Java IOs

RES, Lecture 2 (second part)

Olivier Liechti
Juergen Ehrensberger

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Agenda

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **Week 1**
  - Universal API
  - Sources, Sinks and Streams
  - Performance and Buffering
- **Week 2**
  - The Decorator Pattern and The Mighty Filter Classes
  - Binary vs. Character-Oriented IOs
  - Shit Happens... Dealing with IO Exceptions

The Mighty Filter Classes

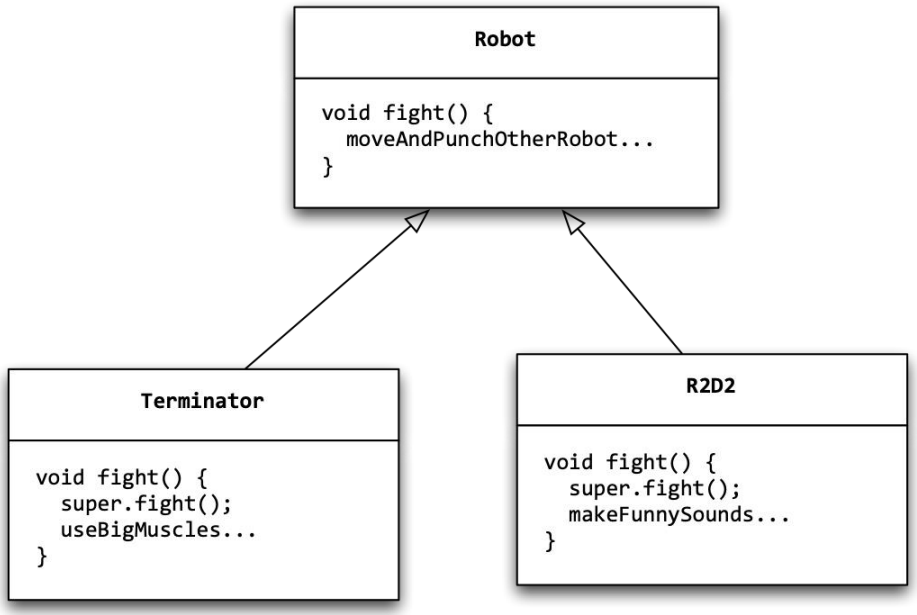
The Decorator Design Pattern

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

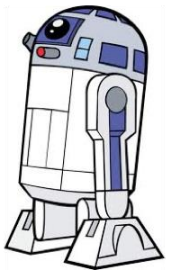
- The **decorator design pattern** is a solution that is often used when creating object-oriented models.
- It makes it possible to **add behavior to an existing class**, without modifying the code of this class. In other words, it makes it possible to **decorate** an existing class with additional behaviors.
- The design pattern also makes it possible to **define a collection of decorators**, and to **combine** them in arbitrary ways at runtime. In other words, it is possible to decorate a class with several behaviors.

Example : Fighting Robots

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

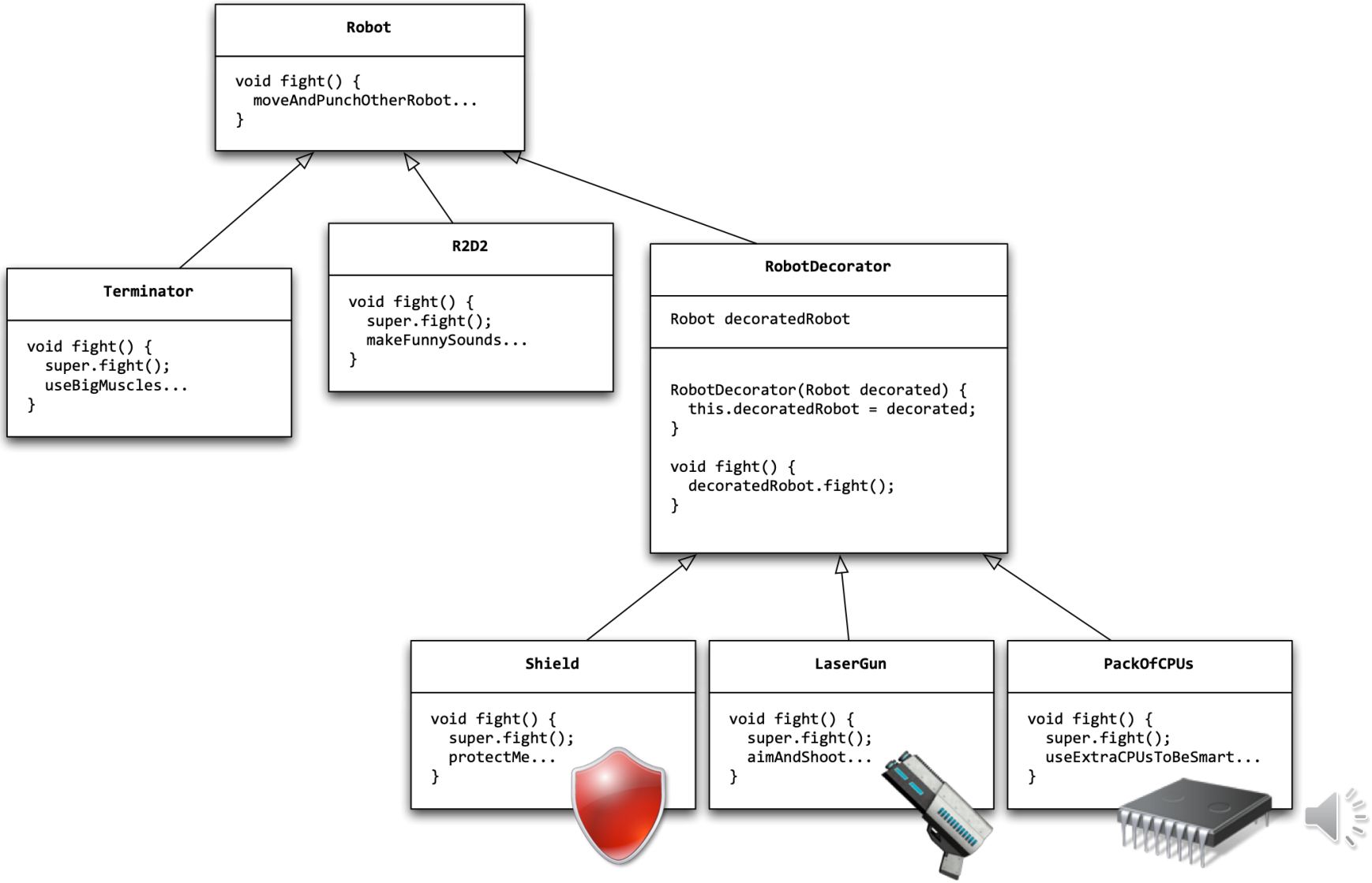



```
Robot bigOne = new Terminator();
Robot smallOne = new R2D2();

bigOne.fight();
smallOne.fight();

--
bigOne   > useBigMuscles
smallOne > makeFunnySounds

bigOne wins.
```



