Tecken och strängar i Java

jonas.kvarnstrom@liu.se – 2017

Introduktion till tecken

Begrepp: Tecken, kodpunkter, kodningar, …

Unicode: A till Z… och mer

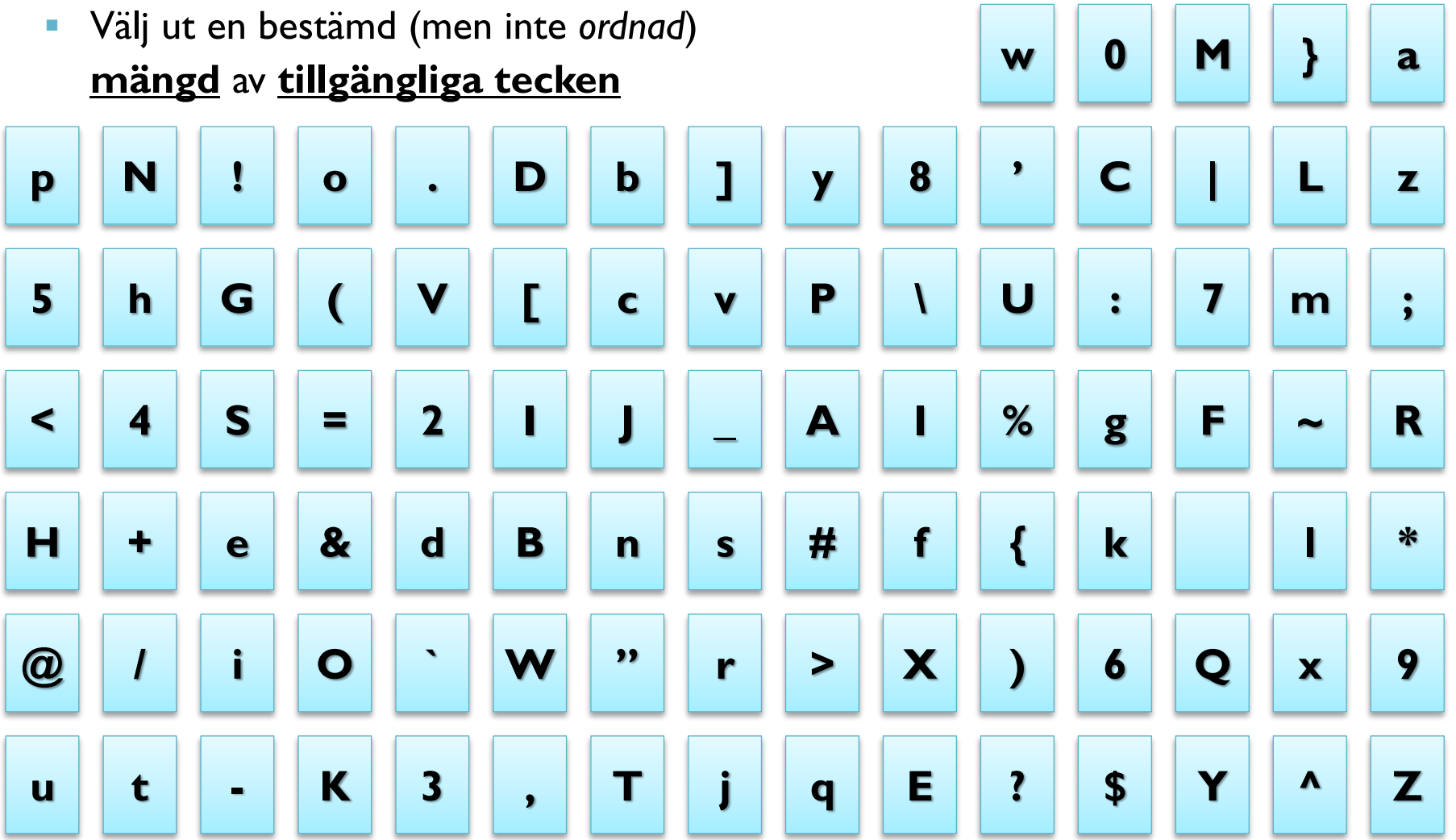
Tecken

- Steg 1: Ett **<u>tecken</u>** (en **<u>symbol</u>**)
  - 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 ї̆
    - ਈ ਉ ਊ ਏ ਓ ਐ ਕ ਖ ਗ ਘ ਙ
    - € e ∞ ∫ ℝ 𝓂 ∇ ∰ ∺ ≩ ⊛ ☺ ♠ ◆ ◬
    - ௵ ⟴ ဍ္ဍ ␓ ريال 略 Ⓧ ⇶ ✷

