Tecken och strängar

med och utan Java

Introduktion till tecken

Begrepp: Tecken, kodpunkter, kodningar, …

Unicode: A till Z… och mer

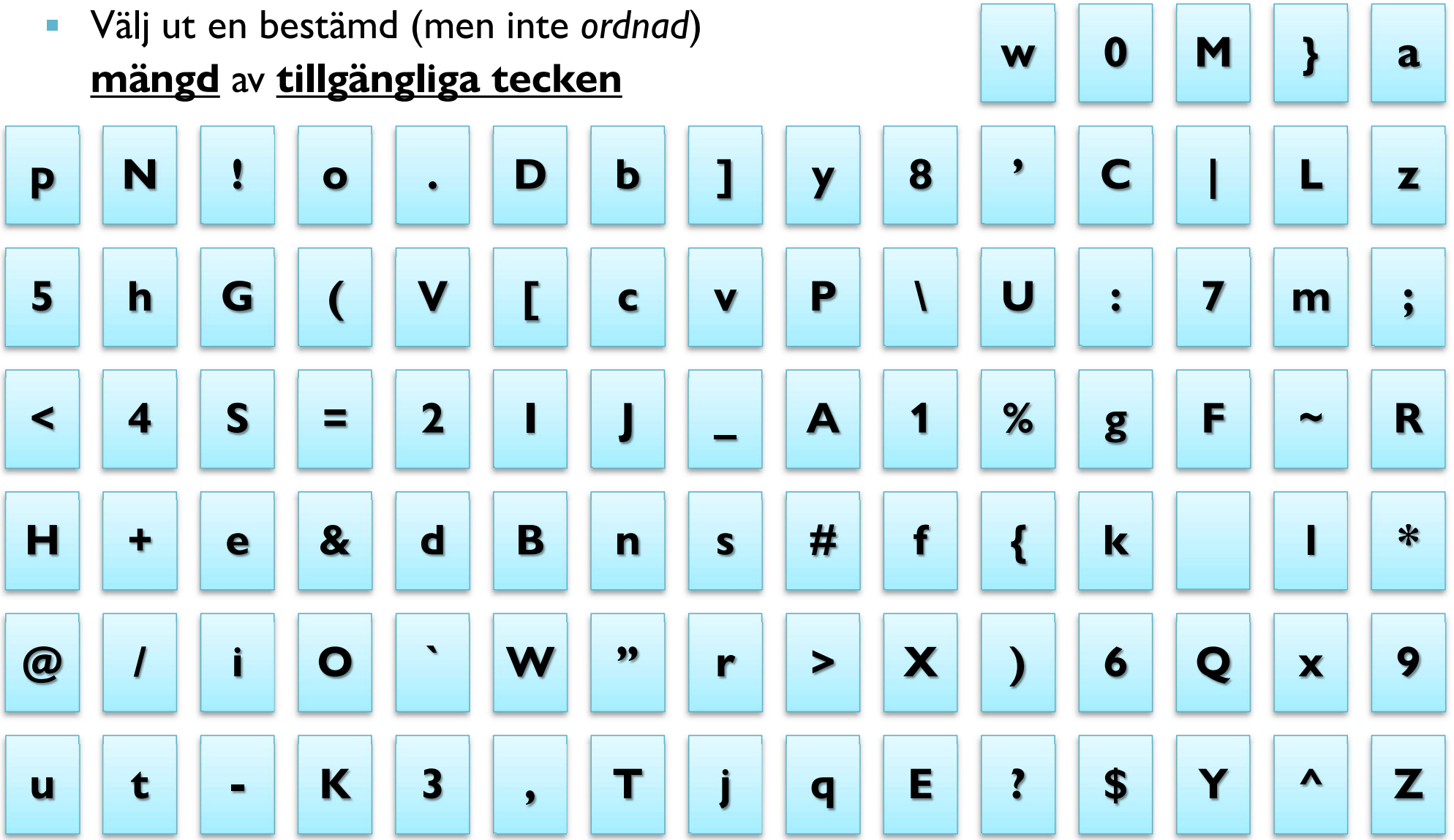
Tecken

- Steg 1: Ett **tecken** (en **symbol**)
  - Odelbar symbol
  - Minsta enheten för information
    - A B C D
    - # " ! ( )
    - å ä ö ċ õ ĕ ķ Ħ Ŗ ǽ Ґ ы Ω Ą Ж ї
    - ਈ ਉ ਊ ਏ ੳ ਐ ਕ ਖ ਗ ਘ ਙ
    - € ℮ ∞ ∫ ℝ 𝓂 ∇ ∰ ∺ ≩ ⊛ ☺ ♠ ◆ ◬
    - ࿐ ⨁ ⌘ ㍲ ﷼ 略 Ⓧ ⇶ ✷

Teckenrepertoar

jonkv@ida

- Steg 2: En **teckenreportoar** (*character repertoire*)
  - Välj ut en bestämd (men inte *ordnad*) **mängd** av **tillgängliga tecken**



Kodad teckenuppsättning

- Steg 3: En **kodad teckenuppsättning** (*coded character set, CCS*)
  - Varje tecken motsvaras av ett **heltal** – exempel: **ASCII** från 0 till 127

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Escape	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	File Separator	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	Group Separator	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	Record Separator	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	Unit Separator	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	Del

Kodad teckenuppsättning (2)

- Annat exempel: **ISO Latin-*x***
  - ISO Latin-1: Adderar 32 kontrolltecken, 96 symboler (västeuropa)
  - ISO Latin-2: Adderar 32 kontrolltecken, 96 symboler (östra/centraleuropa)

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
9-	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A-	00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B-	° 00B0	± 00B1	2 00B2	3 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	1 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C-	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D-	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E-	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F-	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Kodad teckenuppsättning (3)

jonkv@ida

- **EBCDIC**
  (IBM-stordatorer):

EBCDIC Code Page 00037

	−0	−1	−2	−3	−4	−5	−6	−7	−8	−9	−A	−B	−C	−D	−E	−F
