Encodings and Unicode

Ulrich.Schaefer@dfki.de

Motivation

- Multilinguality plays increasing role in LT
- Java supports Unicode natively and provides mappings between character sets
- It's all easy but one must be aware of it
- Don't waste time with detecting encodings and manually converting etc.
- Related in this lecture: Strings and XML (JAXP)

Related: i18n and l10n support in Java

- I18n: Internationalization (Design)
- L10n: Localization (Instantiation)
- country and language codes (ISO 3166, ISO 639)
- date, time formats
- currencies
- number formats
- (composed) messages, plurals

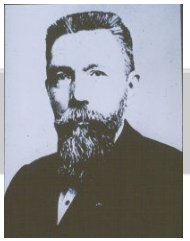
Tutorial: http://java.sun.com/docs/books/tutorial/i18n/index.html

Encodings

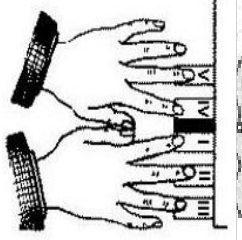
… – – – …

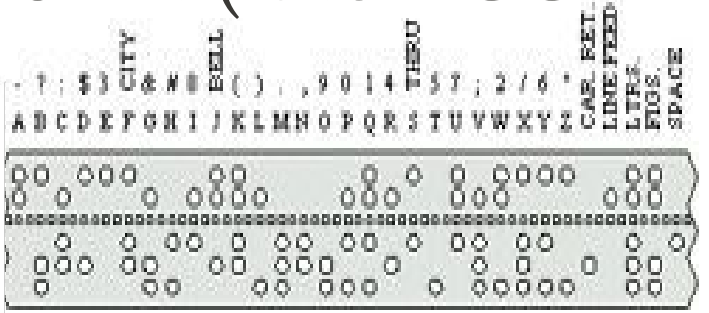
Since 1833 (Samuel Morse): squeeze as many digital codes as possible into a given piece of time or space (memory)

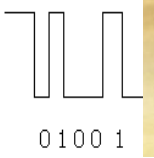
Jean-Maurice-Émile Baudot

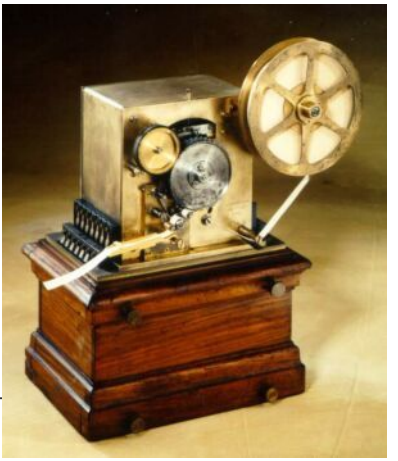


- Teletype (1877: Paris-Bordeaux)
- Baudot-Code: 5 Bit (later: CCITT-2)









- 2 switching codes for letters vs. numbers/punctuation
- no distinction between upper/lower case (2^5=32 codes)
- still today: news agencies and sea wheather forecasts (shortwave transmissions) at 45/50/75 Bd (~6-11 cps)

Definitions

- character: letter, number, punctuation, symbol
- character repertoire: set of distinct characters
- character code: maps a character to a number
- character set: set of character codes (sometimes used for / mixed with repertoire or encoding)
- character encoding: mapping (algorithm) between codes and digital representations
- out of scope: fonts...

ISO-646: 7 bit character sets

- 1968: ASCII (7 bit: 2^7=128 codes from 0-127)
  - A-Z, a-z, 0-9, !"§$%&/()=?`{[]}\#'+*~,.;:-_<>|^@, plus control codes 0-31: EOF, LF, CR, FF, BS, ESC, ...
  - displays/terminals, printers, storage devices, UNIX, email (SMTP...), C, TeX, PostScript, ...
  - Limits: no umlauts, no accents – workarounds:
    1. trade @[\]{|} for umlauts (ISO-646)
    2. escaping: ü=\"u in TeX, ü=ü in HTML
    3. bad solution: ü=u or ü=ue (irreversible)

8 bit character sets: 2^8 = 256 codes

- Codes 0-127: mostly identical with ASCII
  - ISO-8859-1 ("latin1"), -15: Western European (160-255)
  - ISO-8859-2 ("latin2"): East European (160-255), etc.
  - Asian e.g. EUC-JP, GB2312
  - MS-DOS/Windows codepages (128-255):
    - country-specific, graphics, symbols, e.g. CP-1252 ~ latin1

