Lecture 7 - May 27

Exceptions

Tracing Chain of Method Calls via a Stack
Catch-or-Specify Requirement
To Handle or Not to Handle: Version 1

Announcements/Reminders

- Today's class: notes template posted
- ProgTest1 next Friday (June 6) during enrolled session
  - Guide (policies & requirements) to be posted
  - PracticeTest1 to be posted
- Priorities:
  - Lab1
  - Review slides on Classes and Objects

method that calls another method caller

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

callee

caller : C1.m1 → does not imply that m1 is static!

callee : C2. m2

↳ Exercise: make C2.m2 a caller.

YES

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

(Alt I) Make C1.m1 also a callee.

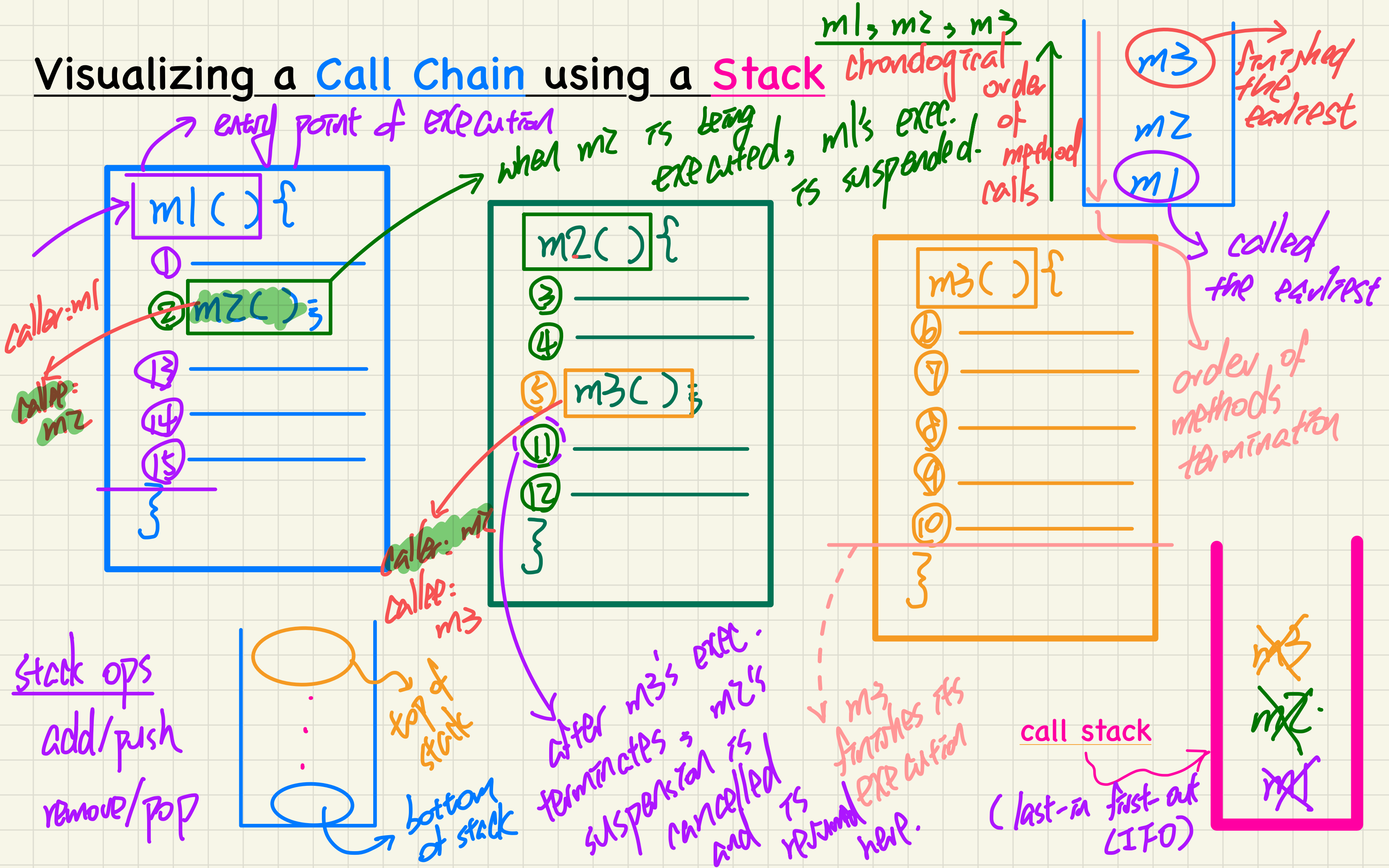
```
class C1 {
  ...
  void m2( ) { this.m1( ); }
}
```

