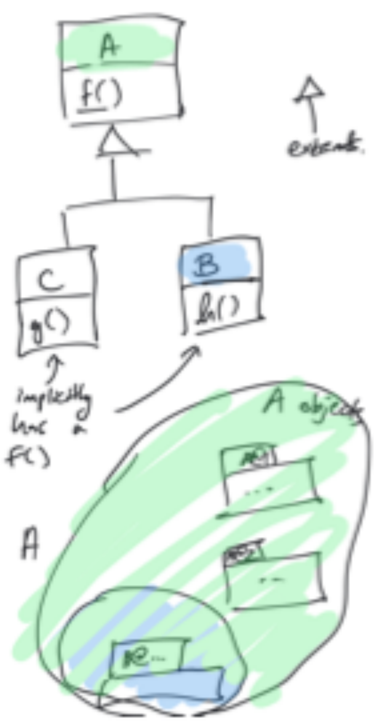
Lecture 6: Subtyping

- Defn of subtype, is-a
- primitive subtypes
- Casts & instanceof
- Object, Comparable, Comparator

Announcements

- ✓ P1 due today, see submission instructions, P2 out soon
- ✓ No lecture/discussion 7/4 or 7/5
- ✓ Guest lectures next week (Prof. G. traveling)
- ✓ Feedback form
- ✓ Anyone w/o partner?

```
class A {.. f(); }
class C extends A {g();}
 class B extends A {h();}
```

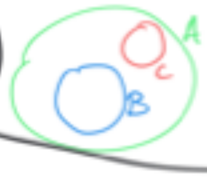


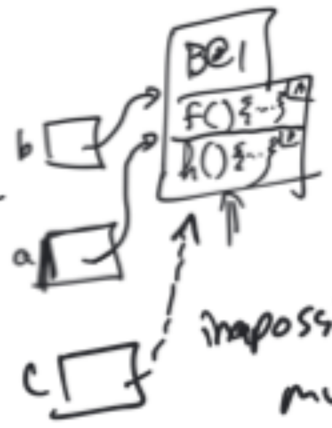
every C <u>is an</u> A
every B <u>is an</u> A.

anything I can do with an A object, I can do with a B.

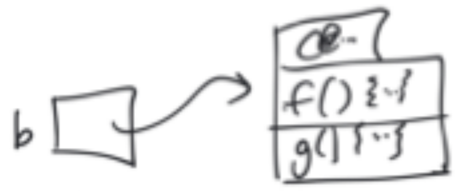
C and B are <u>subtypes</u> of A
we write $C<A$ or $C<:A$ or $C\subseteq A$

```
B b = newB();
A a;
a = b; // ok. (no change to object!)
a.f();
 a = new C();
~~b = a;~~ // not OK.
b.h()
a.h(); // not OK.
```





impossible, C variables must point to C's.



```
// I know (I'm clever)
// that a actually
// points to a B.
```

$\underbrace{b = (B)\,a}_{\text{type B.}};$ ⇐ · <u>check</u> if object is really of type B (run-time)

<u>Cast</u>: converts a reference from a supertype to a subtype.
<u>Can fail!</u>

Don't cast!

```
((B)a).h();
((C)a).g(); // crash
C c = (C)a; // crash
```

$\underbrace{\text{o instanceof B}}$, boolean expression (true or false)

```
if (o instanceof B){
    B b = (B)o;
      ⋮
}
```

Never use!

"Subtyping" for primitive types

char < int < float < double

~7 digits ~16 digits of precision

$$\underset{\text{float}}{\text{float } f} = \underset{\text{int}}{1}; \quad \text{//ok.}$$

$$\underset{\text{int}}{\text{int } i} = \underset{\substack{\text{double} \\ \text{(Java's convention)}}}{1.0}; \quad \text{// not OK} \quad \text{int} \not> \text{float.}$$

int i = 3.2; // doesn't compile

int i = (int) 3.2; // OK, bad idea.
//doesn't give error, approximates.

characters are integers (internally)