Lecture 5: Class & interface hierarchy

- Classes v. interfaces
- Abstract classes
- Inheritance, upside-down, inside out
- Quiz

interfaces

- is a specification
  - list of methods : types
  - Javadoc comments (preconditions, postconditions, conceptual idea, efficiency constraints)

not an implementation.

- can extend other interfaces. (many)
  - can add more methods
  - can also add restrictions on existing methods
  - can't remove methods or restrictions.
- can only contain public methods (no fields, nothing private)

(actually: can contain "final" fields: constants)

```
iface I1 {
  /** no reqs */
  int f();
}

iface I2 extends I1 {
  /** returns a positive int */
  int f();
}
```

classes

- a specification + an implementation.
  - list of fields
  - class invariants
  - code to implement methods.

- can extend (one) other class.

```
class TimeImpl {...}

class SecondsTimeImpl
    extends TimeImpl {
  ...
}
```

- inherits all the details of parent class

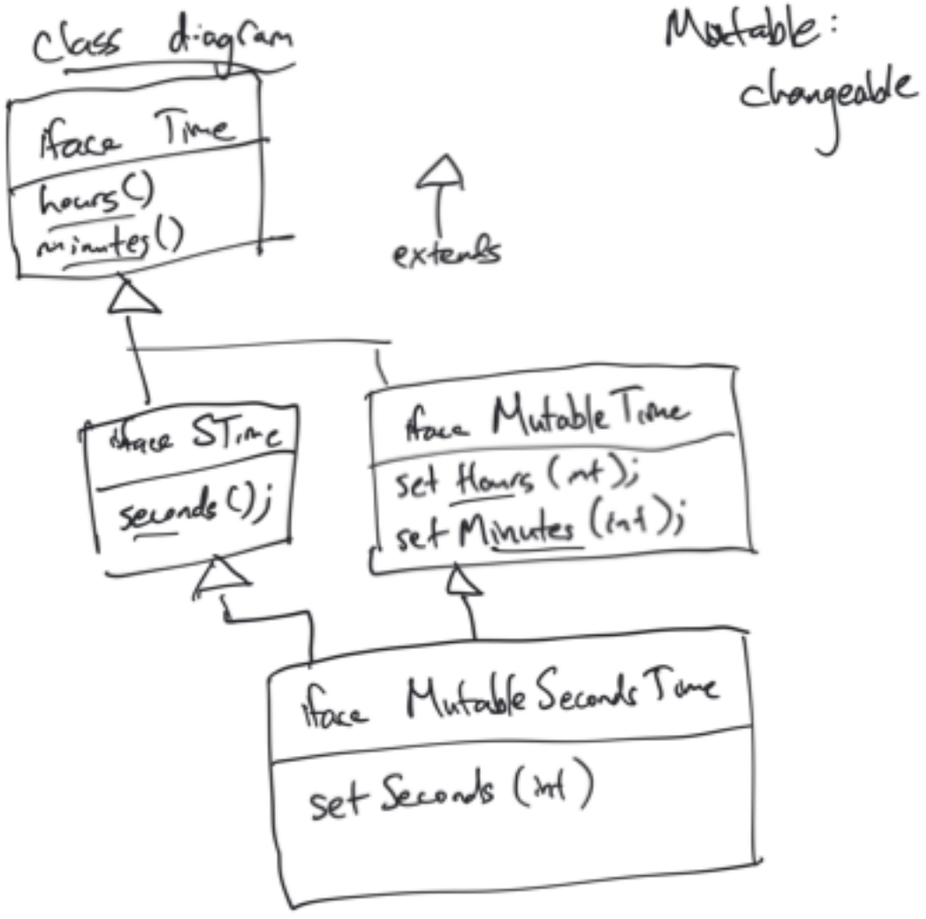
- "subclass" extends "superclass".

