

CSE1720: Primitive Types, Primitive Expression Evaluation

1

1.2.2 The Integer Types

A **type** is a **range** of values and a set of **operations** on these values.

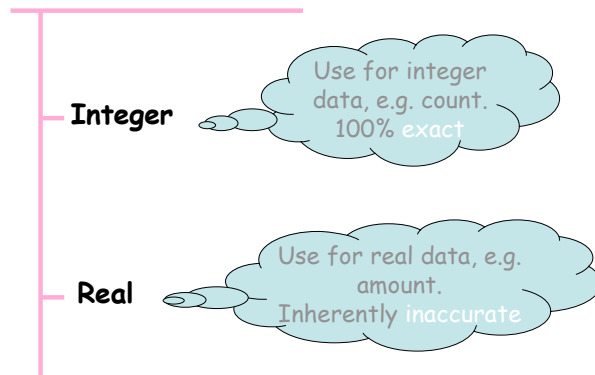
The range of the **int** type consists of all whole numbers between -2 and +2 billions (approx). int supports the four **arithmetic** operations plus the **remainder**.

The **long** type is very similar to int except its range is much bigger, +/-10¹⁹

An integer **literal** has an int type unless suffixed by **L (l)**, in which case it is long.

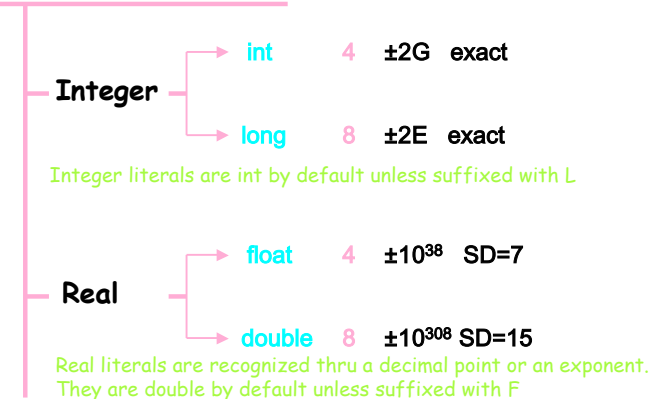
2

1.2.4 Other Data Types



3

Numeric Types



4

The Type `boolean`

- Stores the result on a condition
- Has only two possible values
- `true` and `false` are reserved words
- Boolean variables are not integers

Note: Boolean literals are the easiest to recognize!

5

The Character Type `char`

- A letter, digit, or symbol
- Digits versus Numbers
- Store the code, not the typeface
- The case of English: ASCII
- `char` is thus an (unsigned) integer type
- Unicode has 64K codes

Character literals are recognized by single quotes surrounding one character, e.g. `'A'`

6

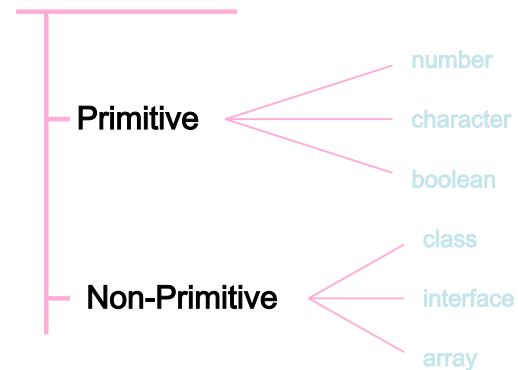
More on Characters

Code	Character
0	
⋮	
32	space
⋮	
48-57	'0'-'9'
⋮	
65-90	'A'-'Z'
⋮	
97-122	'a'-'z'
⋮	
65535	

Escape	Meaning
<code>\uxxxx</code>	The character whose code is (hex) <code>xxxx</code>
<code>\'</code>	Single quote
<code>\"</code>	Double quote
<code>\\</code>	Backslash
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\f</code>	Form Feed
<code>\t</code>	Tab
<code>\b</code>	Backspace

7

1.2.5 Primitive & Non-Primitive



8

Java's Primitive Type

PRIMITIVE TYPES		Type	Size (bytes)	Approximate Range		S.D.
				min	max	
NUMERIC	SIGNED	byte	1	-128	+127	N/A
		short	2	-32,768	+32,767	N/A
		int	4	-2 ₁₀ ³¹	+2 ₁₀ ³¹	N/A
		long	8	-9 ₁₀ ¹⁸	+9 ₁₀ ¹⁸	N/A
	UNSIGNED	char	2	0	65,535	N/A
REAL	SINGLE	float	4	+3.4 ₁₀ ³⁸	+3.4 ₁₀ ³⁸	7
	DOUBLE	double	8	-1.7 ₁₀ ³⁰⁸	+1.7 ₁₀ ³⁰⁸	15
BOOLEAN		boolean	1	true/false		N/A

1.3.1 The int Arithmetic Operators

Precedence	Operator	Kind	Syntax	Operation
-5 →	+	infix	x + y	add y to x
	-	infix	x - y	subtract y from x
-4 →	*	infix	x * y	multiply x by y
	/	infix	x / y	divide x by y
	%	infix	x % y	remainder of x / y
-2 →	+	prefix	+x	identity
	-	prefix	-x	negate x
	++	prefix	++x	x = x + 1; result = x
	--	prefix	--x	x = x - 1; result = x
-1 →	++	postfix	x++	result = x; x = x + 1
	--	postfix	x--	result = x; x = x - 1

10

Examples

```
double price;
price = 17.25;
int quantity = 25;
boolean isValid = false;
double cost;
cost = price;
double extended;
extended = quantity * price;
```

Can combine declaration with assignment.

