Programming in Java – Part 06 - Basics of Input and Output

Paolo Vercesi
ESTECO SpA

Agenda

Input and Output streams

Reading and writing binary data

Data streams

Reading and writing Java types

Readers and Writers

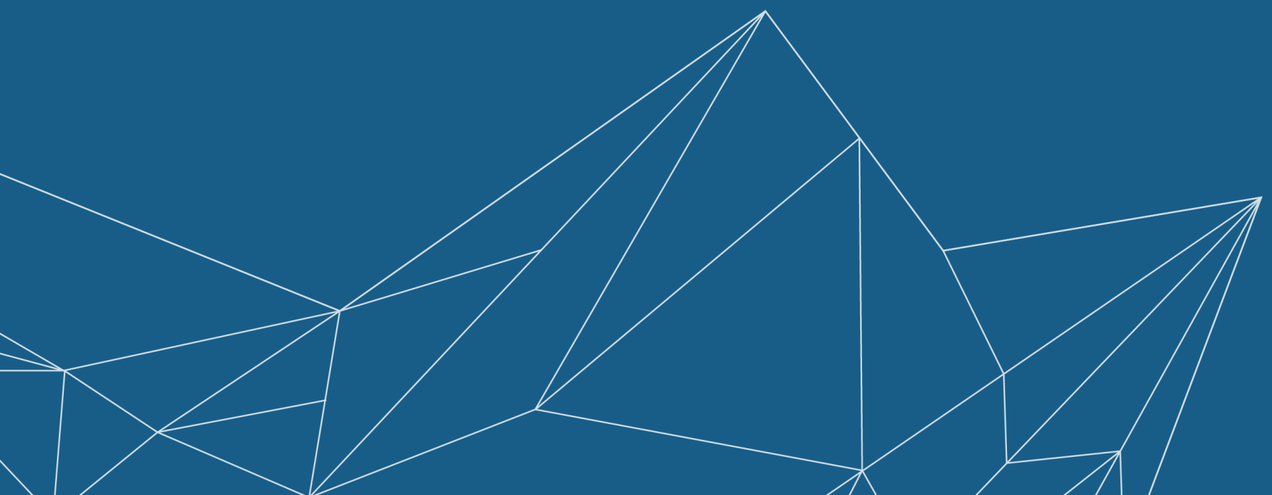
Reading and writing text

Console I/O

Reading and writing from the console

Input and Output streams

Reading and writing binary data



I/O Streams

I/O in Java is based on streams. Not to be confused with the streams in java.util.stream
The abstraction is the same, but the implementation is different

I/O streams represent a flow of binary data

Input streams are used to read from (binary data) sources
Output streams are used to write to (binary data) targets

Introducing InputStream

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

try-with-resources

```
try (InputStream is = …) {
    int read;
    while ((read = is.read()) != -1) {
        System.out.println("Read: " + read);
    }
}
```

Reads the next byte of data from the input stream

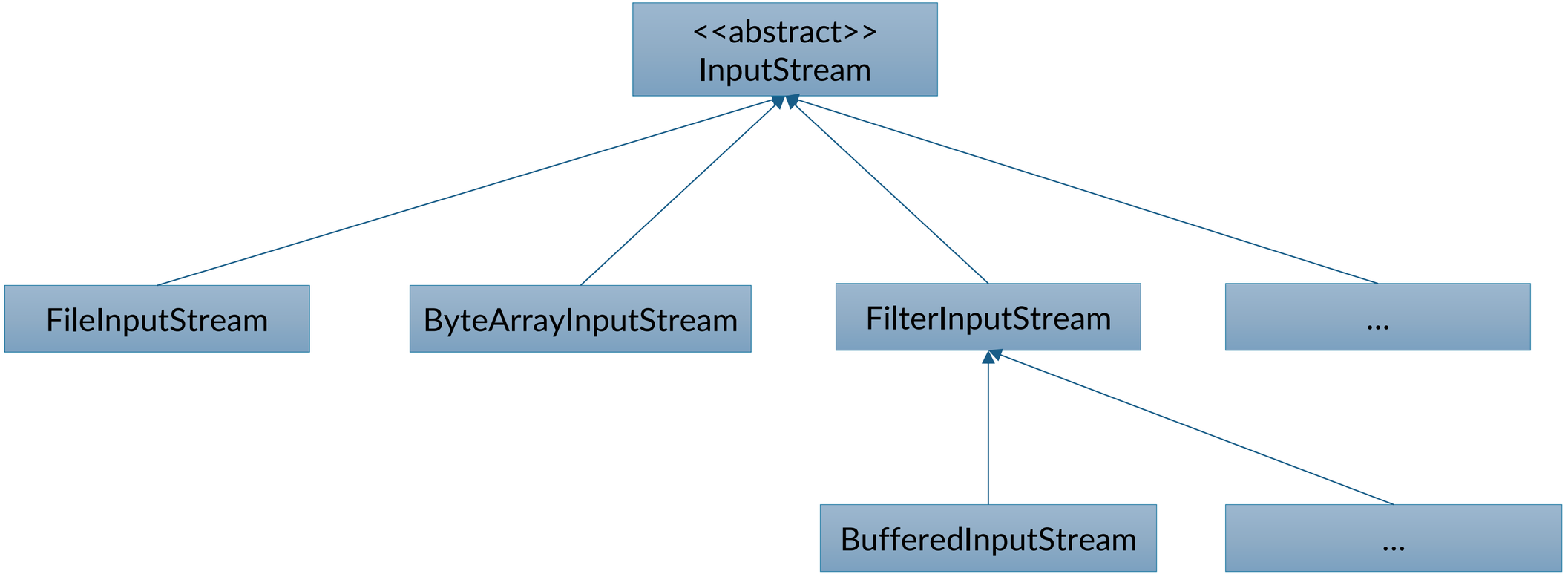
The value byte is returned as an int in the range 0 to 255

If no byte is available because the end of the stream has been reached, the value -1 is returned

The method blocks until

- *input data is available*
- *the end of the stream is detected*
- *an exception is thrown*

The InputStream hierarchy



Examples of InputStream 1/3

```
String fileName = "G:\\My Drive\\ … \\Input and Output.pptx";
try (InputStream fis = new FileInputStream(fileName)) {
    int count = 0;
    while (fis.read() != -1) {
        count++;
    }
    System.out.println("Read: " + count);
}
```

