

# Primitive Data Types

August 28, 2006

ComS 207: Programming I (in Java)  
Iowa State University, FALL 2006  
Instructor: Alexander Stoytchev

© 2004 Pearson Addison-Wesley. All rights reserved

## Quick review of last lecture

© 2004 Pearson Addison-Wesley. All rights reserved

## String Concatenation

- The *string concatenation operator* (+) is used to append one string to the end of another  
`"Peanut butter " + "and jelly"`
- It can also be used to append a number to a string
- A string literal cannot be broken across two lines in a program
- See [Facts.java](#) (page 65)

© 2004 Pearson Addison-Wesley. All rights reserved

## String Concatenation

- The + operator is also used for arithmetic addition
- The function that it performs depends on the type of the information on which it operates
- If both operands are strings, or if one is a string and one is a number, it performs string concatenation
- If both operands are numeric, it adds them
- The + operator is evaluated left to right, but parentheses can be used to force the order
- See [Addition.java](#) (page 67)

© 2004 Pearson Addison-Wesley. All rights reserved

## Escape Sequences

- What if we wanted to print a the quote character?
- The following line would confuse the compiler because it would interpret the second quote as the end of the string

```
System.out.println ("I said "Hello" to you.");
```

- An *escape sequence* is a series of characters that represents a special character
- An escape sequence begins with a backslash character (\)

```
System.out.println ("I said \"Hello\" to you.");
```

© 2004 Pearson Addison-Wesley. All rights reserved

## Escape Sequences

- Some Java escape sequences:

Escape Sequence	Meaning
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\"</code>	double quote
<code>'</code>	single quote
<code>\\</code>	backslash

- See [Roses.java](#) (page 68)

© 2004 Pearson Addison-Wesley. All rights reserved

## Variables

- A *variable* is a name for a location in memory
- A variable must be *declared* by specifying the variable's name and the type of information that it will hold

data type                      variable name

```
int total;
```

int count, temp, result;

Multiple variables can be created in one declaration

© 2004 Pearson Addison-Wesley. All rights reserved

## Assignment

- An *assignment statement* changes the value of a variable
- The assignment operator is the = sign

```
total = 55;
```

- The expression on the right is evaluated and the result is stored in the variable on the left
- The value that was in `total` is overwritten
- You can only assign a value to a variable that is consistent with the variable's declared type
- See [Geometry.java](#) (page 71)

© 2004 Pearson Addison-Wesley. All rights reserved

## Constants

- A constant is an identifier that is similar to a variable except that it holds the same value during its entire existence
- As the name implies, it is constant, not variable
- The compiler will issue an error if you try to change the value of a constant
- In Java, we use the `final` modifier to declare a constant

```
final int MIN_HEIGHT = 69;
```

© 2004 Pearson Addison-Wesley. All rights reserved

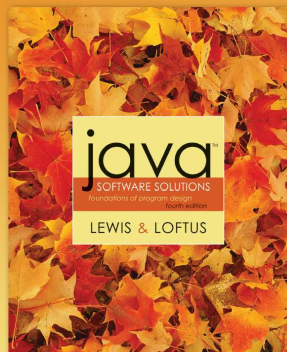
## Constants

- Constants are useful for three important reasons
- First, they give meaning to otherwise unclear literal values
  - For example, `MAX_LOAD` means more than the literal 250
- Second, they facilitate program maintenance
  - If a constant is used in multiple places, its value need only be updated in one place
- Third, they formally establish that a value should not change, avoiding inadvertent errors by other programmers

© 2004 Pearson Addison-Wesley. All rights reserved

## Chapter 2

### Sections 2.3 & 2.4



PEARSON  
Addison  
Wesley

© 2005 Pearson Addison-Wesley. All rights reserved.

## Primitive Data

- There are eight primitive data types in Java
- Four of them represent integers:
  - `byte`, `short`, `int`, `long`
- Two of them represent floating point numbers:
  - `float`, `double`
- One of them represents characters:
  - `char`
- And one of them represents boolean values:
  - `boolean`

© 2004 Pearson Addison-Wesley. All rights reserved

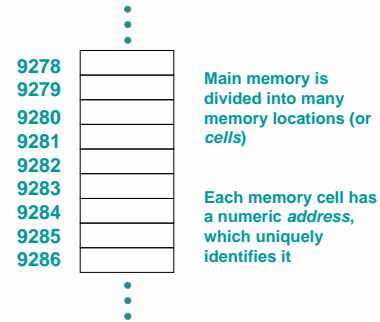
## Numeric Primitive Data

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

Type	Storage	Min Value	Max Value
byte	8 bits	-128	127
short	16 bits	-32,768	32,767
int	32 bits	-2,147,483,648	2,147,483,647
long	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
float	32 bits	+/- $3.4 \times 10^{38}$ with 7 significant digits	
double	64 bits	+/- $1.7 \times 10^{308}$ with 15 significant digits	

