Lecture 7 - Sep. 26

Review of OOP, Exceptions

Tracing Method Call Chain via Call Stack
Chronological Order of Method Calls
How Exception Disrupts Execution Flow
Catch-or-Specify Requirement
Example: To Handle or Not to Handle (V1)

Announcements/Reminders

- Lab1 released
- In-Lab demo: Incremental Development for Lab1
- Mockup Programming Test tomorrow (5pm or 6pm)
- Guides for WrittenTest1 and ProgTest1 released
- WrittenTest1 review (Zoom) on Monday, time TBA

Caller vs. Callee

- **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 */
  }
}
```

calling context (caller)

C1.m1 – being called (callee)

(2) Exercise: Make C2.m2 a caller.

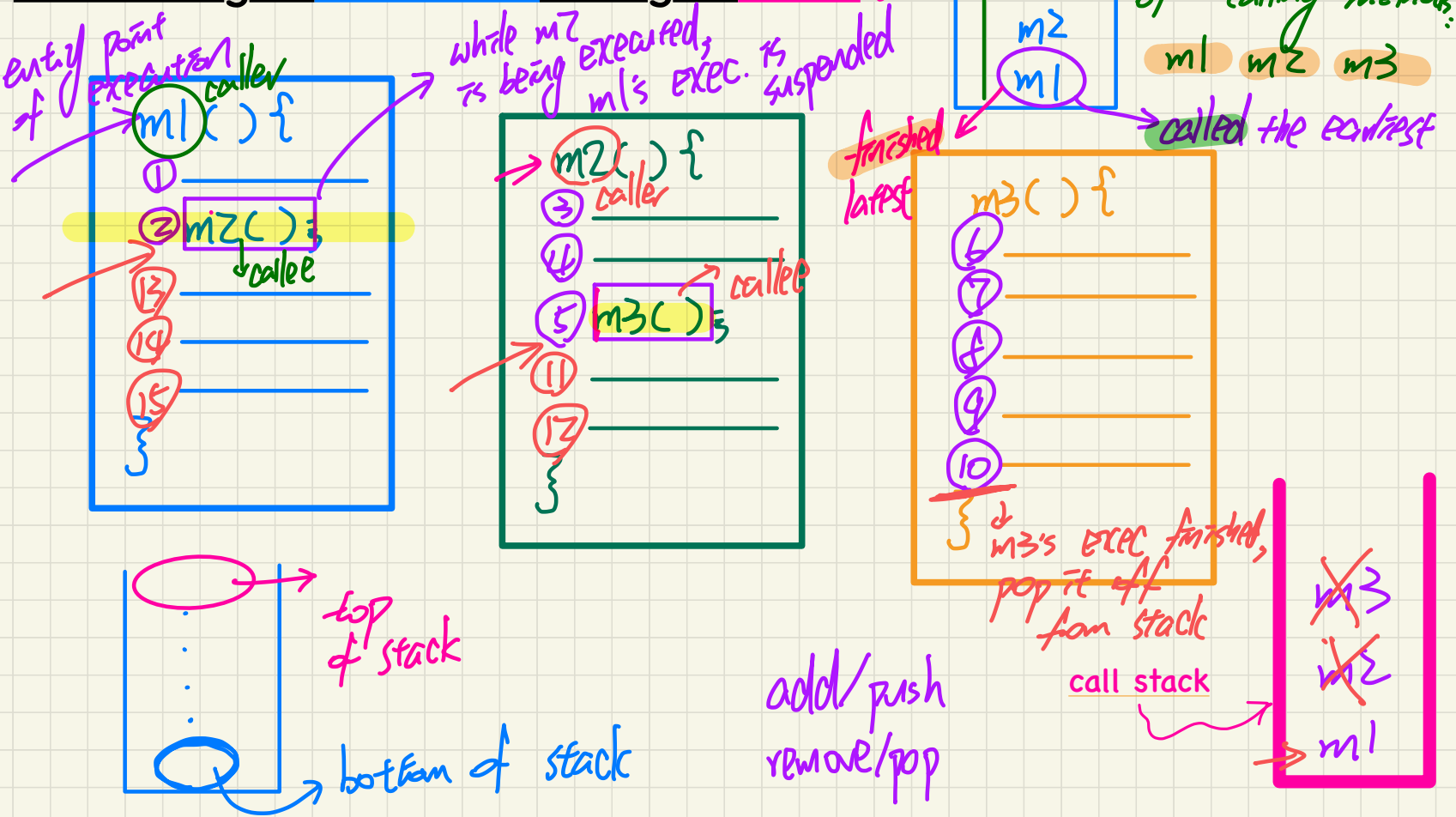
Q: Can a method be a caller and a callee simultaneously?

(1) Make C1.m1 a callee

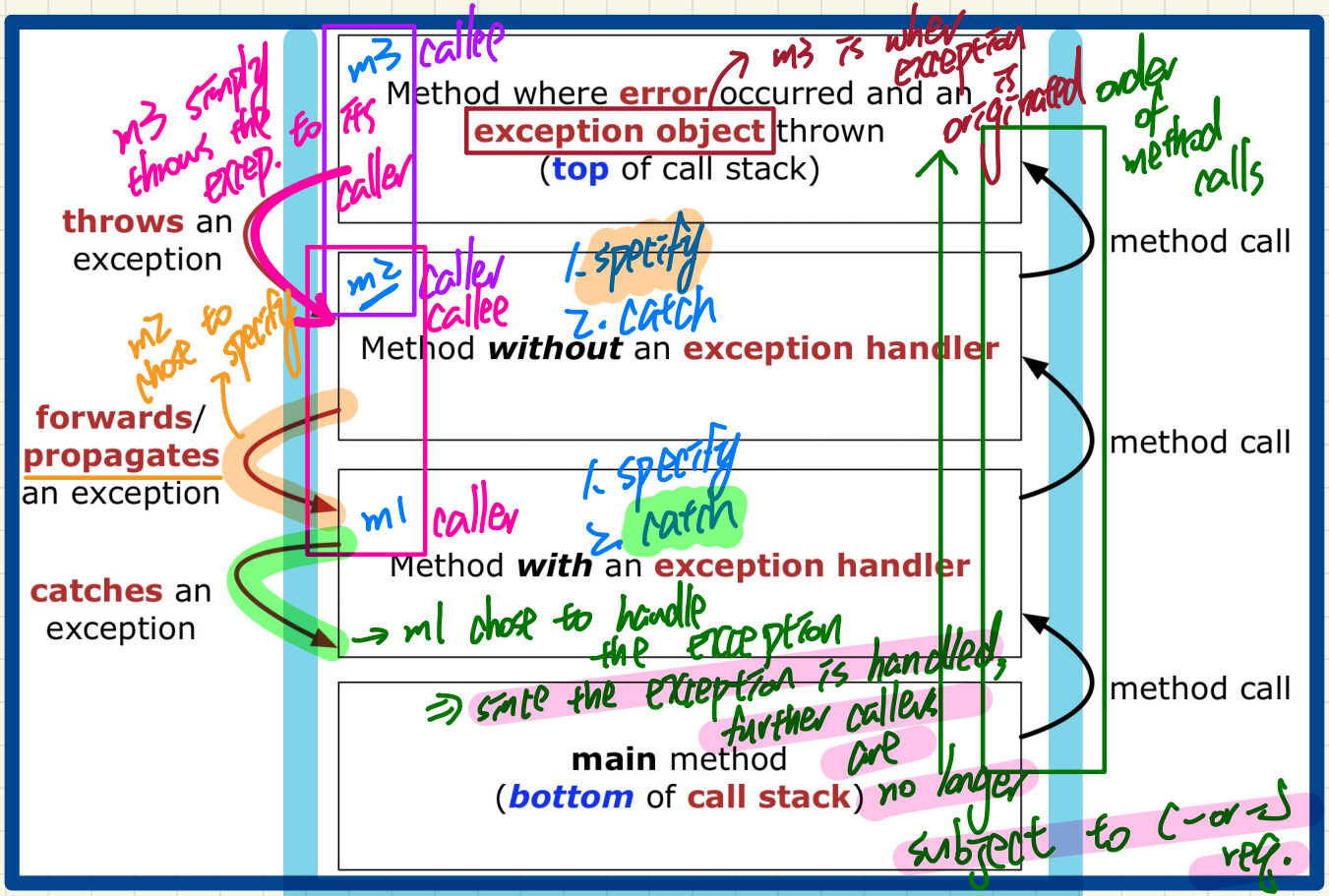
(1.a) class C1 { void m2() { this.m1(); } } — m2: caller ✓, m1: callee

(1.b) class C3 { void m4() { C1 o = new C1(); o.m1(); } } — m4: caller, m1: callee

Visualizing a Call Chain using a Stack

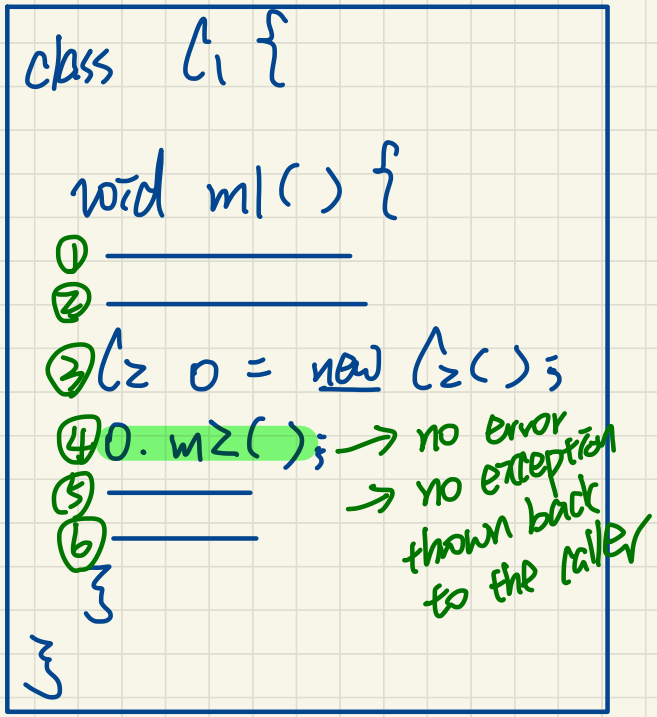


What to Do When an Exception is Thrown: Call Stack

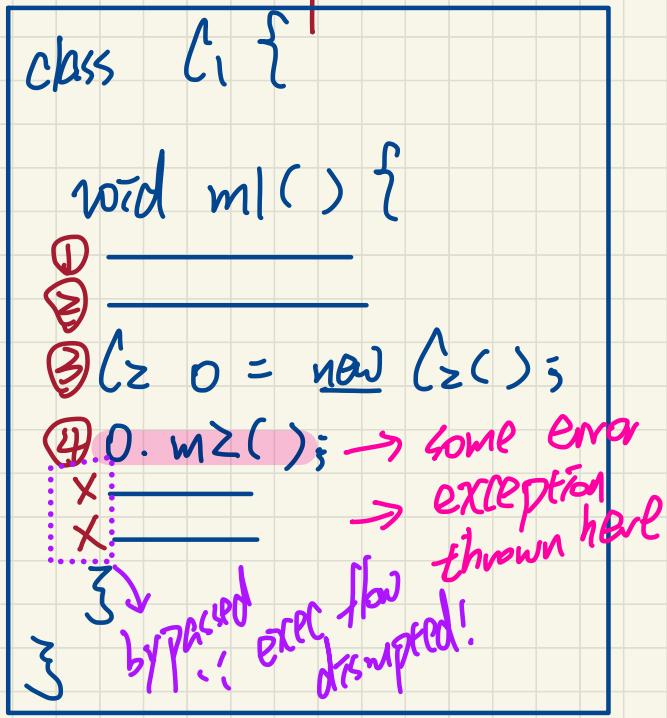


Exceptions : When occurred, Normal Flow of Exec. Disrupted

Normal



Abnormal/Exception



Example: To Handle or **Not** To Handle?

