COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

For loop pitfalls

```
for (int i= 0; i>= 10; i++)
{
// do something
}
```

Loop is never run, because the test is false

For loop pitfalls

```
for (int i= 0; i<= 10; i--)
{
// do something
}
```

Loop keeps running because i doesn’t get > 10

For loop pitfalls

```
for (i= 0; i<= 10; i++)
{
// do something
}
```

The variable i is never declared

sum the odd numbers up until 50

→ iterate over all the odd numbers from 1 to 50

for (**initializer; condition; update**)

initializer?

set a variable to 1, the first odd number

int i = 1;

condition?

loop while the variable is less than or equal to 50

i <= 50;

update?

increment i by 2 to get the next odd number

i+=2

```
for (int i = 1; i <= 50; i += 2)
        sum += i;
```

count backwards with a for loop!!

what if you want to do…

for i from 20..1?

initializer: set i to the largest number

int i = 20;

condition: loop while i is bigger than or equal to 1.

i >= 1

update: reduce i by 1

i--

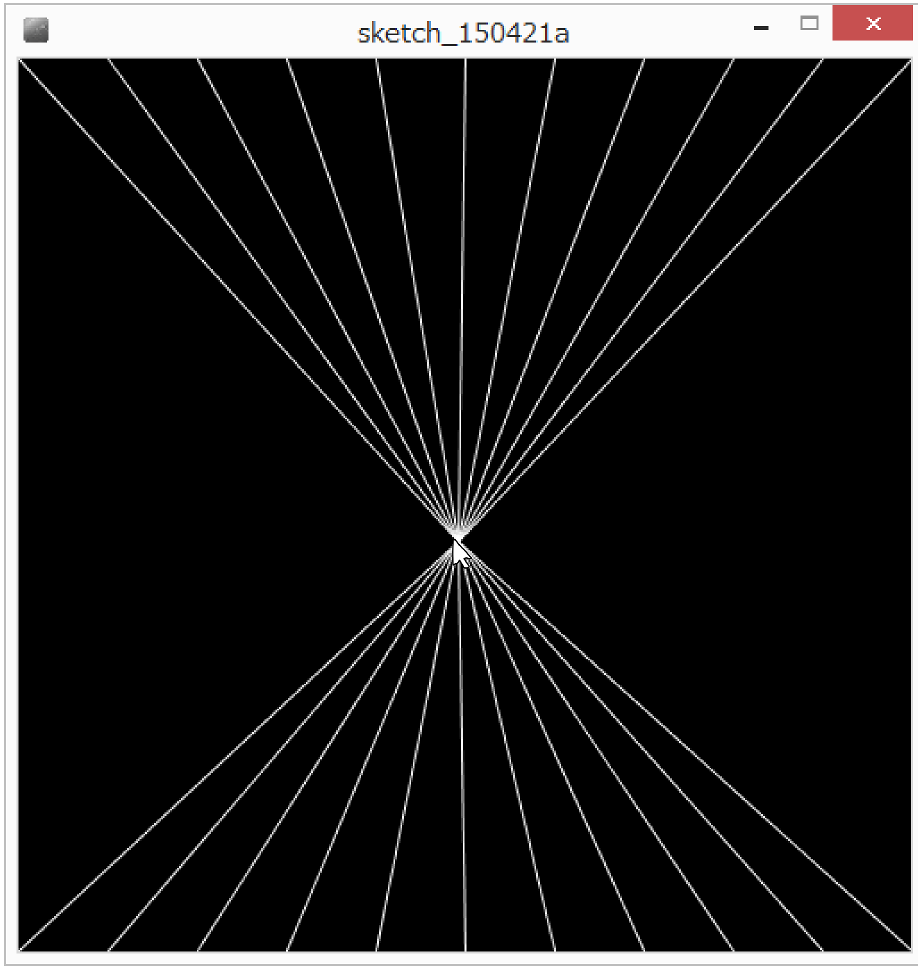
for (**initializer**; **condition**; **update**)

```
for (int i = 20; i >= 1; i--)
        pritnln(i);
```

Exercise: Use a for loop to implement the following:

Space 50 pixels apart

Make x get bigger by 50 each time in the loop



Data types and memory

bits and bytes and nibbles… (don’t memorize)

A computer stores everything as switches (**bits**)

represent 0 (off) and 1 (on).