Examples of InputStream 2/3

```
URL url = new URL("https://www.google.it");
try (InputStream urlStream = url.openStream()) {
    int read;
    while ((read = urlStream.read()) != -1) {
        System.out.print((char) read);
    }
}
```

WARNING we are converting a stream of bytes into chars

Examples of InputStream 3/3

```
byte[] byteArray = …
try (InputStream is = new ByteArrayInputStream(byteArray)) {
    int read;
    while ((read = is.read()) != -1) {
        System.out.print(read);
    }
}
```

Other methods in InputStream

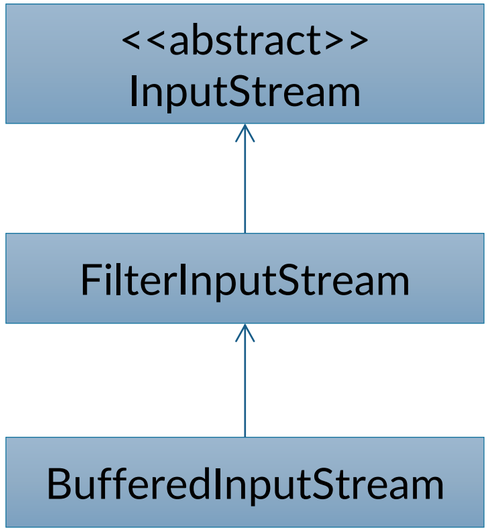
```
public int read(byte b[]) throws IOException
public int read(byte b[], int off, int len) throws IOException
public byte[] readNBytes(int len) throws IOException
public int readNBytes(byte[] b, int off, int len) throws IOException
public byte[] readAllBytes() throws IOException
public long skip(long n) throws IOException
public void skipNBytes(long n) throws IOException
public long transferTo(OutputStream out) throws IOException
public int available() throws IOException
public synchronized void mark(int readlimit)
public synchronized void reset() throws IOException
public boolean markSupported()
public void close() throws IOException
```

BufferedInputStream

When reading from the filesystem or from the network, the reading of small chunks of data can be very inefficient

Java offers buffered input to speedup the reading of small chunks of data

The BufferedInputStream reads data in advance in a buffer of a specified size




```
public class BufferedInputStream extends FilterInputStream {

    public BufferedInputStream(InputStream in)
    public BufferedInputStream(InputStream in, int size)
    …
}
```

A BufferedInputStream is an InputStream wrapping another input stream

Working with BufferedInputStream

```
String fileName = "G:\\My Drive\\ … \\Input and Output.pptx";
try (InputStream fis = new BufferedInputStream(new FileInputStream(fileName))) {
    int count = 0;
    while (fis.read() != -1) {
        count++;
    }
    System.out.println("Read: " + count);
}
```

```
URL url = new URL("https://www.google.it"):
try (InputStream urlStream = new BufferedInputStream(url.openStream())) {
    int read;
    while ((read = urlStream.read()) != -1) {
        System.out.print((char) read);
    }
}
```

Introducing OutputStream

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

Writes the specified byte to this output stream

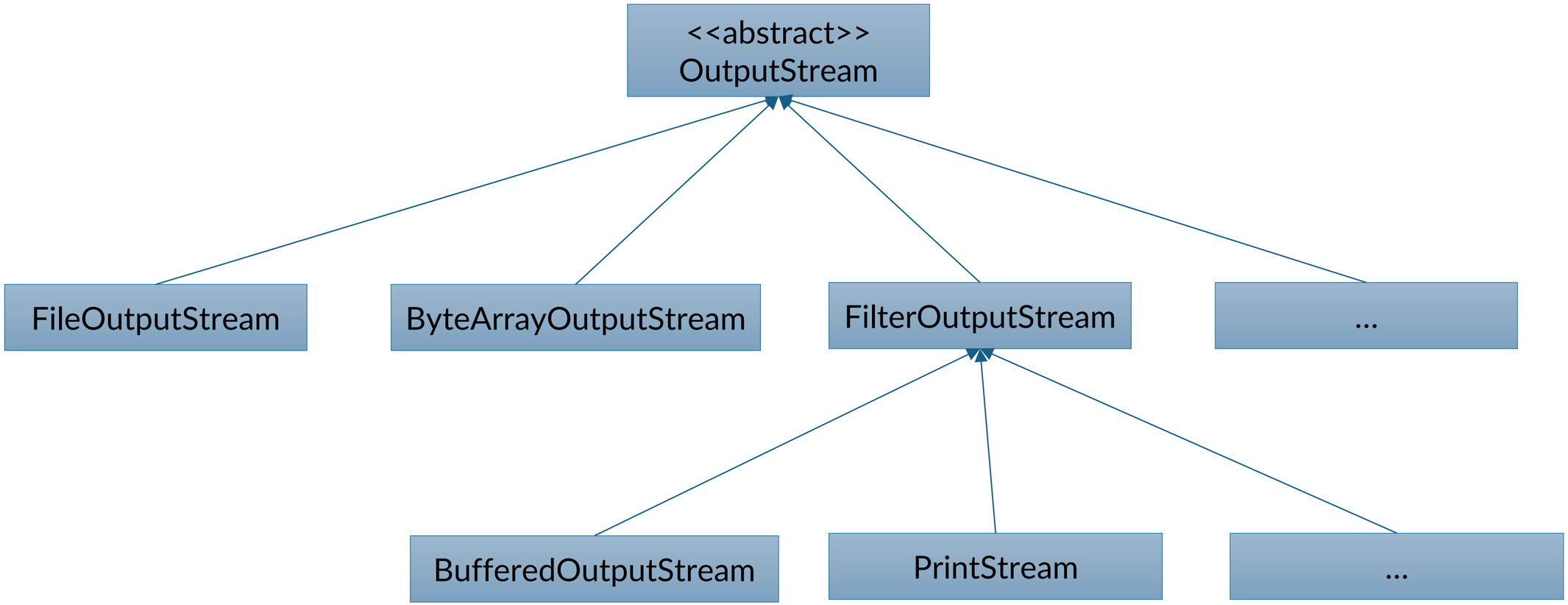
The byte to be written is the 8 low-order bits of the argument b

The 24 high-order bits of b are ignored

try-with-resources

```
try (OutputStream os = …) {
    int[] data = …;
    for (int datum : data) {
        os.write(datum);
    }
}
```

