



Tecken och strängar med och utan Java

Introduktion till tecken

Begrepp: Tecken, kodpunkter, kodningar, ...

Unicode: A till Z... och mer

- Steg 1: Ett tecken (en symbol)

- Odelbar symbol
- Minsta enheten för information

- A B C D

- # " ! ()

- å ä ö ÷ ã ě ħ Ĩ æ Γ Ъ Ω Α Ж ĩ

- ੲ ੳ ੴ ੵ ੶ ੷ ੸ ੹ ੺ ੻ ੼ ੽ ੾ ੿

- € e ∞ ∫ ℝ *m* ∇ fff :: ≧ ⊛ ☺ ♠ ♦ △

- ॐ ⊕ → ॐ⁰ ॐ₃ ۱۱۱ 略 ⊗ ≡ ✱

- Steg 2: En **teckenreportoar** (*character repertoire*)

- Välj ut en bestämd (men inte *ordnad*)

mängd av **tillgängliga tecken**

											w	0	M	}	a
p	N	!	o	.	D	b	]	y	8	'	C		L	z	
5	h	G	(	V	[	c	v	P	\	U	:	7	m	;	
<	4	S	=	2	I	J	_	A	1	%	g	F	~	R	
H	+	e	&	d	B	n	s	#	f	{	k		l	*	
@	/	i	O	`	W	”	r	>	X	)	6	Q	x	9	
u	t	-	K	3	,	T	j	q	E	?	\$	Y	^	Z	

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		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-		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
2-		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6-	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8-																
9-																
A-		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	®	¯	
B-	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C-	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D-	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E-	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F-	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

jonkv@ida

[illegible]

Kodad teckenuppsättning (4)



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

我 wǒ I; me	的 de possessive p.	你 nǐ you	是 shì to be; is	了 le completed	不 bù no; not	们 men plural p.	这 zhè this	一 yī one; a	他 tā he; him	么 me interrogative p.
子 zǐ child	生 shēng life; raw	时 shí time	样 yàng manner	也 yě also; too	和 hé and	下 xià below; down	真 zhēn really; truly	现 xiàn present	做 zuò to do	大 dà big
觉 jué to feel	太 tài too (much)	该 gāi should	当 dāng to be; just at	经 jīng pass through	妈 mā mum	用 yòng to use	打 dǎ to hit	地 de -ly structural p.	再 zài again; then	因 yīn reason
法 fǎ law	电 diàn electric	间 jiān between; room	哪 nǎ which	西 xī West	己 jǐ oneself	候 hòu to wait; season	次 cì mw. for time	信 xìn letter; to trust	欢 huān joyous	正 zhèng just (right); correct
工 gōng work	许 xǔ to permit	东 dōng East	名 míng name	同 tóng same; similar	长 cháng long; length	亲 qīn parent; relative	种 zhǒng kind	者 zhě one who (is)	嘿 hēi hey	白 bái white; pure
更 gèng more	钱 qián money	马 mǎ horse	思 sī to think; to consider	部 bù section; part	场 cháng open area	嗯 ēn approval interjection	计 jì to plan	任 rèn to appoint; office	确 què solid; real	吃 chī to eat
少 shǎo few; little	切 qiē to cut; to slice	失 shī to lose	算 suàn to calculate	性 xìng nature; gender	此 cǐ this; these	必 bì must; will	备 bèi get ready	合 hé to close; together	德 dé virtue; ethics	队 duì team; group

Kodad teckenuppsättning (5)

- Och alla nya emojis?



Kodad teckenuppsättning (6)

- Java 21 använder **Unicode 15**

- **1,114,112** kodpunkter, 0_{hex} till $10\text{FFFF}_{\text{hex}}$, **149,186** namngivna tecken!

Code Point Type	Count	Delta
Alphabets, 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

Räcker inte med 8 bitar – eller 16...

- 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 "rÃœksmÃ¶rgÃ¥s"
 - 72 → r
 - c3 → Ã
 - a4 → œ, ...