A group of 8 **bits** (switches) makes a **byte**

00110110 ← one byte of data

A group of 4 **bits** (half a bite) make a **nibble**

(I kid you not!) 1101 is a **nibble** of data

1024 **bytes** (2^{10}) make a **kilobyte**

1024 **kilobytes** (2^{20} bytes) make a **megabyte**

1024 **megabytes** (2^{30} bytes) make a **gigabyte**

1024 **gigabytes** (2^{40} bytes) make a **terabyte**

A **terabyte** has 1,099,511,627,776 bytes or

8,796,093,022,208 switches

using a 7cm standard light switch… 615 million KM

4 times the distance to the sun!!!!!!!!!!!!!

(aside: new standard units are moving to even powers of ten where 1 terabte = 1,000,000,000,000 bytes)

Counting with bits!!!!! (not covered in class, not testable)

0 -> 0

1 -> 1

How to represent “2”?

we need another bit. Put it in front

Start over..

00 // right column is 2^0 place

01

10 -> 2 // left column is 2^1 place

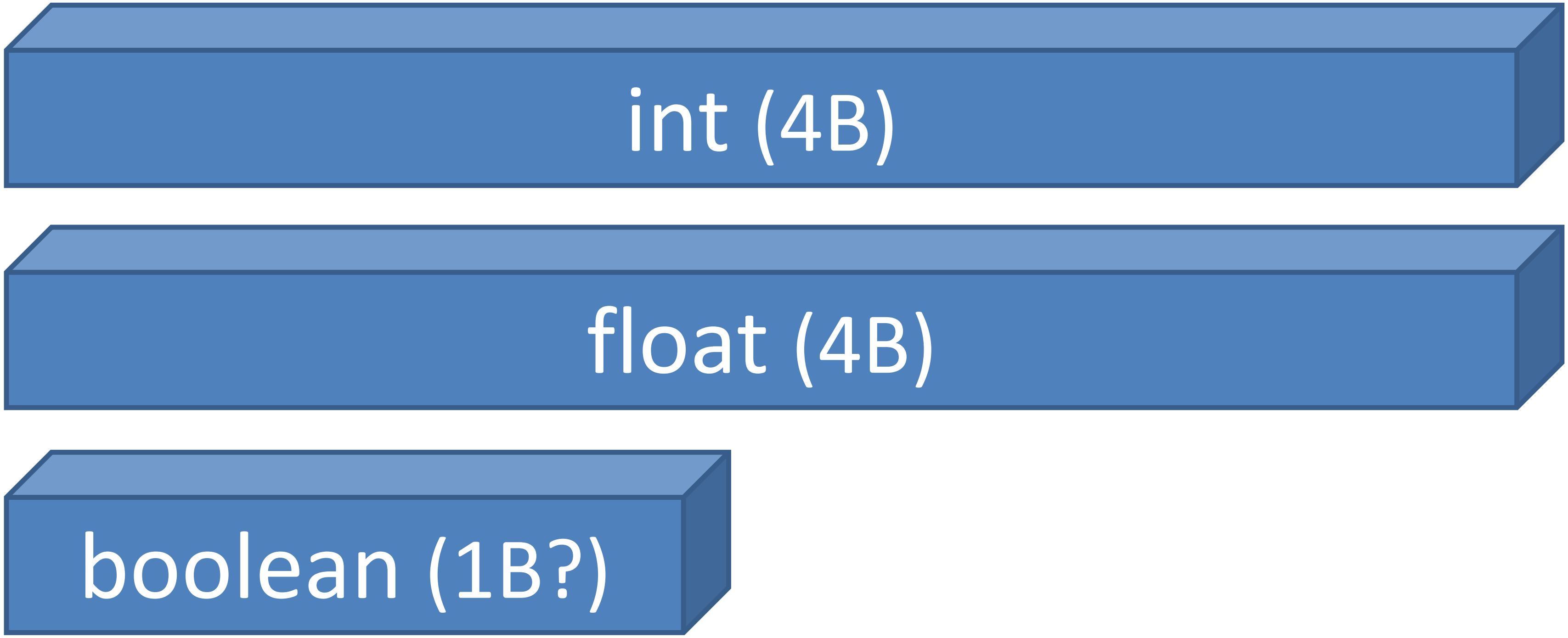
11 -> 3

100 -> 4 // left column is 2^2 place

What is 6?