80 €		82 ‚	83 ƒ	84 „	85 …	86 †	87 ‡	88 ˆ	89 ‰	8A Š	8B ‹	8C Œ		8E Ž	
	91 ‘	92 ’	93 “	94 ”	95 •	96 –	97 —	98 ˜	99 ™	9A š	9B ›	9C œ		9E ž	9F Ÿ
A0	A1 ¡	A2 ¢	A3 £	A4 ¤	A5 ¥	A6 ¦	A7 §	A8 ¨	A9 ©	AA ª	AB «	AC ¬	AD -	AE ®	AF ¯
B0 °	B1 ±	B2 ²	B3 ³	B4 ´	B5 µ	B6 ¶	B7 ·	B8 ¸	B9 ¹	BA º	BB »	BC ¼	BD ½	BE ¾	BF ¿
C0 À	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Ç	C8 È	C9 É	CA Ê	CB Ë	CC Ì	CD Í	CE Î	CF Ï
D0 Ð	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD Ý	DE Þ	DF ß
E0 à	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 ç	E8 è	E9 é	EA ê	EB ë	EC ì	ED í	EE î	EF ï
F0 ð	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ý	FE þ	FF ÿ

Codepage 1252 ("ANSI")

A0	A1 ¡	A2 ¢	A3 £	A4 €	A5 ¥	A6 Š	A7 §	A8 š	A9 ©	AA ª	AB «	AC ¬	AD -	AE ®	AF ¯
B0 °	B1 ±	B2 ²	B3 ³	B4 Ž	B5 µ	B6 ¶	B7 ·	B8 ž	B9 ¹	BA º	BB »	BC Œ	BD œ	BE Ÿ	BF ¿
C0 À	C1 Á	C2 Â	C3 Ã	C4 Ä	C5 Å	C6 Æ	C7 Ç	C8 È	C9 É	CA Ê	CB Ë	CC Ì	CD Í	CE Î	CF Ï
D0 Ð	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD Ý	DE Þ	DF ß
E0 à	E1 á	E2 â	E3 ã	E4 ä	E5 å	E6 æ	E7 ç	E8 è	E9 é	EA ê	EB ë	EC ì	ED í	EE î	EF ï
F0 ð	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ý	FE þ	FF ÿ

ISO-8859-15

Unicode (ISO/IEC 10646 in 1993)

- motivation: multilingual documents, ambiguity
- world-wide, platform-independent character repertoir comprising "all" (i.e., many) languages
- one repertoir, multiple 'encodings'
- ISO 10646 (UCS= Universal Multiple-Octet Coded Character Set)
  - 2 or 4 bytes for a character (UCS-2, UCS-4)
  - UCS-2: $2^{16} = 65536$ codes
  - UCS-4: $2^{31} = 2.147.483.648$ codes

"The" Unicode: UCS-2 and UCS-4

- UCS-2 (2 byte encoding) contains most important characters from all over the world ~> Java char
- aka BMP (Basic Multilingual Plane)
- each code also has a unique textual name like (ϋ):
  "GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND TONOS"
- codes 0-255 are identical with ISO-8859-1 !
  - ISO8859-1: 'U'= 01010101 (8 bits)
  - UCS-2: 'U'= 00000000 01010101 (16 bits) (31 bits)
  - UCS-4: 'U'= 00000000 00000000 00000000 01010101

Unicode BMP (Basic Multilingual Plane)

- Character Blocks (examples):
  - \u0000-\u007F: basic (ASCII) characters
  - \u0080-\u00FF: latin1, i.e. codes 0-255 as in ISO8859-1
  - \u0100-\u024F: extended latin A/B ...
  - \u0250-\u02AF: IPA (phonetic)
  - \u0370-\u04FF: Greek, Coptic, Cyrillic
  - \u2190-\u22FF: Arrows, Mathematical Operators
  - \u3040-\u30FF: Hiragana, Katakana
- BTW: java.lang.Character(.UnicodeBlock) defines constants

The Unicode Standard: Documentation

- From http://www.unicode.org/charts/PDF/U0000.pdf :

ASCII punctuation and symbols

Based on ISO/IEC 646.