The OutputStream hierarchy



Examples of OutputStream

```
try (OutputStream fos = new FileOutputStream("A:\\git\\sdm\\pippo.dat")) {
    for (int i = 0; i < 10; i++) {
        fos.write(i);
    }
}
```

```
byte[] byteBuffer = new byte[10];
try (OutputStream os = new ByteArrayOutputStream(byteBuffer)) {
    for (int i = 0; i < 10; i++) {
        os.write(i);
    }
}
```

Other methods of OutputStream

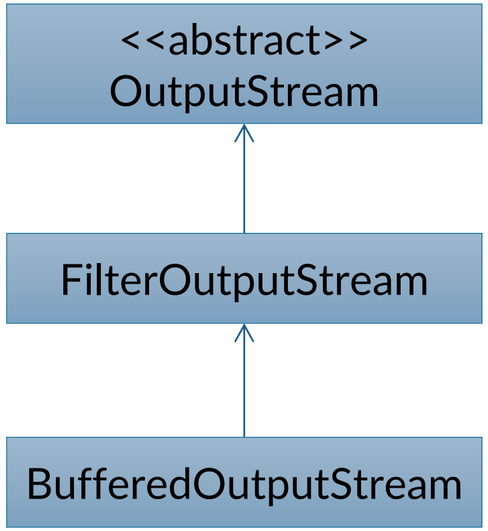
```
public void write(byte b[]) throws IOException
public void write(byte b[], int off, int len) throws IOException
public void flush() throws IOException
public void close() throws IOException
```

BufferedOutputStream

When writing to the filesystem or to the network, the writing of small chunks of data can be very inefficient

Java offers buffered output to speedup the writing of small chunks of data

The BufferedOutputStream writes data to the wrapped stream only when the buffer is full or when flush() is invoked




```
public class BufferedOutputStream extends FilterOutputStream {

    public BufferedOutputStream(OutputStream out)
    public BufferedOutputStream(OutputStream out, int size)
    …
}
```

A BufferedOutputStream is an OutputStream wrapping another output stream

Working with BufferedOutputStream

```
String fileName = "A:\\git\\sdm\\pippo.dat";
try (OutputStream fos = new BufferedOutputStream(new FileOutputStream("…")) {
    for (int i = 0; i < 10; i++) {
        fos.write(i);
    }
}
```

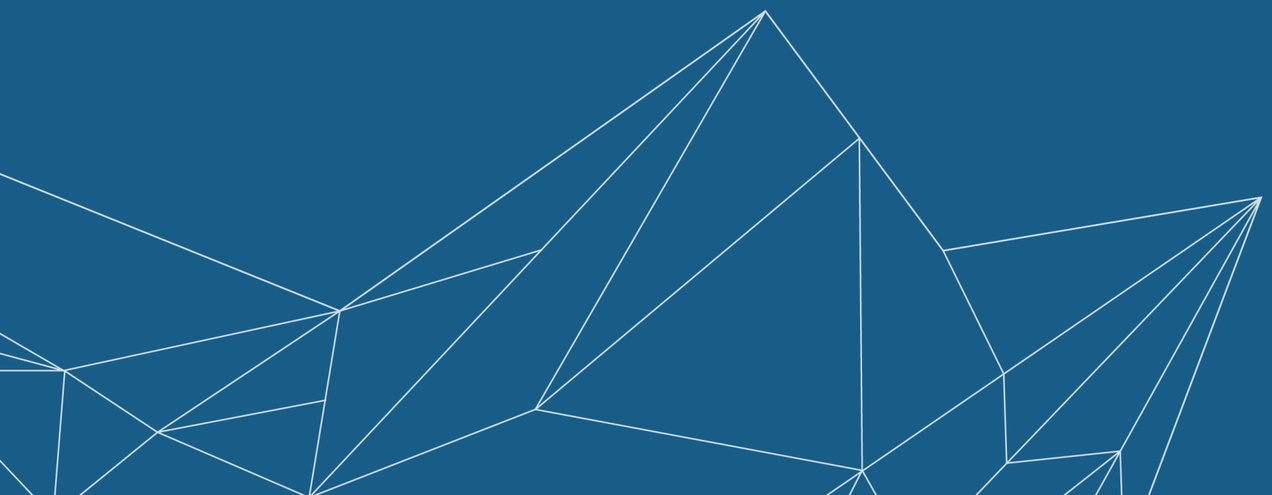
Streams must be closed

Use try-with-resources if you open (create)
and use the stream from the same method

Explicitly invoke close() if you open (create)
and use the stream in different methods

Data streams

Reading and writing Java types

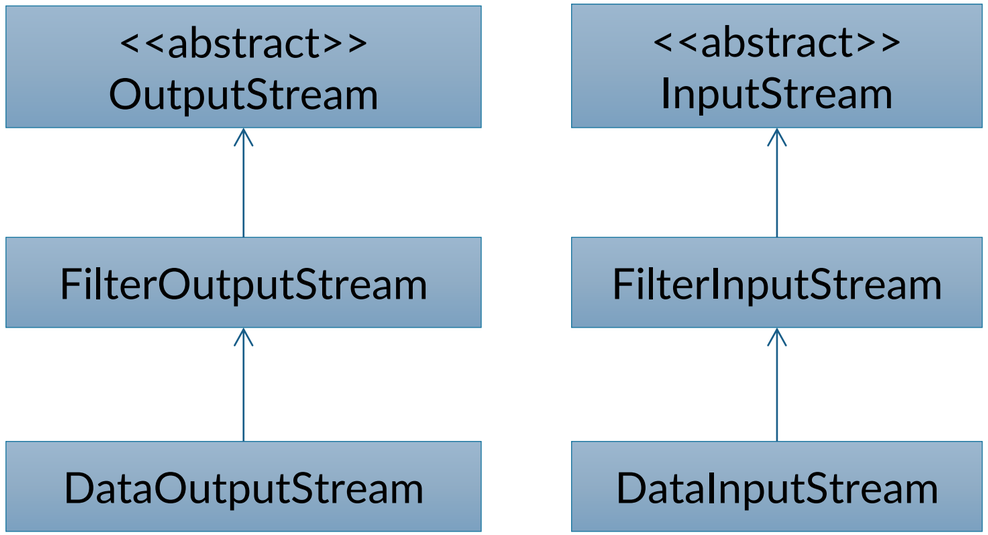


Primitive types I/O

DataOutputStream and DataInputStream enable you to write or read primitive data to or from a stream

They implement the DataOutput and DataInput interfaces, respectively. These interfaces define methods that convert primitive values to or from a sequence of bytes

These streams make it easy to store binary data, such as integers or floating-point values, in a file