RHS is a variable

RHS is an expression

11

Example

$$5 + (4 - 3) / 5 - 2 * 3 \% 4$$

12

Example


$$5 + (4 - 3) / 5 - 2 * 3 \% 4$$

$$= 5 + 1 / 5 - 2 * 3 \% 4$$

13

Example

$$5 + (4 - 3) / 5 - 2 * 3 \% 4$$

$$= 5 + 1 / 5 - 2 * 3 \% 4$$


14

Example

$$5 + (4 - 3) / 5 - 2 * 3 \% 4$$

$$= 5 + 1 / 5 - 2 * 3 \% 4$$


$$= 5 + 0 - 2 * 3 \% 4$$

15

Example

$$5 + (4 - 3) / 5 - 2 * 3 \% 4$$

$$= 5 + 1 / 5 - 2 * 3 \% 4$$

$$= 5 + 0 - 2 * 3 \% 4$$


16

Example

$$\begin{aligned}
 & 5 + (4 - 3) / 5 - 2 * 3 \% 4 \\
 = & 5 + 1 / 5 - 2 * 3 \% 4 \\
 = & 5 + 0 - 2 * 3 \% 4 \\
 = & 5 + 0 - 6 \% 4
 \end{aligned}$$

17

Example

$$\begin{aligned}
 & 5 + (4 - 3) / 5 - 2 * 3 \% 4 \\
 = & 5 + 1 / 5 - 2 * 3 \% 4 \\
 = & 5 + 0 - 2 * 3 \% 4 \\
 = & 5 + 0 - 6 \% 4
 \end{aligned}$$



18

Example

$$\begin{aligned}
 & 5 + (4 - 3) / 5 - 2 * 3 \% 4 \\
 = & 5 + 1 / 5 - 2 * 3 \% 4 \\
 = & 5 + 0 - 2 * 3 \% 4 \\
 = & 5 + 0 - 6 \% 4 \\
 = & 5 + 0 - 2
 \end{aligned}$$

19

Example

$$\begin{aligned}
 & 5 + (4 - 3) / 5 - 2 * 3 \% 4 \\
 = & 5 + 1 / 5 - 2 * 3 \% 4 \\
 = & 5 + 0 - 2 * 3 \% 4 \\
 = & 5 + 0 - 6 \% 4 \\
 = & 5 + 0 - 2
 \end{aligned}$$



20

Example

```

5 + (4 - 3) / 5 - 2 * 3 % 4
= 5 + 1 / 5 - 2 * 3 % 4
= 5 + 0 - 2 * 3 % 4
= 5 + 0 - 6 % 4
= 5 + 0 - 2
= 5 - 2

```

21

Example

```

5 + (4 - 3) / 5 - 2 * 3 % 4
= 5 + 1 / 5 - 2 * 3 % 4
= 5 + 0 - 2 * 3 % 4
= 5 + 0 - 6 % 4
= 5 + 0 - 2
= 5 - 2
= 3

```

22

1.3.2 Other Arithmetic Operators

Each of `long`, `float`, and `double` come with 11 operators with the same symbols as `int`; i.e. the symbols are **overloaded**. Note:

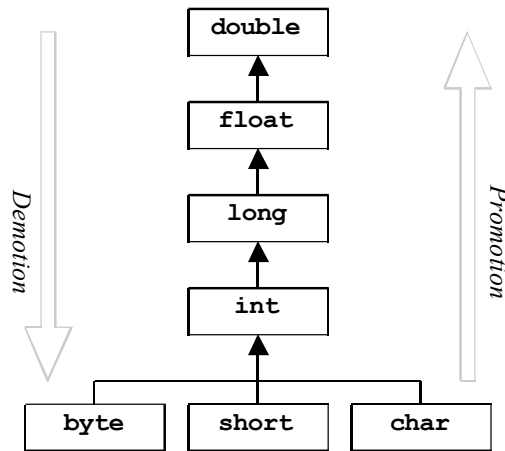
- The `int` operators satisfy **closure** through circular wrapping
- The `/` `int` operator always **rounds** toward 0 and leads to an **exception** if the divisor is zero
- The **sign** of `%` is the same as that of the dividend
- The real operators satisfy closure by adding **Infinity** and **NaN**. Hence, dividing by zero does not lead to exceptions
- $(a * b) / c$ is not the same as $a * (b / c)$ for any type
- $(a + b) - c$ is not the same as $a + (b - c)$ for real types

23

1.3.3 Mixed Types and Casting

- **Promotion** (aka widening conversion) is done automatically **when** needed
- May lead to loss of precision but the order of magnitude is preserved
- **Demotion** is not done automatically. Can be done manually thru a **cast**
- Casting is risky...**avoid it**.

24



25

Note:

- The cast operator has a precedence that is higher than `*` but less than `++`
- The `=` operator has the lowest precedence of all operators
- There are shorthand operators to combine assignment with an operator:

`x op= y` is shorthand for `x = x op y`

Ex: `x +=1` is like `x = x + 1` or `x++`

26

Example

```

int iVar = 15;
long lVar = 2;
float fVar = 7.6f - iVar / lVar;
double dVar = 1L / lVar + fVar / lVar;
int result = 100 * dVar;
  
```

Fix, if need be, and output `result`
The answer may surprise you!

27