0−	NUL 0000 0	SOH 0001 1	STX 0002 2	ETX 0003 3	SEL 4	HT 0009 5	RNL 6	DEL 007F 7	GE 8	SPS 9	RPT 10	VT 000B 11	FF 000C 12	CR 000D 13	SO 000E 14	SI 000F 15
1−	DLE	DC1	DC2	DC3	RES ENP	NL	BS	POC	CAN	EM 19 5	UBS 26	CU1 27	IFS 001C 28	IGS 001D 29	IRS 001E 30	IUS ITB 001F 31
2−										FE 1	SM SW 42	CSP 43	MFA 44	ENQ 0005 45	ACK 0006 46	BEL 0007 47
3−										T 7	RFF 58	CU3 59	DC4 0014 60	NAK 0015 61	62	SUB 001A 63
4−										ñ F1 3	¢ 00A2 74	. 002E 75	< 003C 76	(0028 77	+ 002B 78	| 007C 79
5−										DF 9	! 0021 90	$ 0024 91	* 002A 92	) 0029 93	; 003B 94	¬ 00AC 95
6−										Ñ D1 05	¦ 00A6 106	, 002C 107	% 0025 108	_ 005F 109	> 003E 110	? 003F 111
7−										60 21	: 003A 122	# 0023 123	@ 0040 124	' 0027 125	= 003D 126	" 0022 127
8−	Ø 00D8 128	a 0061 129	b 0062 130	c 0063 131	d 0064 132	e 0065 133	f 0066 134	g 0067 135	h 0068 136	i 0069 137	« 00AB 138	» 00BB 139	ð 00F0 140	ý 00FD 141	þ 00FE 142	± 00B1 143
9−	° 00B0 144	j 006A 145	k 006B 146	l 006C 147	m 006D 148	n 006E 149	o 006F 150	p 0070 151	q 0071 152	r 0072 153	ª 00AA 154	º 00BA 155	æ 00E6 156	, 00B8 157	Æ 00C6 158	¤ 00A4 159
A−	µ 00B5 160	~ 007E 161	s 0073 162	t 0074 163	u 0075 164	v 0076 165	w 0077 166	x 0078 167	y 0079 168	z 007A 169	¡ 00A1 170	¿ 00BF 171	Ð 00D0 172	Ý 00DD 173	Þ 00DE 174	® 00AE 175
B−	^ 005E 176	£ 00A3 177	¥ 00A5 178	· 00B7 179	© 00A9 180	§ 00A7 181	¶ 00B6 182	¼ 00BC 183	½ 00BD 184	¾ 00BE 185	[005B 186	] 005D 187	- 00AF 188	¨ 00A8 189	´ 00B4 190	× 00D7 191
C−	{ 007B 192	A 0041 193	B 0042 194	C 0043 195	D 0044 196	E 0045 197	F 0046 198	G 0047 199	H 0048 200	I 0049 201	SHY 00AD 202	ô 00F4 203	ö 00F6 204	ò 00F2 205	ó 00F3 206	õ 00F5 207
D−	} 007D 208	J 004A 209	K 004B 210	L 004C 211	M 004D 212	N 004E 213	O 004F 214	P 0050 215	Q 0051 216	R 0052 217	¹ 00B9 218	û 00FB 219	ü 00FC 220	ù 00F9 221	ú 00FA 222	ÿ 00FF 223
E−	\ 005C 224	÷ 00F7 225	S 0053 226	T 0054 227	U 0055 228	V 0056 229	W 0057 230	X 0058 231	Y 0059 232	Z 005A 233	² 00B2 234	Ô 00D4 235	Ö 00D6 236	Ò 00D2 237	Ó 00D3 238	Õ 00D5 239
F−	0 0030 240	1 0031 241	2 0032 242	3 0033 243	4 0034 244	5 0035 245	6 0036 246	7 0037 247	8 0038 248	9 0039 249	³ 00B3 250	Û 00DB 251	Ü 00DC 252	Ù 00D9 253	Ú 00DA 254	EO 255