DataInputStream

```
public class DataInputStream extends FilterInputStream implements DataInput {
   public DataInputStream(InputStream in)
    public final boolean readBoolean() throws IOException
    public final byte readByte() throws IOException
    public final int readUnsignedByte() throws IOException
    public final short readShort() throws IOException
    public final int readUnsignedShort() throws IOException
    public final char readChar() throws IOException
    public final int readInt() throws IOException
    public final long readLong() throws IOException
    public final float readFloat() throws IOException
    public final double readDouble() throws IOException
    public final String readUTF() throws IOException
    …
}
```

DataOutputStream

```
public class DataOutputStream extends FilterOutputStream implements DataOutput {
    public DataOutputStream(OutputStream out)
    public void flush() throws IOException
    public final void writeBoolean(boolean v) throws IOException
    public final void writeByte(int v) throws IOException
    public final void writeShort(int v) throws IOException
    public final void writeChar(int v) throws IOException
    public final void writeInt(int v) throws IOException
    public final void writeLong(long v) throws IOException
    public final void writeFloat(float v) throws IOException
    public final void writeDouble(double v) throws IOException
    public final void writeBytes(String s) throws IOException
    public final void writeChars(String s) throws IOException
    public final void writeUTF(String str) throws IOException
    …
}
```

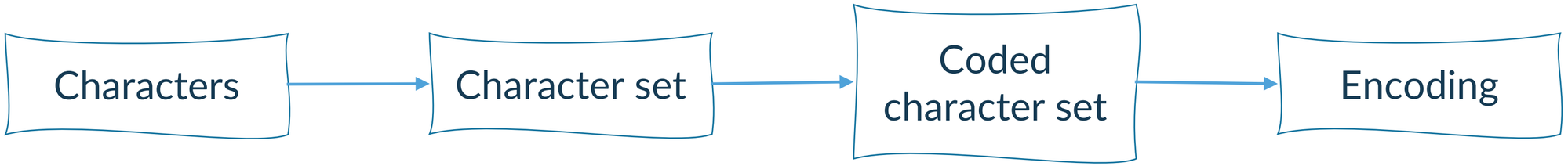
Readers and Writers

Reading and writing text

Text streams

What about reading and writing text?

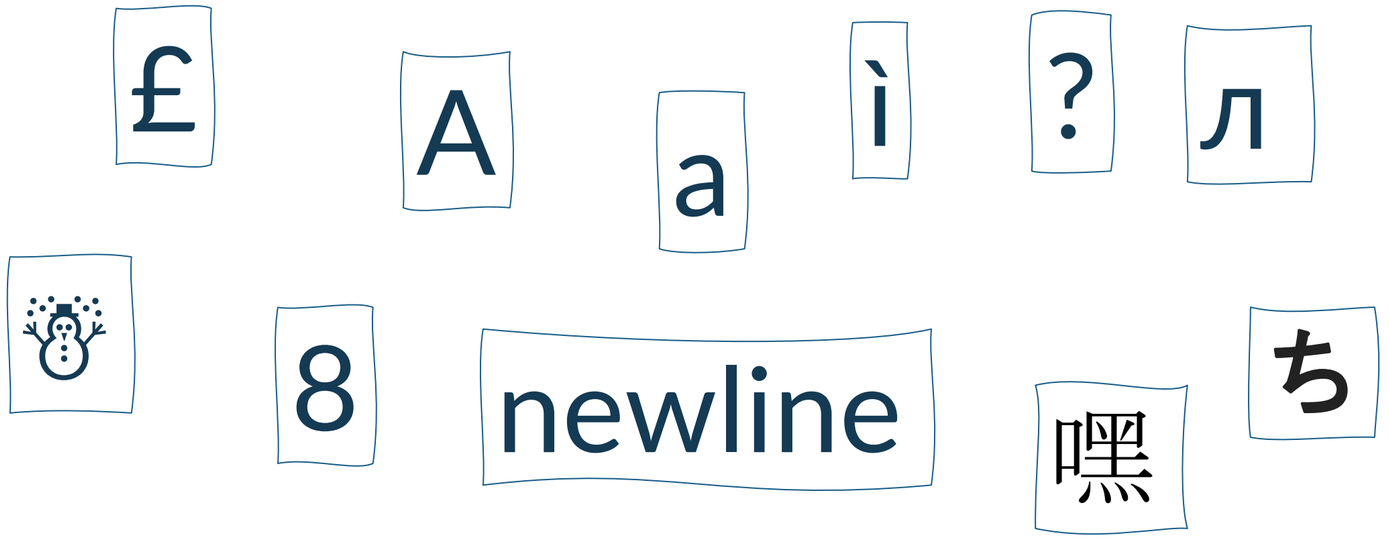
Character sets and encoding



To know everything about character sets and encodings:
https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/

Character

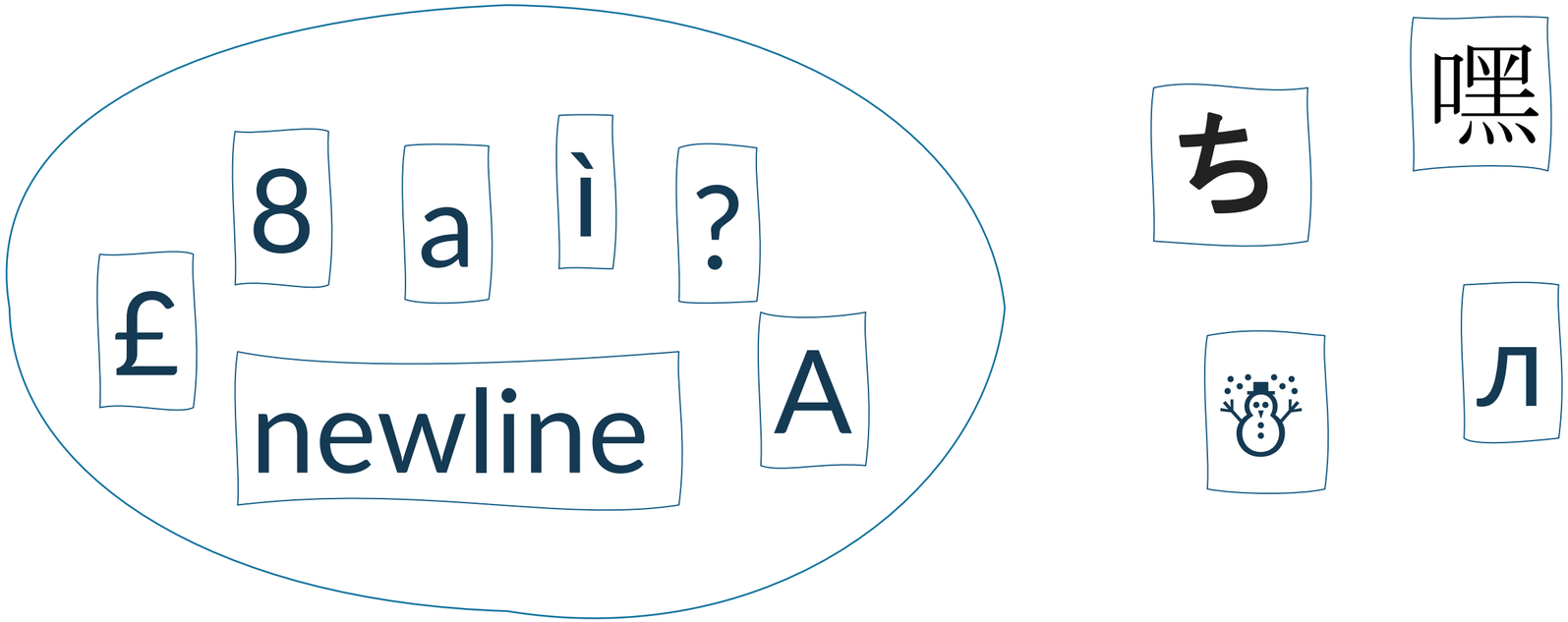
A *character* is a minimal unit of text that has semantic value.