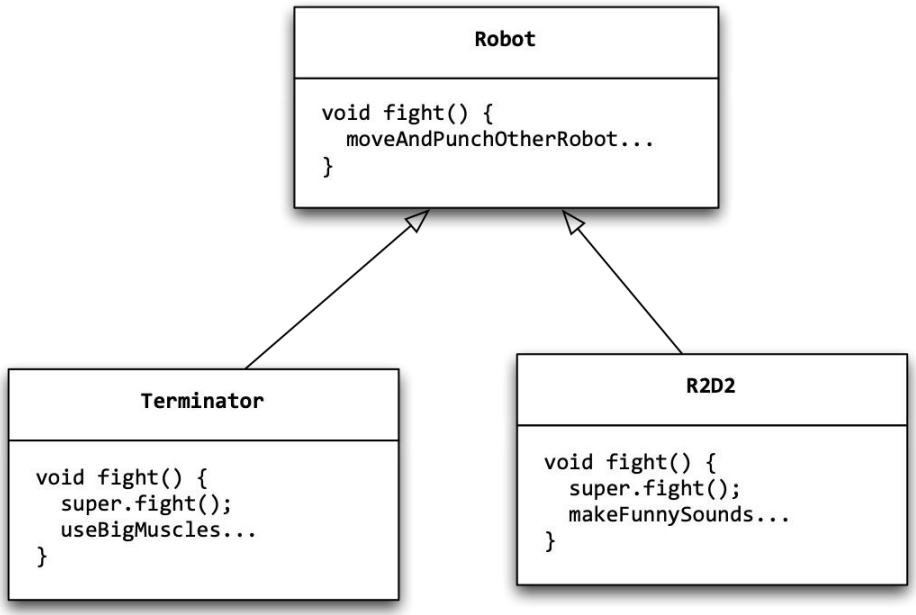
Example : Decorators for Robots

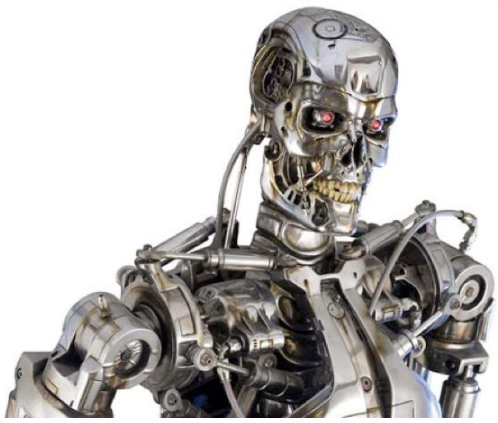
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



Example : Decorated Robots

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

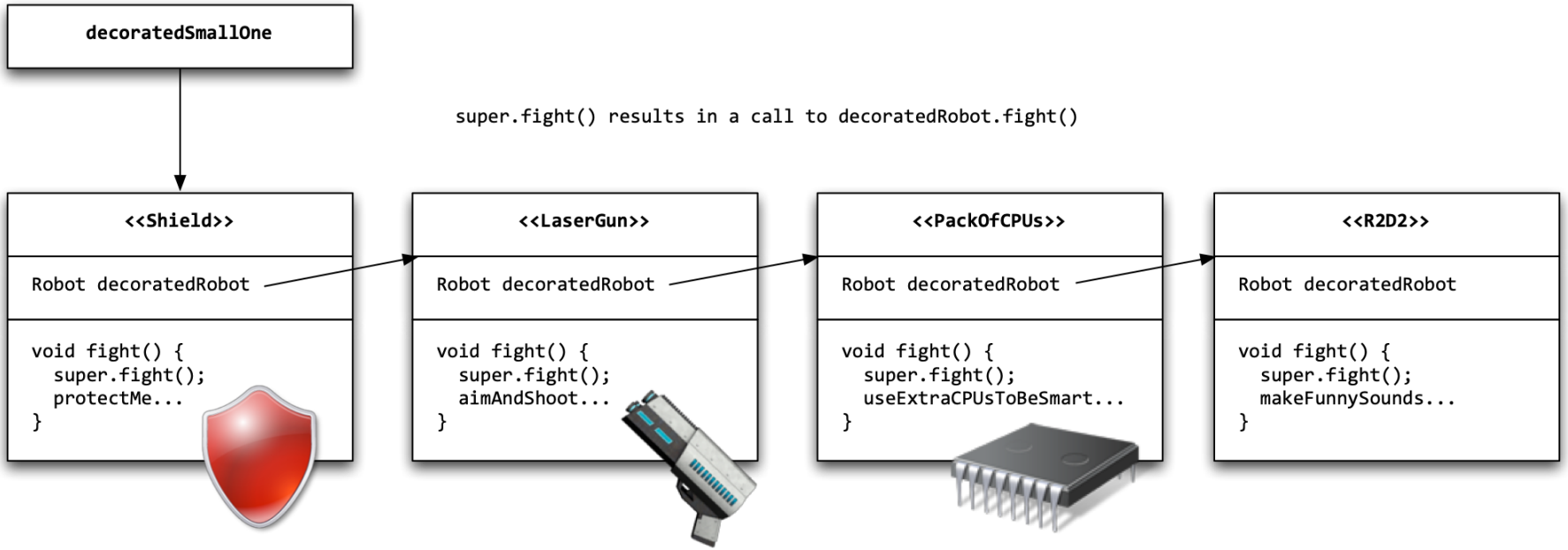




```
Robot bigOne = new Terminator();
Robot decoratedSmallOne =
  new Shield(
    new LaserGun(
      new PackOfCPU(
        new R2D2()
      )
    )
  );

bigOne.fight();
decoratedSmallOne.fight();

--
bigOne    > useBigMuscles
smallOne > makeFunnySounds,
useExtraCPUsToBeSmart, aimAndShoot,
protectMe

decoratedSmallOne wins.
```