Kodad teckenuppsättning (4)

jonkv@ida

- Men hur får vi plats med kinesiska tecken, då?

Kodad teckenuppsättning (5)

- Och alla nya emojis?






Kodad teckenuppsättning (6)

- Java 21 använder **Unicode 15**
  - **1,114,112** kodpunkter, 0_{hex} till $10FFFF_{hex}$, **149,186** namngivna tecken!

Code Point Type	Count	Delta
Alphabetics, Symbols	39,782	289
Han (URO)	20,902	
Han (URO Extension)	90	
Han Extension A	6,592	
Han Extension B	42,720	
Han Extension C	4,154	1
Han Extension D	222	
Han Extension E	5,762	
Han Extension F	7,473	
Han Extension G	4,939	
Han Extension H	4,192	4,192
Han Compatibility	1,014	
Subtotal Han	98,060	4,193
Hangul Syllables	11,172	
Graphic Characters	149,014	4,482
Format Characters	172	7
Graphic + Format	**149,186**	4,489
Controls	65	
Private Use	137,468	
Total Assigned	286,719	4,489
Surrogate Code Points	2,048	
Noncharacters	66	
Total Designated	288,833	4,489
Reserved Code Points	825,279	-4,489

Kodad teckenuppsättning (7)

jonkv@ida

- Exempel:

decodeunicode – https://vimeo.com/48858289

0000..00FF	*Som i ISO Latin-1*	
03D0..03D6	GREEK BETA SYMBOL..GREEK PI SYMBOL	
05B0..05B9	HEBREW POINT SHEVA..HEBREW POINT HOLAM	
0A01	GURMUKHI SIGN ADAK BINDI	ਈਉਊਏਓਅੱਕਖਗਘਛ
0EDC..0EDD	LAO HO NO..LAO HO MO	ໜໝ
2302..237A	HOUSE..APL FUNCT. SYMBOL ALPHA	⌂⌘⌦⌨⍝≠
2623	BIOHAZARD	☣
2E80..2E99	CJK RADICAL REPEAT..CJK RADICAL RAP	
33E0	IDEOGRAPHIC TELEGRAPH SYMBOL FOR DAY ONE	
1039F	UGARITIC WORD DIVIDER	

Räcker inte med 8 bitar – eller 16…

Teckenkodning

jonkv@ida

- Steg 4: *Character encoding form* + *character encoding scheme*
  - *Ungefär*: Hur representerar man dessa tal som *bytes* i en dator?

