



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 ÷ õ ë ð ñ ã æ ÿ Ω Å Ж ĩ

■ ੲ ੳ ੴ ੵ ੶ ੷ ੸ ੹ ੺ ੻ ੼ ੽ ੾ ੿ ੿

■ € e ∞ ∫ ℝ *m* ∇ fff ÷ ≧ ⊛ ☺ ♠ ♦ △

■ ॐ ⊕ ॐ⁰ ^DC₃ 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	I	%	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	001F
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	

Kodad teckenuppsättning (3)



- **EBCDIC**
(IBM-stordatorer):
 - 129 = a, 130 = b, ...,
137 = i,
145 = j, ...

EBCDIC Code Page 00037																
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	NUL 0000	SOH 0001	STX 0002	ETX 0003	SEL 4	HT 0009	RNL 6	DEL 007F	GE 8	SFS 9	RPT 10	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1-	DLE 0010	DC1 0011	DC2 0012	DC3 0013	RES ENP 20	NL 0085	BS 0008	POC 23	CAN 0018	EM 0019	USP 26	CU1 27	IFS 001C	IGS 001D	IRS 001E	IUS ITS 001F
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2-	DS 32	SOS 33	FS 34	WUS 35	BYP INF 36	LF 000A	ETB 0017	ESC 001B	SA 40	SFE 41	SM SW 42	CSP 43	MFA 44	ENQ 0005	ACK 0006	BEL 0007
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3-			SYN 0016	IR 51	FP 52	TRN 53	NBS 54	EOT 0004	SBS 56	IT 57	RFF 58	CU3 59	DC4 0014	NAK 0015		SUB 001A
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4-	SP 0020	RSP 00A0	â 00E2	ä 00E4	à 00E0	á 00E1	ã 00E3	å 00E5	ç 00E7	ñ 00F1	¢ 00A2	· 002E	< 003C	(0028	+	 007C
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5-	& 0026	é 00E9	ê 00EA	ë 00EB	è 00E8	í 00ED	î 00EE	ï 00EF	ì 00EC	í 00DF	! 0021	\$ 0024	* 002A	) 0029	; 003B	¬ 00AC
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6-	- 002D	/ 002F	Â 00C2	Ä 00C4	À 00C0	Á 00C1	Ã 00C3	Å 00C5	Ç 00C7	Ñ 00D1	¡ 00A6	, 002C	% 0025	_ 003F	> 003E	? 003F
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7-	ø 00F8	É 00C9	Ê 00CA	Ë 00CB	È 00C8	Í 00CD	Î 00CE	Ï 00CF	Ì 00CC	˘ 0060	: 003A	# 0023	@ 0040	' 0027	= 003D	" 0022
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8-	ø 00D8	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	« 00AB	» 00BB	ð 00F0	ý 00FD	þ 00FE	± 00B1
	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9-	° 00B0	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F	p 0070	q 0071	r 0072	ª 00AA	º 00BA	æ 00E6	, 00B8	Æ 00C6	¤ 00A4
	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A-	μ 00B5	~ 007E	s 0079	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	¡ 00A1	¿ 00BF	Ð 00D0	Ý 00DD	Þ 00DE	® 00AE
	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B-	^ 005E	£ 00A3	¥ 00A5	· 00B7	© 00A9	§ 00A7	¶ 00B6	‰ 00BC	‰ 00BD	‰ 00BE	[005B	] 005D	- 00AF	“ 00AB	” 00AC	x 00D7
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C-	{ 007B	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	SHY 00AD	ô 00F4	ö 00F6	ò 00F2	ó 00F3	õ 00F5
	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D-	} 007D	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F	P 0050	Q 0051	R 0052	¹ 00B9	û 00FB	ü 00FC	ù 00F9	ú 00FA	ÿ 00FF
	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E-	\ 005C	÷ 00F7	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	² 00B2	ô 00D4	ö 00D6	ò 00D2	ó 00D3	õ 00D5
	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F-	ø 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	³ 00B3	Û 00DB	Ü 00DC	Ù 00D9	Ú 00DA	EO
	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

<https://vimeo.com/48858289>

- I, II, III kodpunkter, 0_{hex} till 10FFFF_{hex}
- >109,242 grafiska (synliga) tecken!
 - 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 ਈਉਊਏਓਅੰਖਗਘਙ
 - 0EDC..0EDD LAO HO NO..LAO HO MO ຫາທ໌ທຸ່ນ
 - 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
 - 100000..
10FFFF <private-use-100000>..<private-use-10FFFF>

- Steg 4: *Character encoding form + character encoding scheme*
 - *Ungefär*: Hur representerar man dessa tal som *bytes* i en dator?
- ISO Latin-I:
 - Värdet 0-255
 - Får plats i en enda byte
 - ➔ Trivialt

■ Unicode:

- >100000 tecken
- Räcker inte ens med 2 bytes (16 bitar)

■ Enkel representation: UTF-32

- Tecken → 32-bitars heltal
- Oftast ineffektivt

■ Exempel:

■ 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

■ 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"



■ Kodningsschema:

- Teckennummer → Lagring som bytes
- 00000 - 0007F → 0xxxxxxx (*samma som ASCII!*)
- 00080 - 007FF → 110xxxxx 10xxxxxx
- 00800 - 0FFFF → 1110xxxx 10xxxxxx 10xxxxxx
- 10000 – FFFFF → 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Informativt
bitmönster!

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`
Unicode-tecken #120 är ett 'x'.