Invocation Chain

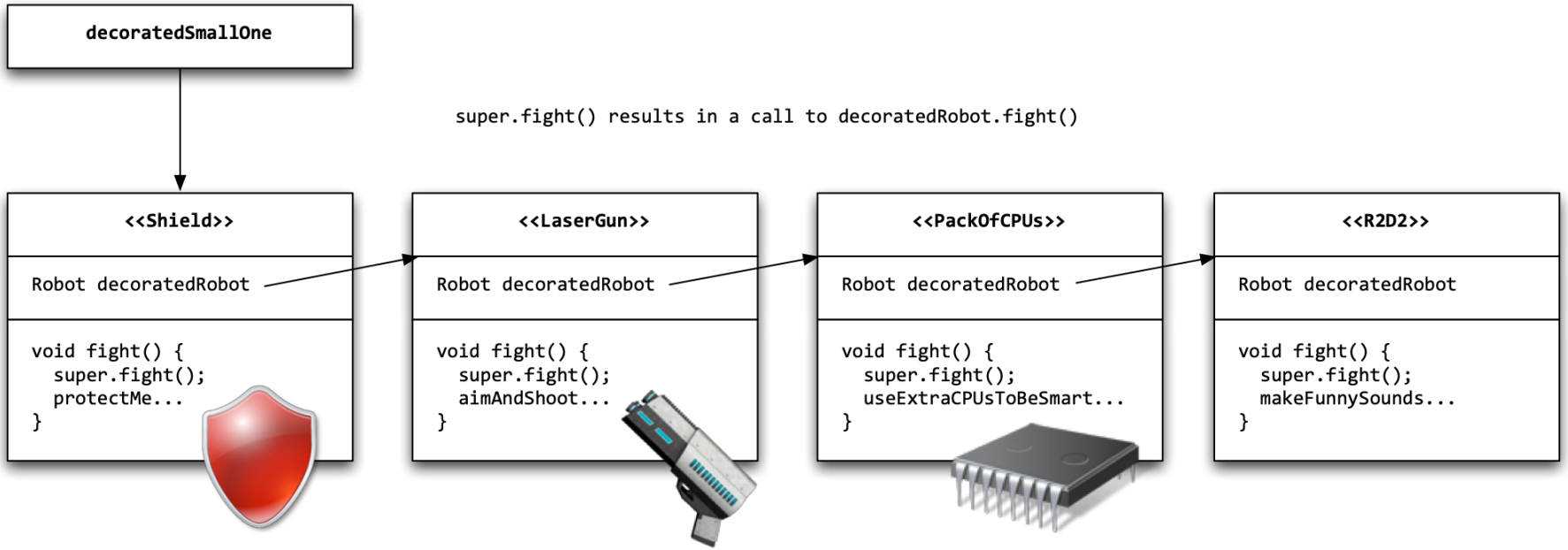
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch




```
decoratedSmallOne.fight()
-> shield.fight();
-> laserGun.fight() + protectMe;
-> packOfCPUs.fight() + aimAndShoot + protectMe
-> R2D2.fight() + useExtraCPUsToBeSmart + aimAndShoot + protectMe
-> makeFunnySounds + useExtraCPUsToBeSmart + aimAndShoot +
protectMe.
```