https://docs.oracle.com/javase/tutorial/i18n/text/terminology.html

Character set

A *character set* is a collection of characters that might be used by multiple languages. For example, the Latin character set is used by English and most European languages, though the Greek character set is used only by the Greek language.



Unicode terminology – Coded character set

A *coded character set* is a character set where each character is assigned a unique number (code point).

US-ASCII

Code point	Character
0	NUL
1	SOH
…	…
65	A
66	B
67	C
…	…
126	~
127	DEL

Windows-1252/ISO-8859-1

Code point	Character
0	NUL
1	SOH
…	…
65	A
66	B
67	C
…	…
254	þ
255	ÿ

Windows-1250

Code point	Character
0	NUL
1	SOH
…	…
65	A
66	B
67	C
…	…
254	ţ
255	˙

Windows-1252 and ISO-8859-1 are not the same character set, but they differs for some code points assigned to control codes.
For HTML5 they can be considered the same https://www.w3.org/TR/encoding/

Unicode

The Unicode standard defines 144,697 characters and their respective code points. This character set is called Universal Coded Character Set (UCS, Unicode).

Positions 0 through 127 of UCS are the same as in US-ASCII.

Positions 0 through 255 of UCS and Unicode are the same as in ISO-8859-1.

Positions 0 through 65535 of UCS (Basic Multilingual Plan) cover all the commonly used languages

How do we encode this information in computers?

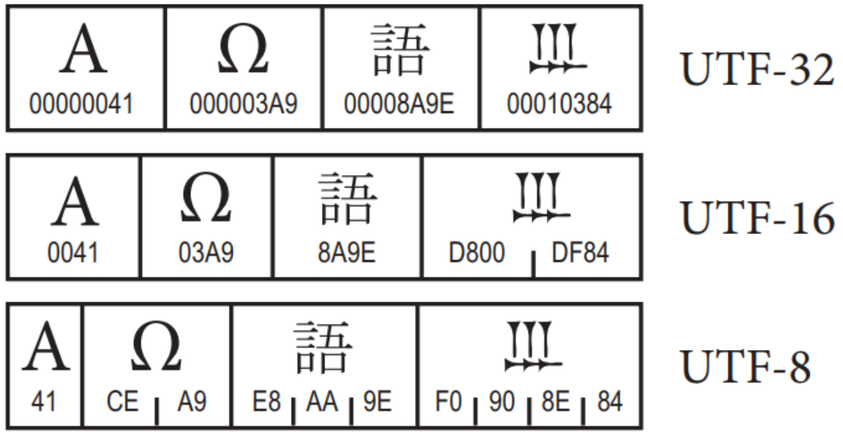
Encodings

1 byte is enough to encode the whole US-ASCII and ISO-8859-1 character sets.

For characters sets with more than 256 characters with need to use multibyte encodings.

Generally, a character sets define its own encoding and so the term charset is used to refer to both the character set and the encoding. E.g., HTTP and HTML define a charset parameter and attribute, respectively, to define the combination character set/encoding.

UCS is currently the most important character sets and it has multiple encodings, so this character set is represented by the name of the encoding, UTF-8, UTF-16, or UTF-32.



Java characters

The Java primitive type char uses 16 bit to represent characters.

So, the char type is not able to represents all Unicode characters. Indeed, Java internally represents text in 16-bits code units using UTF-16.

The char type does not represent characters but code units, this is relevant only when we are using a language outside the Basic Multilanguage Plane. E.g., the cuneiform language, Phoenician, etc.

Encodings supported by Java

Every implementation of the Java platform is required to support the following standard charsets. Usually, every implementation supports many more charsets.

Charset	Description
US-ASCII	Seven-bit ASCII, a.k.a. ISO646-US, a.k.a. the Basic Latin block of the Unicode character set
ISO-8859-1	ISO Latin Alphabet No. 1, a.k.a. ISO-LATIN-1
UTF-8	Eight-bit UCS Transformation Format
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byte order
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byte order
UTF-16	Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark

If in doubt, create documents in UTF-8.

What about reading documents in unknown encodings?

