CS 2110: Inheritance

class C ~~extends~~ implements I creates subtype relationship

C <: I

```
I x = ...
C y = (C) x; // run-time cost: may fail
```

Instanceof: Is dynamic type a subtype of some type

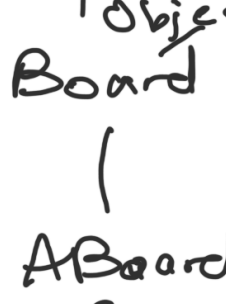
```
Board b = ...
if (b instanceof ABoard) {
    ABoard ab = (ABoard) b;
}
```

Java 7:

```
if (b instanceof ABoard ab) {
    // use ab
}
```

new variable

b instanceof Object → A: true, B: false



Interface subtyping

```
interface Solvable extends Board {
    Direction[] solve();
}
```

"Solvable <: Board

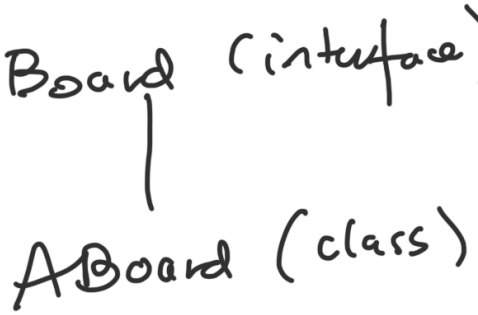
— has all Board methods, plus solve();

interfaces support loose coupling
(but: weak abstraction barrier)
↳ b/c casts can recover dynamic type

OO = encapsulation + subtyping + inheritance

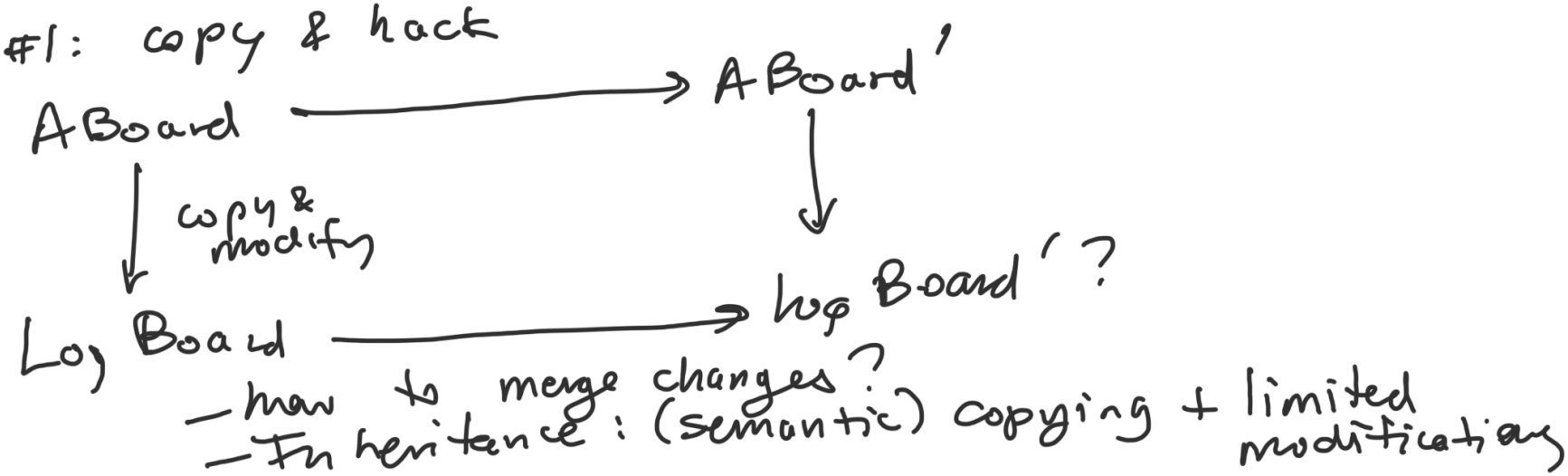
code reuse & extensibility

Ex/

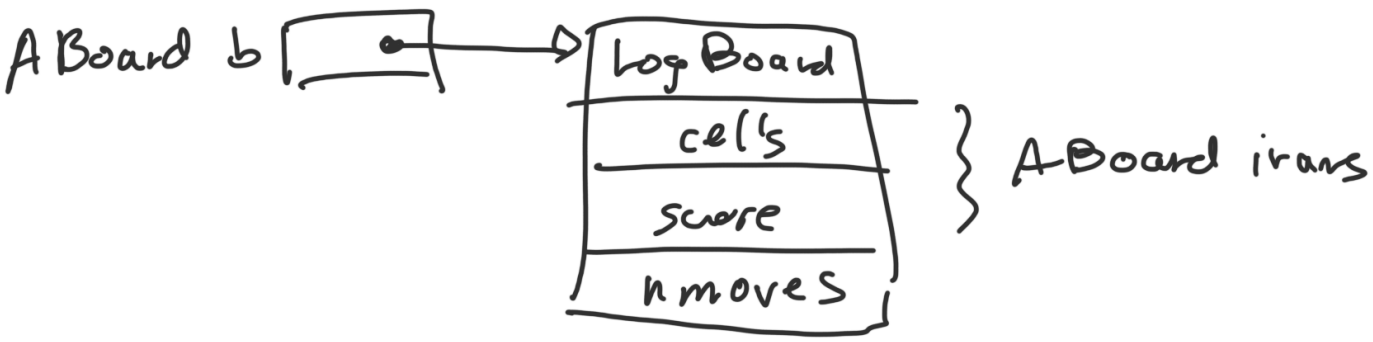


Goal: reuse ABoard code in new class.
new impl. : LogBoard that logs all moves.

Idea #1: copy & hack



- how to merge changes?
- Inheritance: (semantic) copying + limited modifications



Method code is chosen by run-time class
= dynamic dispatch

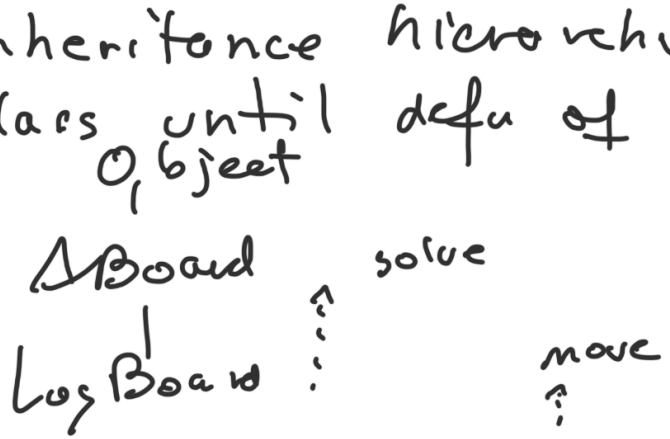
```
class ABoard {
    :
    void solve() {
        move(Direction.UP);
    }
}
```

```
ABoard b
    = new LogBoard();
b.solve();
```

this.move
↳ LogBoard is receiver ⇒
still LogBoard.move()

Two views (same answer)

1) search upward in inheritance hierarchy from runtime class until defn of method.



2) methods of superclass copied to subclass, unless overridden.

Class methods & Inheritance

- static methods can't be overridden, only shadowed.
- no receiver ⇒ method code chosen using static type
- weakens extensibility: no late binding (overriding) of methods
- constructors, static methods, super calls, "final" methods.