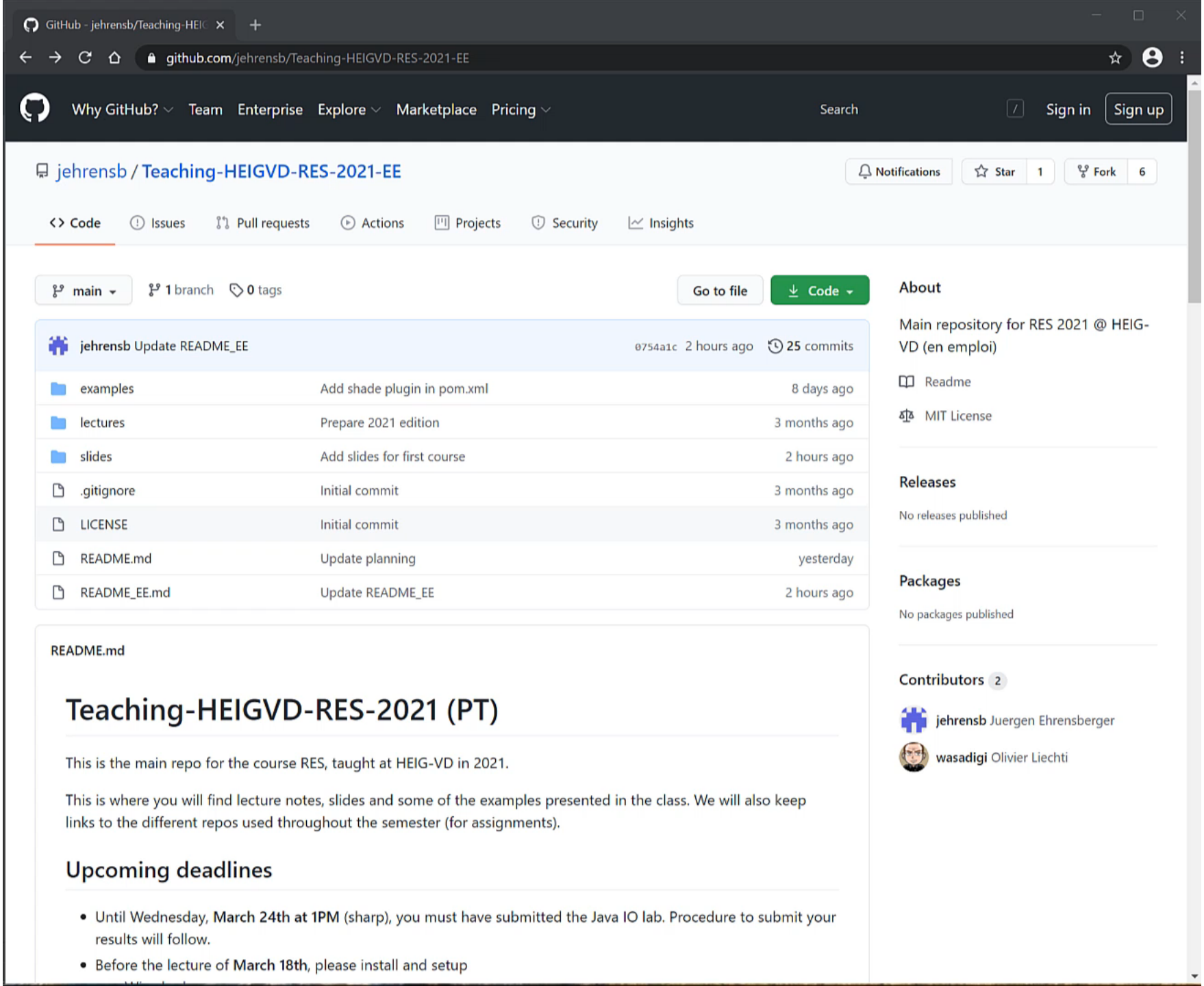
Invocation Chain

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch




```
decoratedSmallOne.fight()
-> shield.fight();
-> laserGun.fight() + protectMe;
-> packOfCPUs.fight() + aimAndShoot + protectMe
-> R2D2.fight() + useExtraCPUsToBeSmart + aimAndShoot + protectMe
-> makeFunnySounds + useExtraCPUsToBeSmart + aimAndShoot +
protectMe.
```

Example: **02-FileIOExample** (2)



README.md

Teaching-HEIGVD-RES-2021 (PT)

This is the main repo for the course RES, taught at HEIG-VD in 2021.

This is where you will find lecture notes, slides and some of the examples presented in the class. We will also keep links to the different repos used throughout the semester (for assignments).

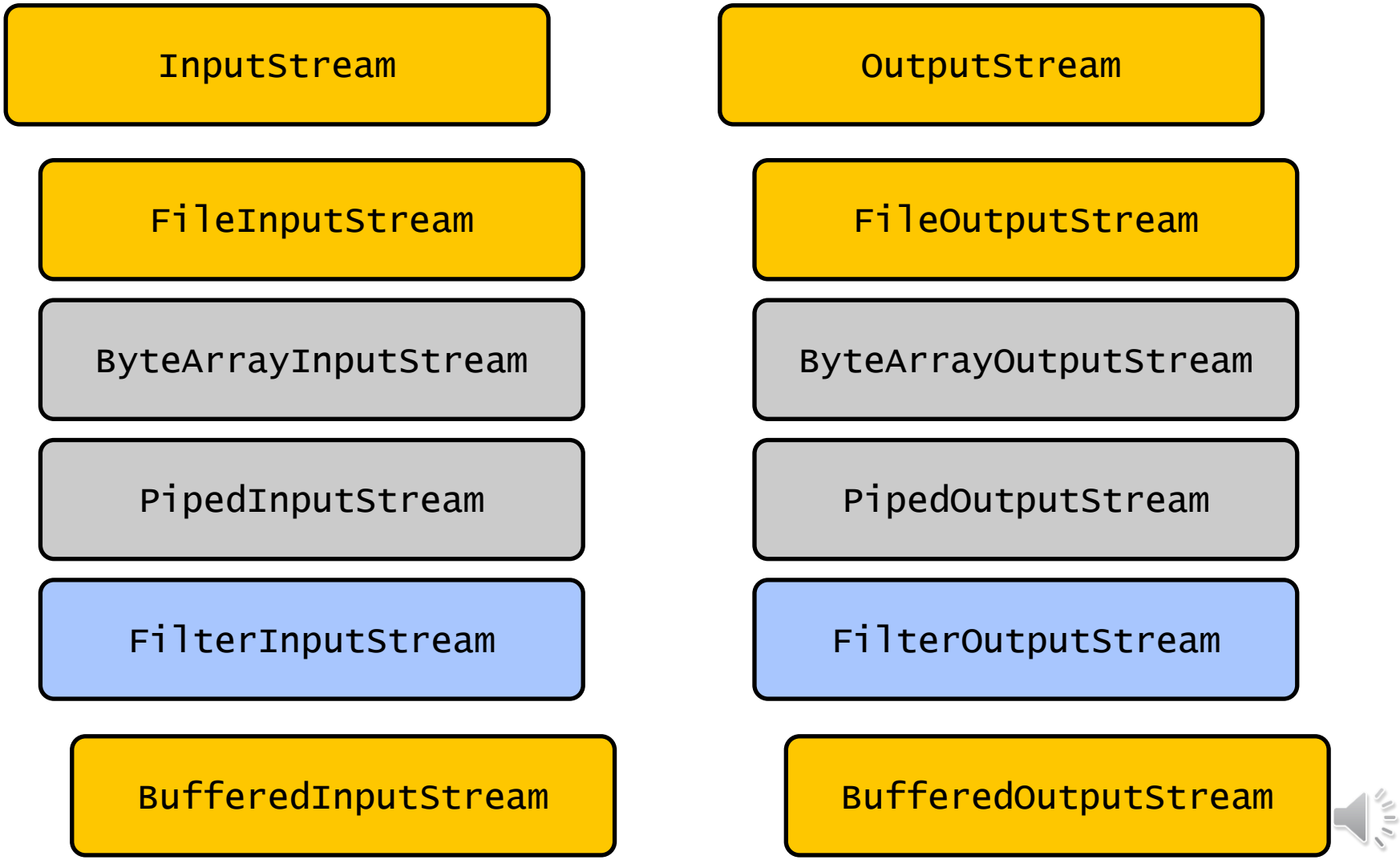
Upcoming deadlines

- Until Wednesday, **March 24th at 1PM** (sharp), you must have submitted the Java IO lab. Procedure to submit your results will follow.
- Before the lecture of **March 18th**, please install and setup

