






Strings in Java

Christoph Hofmann

Albert-Ludwigs Universität Freiburg



Agenda

-  Basics
-  UTF-8
-  Storage Management
-  Performance
-  Tipp's for Java Programming



Basics

Ordinary Object

- Intern implemented as array of chars

Encoding

- Use the default Encoding of the OS
 - Call: `Charset.defaultCharset();`
- Supported Encodings:
 - ASCII, windows-(1250-1254,1257), ISO-8859-(1,2,4,5,7,9,13,15), KOI8-R, UTF-(8,16,16BE, 16LE)...
 - Depends on the library (rt.jar, charsets.jar...)

Stored in the Heap



UTF-8

Unicode

- Each char has a unique code
 - e.g. a = U+0061

UTF-8 (Unicode Transformation Format)

- a standard Encoding Format for all character of the world
- extends the long-standing ASCII / ISO-8859-1
- Variable byte encoding
 - Encoding up to 4 Bytes
- 1.048.576 characters are possible



UTF-8

- ☕ ASCII compatible = a string of characters with ASCII codes < 128 is the same in ASCII as in UTF-8 (1 Byte)
- ☕ frequent special characters (like ä, á, å) need two bytes, only very rare characters need three or four bytes
- ☕ no need to decode from left to right,
- ☕ easy to decode / convert to UTF-32
- ☕ Multibyte sequence
 - the number of leading 1s in the first byte of a sequence encodes the length.



UTF-8

Unicode	UTF-8-Encoding	Byte	Chars
[0, 127]	0xxxxxxx	1	128
[128, 2047]	110xxxxx 10xxxxxx	2	1920
[2048, 65535]	1110xxxx 10xxxxxx 10xxxxxx	3	63.488
[65536, $2^{21} - 1$]	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	4	1.048.576

Char	Unicode	UTF-8-Encoding	Hex
a	U+0061	01100001	61
ä	U+00E4	11000011 10100100	C3 A4
ॐ	U+0904	11100000 10100100 10000100	E0 A4 84
☹	U+10400	11110000 10010000 10010000 10000000	F0 90 90 80



UTF-8

Forbidden code sequences (limited by RFC3629)

- C0-C1 Encoding ASCII with 2 Bytes
- F5-F7 Encoding with 4 Bytes about 140000
- F8-FB (5 Bytes) , FC-FD (6 Bytes) and FE-FF (not specified)
- U+D800–U+DBFF and U+DC00–U+DFFF is only for Special Chars in UTF-16

There is exactly one UTF-8 sequence for a char („Non-shortest-form“ issue)

- Different forms of representations
 - ABC = 0x41 0x42 0x43
= 0xc1 0x81 0xc1 0x82 0xc1 0x83
 - Shortest representation is only legal
- Security issue

STORAGE MANAGEMENT



A simple String

Ordinary object

- `String testString = new String("Hello");`

Heap Storage

Disadvantage

- Each String with the same content needs memory in the heap



String Common Pool

- ☕ Area in the Heap memory
- ☕ String will be initialized with a literal
 - String str1 = "Java is Hot";
- ☕ Strings are shared
- ☕ Strings with same content uses the same memory



Heap -> String Pool

☕ Use the `obj.intern()` to „move“ the String from the Heap to the String Common Pool

- Check if the String Pool contains a String with the same Content
- No => Content will be added to the String Pool and Reference is returned
- Yes => return Reference to the String
- Object will be removed from the Heap by GC

```
String text = new String("text"); (Heap)  
text.intern();
```