Teckenrepertoar

jonkv@ida

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



Kodad teckenuppsättning

jonkv@ida

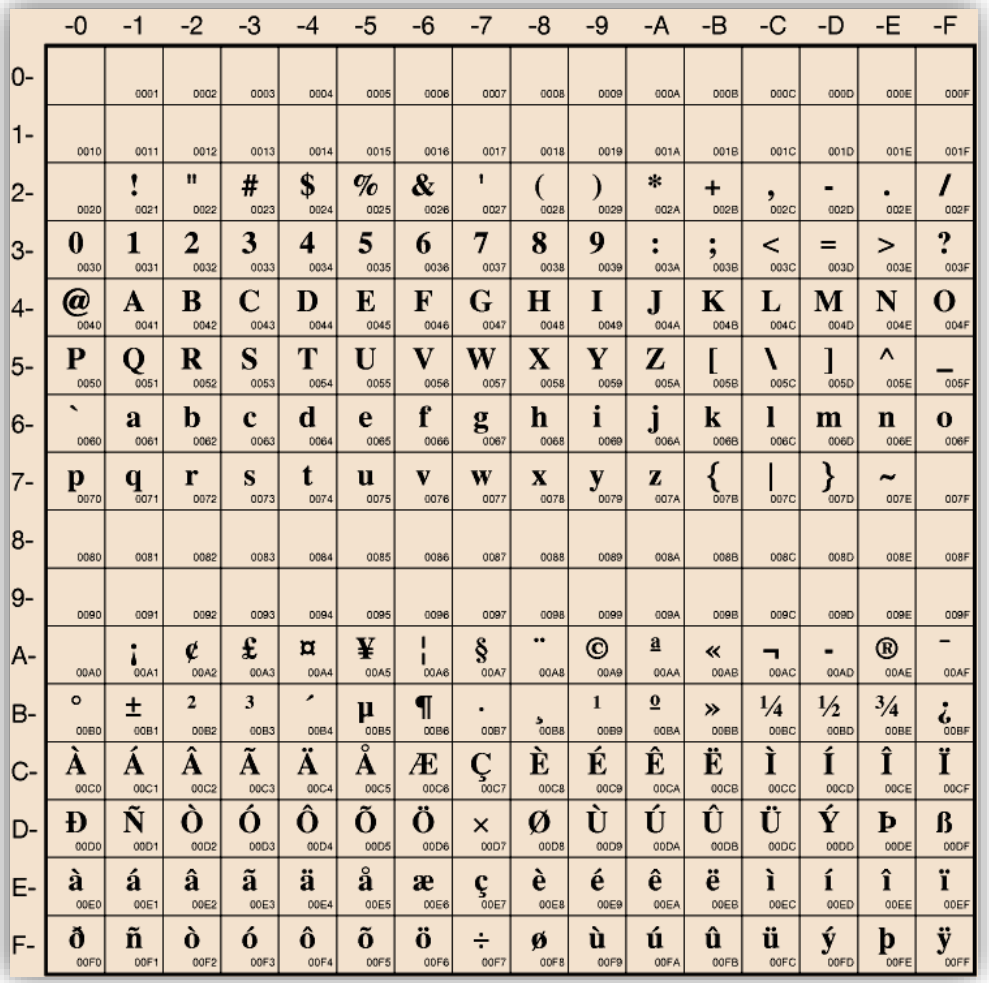
- Steg 3: En **kodad teckenuppsättning** (*coded character set, CCS*)
  - Varje tecken motsvaras av ett **heltal** – exempel: **ASCII** från 0 till 127

