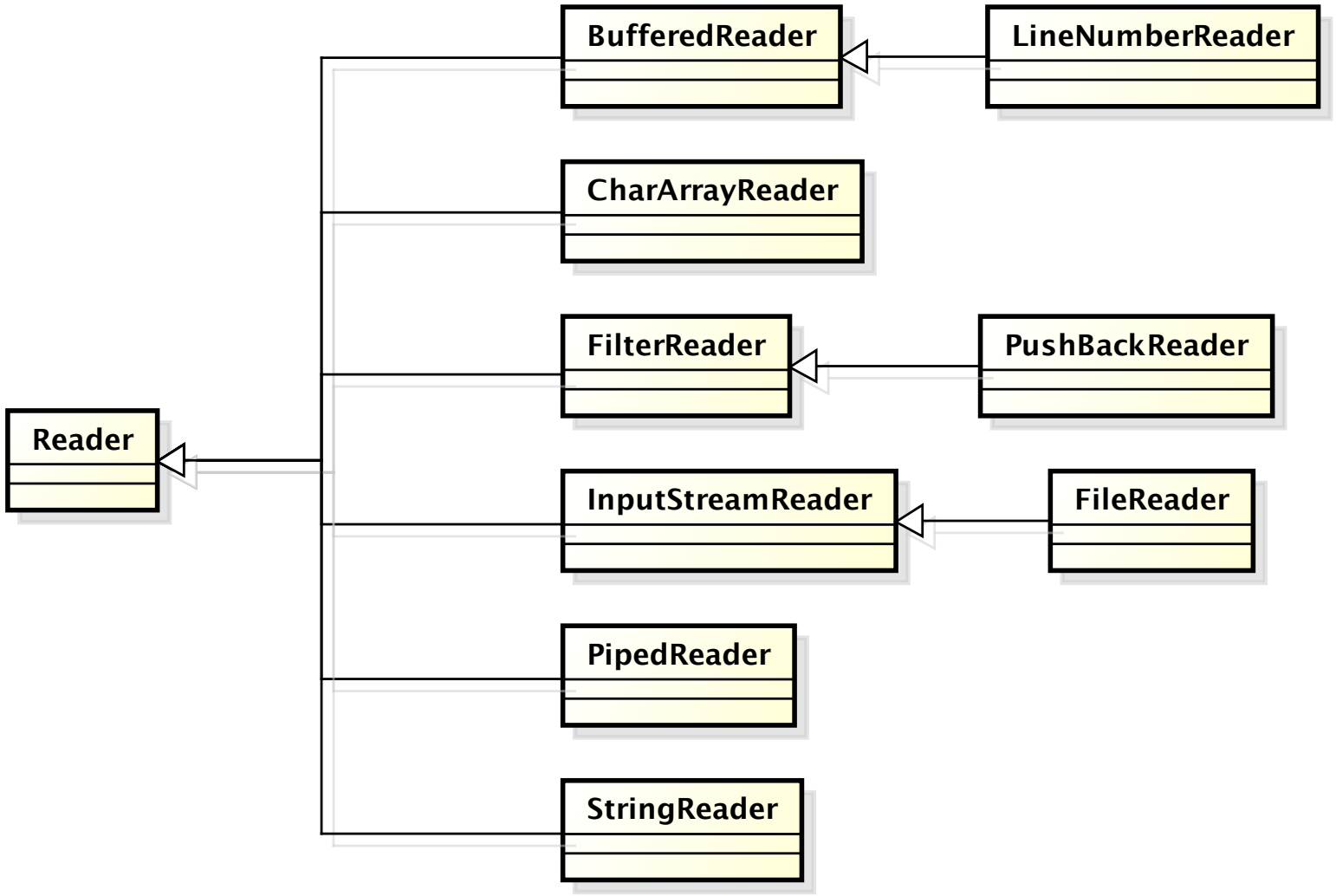
Base classes in java.io



Classes for bytes stream I/O

Classes for Unicode chars stream

Reader (chars)



Reader (abstract)

- int read()
  - Reads a character: -1 when end of stream
- int read(char[] cbuf)
  - Reads characters into an array
- int read(char[] cbuf, int off, int len)
  - Reads characters into a portion of an array

Blocking methods, i.e. stop until
- data available,
- I/O error, or
- end of stream

Reader (abstract)

- boolean ready()
  - Tells whether this stream is ready to be read
- void reset()
  - Resets the stream, restarts from the beginning
- long skip(long n)
  - Skips characters
- void close()
  - Closes the stream