- Kodningsschema:

- **Teckenummer** → **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

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[\]^...¡¢£¤¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿ...

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

Textformatting

Formattering 1: Object → String

- 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

Formatting 2: Object → String



- 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();  
}
```

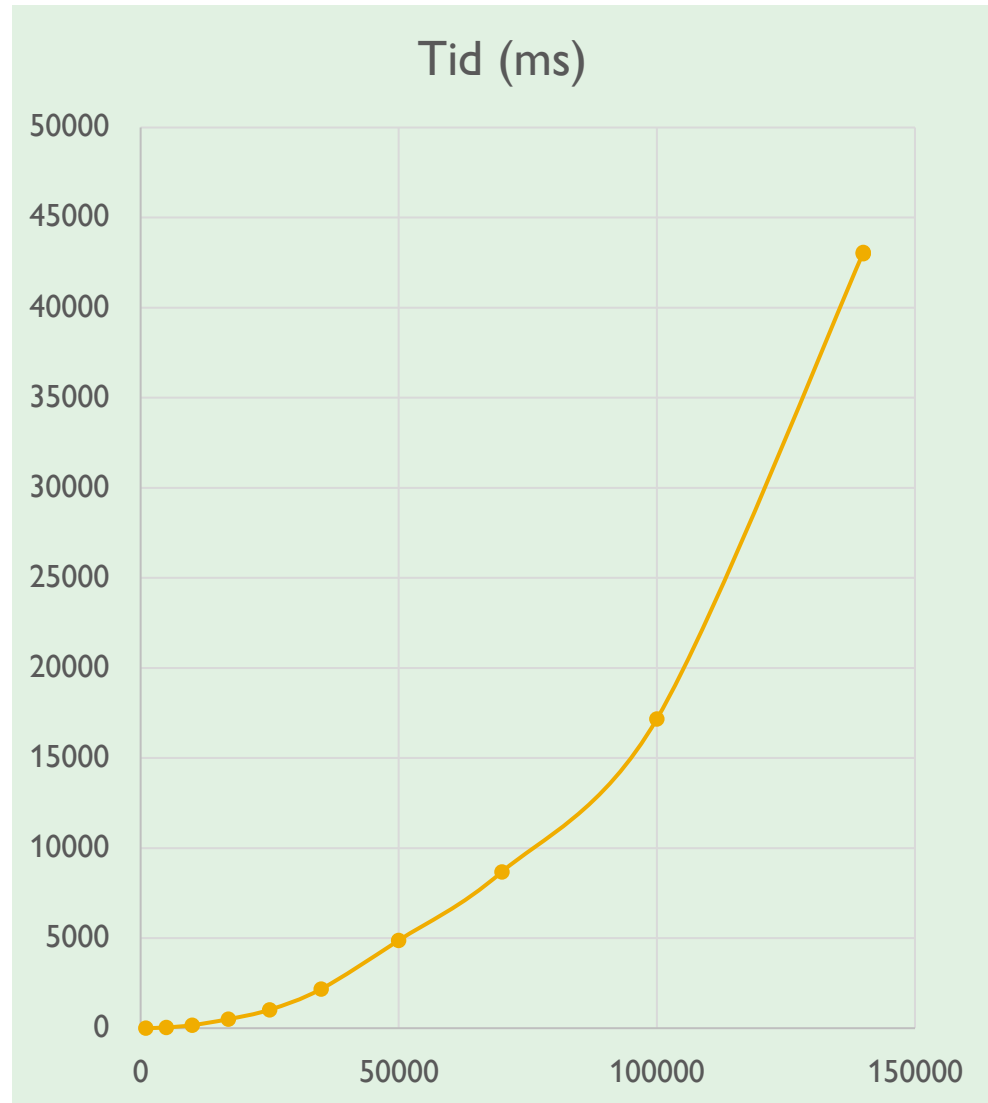
StringBuilder.append(double)
konverterar till tecken

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

Formattering 4: Strängkonkatenering

- 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