There is no such thing as Plain Text

Text streams

To write (read) text to (from) an output(input) stream we need to encode (decode) the text into (from) a binary stream

Fortunately, Java is doing this for us, given we provide a very tiny piece of information, the encoding/charset of the stream

Unfortunately, Java defines default methods that let us skip this step by using by default the default charset

Unfortunately, the default charset might vary depending on the internationalization settings or depending on the operating system

E.g., the default charset on Linux can be UTF-8 while on Windows can be Windows-1252 (in Italy)

Don’t use such methods unless you really know what you are doing

Introducing Reader

```
public abstract class Reader implements Closeable {
    …
    public int read() throws IOException;
    …
}
```

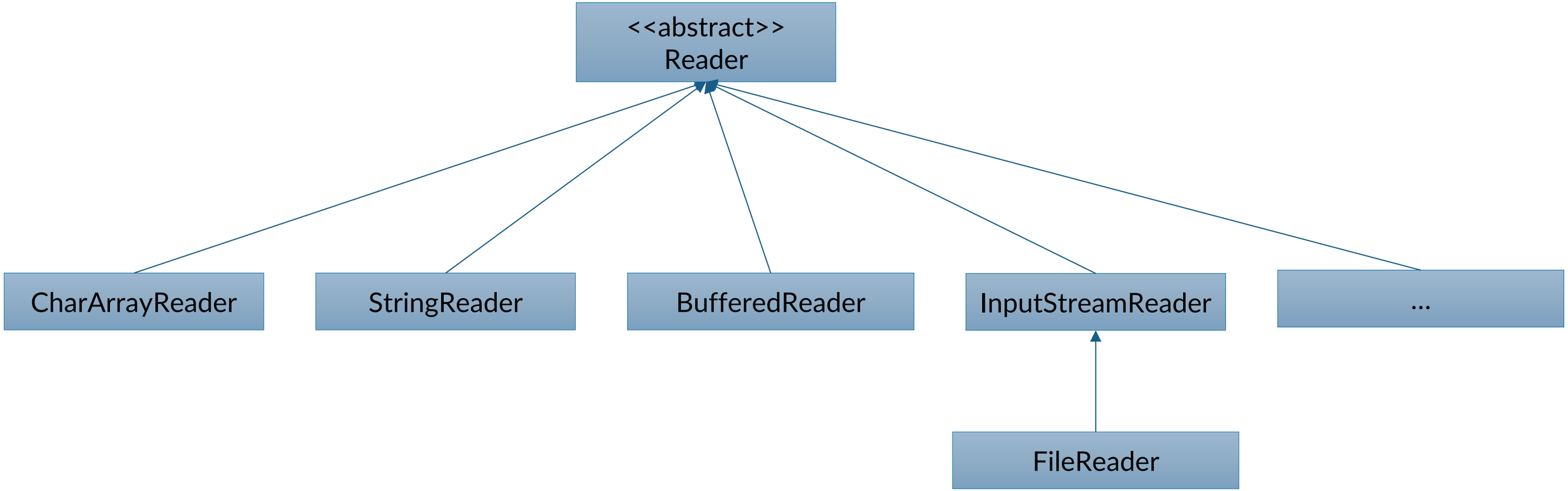
Reads a single character as an integer in the range 0 to 65535 or -1 if the end of the stream has been reached

This method will block until

- *a character is available*
- *an I/O error occurs*
- *or the end of the stream is reached.*

```
try (Reader reader = … )) {
    int ch = -1;
    while ((ch = reader.read()) != -1) {
        System.out.print((char) ch);
    }
}
```

The Reader hierarchy



Examples of Reader 1/2

```
String fileName = "A:\\git\\sdm\\src\\it\\units\\sdm\\iostreams\\Examples.java";
try (Reader reader = new InputStreamReader(new FileInputStream(fileName), UTF_8)) {
    int ch = -1;
    while ((ch = reader.read()) != -1) {
        System.out.print((char) ch);
    }
}
```

```
try (Reader reader = new FileReader(fileName, StandardCharsets.UTF_8)) {
    int ch = -1;
    while ((ch = reader.read()) != -1) {
        System.out.print((char) ch);
    }
}
```

Examples of Reader 2/2

```
URL url = new URL("https://www.google.it"):
try (InputStream urlStream = url.openStream()) {
    int read;
    while ((read = urlStream.read()) != -1) {
        System.out.print((char) read);
    }
}
```

We guess the encoding to be UTF-8

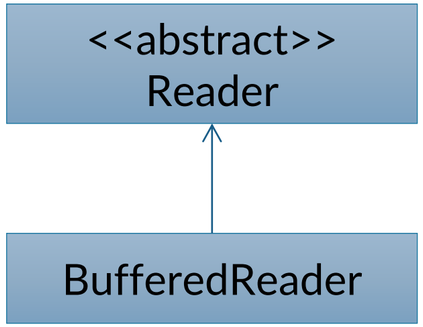
```
URL url = new URL("https://www.google.it");
try (Reader reader = new InputStreamReader(url.openStream(), StandardCharsets.UTF_8)) {
    int ch;
    while ((ch = reader.read()) != -1) {
        System.out.print((char) ch);
    }
}
```

BufferedReader

When reading from the filesystem or from the network, the reading of small chunks of data can be very inefficient

Java offers buffered input to speedup the reading of small chunks of data

The BufferedReader reads data in advance in a buffer of a specified size




```
public class BufferedReader extends Reader {

    public BufferedReader(Reader in)
    public BufferedReader(Reader in, int size)
    …
}
```

A BufferedReader is a Reader wrapping another reader

Working with BufferedReader

```
try (Reader reader = new BufferedReader(new FileReader(fileName, UTF_8))) {
    int ch = -1;
    while ((ch = reader.read()) != -1) {
        System.out.print((char) ch);
    }
}
```

```
try (BufferedReader reader = new BufferedReader(new FileReader(fileName, UTF_8))) {
    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
}
```

```
try (BufferedReader reader = new BufferedReader(new FileReader(fileName, UTF_8))) {
    reader.lines().forEach(System.out::println);
}
```

Introducing Writer

```
public abstract class Writer implements Closeable {
    …
    public void write(int c) throws IOException
    public void write(String str) throws IOException
    …
}
```

