Lecture 8 - Sep 30

Exceptions

To Handle or Not to Handle: Versions 2, 3
Generalizing Exception Handling

Announcements/Reminders

- Today's class: notes template posted
- Priorities:
  - + Complete Lab1; Due: This Tuesday (Sep 30)
  - + Lab2 to be released
- ProgTest1
  - + guide released
  - + PracticeTest1 released
  - + In-Person Review Session at 2 PM, Friday, Oct 3 (CLH C)

Version 1: B.mb Catches

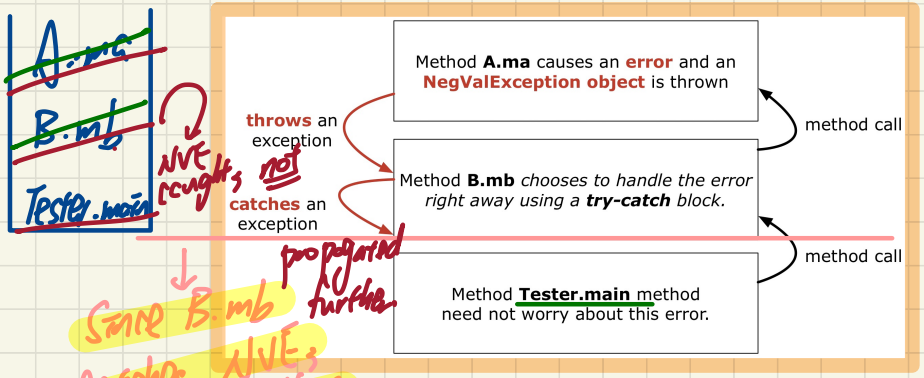
```
class A {
  A() {}

  void ma(int i) throws NegValException {
    if(i < 0) {
      println("abnormal exec of A.ma");
      throw new NegValException("Neg Val: " + i);
    }
    else {
      println("normal exec of A.ma");
    }
  }
}
```

```
class B {
  B() {}

  void mb(int i) {
    A oa = new A();
    try {
      oa.ma(i);
      println("From B.mb: Calling A.ma did not cause NVE.");
    }
    catch(NegValException nve) {
      println("From B.mb: Calling A.ma caused NVE.");
    }
  }
}
```

```
class Tester {
  main(...) {
    int i = 5;
    B ob = new B();
    ob.mb(i);
    println("From Tester.main: After calling B.mb.");
  }
}
```

normal exec of A.ma
B.mb: calling A.ma did not cause NVE.
Tester.main: after calling B.mb.

abnormal exec. of A.ma
B.mb: calling A.ma caused NVE.
Tester.main: After calling B.mb.



Since B.mb catches NVE, subsequent callers no longer subject to C-or-S req.

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

Yes ∵ its callee A.ma specifies it.

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

No ∵ its callee B.mb catches that.

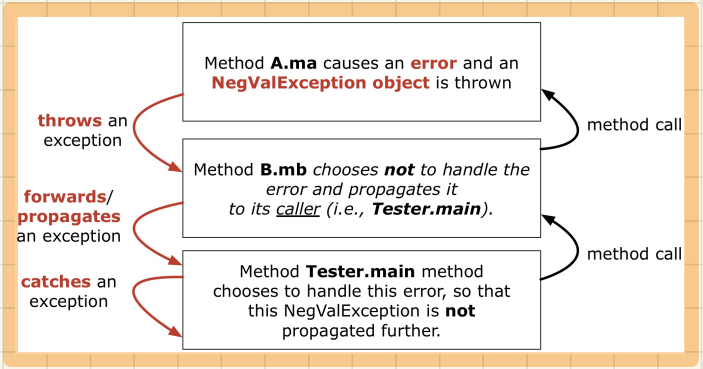
Version 2: B.mb Specifies, Tester.main Catches

```
class A {
  A() {}

  void ma(int i) throws NegValException {
    if(i < 0) {
      println("abnormal exec of A.ma");
      throw new NegValException("Neg Val: " + i);
    }
    else {
      println("normal exec of A.ma");
    }
  }
}
```

```
class B {
  B() {}

  void mb(int i) throws NegValException {
    A oa = new A();
    oa.ma(i);
    println("From B.mb: Calling A.ma did not cause NVE.");
  }
}
```

```
class Tester {
  main(...) {
    int i;
    B ob = new b();
    try {
      ob.mb(i);
      println("Tester.main: Calling B.mb did not cause NVE.");
    }
    catch(NegValException nve) {
      println("Tester.main: Calling B.mb caused NVE.");
    }
  }
}
```

could be specify option (version 3)



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

Yes :: A.ma specifies it.

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

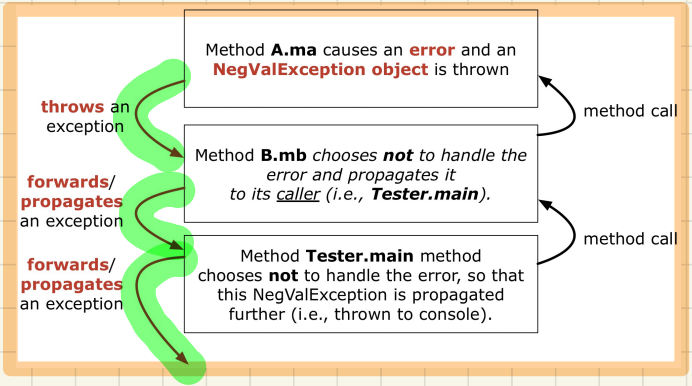
Yes :: B.mb specifies it

Version 3: B.mb Specifies, Tester.main Specifies

```
class A {
  A() {}

  void ma(int i) throws NegValException {
    if(i < 0) {
      println("abnormal exec of A.ma");
      throw new NegValException("Neg Val: " + i);
    }
    else {
      println("normal exec of A.ma");
    }
  }
}
```

```
class B {
  B() {}

  void mb(int i) throws NegValException {
    A oa = new A();
    oa.ma(i);
    println("From B.mb: Calling A.ma did not cause NVE.");
  }
}
```

```
class Tester {
  main(...) throws NegValException {
    int i;
    B ob = new b();
    ob.mb(i);
    println("Tester.main: Calling B.mb did not cause NVE.");
  }
}
```



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

Yes. A.ma specifies it

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

Yes. B.mb specifies it.

Catch-or-Specify Requirement: Call Stack

Q1. Origin of Exception:

$C_1.m_1$

Q2. Methods subject to CoS Req:

i methods ($i-1$ methods specify; 1 method catch).

Q3. Methods free from CoS Req:

$n-i$ methods.

Q4. Extreme Case 1 (exception handled earliest):

$C_2.m_2$ catches the exception.

Q5. Extreme Case 2 (exception never handled):

$C_1.m_1 \sim C_n.m_n$ all specify.