110

data types so far



int (4B)

float (4B)

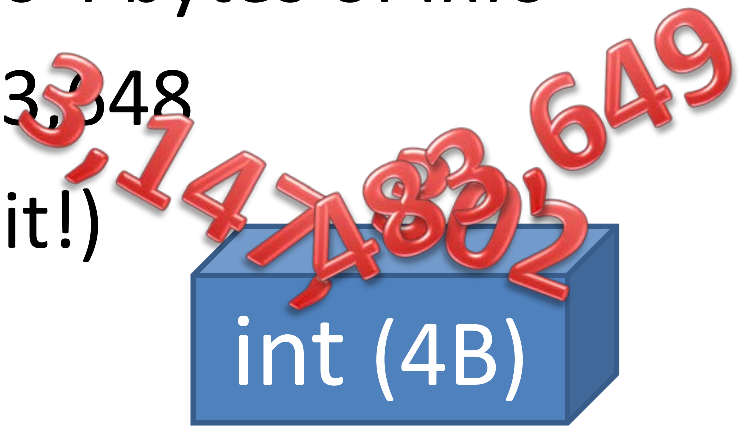
boolean (1B?)

integer overflows

"int" type in Processing stores 4 bytes of info

smallest number is -2,147,483,648

largest is 2,147,483,647 (try it!)



what happens when you go past these numbers accidentally?

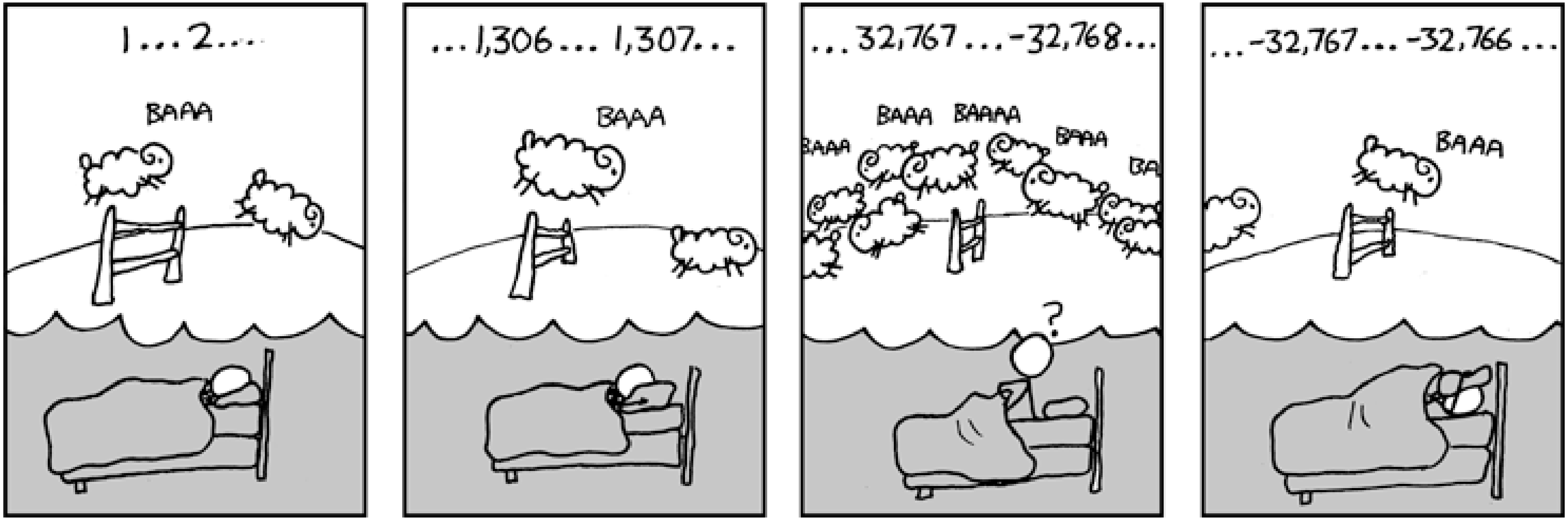
variable overflow

note: when you go over the specified maximum value of an integer variable, the value *wraps around* to the smallest value.

