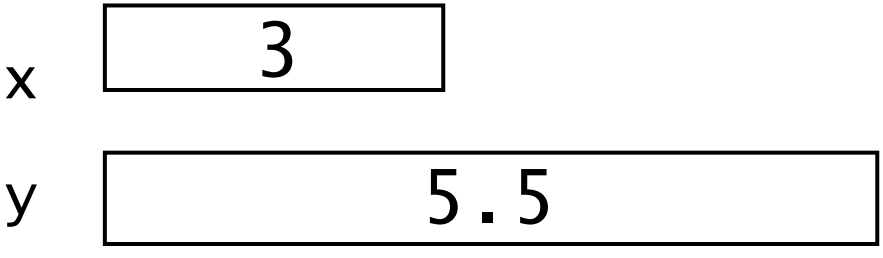
Chapter 3

Primitive Types and Casts

Some examples of primitive types

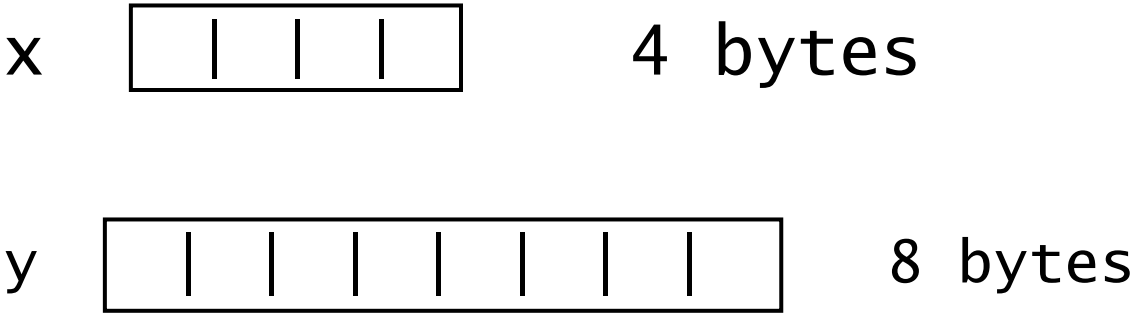
```
int x;
double y;
x = 3;
y = 5.5;
```



Bits and bytes

A `bit` is a 0 or 1 in a binary number.
A `byte` is 8 bits.

```
int x;
double y;
```



All eight primitive types

Type	Size	Range	
`byte`	one byte	-128 to 127	Arithmetic types
`short`	two bytes	-32768 to 3276	
`int`	four bytes	approximately -2 billion to 2 billion	
`long`	eight bytes	approximately -10^{19} to 10^{19}	
`float`	four bytes	approximately -10^{38} to 10^{38}, 8 digits	
`double`	eight bytes	approximately -10^{308} to 10^{308}, 16 digits	
`char`	two bytes	can hold any single character	
`boolean`	one byte	holds either `true` or `false`	

Mixed types

```
byte b = 5;
int i;
short s;
long l;
float f;
double d;

i = b;      // i is int, b is byte
s = b;      // s is short, b is byte
l = b;      // l is long, b is byte
f = b;      // f is float, b is byte
d = b;      // d is double, b is byte
```

Casts

```
i = 5;

b = i;          // illegal statement

b = (byte)i;  // legal statement

b = (byte)d; // legal statement

x = (int)"hello"; // illegal cast
```

Truncation

```
i = 257;        // i is 4 bytes
b = (byte)i;  // b is 1 byte
System.out.println(b);
```

i

00000000 00000000 00000001 00000001

b

Promotion

```
// 2 is promoted in next statement
System.out.println(2 + 3.5);
System.out.println(1.0 + 7/2);
System.out.println(1.0 + 7.0/2);
System.out.println(1.0 + (double)7/2);
```

`char` variable

```
char c;
c = 'A';
```

c

'A'

Escape sequences

`\'`	ordinary single quote
`\"`	ordinary double quote
`\\`	ordinary backslash
`\n`	newline character
`\r`	carriage return character
`\t`	tab character

```
char c;
c = '\'';
System.out.println(c);
```

displays

```
'
```

```
System.out.println
    ("I read \"War and Peace.\"");
```

displays

```
I read "War and Peace."
```

boolean variables

```
boolean boo1, bool2;

bool1 = true;
boo2 = boo1;
boo1 = "true";     // illegal
```

bool1	bool2
true	true

Relational operators

<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
==	equal (don't use =)
!=	not equal

Using relational operators

```
System.out.println(2 < 3);

boolean boo;
boo = 2 < 3;
System.out.println(boo);

int x, y;
x = 2;
y = 3;
System.out.println(x < y);
```

Illegal

```
int x;
x = 2;

System.out.println(1 < x < 4);

System.out.println(2 < = 3);
```

Equality vs assignment

```
int x = 1, y = 2, z;
boolean boo;

x == y        // testing for equality

x = y         // assignment of y to x

boo = x == y;

z = x = y;
```

Boolean operators

`&&`	AND
`||`	OR
`!`	NOT

```
boolean p = true, q = false, r;
r = p && q; // false assigned
```

`p && q` is true only if both `p` and `q` are `true`.

AND truth table

p	q	p && q
false	false	false
false	true	false
true	false	false
true	true	true

OR truth table

p	q	p || q
false	false	false
false	true	true
true	false	true
true	true	true

NOT truth table

p	!p
false	true
true	false

Fix for `1 < x < 4`

```
boo = x > 1 && x < 4;
```

```
 1 class PrimitiveTypes
 2 {
 3    public static void main(String[] args)
 4    {
 5        int x = 1, y = 2;
 6        System.out.println(x < y);        // displays true
 7        boolean boo1, boo2;
 8        boo1 = x < y;                     // boo1 assigned true
 9        boo2 = false;                     // boo2 assigned false
10        System.out.println(boo1 && boo2); // displays false
11        System.out.println(boo1 || boo2); // displays true
12        System.out.println(!boo2);        // displays true
13
14        byte b;
15        short s = 257;
16        b = (byte)s;             // truncated value assigned
17        System.out.println(b);   // displays 1
18
19        float f = 1.0f/3.0f;     // f suffix means float constant
20        double d = 1.0/3.0;      // no suffix means double constant
21
22        System.out.println(f);   // displays 0.33333334
23        System.out.println(d);   // displays 0.3333333333333333
24
25        d = 3.99999999;
26        long lg;
27        lg = (long)d;            // fractional part truncated
28        System.out.println(lg);  // displays 3
29    }
```

Precedence

```
b = x > 2 && x < 4;
```

Do `>` before `&&`.

High to low precedence

```
!, ++, --,+, -
new, (type)
*, /, %
+, -
<, > <=, >=
==, !=
&&
||
=
```

Is this legal?

```
int x;
x = 2 + 3 < 4 + 5;
```