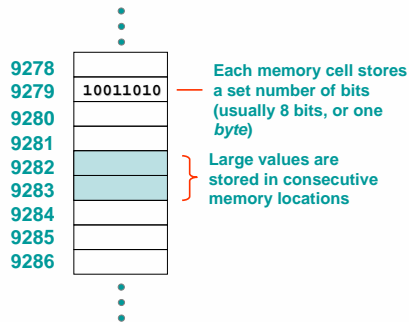
© 2004 Pearson Addison-Wesley. All rights reserved

## Computer Memory



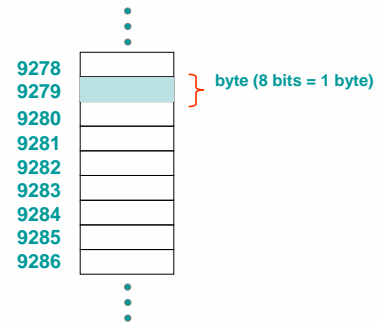
© 2004 Pearson Addison-Wesley. All rights reserved

## Storing Information



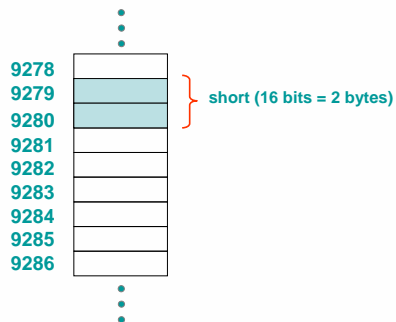
© 2004 Pearson Addison-Wesley. All rights reserved

## Storing a byte



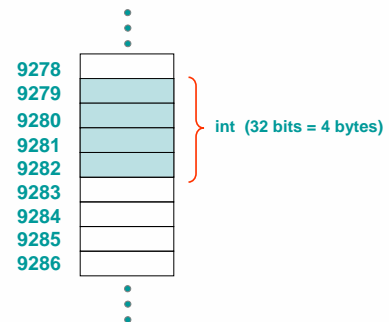
© 2004 Pearson Addison-Wesley. All rights reserved

## Storing a short



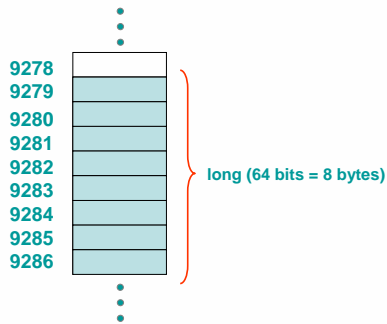
© 2004 Pearson Addison-Wesley. All rights reserved

## Storing an int



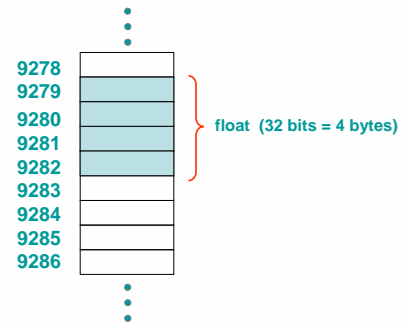
© 2004 Pearson Addison-Wesley. All rights reserved

## Storing a long



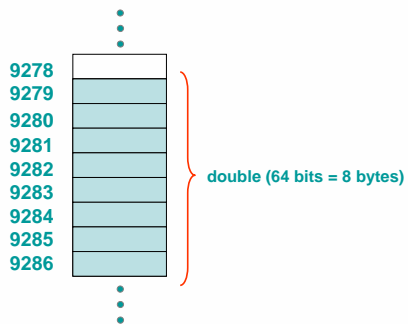
© 2004 Pearson Addison-Wesley. All rights reserved.

## Storing a float



© 2004 Pearson Addison-Wesley. All rights reserved.

## Storing a double



© 2004 Pearson Addison-Wesley. All rights reserved.

## Characters

- A `char` variable stores a single character
- Character literals are delimited by single quotes:  
`'a'` `'x'` `'7'` `'$'` `'.'` `'\n'`
- Example declarations:  

```
char topGrade = 'A';  
char terminator = ';', separator = ',';
```
- Note the distinction between a primitive character variable, which holds only one character, and a `String` object, which can hold multiple characters

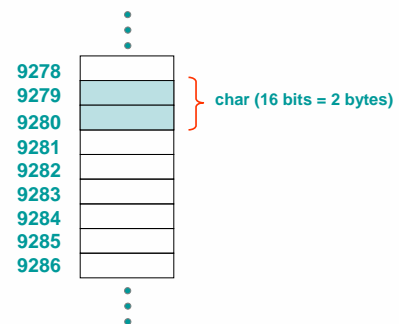
© 2004 Pearson Addison-Wesley. All rights reserved.