0020 SP SPACE
- sometimes considered a control code
- other space characters: 2000 NQSP –200A HSP

→ 00A0 NBSP no-break space
→ 200B ZWSP zero width space
→ 2060 WJ word joiner
→ 3000 IDSP ideographic space
→ FEFF ZWNBSP zero width no-break space

0021 ! EXCLAMATION MARK
= factorial
= bang
→ 00A1 ¡ inverted exclamation mark
→ 01C3 ǃ latin letter retroflex click
→ 203C ‼ double exclamation mark
→ 203D ⁉ interrobang
→ 2762 ❢ heavy exclamation mark ornament

0022 " QUOTATION MARK
- neutral (vertical), used as opening or closing quotation mark
- preferred characters in English for paired quotation marks are 201C “ & 201D ”

→ 02BA ″ modifier letter double prime
→ 030B ◌̋ combining double acute accent
→ 030E ◌̎ combining double vertical line above
→ 2033 ″ double prime
→ 3003 〃 ditto mark

UTF (UCS Transformation Format) encodings

- UCS-2 and UCS-4 waste memory. Idea: compact variable-length multi-byte format for UCS-2 and UCS-4, with eight bit compatibility (in UTF-8 only)
- UTF-8: codes 0-127 identical with ASCII

Unicode Char	UTF-8 representation
\u00000000 - \u0000007F:	0xxxxxxx (ASCII)
\u00000080 - \u000007FF:	110xxxxx 10xxxxxx
\u00000800 - \u0000FFFF:	1110xxxx 10xxxxxx 10xxxxxx
\u00010000 - \u001FFFFF:	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
\u00200000 - \u03FFFFFF:	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
\u04000000 - \u7FFFFFFF:	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

UCS-2

16 'x' =16 bits

UCS-4

31 'x'=31 bits

UTF-8 is not ISO-8859-1 !

- Example: character 'ü' (UCS code: \u00FC)
- UCS: two bytes
  - ü = 00000000 11111100 (0x00, 0xFC)
- ISO-8859-1: single byte:
  - ü = 11111100 (0xFC)
- UTF-8: all characters in Latin1 extension block (ISO-8859-1-equiv.) of BMP need two bytes:
  - ü = 11000011 10111100 (0xC3, 0xBC)

UTF-16, UTF-32

- UTF-16: extension of UCS-2 to 21 bits for characters beyond \uFFFF (UCS-4) -> JAVA char. Codes \uD800 to \uDFFF indicate 1024×1024 non-BMP characters to be represented as a sequence of two 16-bit `surrogate' characters
- UTF-32: same as UCS-4 with less characters (21 bits)

Endians and BOMs (Byte Order Marks)

Why do Unicode files sometimes start with FFFE or FEFF ? -> Endian byte order mark (BOM)

(\uFEFF = ZERO WIDTH NO-BREAK SPACE)

purpose: autodetection of byte order in files

Endian variants in memory and in files

- Big (SPARC, PowerPC) vs. Little (Intel, AMD) Endian processor architectures:
  1. ü in UCS-2LE: 11111100 00000000 (little endian)
  2. ü in UCS-2BE: 00000000 11111100 (big endian)
  3. either LE or BE, but with byte order mark (BOM)
- 3 file variants for each UCS and UTF encoding: UCS-2, UCS-2BE, UCS-2LE, UCS-4, UCS-4LE, UCS-4BE, UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32, UTF-32BE, UTF-32LE

'Déjà 5€?' in different encodings


```
 D| é| j| à|  | 5| €| ?| (ISO-8859-15)
44|E9|6A|E0|20|35|A4|3F|

 D| é| j| à|  | 5| €| ?| (windows-1252)
44|E9|6A|E0|20|35|80|3F|

 D| é| j| à|  | 5| €| ?| (IBM500/EBCDIC)
C4|51|91|44|40|F5|3F|6F|

 D|    é| j|    à|  | 5|       €| ?| (UTF-8)
44|C3 A9|6A|C3 A0|20|35|E2 82 AC|3F|

    D|    é|    j|    à|     |    5|    €|    ?| (UTF-16BE)
00 44|00 E9|00 6A|00 E0|00 20|00 35|20 AC|00 3F|

    D|    é|    j|    à|     |    5|    €|    ?| (UTF-16LE)
44 00|E9 00|6A 00|E0 00|20 00|35 00|AC 20|3F 00|

          D|          é|          j|          à|             |          5|…(UTF-32)
00 00 00 44|00 00 00 E9|00 00 00 6A|00 00 00 E0|00 00 00 20|00 00 00 35|…
```