Binary vs Character-Oriented IOs

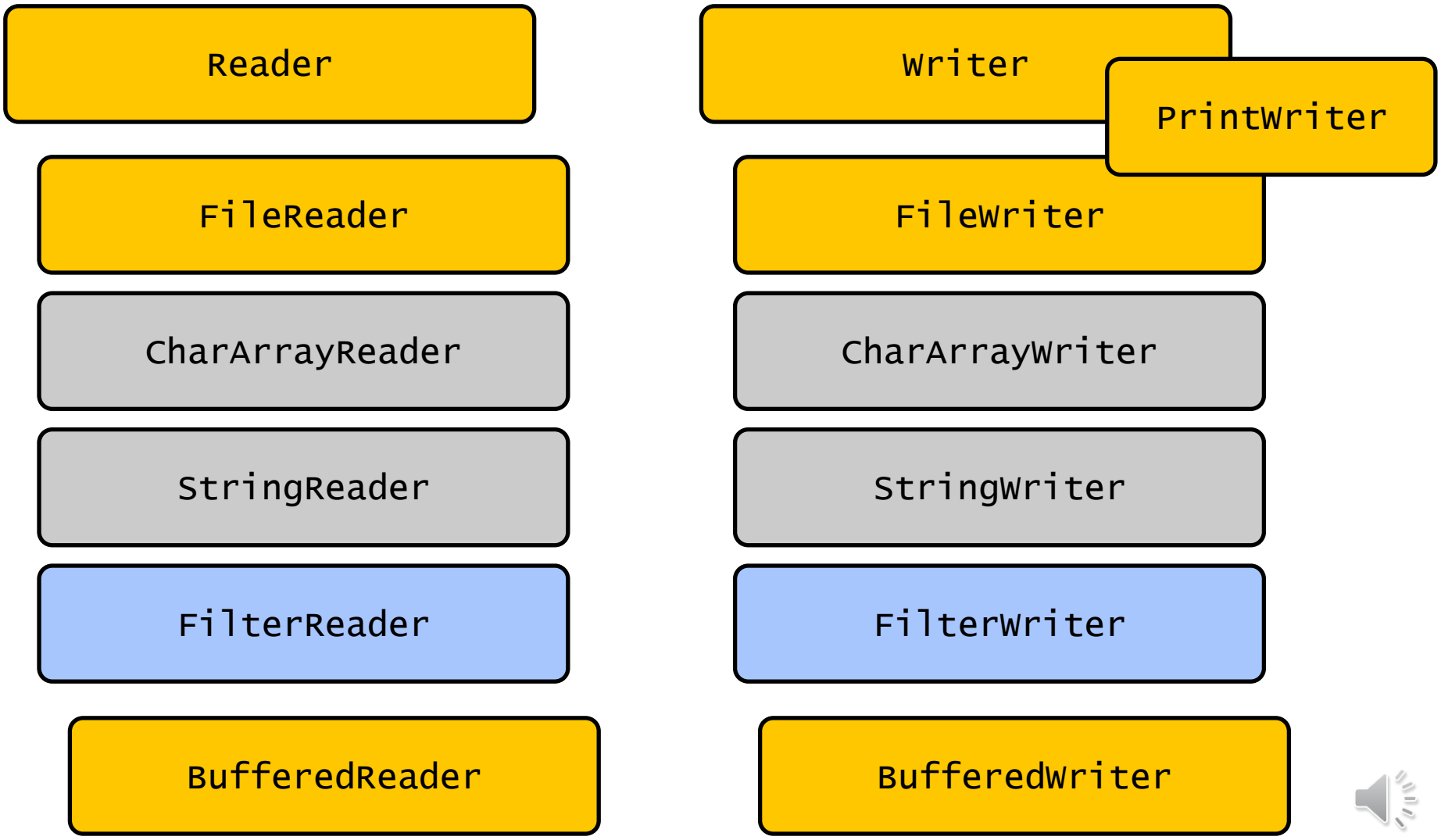
Classes in the `java.io` package (1)

heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch



Classes in the `java.io` package (2)

heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch



Bytes vs. Characters

“A byte is a sequence of 8 bits. Period.”

1000001 → 1000001 → 1000001

*“A character is the **interpretation** of a sequence of n bits. Producer and consumer need to **agree** on how to do this interpretation.”*

✓ ‘A’ ⇝ 1000001 ⇝ ‘A’

✗ ‘A’ ⇝ 1000001 ⇝ ‘▣’

Example : the ASCII encoding

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

'A' 1000001 65

64-32-16-08-04-02-01

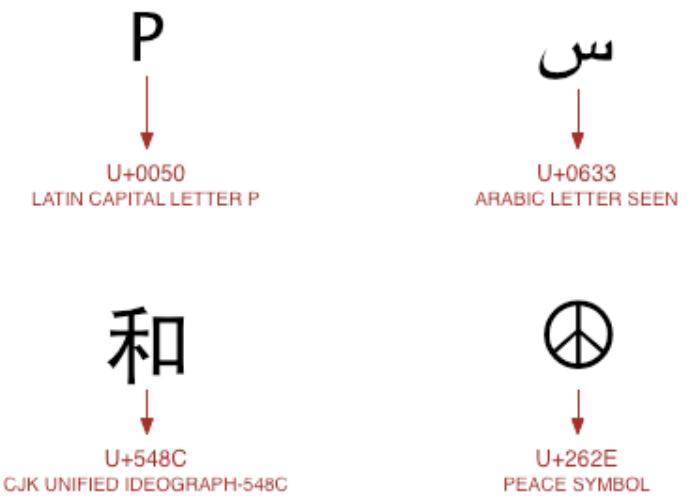
USASCII code chart

$b_7 b_6 b_5$ → / $b_4 b_3 b_2 b_1$ ↓	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Example : Unicode & UTF-8 encoding

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **Unicode** is an industry standard for representing text in almost all world languages (e.g. japanese, hebrew, arabic, etc.).
- With Unicode, **every character is associated with a "code point"**, which is nothing else than a number. It is expressed as 'U+' followed by an **hexadecimal** value.
- For instance, 'A' has the code point **U+41** (41 is 65 in hexadecimal).
- The code points for **latin characters** have the same value as in the ASCII encoding.
- **UTF-8** is **one of the encoding systems** used to represent characters of the Unicode character set.
- UTF-8 is a **variable-length encoding system**. Some characters are encoded with 1 byte, others with 2 bytes, etc.



