Lecture 7 - Sep 24

OOP Review

Caller vs. Callee

Tracing Chain of Method Calls via a Stack

Catch-or-Specify Requirement

Announcements/Reminders

- Today's class: notes template posted
- Priorities:
  - + Review Lab0
  - + Complete Lab1; Due: Next Tuesday (Sep 30)
- Today's class:
  - + We'll finish at about 12:30.
  - + 20 minutes to be covered in Section G's recording

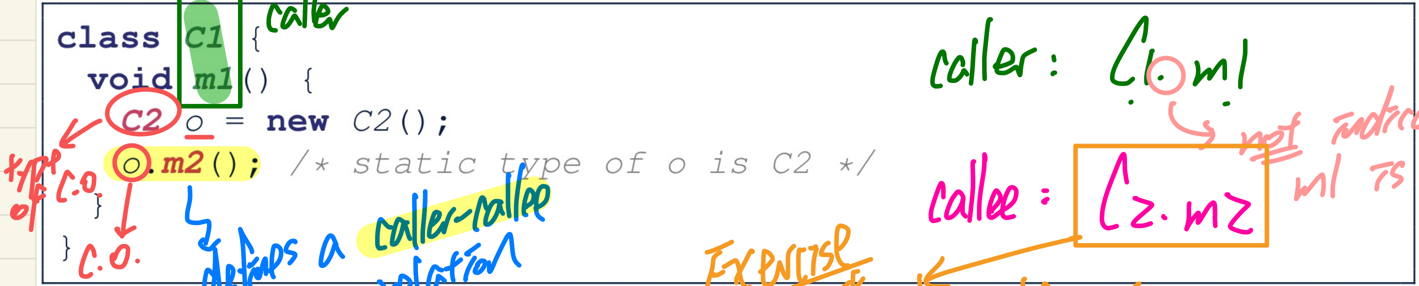
a method that calls another method

Caller vs. Callee

callee

method that's called by ~~another method~~ caller

- **caller** is the **client** using the service provided by another method.
- **callee** is the **supplier** providing the service to another method.



```
class C1 {
  void m1() {
    C2 o = new C2();
    o.m2(); /* static type of o is C2 */
  }
}
```

caller

type of C.O.

C.O.

defines a caller-callee relation

caller : C1.m1

not indicating that m1 is static

callee : C2.m2

Exercise

Make it a caller in another context.

Q: Can a method be a **caller** and a **callee** simultaneously?

YES. Make C1.m1 a callee in another context

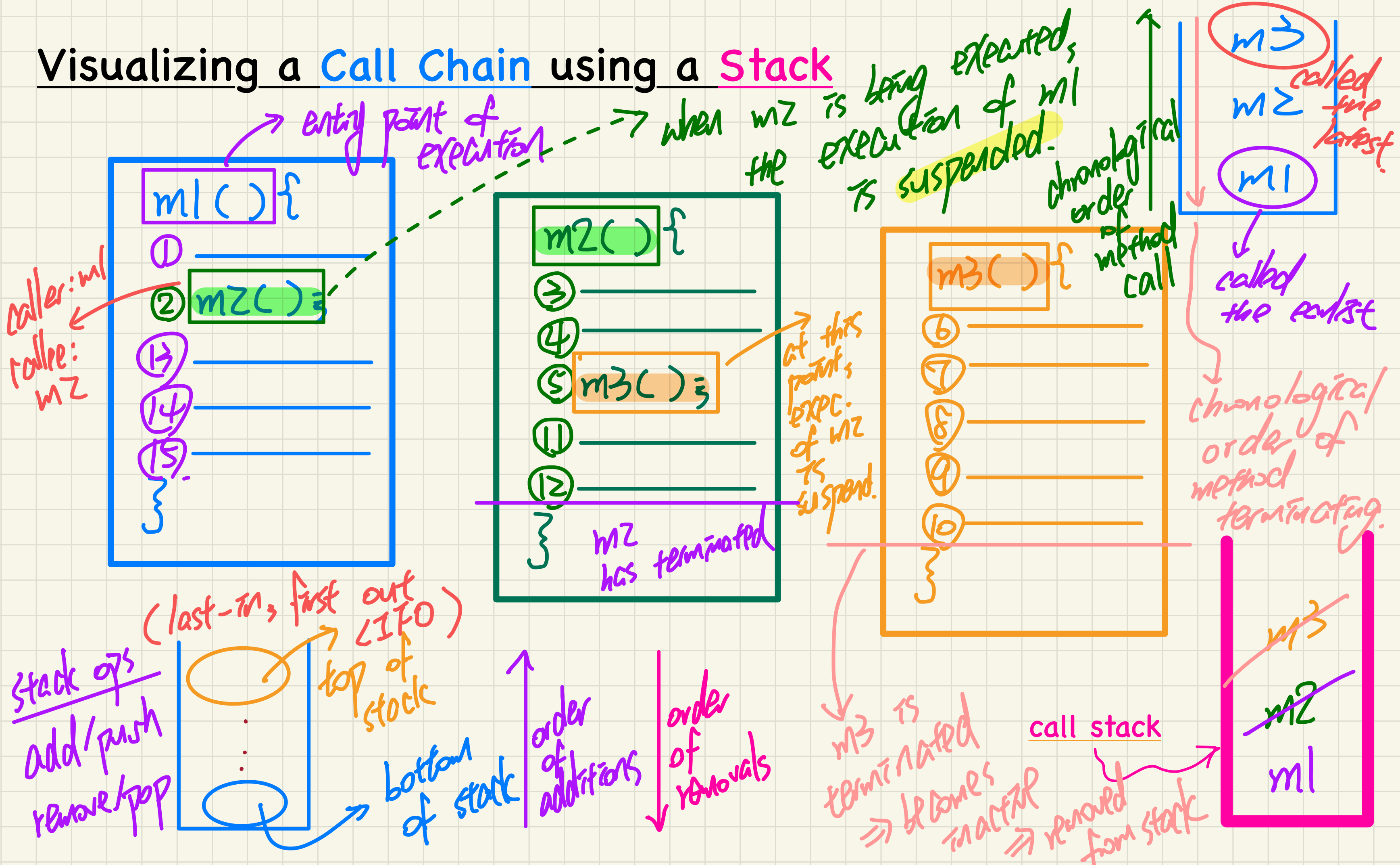
(Alt 1) class C1 {
  void m3() { this.m1(); }
}