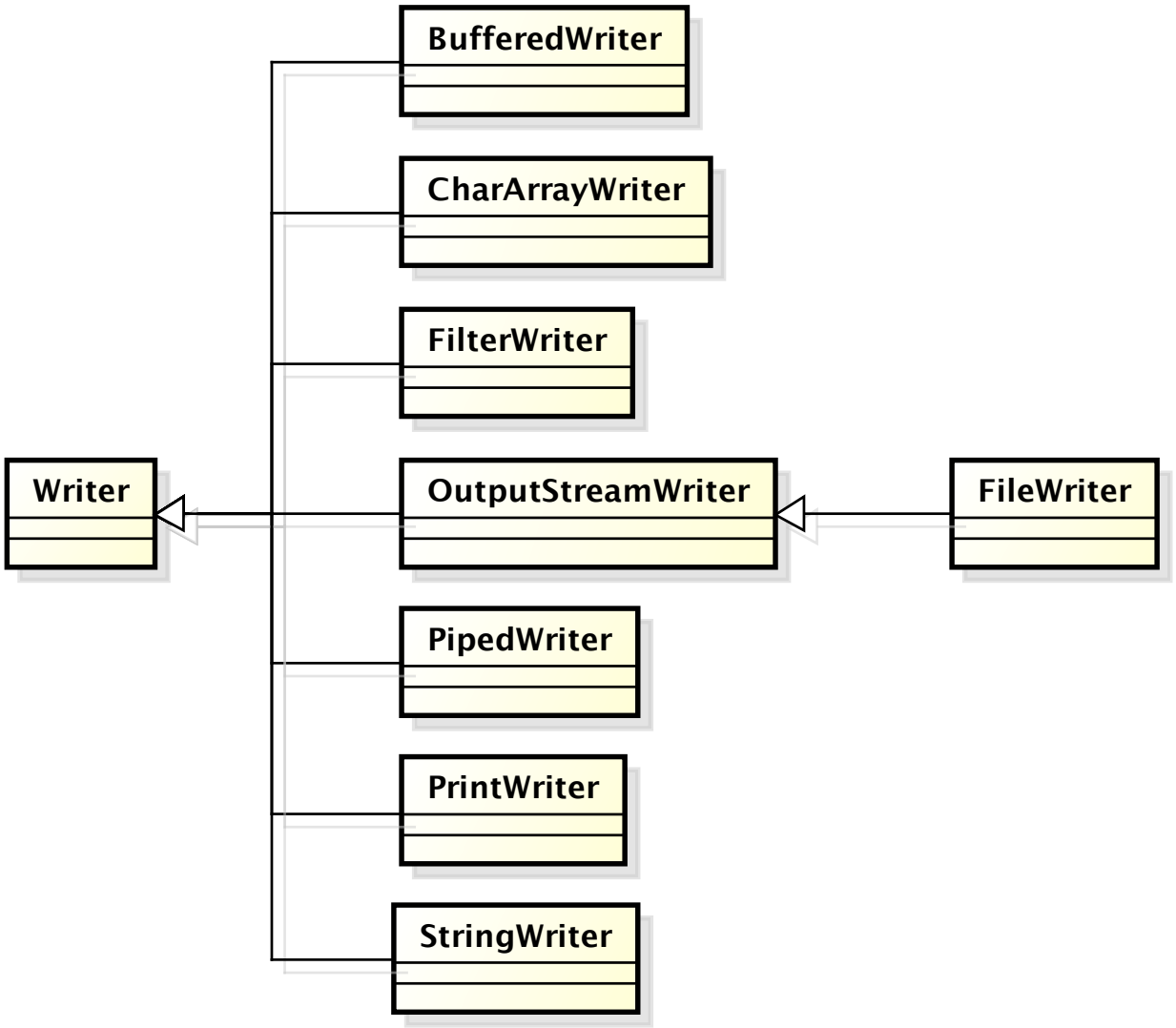
Read a char

```
int ch = r.read();
char unicode = (char) ch;
System.out.print(unicode);
r.close();
```

Read a line

```
public static String readLine(Reader r)
                              throws IOException{
   StringBuffer res= new StringBuffer();
   int ch = r.read();
   if(ch == -1) return null; // END OF FILE!
   while( ch != -1 ){
      char unicode = (char) ch;
      if(unicode == '\n') break;
      if(unicode != '\r')
         res.append(unicode);
         ch = r.read();
   }
return res.toString();
}
```