- 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();  
        }  
        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

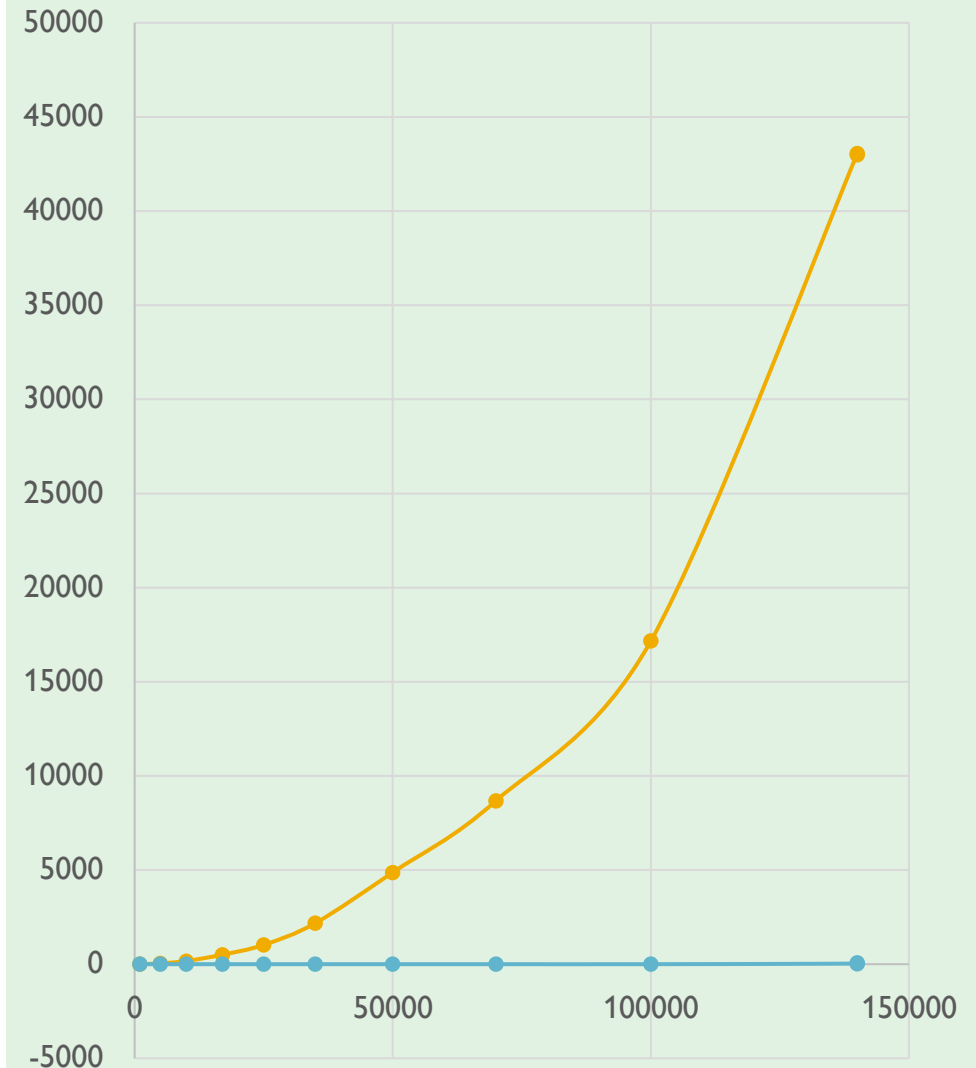
```
public class ArrayList {  
    private Object[] elements;  
    private int size;  
    // ...  
    public String toString() {  
        String res = "";  
        for (int i = 0; i < size; i++) {  
            StringBuilder buf = new ...();  
            buf.append(res);  
            buf.append(elements[i].toString());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



Formatting 6: Strängkonkatenering

- 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



- 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);
```