String Common Pool vs. Heap

```
String s1 = "Hello";
```

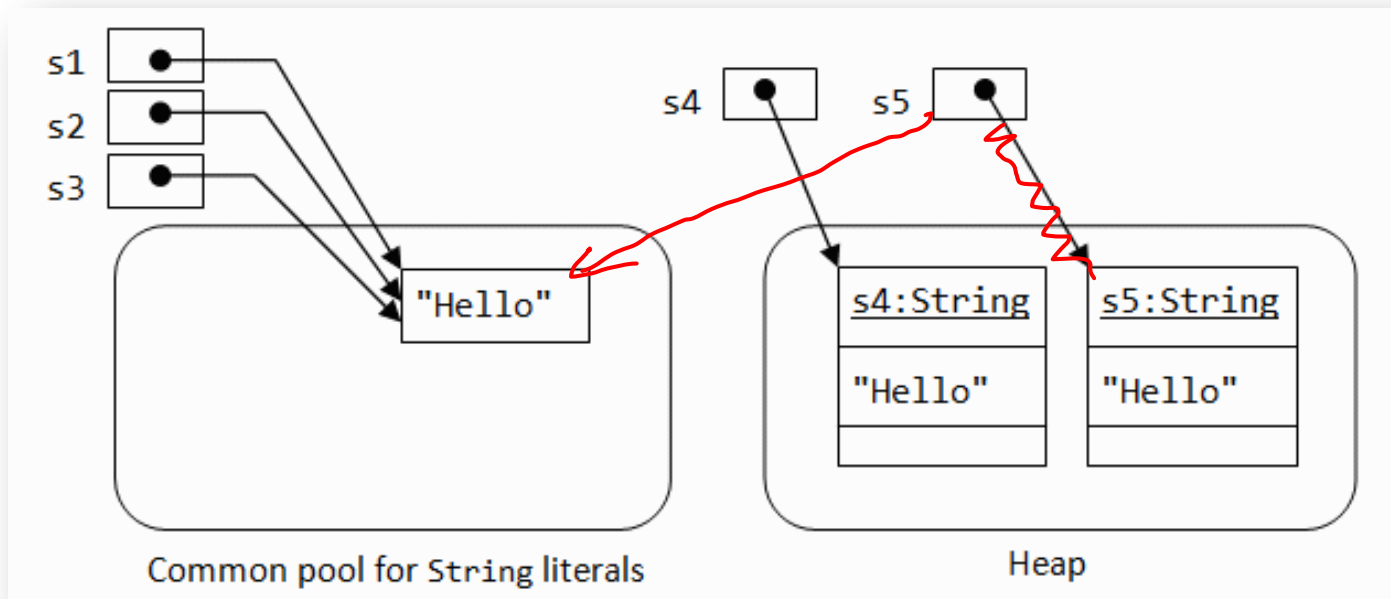
```
String s2 = "Hello";
```

```
String s3 = s1;
```

```
String s4 = new String("Hello");
```

```
String s5 = new String("Hello");
```

s5 = s5.intern();



PERFORMANCE



Append Strings

```
String text1 = "This is an ";  
String text2 = "example!";  
String text3 = text1 + text2;
```

T	h	i	s		i	s		a	n	
---	---	---	---	--	---	---	--	---	---	--

text 1

e	x	a	m	p	l	e	!
---	---	---	---	---	---	---	---

text 2

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

T	h	i	s		i	s		a	n		e	x	a	m	p	l	e	!
---	---	---	---	--	---	---	--	---	---	--	---	---	---	---	---	---	---	---



Find

- ☕ Call `obj.indexOf(string)`
- ☕ Starts to compare if the first char from argument is found in the source string
- ☕ If string contains the substring the index of the first char will be returned

```
String text1 = "This is an example!";  
String text2 = "example";  
int result = text1.indexOf(text2);
```

TIPPS FOR JAVA PROGRAMMING



String Builder

Ordinary object

- Stored in the Heap
- Not Shared

Advantages:

- Reduce side-effects
- Less Memory Transactions
- Raise Performance



String Builder - Initialization

Initialization

- Minimum Size when initialized is 16 (also default size)
- Initialize with a size
 - `StringBuilder sb = new StringBuilder(int)`
 - Initial capacity = int
- Initialize with a string
 - `StringBuilder sb = new StringBuilder(string)`
 - Capacity = `string.length + 16`



String Builder - Initialization

```
StringBuilder sb = new StringBuilder();  
StringBuilder sb2 = new StringBuilder(20);  
StringBuilder sb3 = new StringBuilder("Hello");
```



String Builder - Resizing

Resizing

- Necessary if $\text{length} > \text{capacity}$
- Creates new array and copies content
- $\text{newCapacity} = (\text{oldCapacity} + 1) * 2$



String Builder - Append

Concatenate with StringBuilder

```
StringBuilder sb = new StringBuilder();  
sb.append("This is an ");  
sb.append("example!");
```



String Buffer vs. String Builder

☕ String Buffer is almost equal to the String Builder

- String Builder will be used in a single thread context
- String Builder isn't synchronized (thread-safe)



References



Java Standard

- http://java.sun.com/docs/books/jls/third_edition/html/lexical.html#101083



String Basics + StringBuilder

- <http://download.oracle.com/javase/1.4.2/docs/api/java/lang/String.html>
- <http://download.oracle.com/javase/tutorial/java/data/strings.html>
- <http://java.sun.com/docs/books/jni/html/objtypes.html>



Storage Management

- http://en.wikipedia.org/wiki/String_interning
- http://www3.ntu.edu.sg/home/ehchua/programming/java/J3d_String.html



UTF-8

- <http://en.wikipedia.org/wiki/UTF-8>
- <http://developers.sun.com/dev/gadc/technicalpublications/articles/utf8.html>
- http://blogs.sun.com/CoreJavaTechTips/entry/the_overhaul_of_java_utf



Charsets:

- <http://download.oracle.com/javase/1.4.2/docs/guide/intl/encoding.doc.html>

The End

Any Questions ...?