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)

jonkv@ida

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



Kodad teckenuppsättning (3)

jonkv@ida

- **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 *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 0010 *16*	DC1 0011 *17*	DC2 0012 *18*	DC3 0013 *19*	RES ENP *20*	NL 0085 *21*	BS 0008 *22*	POC *23*	CAN 0018 *24*	EM 0019 *25*	UBS *26*	CU1 *27*	IFS 001C *28*	IGS 001D *29*	IRS 001E *30*	IUS ITB 001F *31*
2–	DS *32*	SOS *33*	FS *34*	WUS *35*	BYP INP *36*	LF 000A *37*	ETB 0017 *38*	ESC 001B *39*	SA *40*	SFE *41*	SM SW *42*	CSP *43*	MFA *44*	ENQ 0005 *45*	ACK 0006 *46*	BEL 0007 *47*
3–	*48*	*49*	SYN 0016 *50*	IR *51*	PP *52*	TRN *53*	NBS *54*	EOT 0004 *55*	SBS *56*	IT *57*	RFF *58*	CU3 *59*	DC4 0014 *60*	NAK 0015 *61*	*62*	SUB 001A *63*
4–	SP 0020 *64*	RSP 00A0 *65*	â 00E2 *66*	ä 00E4 *67*	à 00E0 *68*	á 00E1 *69*	ã 00E3 *70*	å 00E5 *71*	ç 00E7 *72*	ñ 00F1 *73*	¢ 00A2 *74*	. 002E *75*	< 003C *76*	(0028 *77*	+ 002B *78*	| 007C *79*
5–	& 0026 *80*	é 00E9 *81*	ê 00EA *82*	ë 00EB *83*	è 00E8 *84*	í 00ED *85*	î 00EE *86*	ï 00EF *87*	ì 00EC *88*	ß 00DF *89*	! 0021 *90*	$ 0024 *91*	* 002A *92*	) 0029 *93*	; 003B *94*	¬ 00AC *95*
6–	- 002D *96*	/ 002F *97*	Â 00C2 *98*	Ä 00C4 *99*	À 00C0 *100*	Á 00C1 *101*	Ã 00C3 *102*	Å 00C5 *103*	Ç 00C7 *104*	Ñ 00D1 *105*	¦ 00A6 *106*	, 002C *107*	% 0025 *108*	_ 005F *109*	> 003E *110*	? 003F *111*
7–	ø 00F8 *112*	É 00C9 *113*	Ê 00CA *114*	Ë 00CB *115*	È 00C8 *116*	Í 00CD *117*	Î 00CE *118*	Ï 00CF *119*	Ì 00CC *120*	` 0060 *121*	: 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

https://vimeo.com/48858289

- Java 8 använder **Unicode 6.2.0**
  - 1,114,112 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 ੁੂੇੈੋੌਅਆਖਗਘਛ
    - 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
    - 100000..10FFFD <private-use-100000>..<private-use-10FFFD>

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:
  - Värden 0-255
  - Får plats i en enda byte
  - ➔ Trivialt

Teckenkodning (2)

- **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

Teckenkodning (3)

jonkv@ida

- **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"
  - 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!

Text i Java

Tecken och strängar

jonkv@ida

Java skiljer på:

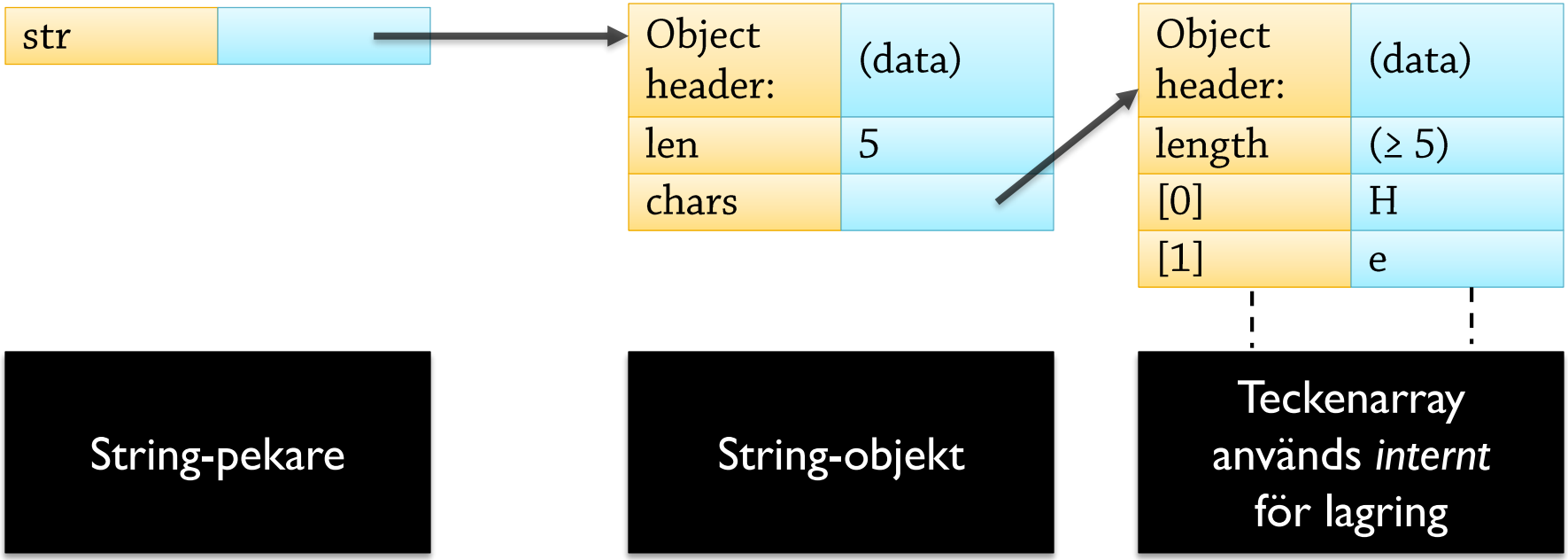
- **Tecken**
  - Exakt en symbol
  - Inom apostrofer: ’x’
  - Primitiv datatyp, **char**
    - 16 bitar,
    - Unicode-tecken 0–65535
    - `char c = 'x';`
    - `char c = 120;  // c=='x', eftersom Unicodetecken #120 är ett 'x'.`
- **Strängar**
  - Noll eller flera symboler
  - Inom citat: "Hello World"
  - Objekttyp, **java.lang.String**