http://wiki.secondlife.com/wiki/Unicode_In_5_Minutes

Java & Unicode

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Java uses the **unicode character encoding system**. This means that your program can manipulate characters in different languages and alphabets.
- Every **char** variable is defined by **2 bytes**, i.e. 16 bits.
- Think about **what happens when you read data from a source**. You will see a series of 1's and 0's. You know that these bits represent characters, but how do you know how to interpret them? The answer will depend on the source! Or more precisely, **it will depend on the encoding system used by the source**.
- **Same problem when you produce data**. You have text data in memory (char and String variables). You want to understand and control how this data is transformed in a series of bits. This is important if you want that other parties are able to read what you have produced!

Exploring the Unicode Charset

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

http://www.fileformat.info/info/unicode/char/41/index.htm

Unicode Character 'LATIN CAPITAL LETTER A' (U+0041)

A

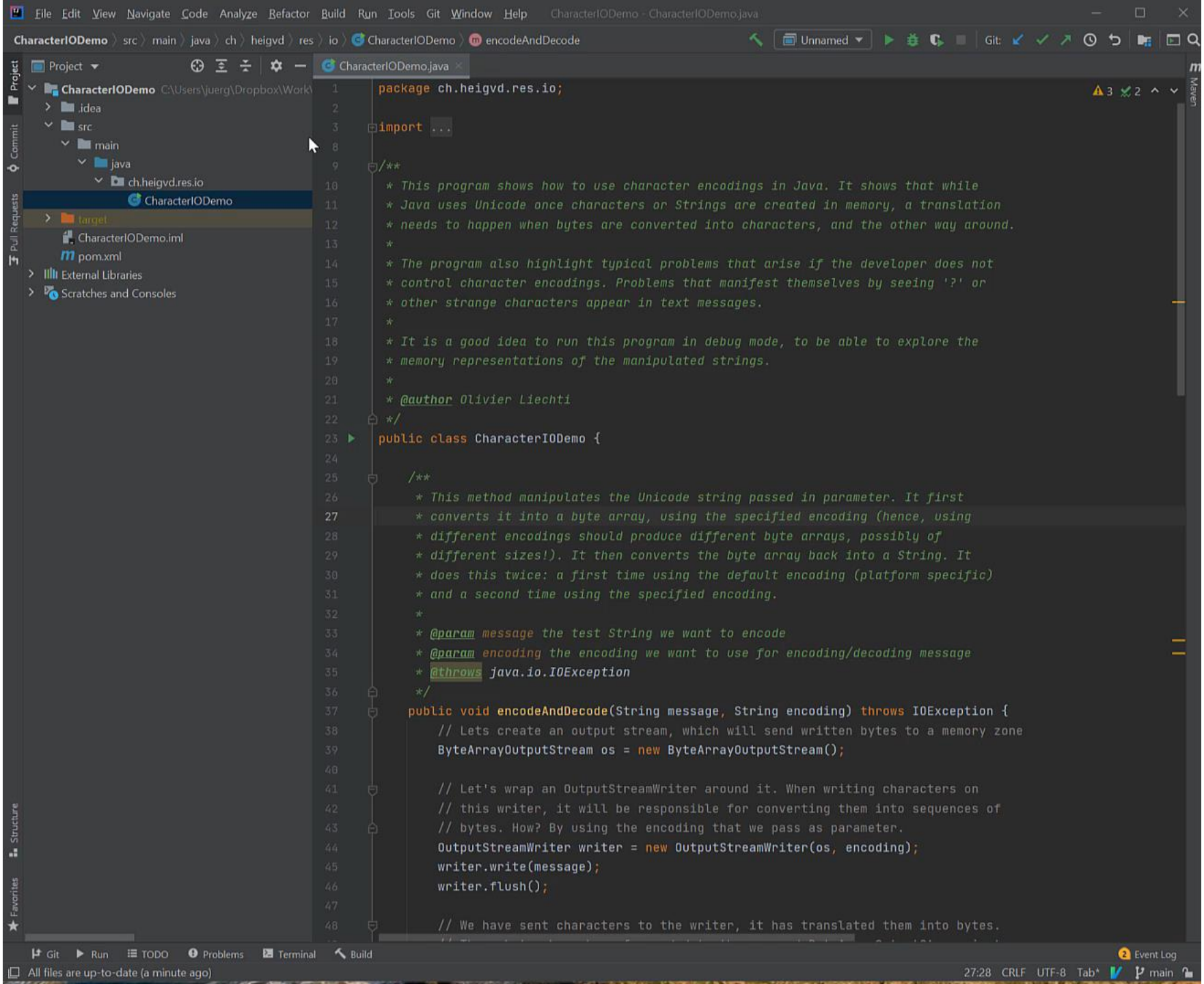
Browser Test Page
Outline (as SVG file)
Fonts that support U+0041

Unicode Data	
Name	LATIN CAPITAL LETTER A
Block	Basic Latin
Category	Letter, Uppercase [Lu]
Combine	0
BIDI	Left-to-Right [L]
Mirror	N
Index entries	Latin Uppercase Alphabet Uppercase Alphabet, Latin Capital Letters, Latin
Lower case	U+0061
Version	Unicode 1.1.0 (June, 1993)

Encodings	
HTML Entity (decimal)	A
HTML Entity (hex)	A
How to type in Microsoft Windows	Alt +41 Alt 065 Alt 65
UTF-8 (hex)	0x41 (41)
UTF-8 (binary)	01000001
UTF-16 (hex)	0x0041 (0041)
UTF-16 (decimal)	65
UTF-32 (hex)	0x00000041 (41)
UTF-32 (decimal)	65
C/C++/Java source code	"\u0041"
Python source code	u"\u0041"
More...	