In Java: `byte[] bytes = Character.toString('€').getBytes("UTF-16BE");`
`returns byte array of length 2: 20 AC`
(may throw `java.io.UnsupportedEncodingException`)

Java Character Encodings

- internally, a Java char always represents 16 bit Unicode (=UCS-2 / UTF-16)
- as a consequence, \u0000-\u00FF is identical with iso-8859-1
- char byte order in Java is architecture-independent (always Big-Endian as suggested by Unicode) like int, and in contrast to C/C++!
- \uXXXX specifies UCS code, not UTF encoding!

- use for File, Stream I/O, byte array mapping
- java.nio.charset package contains encoders/decoders for many character sets
- selection through encoding ("Charset") name
- character I/O methods without encodings are deprecated!!
- remember (i.e. store) encoding information, or make input and output encoding configurable!

Files, Streams, Writers, Readers in java.io

- File: object for file properties like name, path, directory, random or sequential access
- InputStream, OutputStream: abstract class for sequential read and write access, e.g., files, processes, arrays, buffers
- Reader, Writer: abstract class for character streams, e.g. to/from File or Process
- Combination via constructors

Reading from a file with a specified encoding

- There are alternatives to do this (e.g. in java.nio) – but if you can't specify an encoding (charset), something is wrong or missing.
- FileInputStream(Filename)
- InputStreamReader(InputStream, Charsetname)
- br = new BufferedReader (Reader)
- while ((String line = br.readLine())!=null) { ... }
- ex.: implement this + vice versa (writer)

Encodings of Java source code

- other encodings than latin1 are possible, but dangerous (not all compilers support it)
- compilation (Sun JDK):
  - javac -encoding GB2312 ...
    - default is iso-8859-1, with \uXXXX Unicode extensions (can be used anywhere, also as part of variable names)
- JDK-tool 'native2ascii' converts chars > 127 into javac-congestible format (\uXXXX)
  http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/native2ascii.html

- XML standard says: all XML names *must* comprise Unicode BMP (the 16 bit Unicode 'subset')
- XML without encoding declaration must be one of
  - UTF-8: 3C 3F 78 6D 6C for <?xml
  - UTF-16: **FE FF** 00 3C 00 3F 00 78 00 6D 00 6C (Big Endian)
  - **FF FE** 3C 00 3F 00 78 00 6D 00 6C 00 (LittleEndian)
  - + further variants, see App. F in XML recommendation
  - others encodings (including ISO-8859-1!) must be declared and an XML processor may not implement them

XML and Unicode

- Unicode characters may be transcribed as entities in XML documents
- ü (decimal)
- &x00FC; (hexadecimal)
- using this transcription, an XML document containing special Unicode characters may be encoded in pure 7 bit ASCII

DO NOT USE ä etc. - this is HTML!

XML special: convert Unicode text to ASCII repres.

Idea: all codes > 127 can be rewritten as entities,
e.g. ü as ü (this is not UTF-7!)

```
public static String cleanXMLString(String strIn) {
  StringBuffer strBuffer = new StringBuffer();
  for (int i = 0; i < strIn.length(); i++) {
    char c = strIn.charAt(i);
    int s = (int) c;
    if (c == '&')        strBuffer.append("&amp;");
    else if (c == '\'')  strBuffer.append("&apos;");
    else if (c == '>')   strBuffer.append("&gt;");
    else if (c == '<')   strBuffer.append("&lt;");
    else if (c == '"')   strBuffer.append("&quot;");
    else if (s > 127)    strBuffer.append("&#" + s + ";");
    else strBuffer.append(c);
  }
  return strBuffer.toString();
}
```

Literature

- Unicode
  - http://www.unicode.org/charts/
  - http://www.unicodecharacter.com/charsets/iso8859.html
  - http://www.cs.tut.fi/~jkorpela/chars.html
  - http://www.cl.cam.ac.uk/~mgk25/unicode.html
  - http://www.unicode.org/unicode/faq/
  - http://de.wikipedia.org/wiki/Unicode
- Encodings supported by Sun JDK
  http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html
- Files, Streams: www.javabuch.de,
  http://www.galileocomputing.de/openbook/javainsel7/