## Character Sets

- A *character set* is an ordered list of characters, with each character corresponding to a unique number
- A `char` variable in Java can store any character from the *Unicode character set*
- The Unicode character set uses sixteen bits per character, allowing for 65,536 unique characters
- It is an international character set, containing symbols and characters from many world languages

© 2004 Pearson Addison-Wesley. All rights reserved.

## Storing a char



© 2004 Pearson Addison-Wesley. All rights reserved.

## Characters

- The **ASCII character set** is older and smaller than **Unicode**, but is still quite popular
- The ASCII characters are a subset of the Unicode character set, including:

uppercase letters    A, B, C, ...  
 lowercase letters    a, b, c, ...  
 punctuation        period, semi-colon, ...  
 digits                0, 1, 2, ...  
 special symbols     &, |, \, ...  
 control characters    carriage return, tab, ...

© 2004 Pearson Addison-Wesley. All rights reserved

## ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Html	Chr	Dec	Hex	Oct	Html	Chr	Dec	Hex	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	#32;	Space	64	40	100	#64;	@	96	60	140	#96;	`
1	1	001	SOH (start of heading)	33	21	041	#33;	!	65	41	101	#65;	A	97	61	141	#97;	a
2	2	002	STX (start of text)	34	22	042	#34;	"	66	42	102	#66;	B	98	62	142	#98;	b
3	3	003	ETX (end of text)	35	23	043	#35;	#	67	43	103	#67;	C	99	63	143	#99;	c
4	4	004	EOT (end of transmission)	36	24	044	#36;	\$	68	44	104	#68;	D	100	64	144	#100;	d
5	5	005	ENQ (enquiry)	37	25	045	#37;	%	69	45	105	#69;	E	101	65	145	#101;	e
6	6	006	ACK (acknowledge)	38	26	046	#38;	&	70	46	106	#70;	F	102	66	146	#102;	f
7	7	007	BEI (bell)	39	27	047	#39;	'	71	47	107	#71;	G	103	67	147	#103;	g
8	8	010	BS (backspace)	40	28	050	#40;	(	72	48	110	#72;	H	104	68	150	#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	#41;	)	73	49	111	#73;	I	105	69	151	#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	#42;	*	74	4A	112	#74;	J	106	70	152	#106;	j
11	B	013	VT (vertical tab)	43	2B	053	#43;	+	75	4B	113	#75;	K	107	71	153	#107;	k
12	C	014	FF (NF form feed, new page)	44	2C	054	#44;	,	76	4C	114	#76;	L	108	72	154	#108;	l
13	D	015	CR (carriage return)	45	2D	055	#45;	-	77	4D	115	#77;	M	109	73	155	#109;	m
14	E	016	SO (shift out)	46	2E	056	#46;	.	78	4E	116	#78;	N	110	74	156	#110;	n
15	F	017	SI (shift in)	47	2F	057	#47;	:	79	4F	117	#79;	O	111	75	157	#111;	o
16	10	020	DLE (data link escape)	48	30	060	#48;	;	80	50	120	#80;	P	112	76	160	#112;	p
17	11	021	DC1 (device control 1)	49	31	061	#49;	<	81	51	121	#81;	Q	113	77	161	#113;	q
18	12	022	DC2 (device control 2)	50	32	062	#50;	=	82	52	122	#82;	R	114	78	162	#114;	r
19	13	023	DC3 (device control 3)	51	33	063	#51;	>	83	53	123	#83;	S	115	79	163	#115;	s
20	14	024	DC4 (device control 4)	52	34	064	#52;	?	84	54	124	#84;	T	116	7A	164	#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	#53;	@	85	55	125	#85;	U	117	7B	165	#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	#54;	[	86	56	126	#86;	V	118	7C	166	#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	#55;	\	87	57	127	#87;	W	119	7D	167	#119;	w
24	18	030	CAN (cancel)	56	38	070	#56;	]	88	58	130	#88;	X	120	7E	170	#120;	x
25	19	031	EM (end of medium)	57	39	071	#57;	^	89	59	131	#89;	Y	121	7F	171	#121;	y
26	1A	032	SUB (substitute)	58	3A	072	#58;	_	90	5A	132	#90;	Z	122	7A	172	#122;	z
27	1B	033	ESC (escape)	59	3B	073	#59;	`	91	5B	133	#91;	[	123	7B	173	#123;	[
28	1C	034	FS (file separator)	60	3C	074	#60;	{	92	5C	134	#92;	\	124	7C	174	#124;	\
29	1D	035	GS (group separator)	61	3D	075	#61;		93	5D	135	#93;	]	125	7D	175	#125;	]
30	1E	036	RS (record separator)	62	3E	076	#62;	~	94	5E	136	#94;	^	126	7E	176	#126;	^
31	1F	037	US (unit separator)	63	3F	077	#63;	?	95	5F	137	#95;	_	127	7F	177	#127;	_