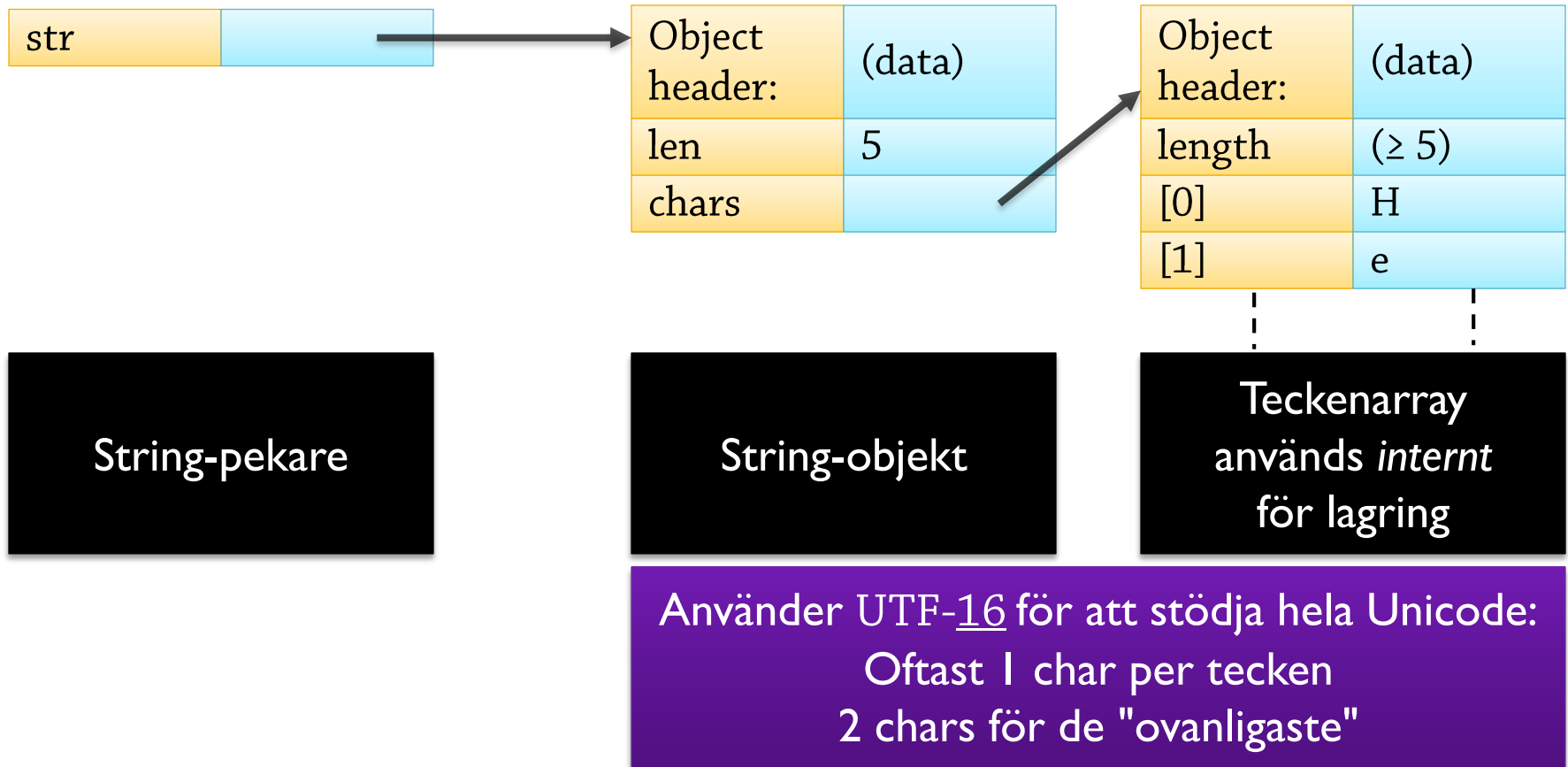
■ Strängar

- Noll eller flera symboler
- Inom citat: "Hello World"
- Objekttyp, **java.lang.String**

Strängar som objekt

String 1: Strängar är objekt

- Strängar i Java är **objekt**
 - `String str = "Hello";`



- Även strängliteraler representerar objekt

- Kan användas "direkt"

- `System.out.println("Hello".length());` `// 5`

Resulterar i ett String-objekt med 5 tecken

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

== jämför pekare!

```
if (str == "hello") {  
    // Jämför om det är samma strängobjekt  
}
```

```
if (str.equals("hello")) {  
    // 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]
```

Detta är en override – diskuteras senare

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(int)
konverterar till tecken

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

Formattering 4: Strängkonkatenering

- Konkatenering kan bli ineffektivt...

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

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

Kopiera alla tecken till StringBuilder
Lägg till några tecken
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 = "ArrayList<";  
        for (int i = 0; i < size; i++) {  
            if (i != 0) {  
                StringBuilder buf = new ...();  
                buf.append(res);  
                buf.append(", ");  
                res = buf.toString();  
            }  
            StringBuilder buf2 = new ...();  
            buf2.append(res);  
            buf2.append(elements[i].toString());  
            res = buf2.toString();  
        }  
        ...  
    }  
}
```

Formattering 5: Strängkonkatenering

- Lösning: Använd **StringBuilder** själv!

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



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

Formattering 6: Detaljerad kontroll

- För mer detaljerad kontroll: **String.format()**
 - Liknar **printf()** i C/C++

%s=sträng

%d=heltal

```
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);
```

```
// Ange en locale...
```

```
String s3 = String.format(Locale.FRANCE, "e = %+10.4f", Math.E);
```

```
// ➔ "e = +2,7183"
```

- I **PrintStream** finns både **println()** och **printf()**
 - `System.out.printf("Hello, %s!\n", person.name);` // Och så vidare...