Example: **03-CharacterIODemo**

*How do I avoid these damn **?l?ves** when I want to see **élèves**?*


File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help
CharacterIODemo - CharacterIODemo.java
CharacterIODemo › src › main › java › ch › heigvd › res › io › CharacterIODemo › encodeAndDecode
Unnamed
Git:
Project
CharacterIODemo C:\Users\juerg\Dropbox\Work\
.idea
src
main
java
ch.heigvd.res.io
CharacterIODemo
target
CharacterIODemo.iml
pom.xml
External Libraries
Scratches and Consoles
CharacterIODemo.java
package ch.heigvd.res.io;
import ...
/**
* This program shows how to use character encodings in Java. It shows that while
* Java uses Unicode once characters or Strings are created in memory, a translation
* needs to happen when bytes are converted into characters, and the other way around.
*
* The program also highlight typical problems that arise if the developer does not
* control character encodings. Problems that manifest themselves by seeing '?' or
* other strange characters appear in text messages.
*
* It is a good idea to run this program in debug mode, to be able to explore the
* memory representations of the manipulated strings.
*
* @author Olivier Liechti
*/
public class CharacterIODemo {
/**
* This method manipulates the Unicode string passed in parameter. It first
* converts it into a byte array, using the specified encoding (hence, using
* different encodings should produce different byte arrays, possibly of
* different sizes!). It then converts the byte array back into a String. It
* does this twice: a first time using the default encoding (platform specific)
* and a second time using the specified encoding.
*
* @param message the test String we want to encode
* @param encoding the encoding we want to use for encoding/decoding message
* @throws java.io.IOException
*/
public void encodeAndDecode(String message, String encoding) throws IOException {
// Lets create an output stream, which will send written bytes to a memory zone
ByteArrayOutputStream os = new ByteArrayOutputStream();
// Let's wrap an OutputStreamWriter around it. When writing characters on
// this writer, it will be responsible for converting them into sequences of
// bytes. How? By using the encoding that we pass as parameter.
OutputStreamWriter writer = new OutputStreamWriter(os, encoding);
writer.write(message);
writer.flush();
// We have sent characters to the writer, it has translated them into bytes.
Git Run TODO Problems Terminal Build
Event Log
All files are up-to-date (a minute ago)
27:28 CRLF UTF-8 Tab* main

Beware of the FileReader and FileWriter!

heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

no way to specify an encoding!!!

Constructor Summary

Constructors

Constructor and Description
`FileReader(File file)` Creates a new `FileReader`, given the `File` to read from.
`FileReader(FileDescriptor fd)` Creates a new `FileReader`, given the `FileDescriptor` to read from.
`FileReader(String fileName)` Creates a new `FileReader`, given the name of the file to read from.

InputStreamReader to the rescue

Constructor Summary

Constructors

Constructor and Description
`InputStreamReader(InputStream in)` Creates an InputStreamReader that uses the default charset.
`InputStreamReader(InputStream in, Charset cs)` Creates an InputStreamReader that uses the given charset.
`InputStreamReader(InputStream in, CharsetDecoder dec)` Creates an InputStreamReader that uses the given charset decoder.
`InputStreamReader(InputStream in, String charsetName)` Creates an InputStreamReader that uses the named charset.

```
Reader reader = new InputStreamReader(new FileInputStream(file), "UTF-8");
```
is safer than
```
Reader reader = new FileReader(file); // what is the "default" encoding???
```

Bytes vs. Characters

“A byte is a sequence of 8 bits. Period.”

1000001 → 1000001 → 1000001

*“A character is the **interpretation** of a sequence of n bits. Producer and consumer need to **agree** on how to do this interpretation.”*

✓ ‘A’ ⇝ 1000001 ⇝ ‘A’

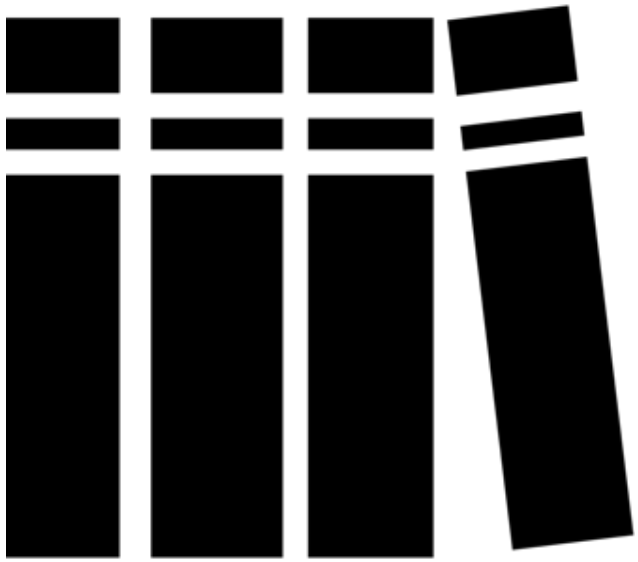
✗ ‘A’ ⇝ 1000001 ⇝ ‘▣’

Shit Happens... Dealing with IO Exceptions

Exception Summary

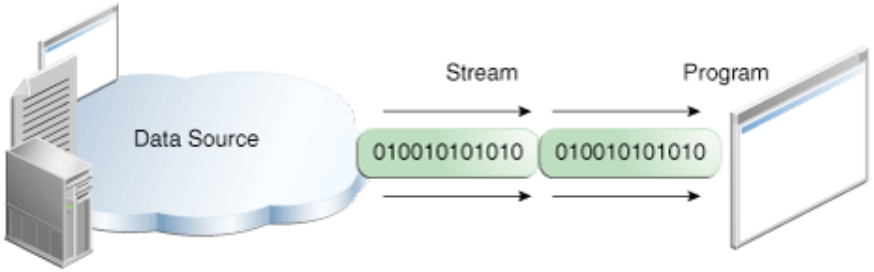
Exception	Description
CharConversionException	Base class for character conversion exceptions.
EOFException	Signals that an end of file or end of stream has been reached unexpectedly during input.
FileNotFoundException	Signals that an attempt to open the file denoted by a specified pathname has failed.
InterruptedIOException	Signals that an I/O operation has been interrupted.
InvalidClassException	Thrown when the Serialization runtime detects one of the following problems with a Class.
InvalidObjectException	Indicates that one or more deserialized objects failed validation tests.
IOException	Signals that an I/O exception of some sort has occurred.
NotActiveException	Thrown when serialization or deserialization is not active.
NotSerializableException	Thrown when an instance is required to have a Serializable interface.
ObjectStreamException	Superclass of all exceptions specific to Object Stream classes.
OptionalDataException	Exception indicating the failure of an object read operation due to unread primitive data, or the end of data belonging to a serialized object in the stream.
StreamCorruptedException	Thrown when control information that was read from an object stream violates internal consistency checks.
SyncFailedException	Signals that a sync operation has failed.
UnsupportedEncodingException	The Character Encoding is not supported.
UTFDataFormatException	Signals that a malformed string in modified UTF-8 format has been read in a data input stream or by any class that implements the data input interface.
WriteAbortedException	Signals that one of the ObjectStreamExceptions was thrown during a write operation.