- ISO Latin-1:
  - Tecken ➔ värden 0-255, får plats i 1 byte
  - Trivialt!

- **Unicode:** Exempelvis **UTF-32** – enkelt men ineffektivt
  - Tecken ➔ 32-bitars heltal, 4 bytes
    - Räksmörgås ➔ 00 00 00 **52** 00 00 00 **E4** 00 00 00 **6B**
      00 00 00 **73** 00 00 00 **6D** 00 00 00 **F6** 00 00 00 **72**
      00 00 00 **67** 00 00 00 **E5** 00 00 00 **73**

Teckenkodning (2)

- Ett av flera alternativ: **UTF-8**
  - **Varierande antal bytes** per tecken!
    - Räksmörgås → 72 **c3 a4** 6b 73 6d **c3 b6** 72 67 **c3 a5** 73 (hexadecimalt)
      → 13 bytes
      → Ganska effektivt för engelska och liknande språk
    - I ISO Latin-1 skulle dessa 13 bytes betyda "rÃ¤ksmÃ¶rgÃ¥s“
      - 72 → r
      - c3 → Ã
      - a4 → ¤, …
  - Kodningsschema:
    - **Teckennummer** → **Lagring som bytes**
    - 00000 - 0007F → **0***xxxxxxx (samma som ASCII!)*
    - 00080 - 007FF → **110***xxxxx* **10***xxxxxx*
    - 00800 - 0FFFF → **1110***xxxx* **10***xxxxxx* **10***xxxxxx*
    - 10000 – FFFFF → **11110***xxx* **10***xxxxxx* **10***xxxxxx* **10***xxxxxx*

Informativt bitmönster!

Börjar med 10 → är en fortsättning på ett tecken

Text i Java

Tecken och strängar

Java skiljer på:

- **Tecken**
  - Exakt en symbol
  - Inom apostrofer: ’x’
  - Primitiv datatyp, **char**
    - 16 bitar,
    - Unicode-tecken 0–65535
    - `char c = 'x';`
    - `char c = 120;  // c=='x', eftersom Unicodetecken #120 är ett 'x'.`

- **Strängar**
  - Noll eller flera symboler
  - Inom citat: "Hello World"
  - Objekttyp, **String**

- Teckenkodning: **UTF-16**
  - De flesta tecken ➔ 1 char
  - Vissa tecken ➔ 2 chars

- Varför UTF-16?
  - Dålig planering…
  - Java 1.0: Unicode 1.1.5, med 40,635 tecken – fick plats i 16 bitar

16 bitar överallt?

- Men kunde det inte räcka med 8 bitar ibland?
  - Fram till Java 8 (2014):
    - private final **char**[] value; // UTF-16
  - Från och med Java 9 (2017): 8-bitar **om** strängens tecken finns i ISO Latin-1
    - private final **byte**[] value; // UTF-16 eller ISO Latin-1
    - private final **byte** coder;
    - static final **byte** LATIN1 = 0;
    - static final **byte** UTF16 = 1;

Kunde ändras eftersom representationen var privat, inkapslad

Att jämföra strängar

- Har två strängar samma innehåll?
  - **equals(String other), equalsIgnoreCase(String other)**

== jämför pekare!

```
if (str == str2) {
   // Jämför om det är samma objekt
}
if (str.equals(str2)) {
   // Jämför om strängarna innehåller samma tecken
}
```

```
>>> x = "a"
>>> "aa" is x * 2
False
>>> "aa" == x * 2
True
```

  - **compareTo(String other), compareToIgnoreCase(String other)**
    - -1 om **this** ska vara före **other**, 0 om lika, 1 om **this** ska vara efter **other**
    - Jämför enligt Unicode-ordning:
      …ABCD…XYZ[\]^…¡¢£¤¥…¾¿ÀÁÂÃÄÅÆÇÈ…ÔÕÖ×ØÙ…ĀĄ…Ʀ…Ж…چ…