C1.m1 a callee

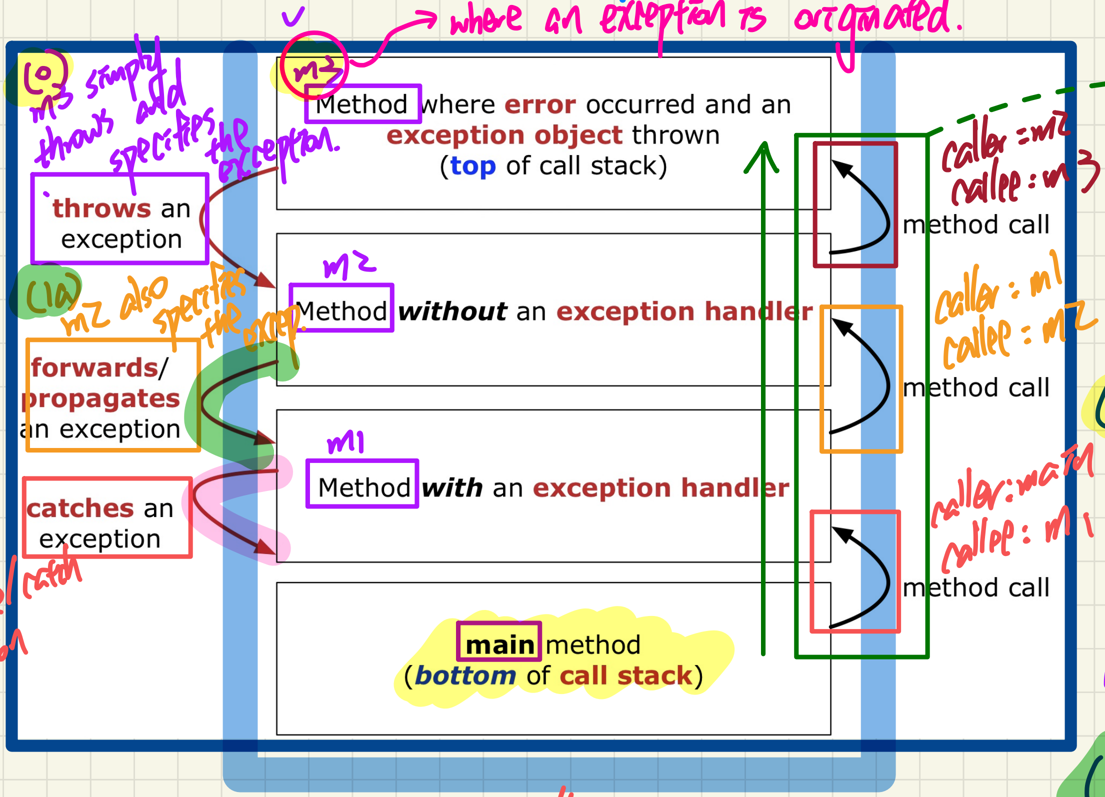
```
class C3 {
  void m( ) { C1 obj = new C1( ); obj.m1( ); }
}
```

C1.m1 a callee

Visualizing a Call Chain using a Stack



What to Do When an Exception is Thrown: Call Stack



where an exception is originated.
m3
Method where error occurred and an exception object thrown (top of call stack)
m2
Method without an exception handler
m1
Method with an exception handler
main method (bottom of call stack)
(0) m3 simply throws and specifies the exception.
throws an exception
(1a) m2 also specifies the excep.
forwards/ propagates an exception
(1b) m3 chooses to handle/catch exception
catches an exception
method call
method call
method call
caller : m2
callee : m3
caller : m1
callee : m2
caller : main
callee : m1
order of method calls

handle error

propagate error

Catch-or-Specify Principle

(0) Where an exception is thrown, just specify it. (this will force the caller to catch or specify it).

(1a) a caller may specify the exception.

(1b) a caller may catch/handle the exception.

↳ beyond this point, no further callers subject to C-O-S. req.

Example: To Handle or Not To Handle?

context	caller	callee

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

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

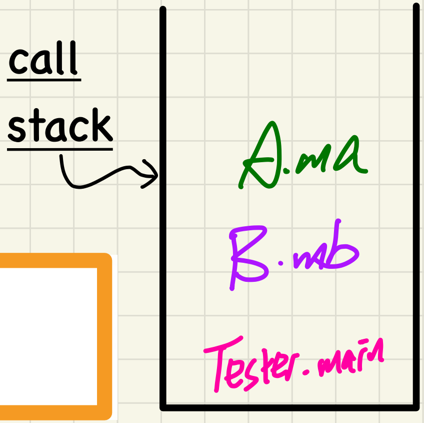
A.ma callee B.mb: caller

```java
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? */
  } }
```

callee: B.mb caller: Tester.main

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.

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


call stack
A.ma
B.mb
Tester.main