interface Time { ... }

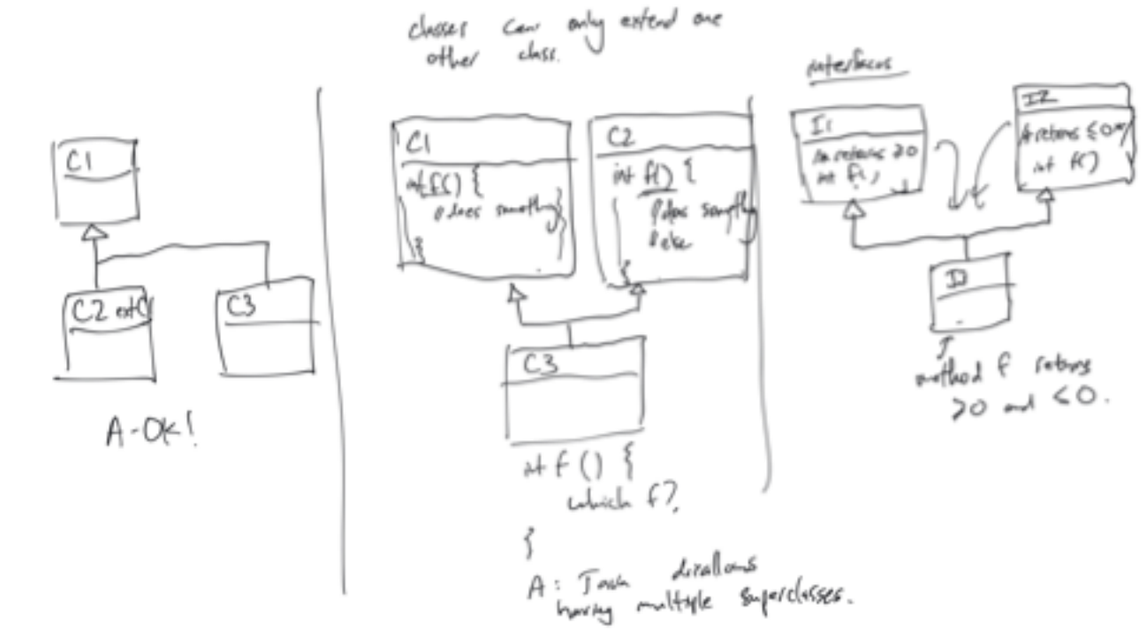
interface SecondsTime ext. Time { ... }

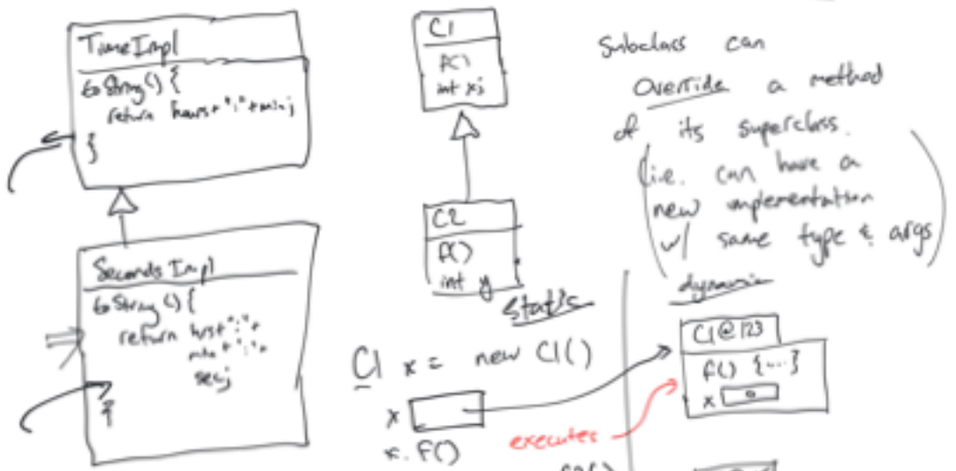
every SecondsTime is-a Time.

```
Time t;
SecondsTime st;

t = st; // makes sense,
        because every ST is-a T.

t.minutes();

st = t; // all I know is
        that t is a Time.
        - minutes, seconds.
st.seconds(); error!
```




```
interface MutableST
    extends ST, MT {
    :
}
```





keyword
super: means use superclass implementation.

```
class Sub ext Superclass {
  int f() {
    super.f();
  }
  Sub(int x) {
    super(...);
    this.x = x;
  }
```