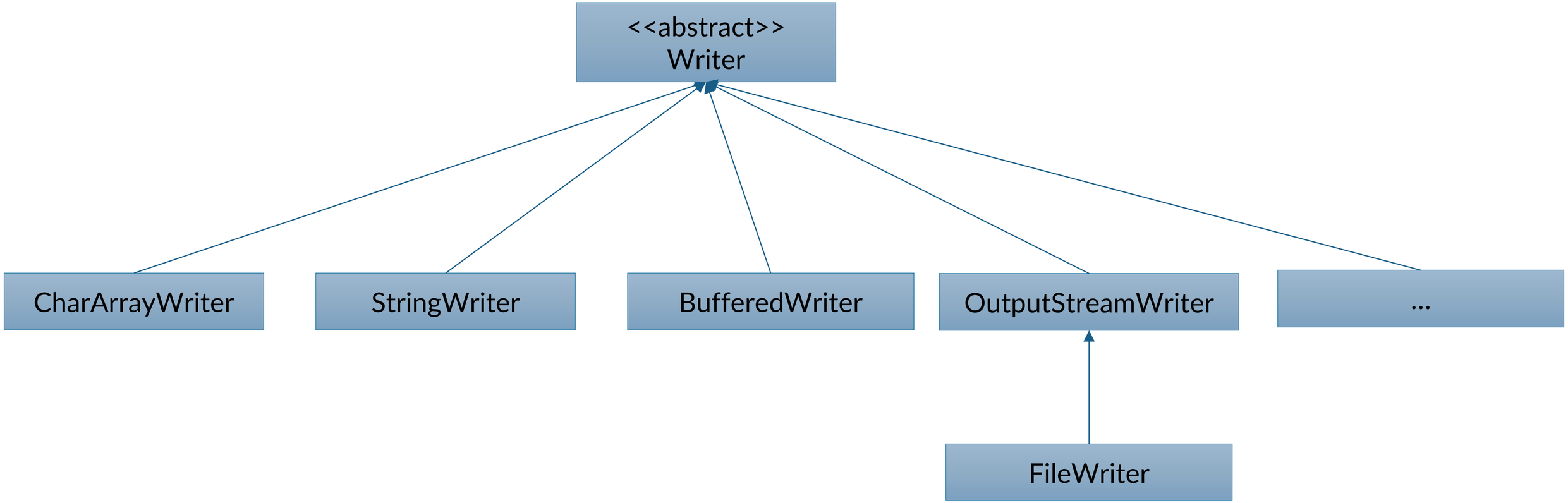
Writes a single character. The character to be written is contained in the 16 low-order bits of the given integer value; the 16 high-order bits are ignored

```
String data = "some data";

try (Writer writer = …) {
    writer.write(data);
}

try (Writer writer = …) {
    for (int i = 0; i < data.length(); i++) {
        writer.write(data.charAt(i));
    }
}
```

The Writer hierarchy



Examples of Writer

```
String data = "some data";
try (Writer writer = new FileWriter("A:\\git\\sdm\\pippo.txt", StandardCharsets.UTF_8)) {
    writer.write(data);
}
```

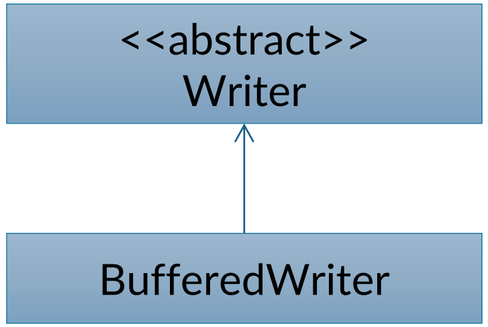
```
try (Writer writer = new OutputStreamWriter(new FileOutputStream(fileName1), UTF_8)) {
    writer.write(data);
}
```

BufferedWriter

When writing to the filesystem or to the network, the writing of small chunks of data can be very inefficient

Java offers buffered output to speedup the writing of small chunks of data

The BufferedWriter writes data to the wrapped writer only when the buffer is full or when flush() is invoked




```
public class BufferedWriter extends Writer {

    public BufferedWriter(Writer writer)
    public BufferedWriter(Writer writer, int size)
    …
}
```

A BufferedWriter is a Writer wrapping another writer

Readers and Writers must be closed

Use try-with-resources if you open (create) and use the stream from the same method

Explicitly invoke close() if you open (create) and use the stream from different methods

Console I/O

Reading and writing from the console

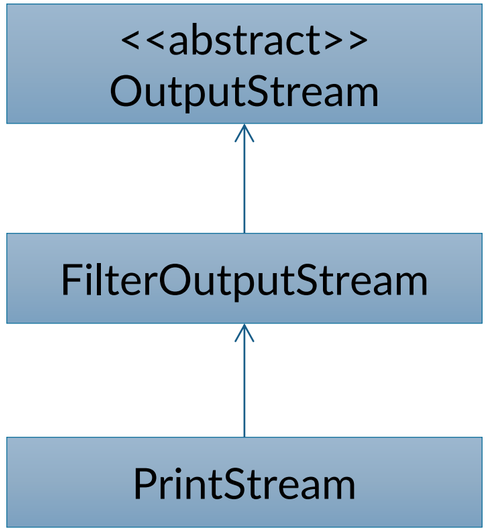
Console I/O

System.in is an object of type InputStream

System.out and System.err are objects of type PrintStream.

These are byte streams, even though they are typically used to read and write characters from and to the console

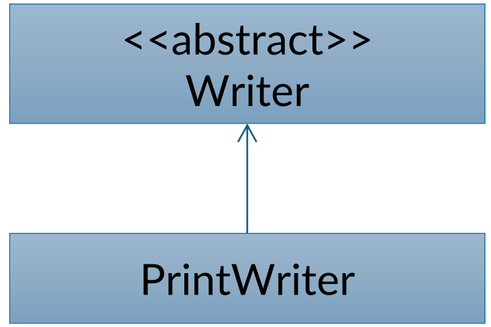
Digression - PrintStream & PrintWriter



PrintStream & PrintWriter are a stream and a writer providing functionalities to conveniently print Java data types in text format into streams

I.e., printing a byte to a PrintStream or a PrintWriter results in writing its textual representation rather than its binary representation

PrintStream and PrintWriter are directly used to write text



PrintXXX API

```
void println()
void print/println(boolean x)
void print/println(char x)
void print/println(char[] x)
void print/println(double x)
void print/println(float x)
void print/println(int x)
void print/println(long x)
void print/println(Object x)
void print/println(String x)

PrintWriter format(String format, Object... args)
PrintWriter format(Locale l, String format, Object... args)
```

None of these methods throws any IOException, use checkError() to test the error status.

Reading from the console

```
public class Echo {

    public static void main(String[] args) throws IOException {
        BufferedReader reader = new BufferedReader(
                new InputStreamReader(System.in, System.console().charset()));

        String line;
        while (!(line = reader.readLine()).isEmpty()) {
            System.out.println("READ: " + line);
        }
    }
}
```

DO NOT CLOSE System.in

The `java.lang.io.Console` class

```
Charset charset()
void flush()
Console format(String format, Object... args)
Console printf(String format, Object... args)
Reader reader()
String readLine()
String readLine(String format, Object... args)
char[] readPassword()
char[] readPassword(String format, Object... args)
PrintWriter writer()
```

The methods format() and printf() are exactly the same
Remember to flush when you use writer()

Reading from the Console

```
public class Echo {

    public static void main(String[] args) throws IOException {
        String line;
        while (!(line = System.console().readLine()).isEmpty()) {
            System.console().printf("READ: %s\n", line);
        }
    }
}
```

Assignments

Assignment 1

Write a program that given a file writes the hexdump of the file in another file.