© 2004 Pearson Addison-Wesley. All rights reserved

Source: [www.LookupTables.com](http://www.LookupTables.com)

## Extended ASCII Codes

128	Ç	144	É	161	í	177	ÿ	193	ƒ	209	ƒ	225	ß	241	±
129	à	145	ê	162	ó	178	ÿ	194	g	210	g	226	À	242	²
130	á	146	ë	163	ô	179		195	h	211	h	227	Á	243	³
131	â	147	ì	164	ù	180	†	196	i	212	i	228	Â	244	´
132	ã	148	í	165	ú	181	‡	197	ª	213	ª	229	Ã	245	µ
133	ä	149	î	166	û	182	§	198	«	214	«	230	Ä	246	¶
134	å	150	ï	167	ü	183	¶	199	»	215	»	231	Å	247	·
135	ç	151	ú	168	ý	184	·	200	¼	216	¼	232	Æ	248	¸
136	è	152	û	169	ÿ	185	¸	201	½	217	½	233	Ë	249	¹
137	é	153	ü	170	ÿ	186	¹	202	¾	218	¾	234	Ì	250	º
138	ê	154	ý	171	ÿ	187	º	203	¾	219	¾	235	Í	251	»
139	ë	156	ÿ	172	¼	188	»	204	¾	220	¾	236	Î	252	¼
140	ì	157	ÿ	173	½	189	¼	205	¾	221	¾	237	Ï	253	½
141	í	158	ÿ	174	¾	190	½	206	¾	222	¾	238	Ð	254	¾
142	î	159	ÿ	175	¾	191	¾	207	¾	223	¾	239	Ñ	255	¾
143	ï	160	á	176	ÿ	192	¾	208	¾	224	¾	240	Ò	256	¾

© 2004 Pearson Addison-Wesley. All rights reserved

Source: [www.LookupTables.com](http://www.LookupTables.com)

## The Unicode Character Code

- <http://www.unicode.org/charts/>

© 2004 Pearson Addison-Wesley. All rights reserved

## Boolean

- A **boolean value** represents a true or false condition
- The reserved words **true** and **false** are the only valid values for a boolean type
 

```
boolean done = false;
```
- A **boolean variable** can also be used to represent any two states, such as a light bulb being on or off

© 2004 Pearson Addison-Wesley. All rights reserved

## Run alphabet examples

© 2004 Pearson Addison-Wesley. All rights reserved

## Expressions

- An *expression* is a combination of one or more operators and operands
- Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

Addition	+
Subtraction	-
Multiplication	*
Division	/
Remainder	%

- If either or both operands used by an arithmetic operator are floating point, then the result is a floating point

© 2004 Pearson Addison-Wesley. All rights reserved

## Division and Remainder

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)

14 / 3 equals 4

8 / 12 equals 0

- The remainder operator (%) returns the remainder after dividing the second operand into the first

14 % 3 equals 2

8 % 12 equals 8

© 2004 Pearson Addison-Wesley. All rights reserved

## Operator Precedence

- Operators can be combined into complex expressions

result = total + count / max - offset;

- Operators have a well-defined precedence which determines the order in which they are evaluated
- Multiplication, division, and remainder are evaluated prior to addition, subtraction, and string concatenation
- Arithmetic operators with the same precedence are evaluated from left to right, but parentheses can be used to force the evaluation order

© 2004 Pearson Addison-Wesley. All rights reserved

## Operator Precedence

- What is the order of evaluation in the following expressions?

a + b + c + d + e      a + b \* c - d / e  
 1 2 3 4                    3 1 4 2

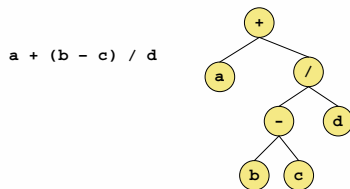
a / (b + c) - d % e  
 2 1 4 3

a / (b \* (c + (d - e)))  
 4 3 2 1

© 2004 Pearson Addison-Wesley. All rights reserved

## Expression Trees

- The evaluation of a particular expression can be shown using an *expression tree*
- The operators lower in the tree have higher precedence for that expression



© 2004 Pearson Addison-Wesley. All rights reserved

## Assignment Revisited

- The assignment operator has a lower precedence than the arithmetic operators


First the expression on the right hand side of the = operator is evaluated

answer = sum / 4 + MAX \* lowest;  
 4 1 3 2




Then the result is stored in the variable on the left hand side

© 2004 Pearson Addison-Wesley. All rights reserved



**Temperature Conversion Example**

© 2004 Pearson Addison-Wesley. All rights reserved



**THE END**

© 2004 Pearson Addison-Wesley. All rights reserved