```
class A {
  ma(int i) {
    if(i < 0) { /* Error */ }
    else { /* Do something. */ }
  } }
```

```
class B {
  mb(int i) {
    A oa = new A();
    oa.ma(i); /* Error occurs if i < 0 */
  } }
```

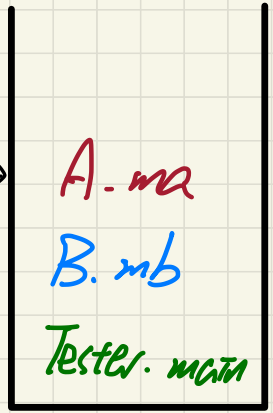
Version 1:
Handle it in `B.mb`

Version 2:
Pass it from `B.mb` and handle it in `Tester.main`

Version 3:
Pass it from `B.mb`, then from `Tester.main`, then throw it to the console.

```
class Tester {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int i = input.nextInt();
    B ob = new B();
    ob.mb(i); /* Where can the error be handled? */
  } }
```

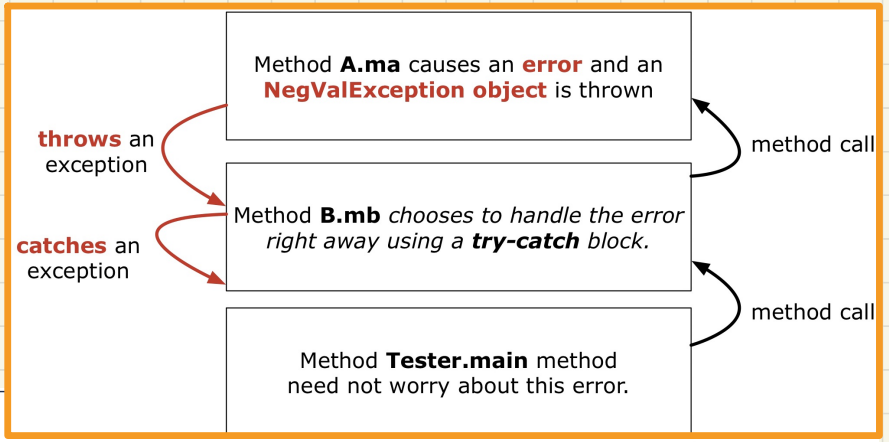
call stack




```
class NegValException extends Exception {
  NegValException(String s) { super(s); }
}
```

Version 1:

Handle the Exception in B.mb




```
class A {
  ma(int i) throws NegValException {
    if(i < 0) { throw new NegValException("Error."); }
    else { /* Do something. */ }
  } }
```

```
class B {
  mb(int i) {
    A oa = new A();
    try { oa.ma(i); }
    catch(NegValException nve) { /* Do something. */ }
  } }
```

```
class Tester {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int i = input.nextInt();
    B ob = new B();
    ob.mb(i); /* Error, if any, would have been handled in B.mb.
  } }
```

Q1. In B.mb, is calling oa.ma subject to **catch-or-specify req**?

Q2. In Tester.main, is calling B.mb subject to **catch-or-specify req**?

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) {
    ...
  }
}
```

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);
    ...
  }
}
```