Se även: java.text.Collator
Sortering enligt *specifika språk* (svenska: …XYZÅÄÖ)

Textformattering

Formattering 1: Object ➔ String

jonkv@ida

- Varje objekt har en **strängrepresentation**
  - Exempel:
    - ```
      public class Circle {
        private double x, y, r;
        private Circle(double x, double y, double r) {
          this.x = x; this.y = y; this.r = r;
        }
        public static void main(String[] args) {
          System.out.println(new Circle(2.0, 4.0, 5.0));
        }
      }
      ```
    - **>> java Circle**
      Circle@a5e17120

Metoden println(Object o)
anropar o.toString()

Circle har ingen egen toString()
➔ använder standardimplementation
➔ klassnamn + "@" + ett unikt ID
      ```

Formattering 2: Object ➔ String

jonkv@ida

- Implementera en **egen** toString()!
  - ```
    public class Circle {
        …
        public String toString() {
            return "Circle[" + x + ", " + y + ", " + r + ']';
        }
        …
    }
    ```

Här adderas strängar – men ett String-objekt är immutable (oföränderligt)!

Gäller även Python…

  - **>> java Circle**
    Circle[2.0, 4.0, 5.0]
    ```

Formattering 3: Strängkonkatenering

- **StringBuilder**: **Manipulering** av teckensekvenser
  - Används av kompilatorn för att implementera "+"

```
public String toString() {
   return "Circle[" + x + ", " + y + ", " + r + ']';
}
```

```
public String toString() {
   final StringBuilder buf = new StringBuilder("");
   buf.append("Circle[");
   buf.append(x);
   buf.append(", ");
   buf.append(y);
   buf.append(", ");
   buf.append(r);
   buf.append(']');
   return buf.toString();
}
```

StringBuilder.append(double) konverterar till tecken

Skapa en **String** med samma tecken som denna **StringBuilder**

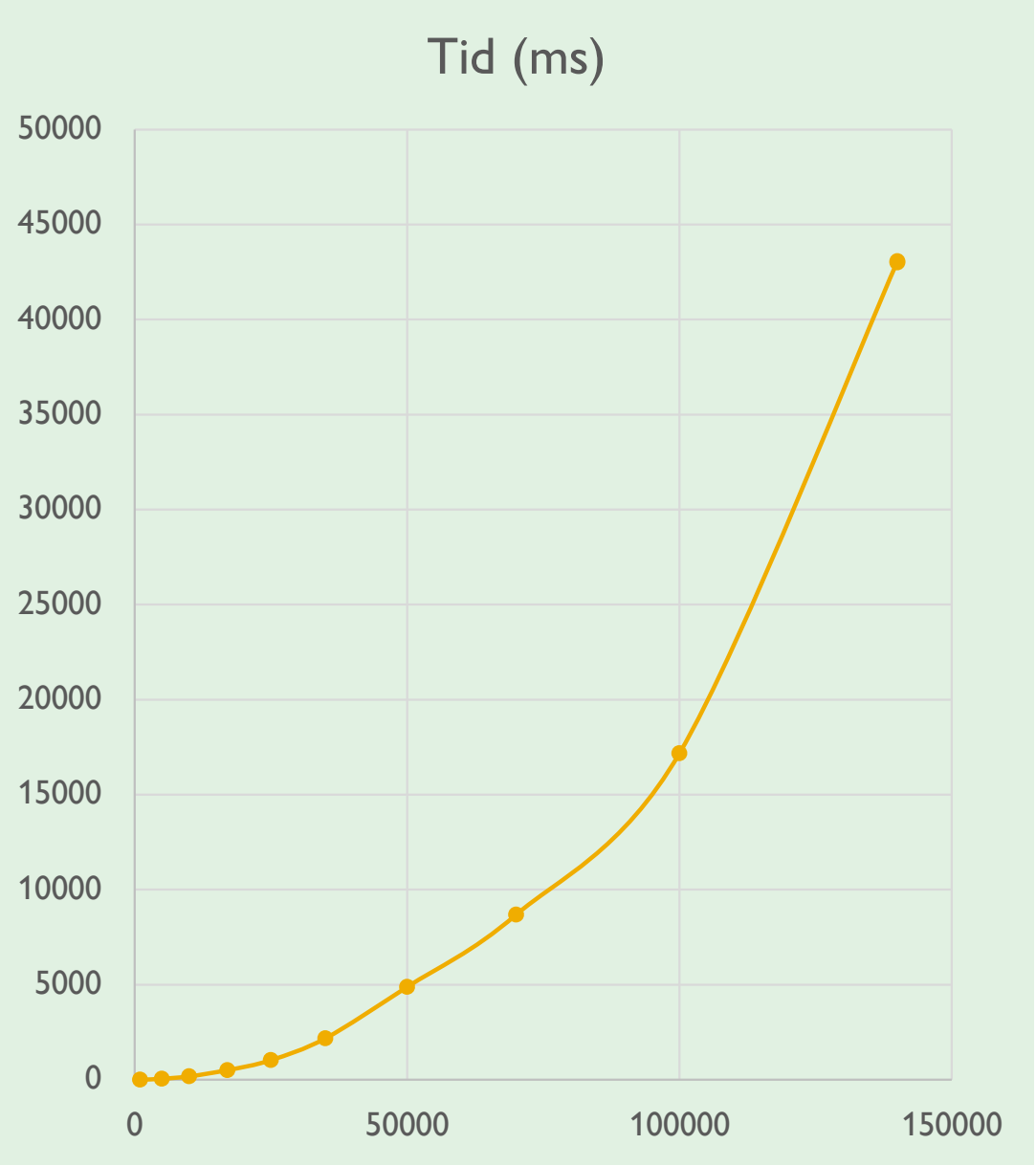
Formattering 4: Strängkonkatenering

jonkv@ida

- Vad händer om man gör detta?