E.g., the command
`java it.units.sdm.HexDump Streams.pdf`

Shall produce a file Streams.pdf.hexdump with the following content (don't print characters outside the US-ASCII charset.

```
2550 4446 2d31 2e37 0d0a 25b5 b5b5 b50d  %PDF-1.7..%.....
0a31 2030 206f 626a 0d0a 3c3c 2f54 7970  .1 0 obj..<</Typ
652f 4361 7461 6c6f 672f 5061 6765 7320  e/Catalog/Pages
3220 3020 522f 4c61 6e67 2865 6e29 202f  2 0 R/Lang(en) /
5374 7275 6374 5472 6565 526f 6f74 2033  StructTreeRoot 3
...
```

Write a program that do the reverse, i.e., creates a files from the hexdump, and verify that the recreated file is still usable!

Assignment 2

Write a program to change the charset of a text file

```
java it.units.sdm.Recode pippo.txt UTF-8 Windows-1252
```

pippo.txt should contain some non US-ASCII character, e.g., ‘è’.

The output file could be named pippo.txt.Windows-1252

Extra step, throw an exception if the program encounter an unmappable byte sequence, e.g., when reading a file containing the ‘嘿’ character by using the Windows-1252 charset. Hint: explore the java.io.Files class

Assignment 3

Write a class (or a set of classes) that given a text file ~~string~~ it produces a Term Frequency table.
Consider the option to provide a list of stop words, normalization, etc.
Provide an option to print the table in alphabetical order and by frequency.

"Term frequency (TF) means how often a term occurs in a document. In the context of natural language, terms correspond to words or phrases …"

Term	Frequency
english	8
language	7
words	12

…

input	7
cactus	1
fireworks	3

Solution of assignments

Assignment 1

Write a program that given a file writes the hexdump of the file in another file.

E.g., the command
`java it.units.sdm.HexDump Streams.pdf`

Shall produce a file Streams.pdf.hexdump with the following content (don't print characters outside the US-ASCII charset.

```
2550 4446 2d31 2e37 0d0a 25b5 b5b5 b50d  %PDF-1.7..%.....
0a31 2030 206f 626a 0d0a 3c3c 2f54 7970  .1 0 obj..<</Typ
652f 4361 7461 6c6f 672f 5061 6765 7320  e/Catalog/Pages
3220 3020 522f 4c61 6e67 2865 6e29 202f  2 0 R/Lang(en) /
5374 7275 6374 5472 6565 526f 6f74 2033  StructTreeRoot 3
...
```

Write a program that do the reverse, i.e., creates a files from the hexdump, and verify that the recreated file is still usable!

HexDump.java

```java
public class HexDump {

    public static void main(String[] args) throws IOException {
        HexDump hexDump = new HexDump();
        try (FileInputStream inputStream = new FileInputStream(args[0])) {
            hexDump.dump(inputStream, new PrintWriter(System.out, true));
        }
    }

    public void dump(InputStream inputStream, Writer writer) throws IOException {
        StringBuilder binary = new StringBuilder();
        StringBuilder text = new StringBuilder();
        String lineSeparator = "";
        int groups = 0;
        int read;
        while ((read = inputStream.read()) != -1) {
            binary.append(HexFormat.of().toHexDigits((byte) read));
            text.append(read < 32 || read > 126 ? '.' : (char) read);
            if (++groups % 2 == 0) {
                binary.append(' ');
            }
            if (groups % 16 == 0) {
                writer.append(lineSeparator).append(binary).append(" ").append(text);
                binary.delete(0, binary.length());
                text.delete(0, text.length());
                lineSeparator = "\n";
            }

        }
        if (!binary.isEmpty()) {
            writer.append(binary).append(" ".repeat(41 - binary.length())).append(text);
        }
    }
}
```

HexDumpTest.java – 1/2

```
@Test
void testOneGroup() throws IOException {
    HexDump hexDump = new HexDump();
    InputStream inputStream = new ByteArrayInputStream(new byte[] {0x25, 0x50});
    StringWriter writer = new StringWriter();

    hexDump.dump(inputStream, writer);

    assertEquals("2550                                                %P", writer.toString());
}

@Test
void testTwoGroups() throws IOException {
    HexDump hexDump = new HexDump();
    InputStream inputStream = new ByteArrayInputStream(new byte[] {0x25, 0x50, 0x44, 0x46});
    StringWriter writer = new StringWriter();

    hexDump.dump(inputStream, writer);

    assertEquals("2550 4446                                           %PDF", writer.toString());
}
```

HexDumpTest.java – 2/2

```
@Test
void testNonPrintables() throws IOException {
    HexDump hexDump = new HexDump();
    InputStream inputStream = new ByteArrayInputStream(new byte[] {0x25, 0x50, 0x44, 0x46, 0x2d, 0x31, 0x2e, 0x37, 0x0d,
0x0a, 0x25, (byte) 0xb5, (byte) 0xb5, (byte) 0xb5, (byte) 0xb5, 0x0d});
    StringWriter writer = new StringWriter();

    hexDump.dump(inputStream, writer);

    assertEquals("2550 4446 2d31 2e37 0d0a 25b5 b5b5 b50d  %PDF-1.7..%.....", writer.toString());
}

@Test
void testMultipleLines() throws IOException {
    HexDump hexDump = new HexDump();
    InputStream inputStream = new ByteArrayInputStream(new byte[] {
            0x25, 0x50, 0x44, 0x46, 0x2d, 0x31, 0x2e, 0x37, 0x0d, 0x0a, 0x25, (byte) 0xb5, (byte) 0xb5, (byte) 0xb5,
(byte) 0xb5, 0x0d,
            0x0a, 0x31, 0x20, 0x30, 0x20, 0x6f, 0x62, 0x6a, 0x0d, 0x0a, 0x3c, 0x3c, 0x2f, 0x54, 0x79, 0x70
    });
    StringWriter writer = new StringWriter();

    hexDump.dump(inputStream, writer);

    assertEquals("""
            2550 4446 2d31 2e37 0d0a 25b5 b5b5 b50d  %PDF-1.7..%.....
            0a31 2030 206f 626a 0d0a 3c3c 2f54 7970  .1 0 obj..<</Typ""", writer.toString());
}
```

Thank you!

esteco.com