Writer (chars)



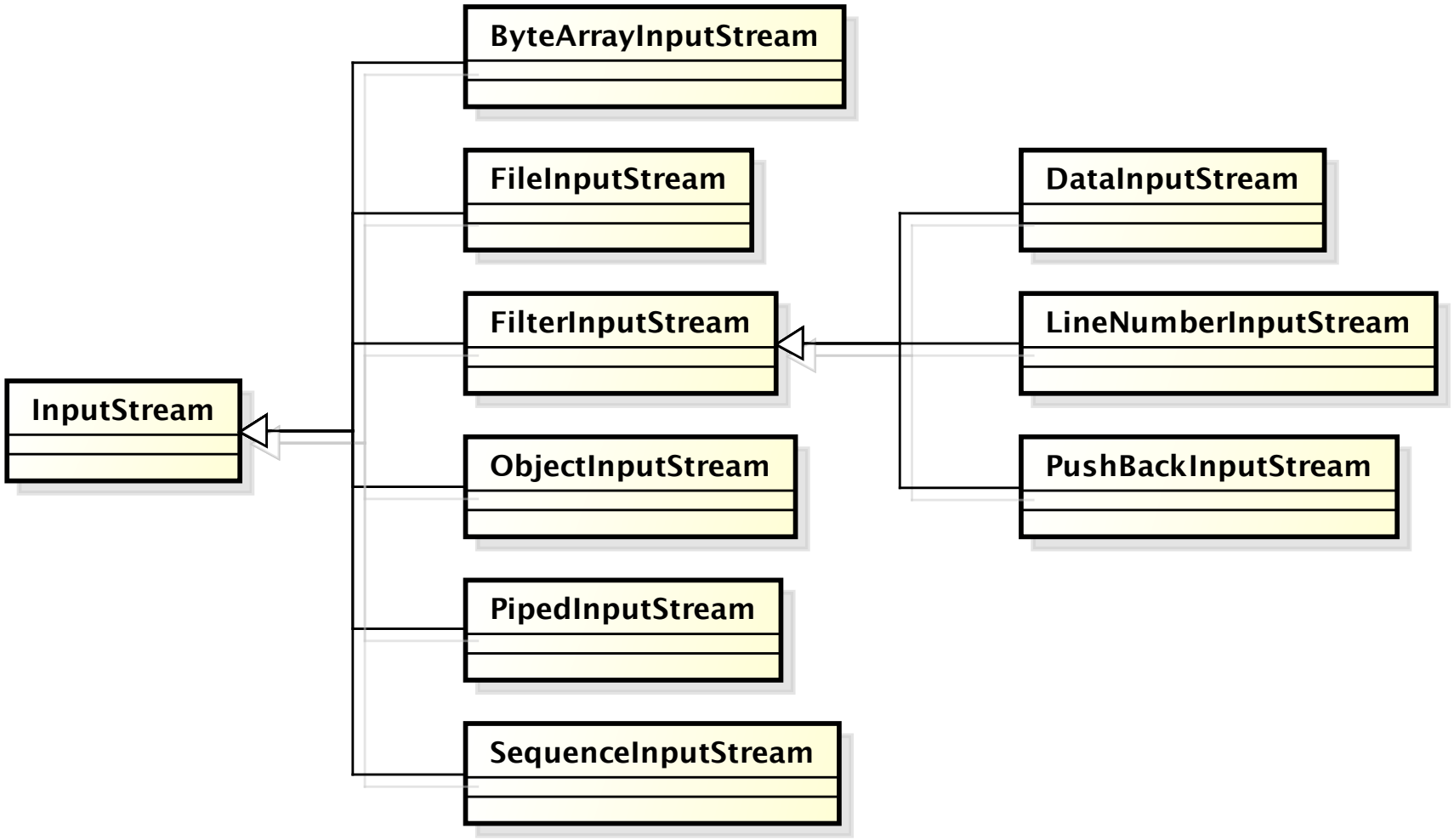
Writer (abstract)

- void write(int c)
  - Writes a single character
- void write(char[] cbuf)
  - Writes an array of characters
- void write(char[] cbuf, int off, int len)
  - Writes a portion of an array of characters
- void write(String str)
  - Writes a string
- void write(String str, int off, int len)
  - Writes a portion of a string

Writer (abstract)

- void flush()
  - Flushes the stream
- close()
  - Closes the stream, flushing it first