when you go below the specified minimum, it wraps in the other direction.

quick comic

a data type that we do not use in this course, called a **short** (a short integer), is 2 bytes (int is 4) and can hold the range -32,768…32,767



XKCD.com

exercise: infinite loop?

```
for (int i = 1; i > 0; i++) // infinite loop?
{
  ; // do nothing
}
background(255);
line(0,0,mouseX,mouseY);
```

is this an infinite loop?

lets test

!(infinite loop):

```
for (int i = 1; i > 0; i++) // infinite loop?
{
  ; // do nothing
}
```

this is not an infinite loop because i cannot get infinitely large. It is limited by the memory of the int data type. Once it hits the largest limit, adding one will make it “roll over” to the smallest value, making it less than 0.

two ways to avoid overflow:

1) use a different data type

2) be clever with your calculations to avoid large numbers

Primitive data types: integers

type	size	minimum	maximum

All used like int

Integer math

Primitive data types: floating point

float – 4 bytes

double – 8 bytes

More memory is more precision, not more range

e.g.,

float - 0.6666667

double - 0.6666666666666666

primitives

boolean – true, false

char – store one character (later!)

Casting

conversion between types (casting)

we have int, long, float, double, etc., how do they relate? how do we go between them?

Try …

```
int i = 1234;
byte b = i;
```

What will happen?

1234 cannot fit into byte?

What about..

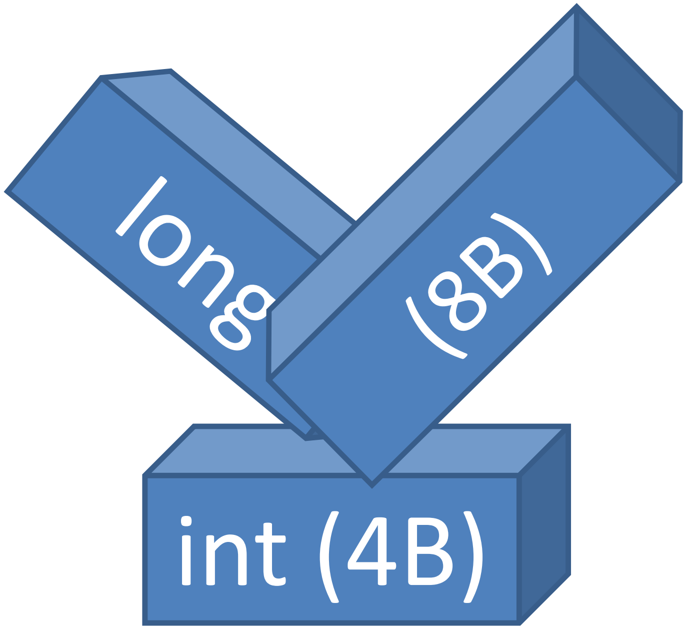
```
long l = 1234;
int i = l;
```

It just doesn’t fit!!

Processing knows that the int only has half the memory. It doesn’t even try

It’s dangerous!