Strängar som objekt

String 1: Strängar är objekt

jonkv@ida

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



Använder UTF-16 för att stödja hela Unicode:
Oftast 1 char per tecken
2 chars för de "ovanligaste"

String 2: Literaler

- Även **strängliteraler** representerar **objekt**
  - Kan användas "direkt"
    - `System.out.println("Hello".length());  // 5`

Resulterar i ett String-**objekt** med 5 tecken

  - Gör inte **onödiga kopior**
    - `new String("Hello")`

String 3: Jämförelser

- 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[\]^…¡¢£¤¥…¾¿ÀÁÂÃÄÅÆÇÈ…ÔÕÖ×ØÙ…ĀĄ…ŘƧ…Ж…ڀ…

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

Textformattering

Formattering 1: Primitiva datatyper

jonkv@ida

- Konvertera **primitiva datatyper till text**:
  - Använd de överlagrade String.valueOf()-metoderna
    - ```
      static String valueOf(char c) { … }
      static String valueOf(int i) { … }
      static String valueOf(boolean b) { … }
      …
      ```
    - ```
      String str = String.valueOf(127);          // "127"
      ```

Formattering 2: Object ➔ String

jonkv@ida

- Varje objekt har en **strängrepresentation**
  - Metoden **toString**() ärvs från **Object**
    - Ska returnera en String som representerar objektet på något sätt
  - 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() ➔ den ärvda implementationen från Object anropas
➔ klassnamn + "@" + ett unikt ID
      ```

Formattering 3: Object ➔ String

jonkv@ida

- Implementera en egen toString()!
  - ```
    public class Circle {
       …
       public String toString() {
          return "Circle[" + x + ", " + y + ", " + r + ']';
       }
       …
    }
    ```
  - **>> java Circle**
    Circle[2.0, 4.0, 5.0]

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

Formattering 4: Strängkonkatenering

jonkv@ida

- 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)
anropar String.valueOf()

Skapa en String med samma tecken
som denna StringBuilder

Formattering 5: Strängkonkatenering

jonkv@ida

- 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?

```
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();
      }
      …
```

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

Formattering 6: Strängkonkatenering

jonkv@ida

- 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 7: Detaljerad kontroll

jonkv@ida

- 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...

Formattering 8: Detaljerad kontroll

jonkv@ida

- http://docs.oracle.com/javase/7/docs/api/java/text/MessageFormat.html

Python

```
>>> 'Use the {}, {}!'.format('power','Amy')
'Use the power, Amy!'
>>> "A {0} {1} is a {1} that's {0}.".format('hot','dog')
"A hot dog is a dog that's hot."
```

Java

```
String s1 = MessageFormat.format("Use the {0}, {1}", "power", "Amy");
String s2 = String.format("Use the %s, %s!", "power", "Amy");

String s3 = MessageFormat.format("A {0} {1} is a {1} that's {0}.", "hot", "dog");
String s4 = String.format("A %0$s %1$s is a %1$s that's %0$s.", "hot", "dog");
```