InputStream (bytes)



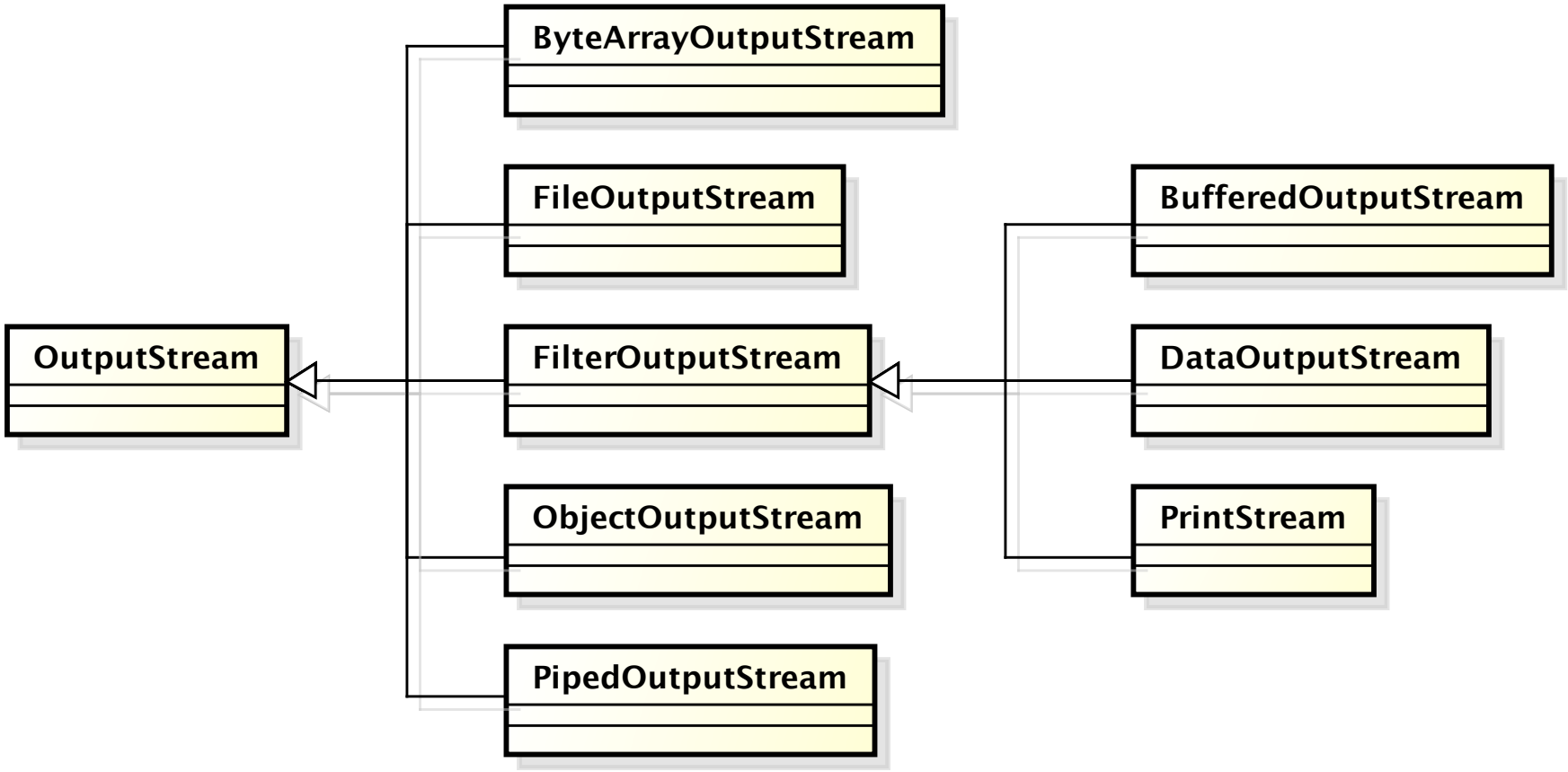
InputStream (abstract)

- int read()
  - Reads the next byte of data from the input stream
- int read(byte[] b)
  - Reads some number of bytes from the input stream and stores them into the buffer array b
- int read(byte[] b, int off, int len)
  - Reads up to len bytes of data from the input stream into an array of bytes

InputStream (abstract)

- int available()
  - Returns the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream
- void reset()
  - Resets the stream, restarts from the beginning
- long skip(long n)
  - Skips over and discards n bytes of data from this input stream
- void close()
  - Closes this input stream and releases any system resources associated with the stream