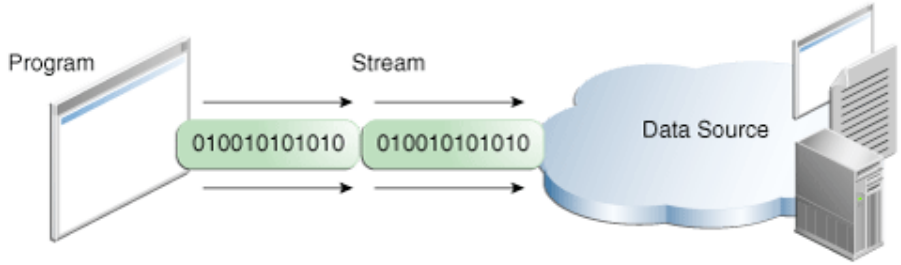
References



The IO Trail in the Java Tutorial

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

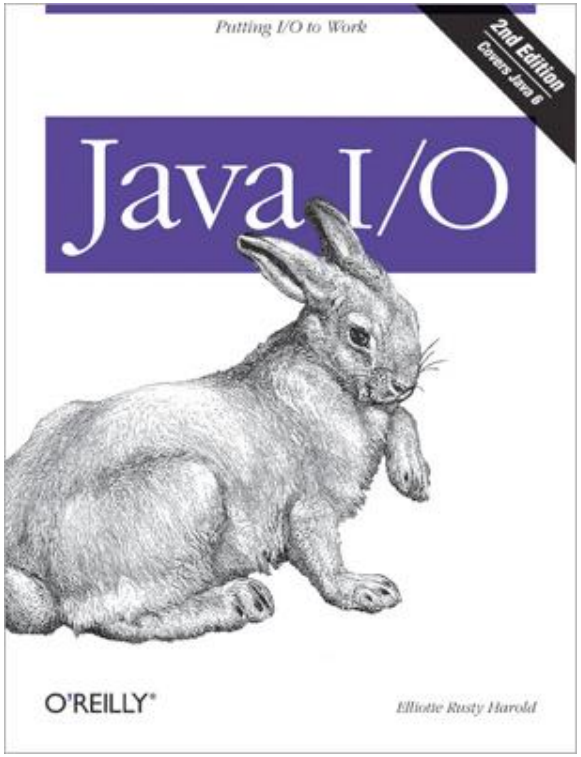


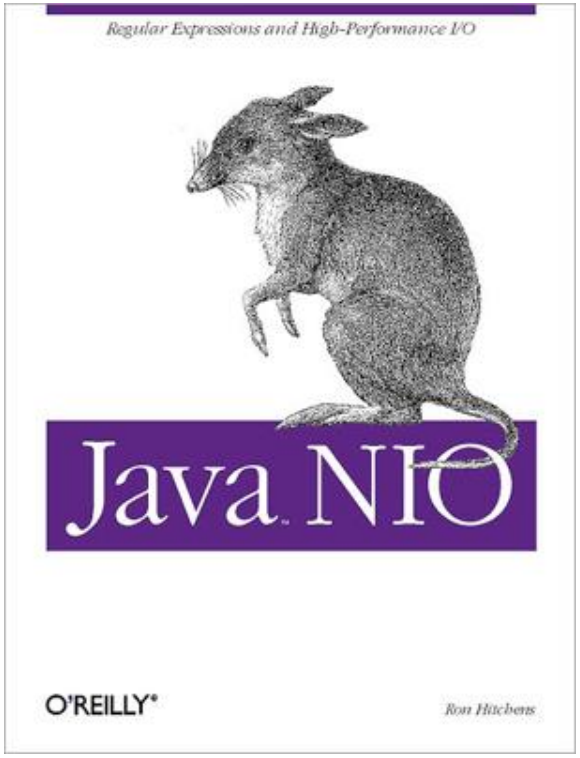


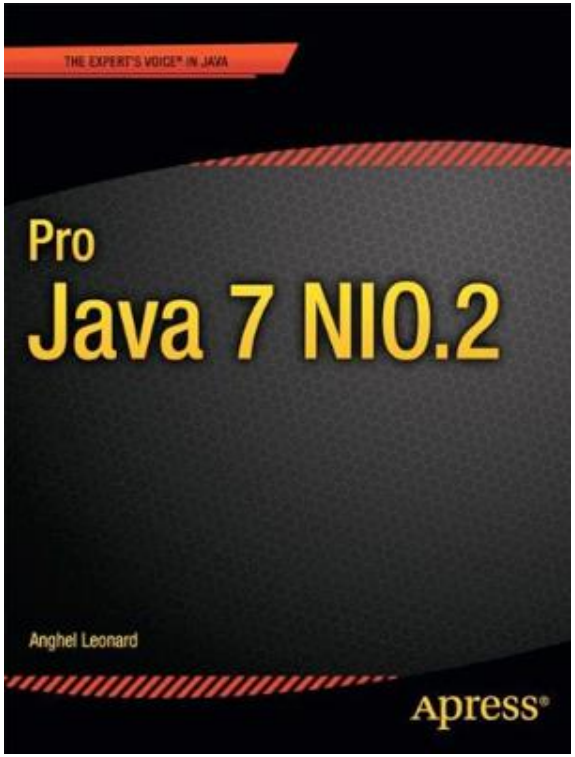
http://docs.oracle.com/javase/tutorial/essential/io/index.html

Books

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch







End of the chapter
Java IO