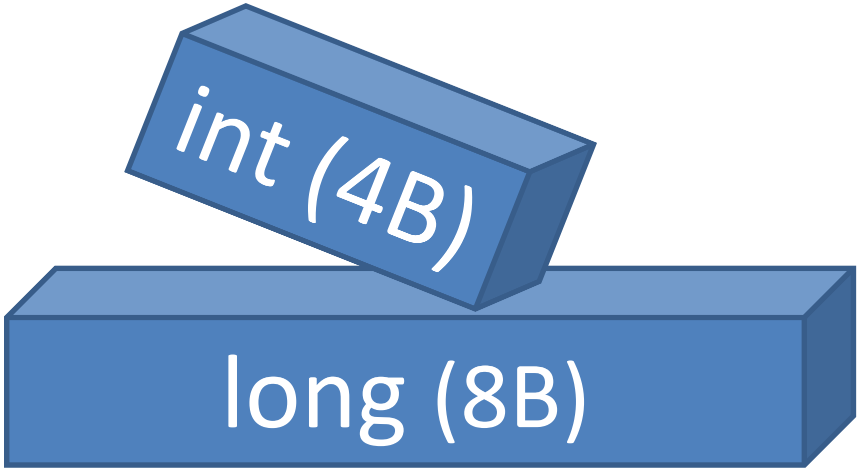
Narrowing conversion



Other direction

```
int small;
long large;
small = 15;
large = small; // Convert an int to a long!!
```

Widening conversion



casts

widening conversions automatically convert (cast) the data types – this is called an **implicit cast**

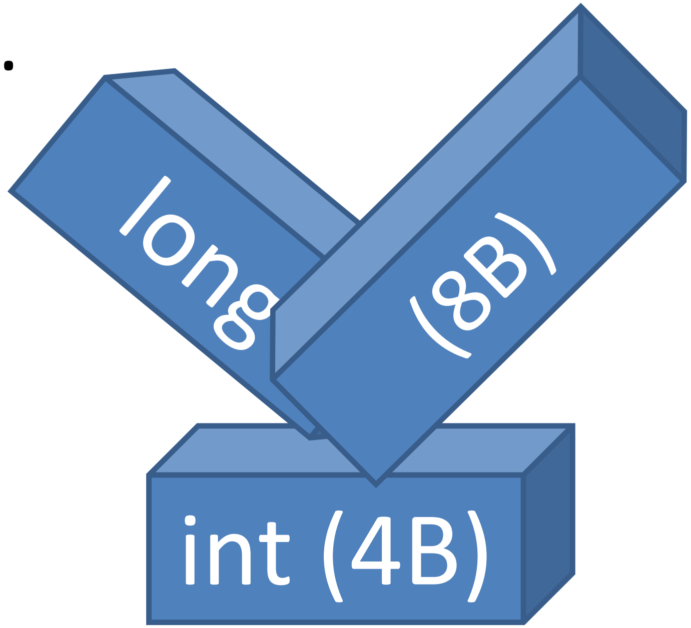
narrowing can result in the loss of data, so Processing requires that you **explicitly cast** the data to the new type

Example:

```
long large = 200;
int small = large;
```

Error: cannot convert

Processing is saying that you may lose data, so it doesn't want to do it.



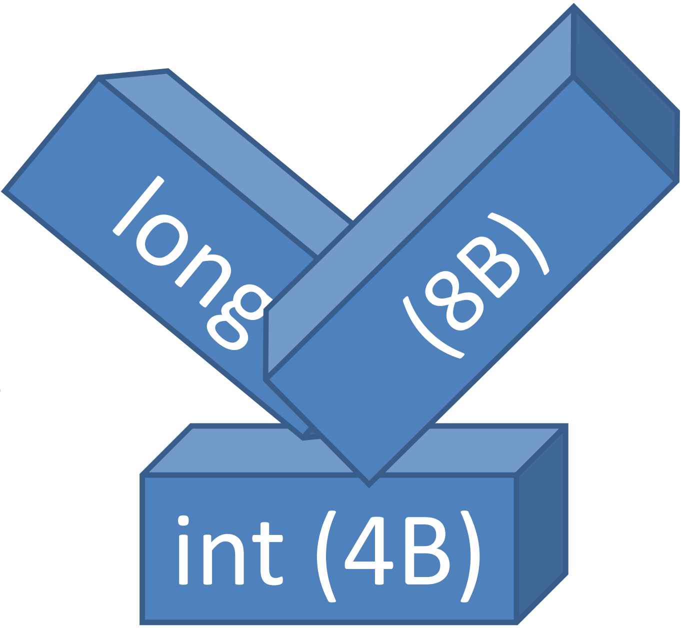
Explicit cast:

put (type) in front of a variable or value

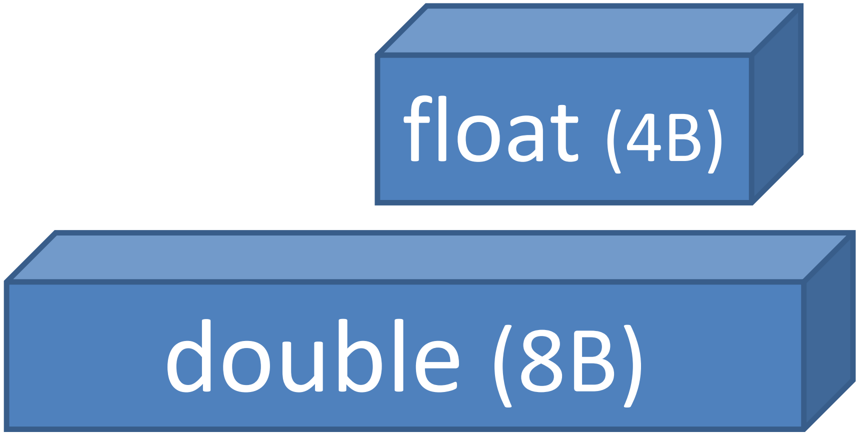
```
long large = 200;
int small = (int)large;
```

This tells Processing to convert

Just do it! I know what I’m doing!



floating point..



float (4B)

double (8B)

float is 4 bytes

double is 8 bytes

widening conversion is an **implicit cast**

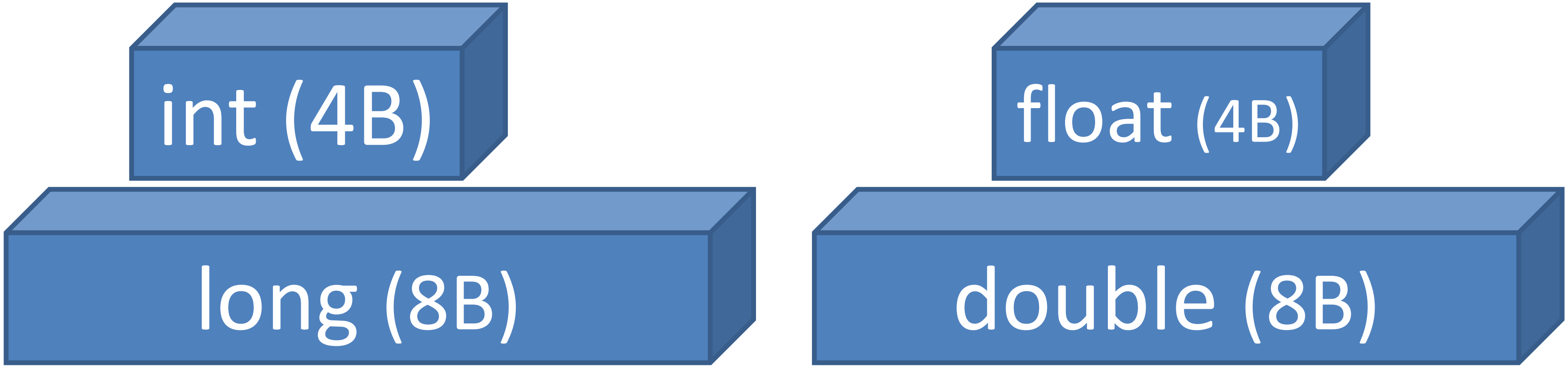
narrowing conversion requires an **explicit cast:**

```
float f = 1.234;
double d = f;
float floatVariable = (float)doubleVariable;
```

converting between integer and floating point types

floating point -> integer, data is lost so an **explicit cast** is needed to make Java happy.

integer -> floating point, **implicit cast** because floating point is more capable and can accommodate the integer.



int (4B)

long (8B)

float (4B)

double (8B)

Float -> integer

When explicitly cast to an integer, a floating point number gets **truncated.** The decimal portion is lost.

```
float f = 123.456;
int i = (int)f;
println(i);
```