OutputStream (bytes)



OutputStream (abstract)

- void write(int b)
  - Writes the specified byte to this output stream
- void write(byte[] b)
  - Writes b.length bytes from the specified byte array to this output stream
- void write(byte[] b, int off, int len)
  - Writes len bytes from the specified byte array starting at offset off to this output stream
- void close()
  - Closes this output stream and releases any system resources associated with this stream
- void flush()
  - Flushes this output stream and forces any buffered output bytes to be written out

Stream specializations

- Memory
- File
- Buffered
- Interpreted
- Others
  - Printed, provide methods like print/ln()
  - Pipe, for inter-thread communication

Standard in & out

- Default input and output streams are defined in class System

```
class System {
  // ...
  static InputStream in;
  static PrintStream out;
  static PrintStream err;
}
```

Conversion bytes to/from chars

- InputStreamReader
  - Bytes to chars
- OutputStreamWriter
  - Chars to byte
- The constructors allow specifying a charset to decode/encode the byte to/from characters

Read/Write in memory

- Read/Write chars or bytes from/to array in memory
  - CharArrayReader
  - CharArrayWriter
  - ByteArrayInputStream
  - ByteArrayOutputStream

Read/Write in memory

- Read/Write chars from/to String
  - StringReader
  - StringWriter
- Read bytes from StringBuffer
  - StringBufferInputString

Read/Write of File

- Read/Write chars from file
  - FileReader
  - FileWriter
- Read/Write bytes from file
  - FileInputStream
  - FileOutputStream
- File
  - Handles filename and pathname

Copying a text file

```
import java.io.*;
public class Copy {
  public static void main(String[] args) throws
  IOException{
    File inputFile = new File("in.txt");
    File outputFile = new File("out.txt");
    FileReader in = new FileReader(inputFile);
    FileWriter out = new FileWriter(outputFile);
    int c;
    while ((c = in.read()) != -1)
       out.write(c); // One char at a time, inefficient
    in.close();
    out.close();
  }
}
```

Copying a text file

```
import java.io.*;
public class Copy {
  public static void main(String[] args) throws
  IOException{
    FileReader in = new FileReader("in.txt");
    FileWriter out = new FileWriter("out.txt");
    int c;
    while ((c = in.read()) != -1)
       out.write(c); // One char at a time, inefficient
    in.close();
    out.close();
  }
}
```

Copying a text file with buffer

```
import java.io.*;
public class Copy {
  public static void main(String[] args) throws
  IOException{
    FileReader in = new FileReader("in.txt");
    FileWriter out = new FileWriter("out.txt");
    char[] buffer = new char[4096];
    int n;
    while ((n = in.read(buffer)) != -1)
       out.write(buffer, 0, n);
    in.close();
    out.close();
  }
}
```

Buffered read/write

- BufferedInputStream
  - BufferedInputStream(InputStream i)
  - BufferedInputStream(InputStream i, int size)
- BufferedOutputStream
- BufferedReader
  - readLine()
- BufferedWriter

Interpreted read/write

- Translate primitive types into/from standard format (generally used in combination with files)
- DataInputStream(InputStream i)
  - readByte()
  - readChar()
  - readDouble()
  - …
- DataOutputStream(OutputStream o)

Stream as a resource

- Streams consume OS resources
  - Should be closed as soon as possible to release resources

```
BufferedReader br=
   new BufferedReader(new FileReader(path));
String l = br.readLine();
br.close();
```

Exceptions

- What happens in case of exception, e.g., in reading a line?

```
BufferedReader br=
 new BufferedReader(new FileReader(path));
try { return br.readLine(); }
catch(IOException ioe) { ... }
finally {
  if(br!=null)
    br.close();
}
```

Tokenizers

- StringTokenizer
  - Works on String objects
  - Set of delimiters (blank, “,”, \t, \n, \r, \f)
  - Blank is the default delimiter
  - Divides a string in tokens (separated by delimiters), returning the token
  - Methods hasMoreTokens(), nextToken()
  - Does not distinguish identifiers, numbers, comments, quoted strings

Tokenizers

- StreamTokenizer
  - Works on Stream (Reader) objects
  - More sophisticated, recognizes identifiers, comments, quoted string, numbers
  - Use symbol table and flag
  - Method nextToken(), TT_EOF if at the end

split()

- Method of the String class
  - Divides the string around matches of the provided regular expression