```
public class ArrayList {
   private Object[] elements;
   private int size;
   // ...

   public String toString() {
      String res = "";
      for (int i = 0; i < size; i++) {
         res = res + elements[i].toString();
         res = res + ", ";
      }
      return res;
   }
}
```



Formattering 5: Strängkonkatenering

jonkv@ida

- Konkatenering kan bli ineffektivt -- *tidskomplexitet*…

```
public class ArrayList {
   private Object[] elements;
   private int size;
   // ...
   public String toString() {
      String res = "";
      for (int i = 0; i < size; i++) {
         res = res + elements[i].toString();
         res = res + ", ";
      }
      return res;
   }
}
```

Vi kan ju inte ändra en sträng – så vad händer här?

Kopiera alla tecken till en ny StringBuilder
Lägg till några tecken på slutet
Kopiera till en ny String
Sätt om res-pekaren
Släng StringBuilder + gammal String

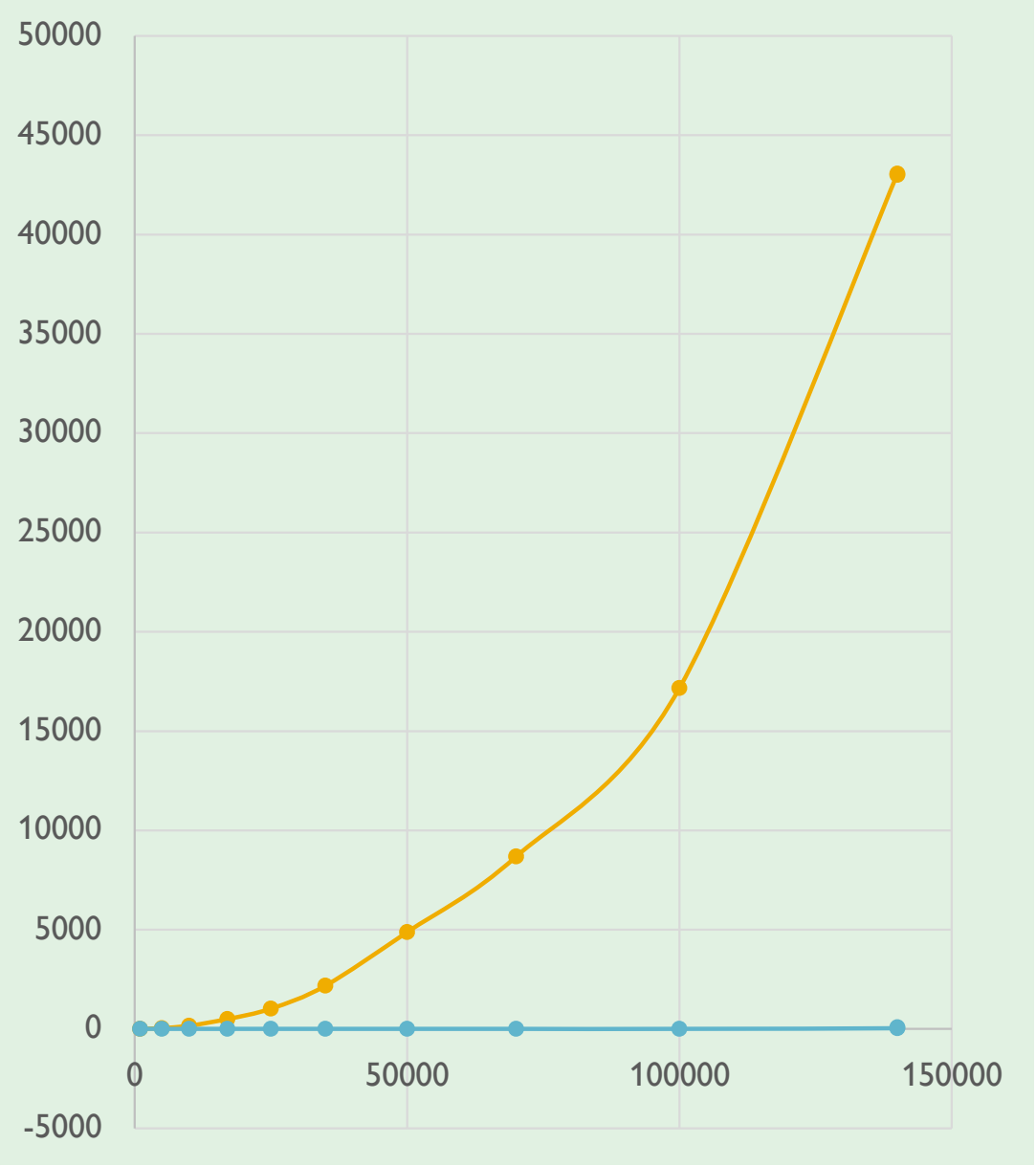
```
public class ArrayList {
   private Object[] elements;
   private int size;
   // ...
   public String toString() {
      String res = "";
      for (int i = 0; i < size; i++) {
         StringBuilder buf = new ...();
         buf.append(res);
         buf.append(elements[i].toString());
         res = buf.toString();
         StringBuilder buf2 = new ...();
         buf2.append(res);
         buf2.append(", ");
         res = buf2.toString();
      }
      ...
```

Formattering 6: Strängkonkatenering

jonkv@ida

- Lösning: Använd **StringBuilder** själv (i loopar/liknande)!

```
public class ArrayList {
   private Object[] elements;
   private int size;
   …
   public String toString() {
      StringBuilder buf = new …();
      for (int i = 0; i < size; i++) {
         buf.append(elements[i]);
         buf.append(", ");
      }
      return buf.toString();
   }
}
```



Formattering 6: Detaljerad kontroll

jonkv@ida

- För mer detaljerad kontroll (överkurs):
  - Se **String.format()** – liknar **printf**() i C/C++

```
String s1 = String.format("Hello, %s!  You are %d years old.\n",
                                        person.name, person.age);
```

```
// Minst 30, högst 40 tecken (utfyllt med mellanslag)
String s2 = String.format("Hello, %30.40s!\n", person.name);
```