C1.m1 is a callee

(Alt 2) class C3 {
  void m() { C1 o = new C1(); o.m1(); }
}

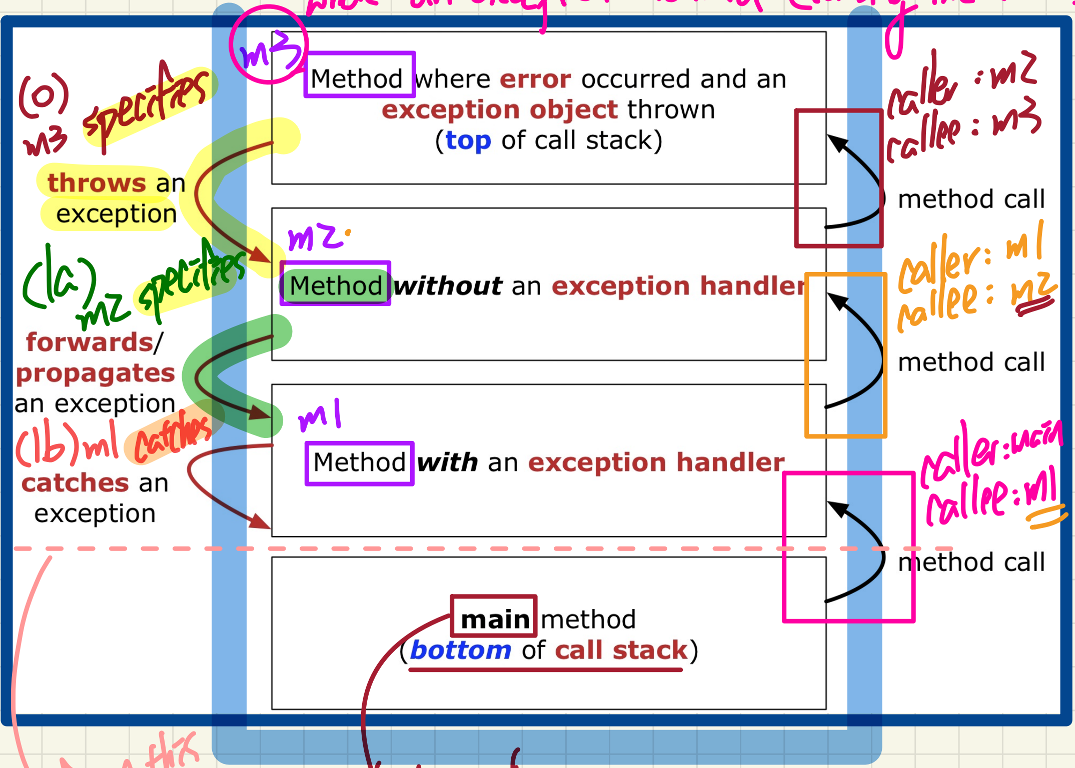
C1.m1 is a callee

Visualizing a Call Chain using a Stack



What to Do When an Exception is Thrown: Call Stack

where an exception occurred (causing the call stack to stop growing)



(0) m3 specifies

throws an exception

(1a) m2 specifies

forwards/ **propagates** an exception

(1b) m1 catches

catches an exception

m3

Method where **error** occurred and an **exception object** thrown (**top** of call stack)

caller: m2
callee: m3

method call

m2:

Method ***without*** an **exception handler**

caller: m1
callee: m2

method call

m1

Method ***with*** an **exception handler**

caller: main
callee: m1

method call

main method (***bottom*** of **call stack**)

entry point of exec.

starting from this point, all callers are not subject to the C-or-S req.

Catch-or-Specify Req.

(0) Where an exception is thrown, just specify and propagate to the next caller.

(1a) a caller may specify an exception (prog. from its callee)

(1b) a caller may catch an exception (prog. from its callee)

Catch-or-Specify Requirement

The "Catch" Solution: A `try` statement that ***catches*** and ***handles*** the ***exception*** (**without** propagating that exception to the method's *caller*).

```
main(...) {
  Circle c = new Circle();
  try {
    c.setRadius(-10);
  }
  catch (NegativeRaidusException e) {
    ...
  }
}
```

≈ parameter decl

store the address of the caught exception

exception to be caught

```
try {
catch (...) {...}
catch (...) {...}
```

handling the caught excep.

The "Specify" Solution: A method that specifies as part of its *header* that it may (or may not) ***throw*** the ***exception*** (which will be thrown to the method's ***caller*** for handling).

```
class Bank {
  Account[] accounts; /* attribute */
  void withdraw (double amount)
    throws InvalidTransactionException {
    ...
    accounts[i].withdraw(amount);
    ...
  }
}
```

part of the method header