Lecture 7 - Sep. 28

Exceptions

To Handle or Not to Handle?
Error Handling using Exceptions

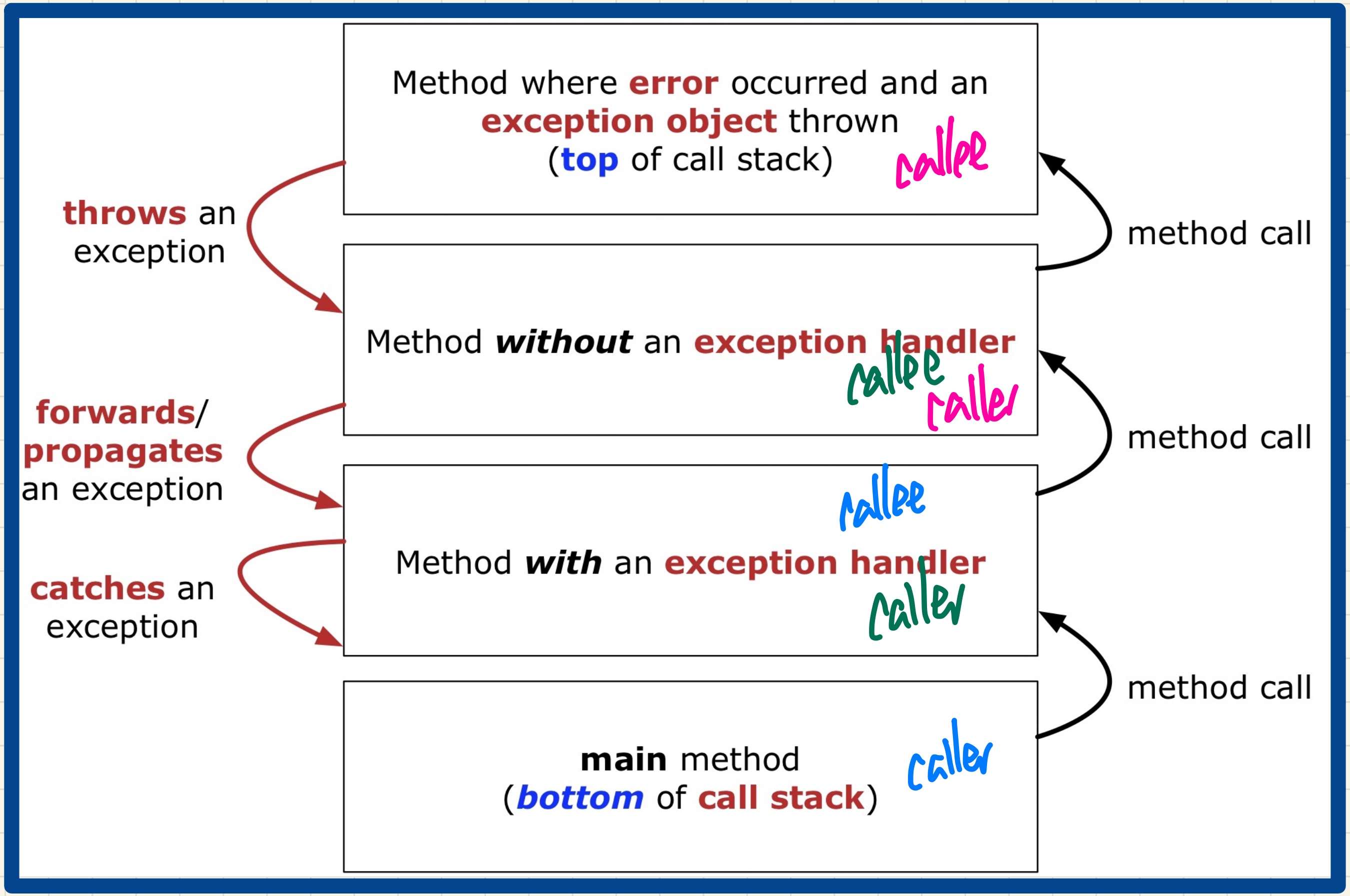
Announcements

- Lab1 due at 2pm today (Wednesday)
- WrittenTest1
  - Marks to be released on Friday
  - Visit my office hours to discuss questions if you wish
- Programming Test 1
  - Guide & Practice Test to be released (bteo Thursday)
  - A Short Mockup Test to be arranged

Exception Handler

```
try {
    ┌──────────────────┐
    │                  │
    └──────────────────┘
}
catch( ____________ ) {
    ________
}
catch( ____________ ) {
    ________
}
```

What to Do When an Exception is Thrown: Call Stack


Method where error occurred and an exception object thrown (top of call stack)
callee
throws an exception
method call
Method without an exception handler
callee
caller
forwards/ propagates an exception
method call
callee
Method with an exception handler
caller
catches an exception
method call
main method (bottom of call stack)
caller

Catch-or-Specify Requirement

to handle

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) {
    ...
  }
}
```

has the potential of throwing an exception

how to handle that exception.

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);
    ...
  }
}
```

1. some line in the body of impl may throw an exception

2. that excep thrown will not be handle in current method.

Example: To Handle or Not To Handle?

context	caller	callee
Tester	main	B.mb
B	mb	A.ma
A	ma	n.a.

```
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 */
  } }
```

```
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? */
  } }
```

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.

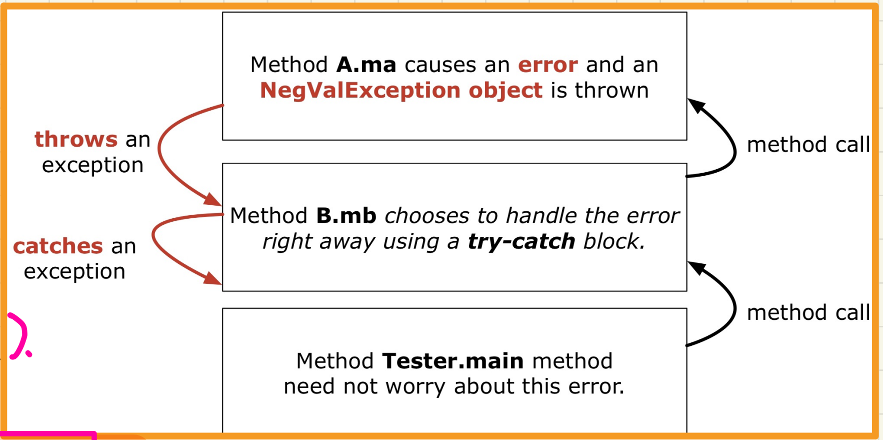
```
class NegValException extends Exception {
  NegValException(String s) { super(s); }
}
```

call stack

A.ma
B.mb
Tester.main

Version 1:

Handle the Exception in B.mb



Method **A.ma** causes an **error** and an **NegValException object** is thrown
throws an exception
method call
Method **B.mb** *chooses to handle the error right away using a* ***try-catch*** *block.*
catches an exception
method call
Method **Tester.main** method need not worry about this error.

to satisfy the catch or specify req. (specify).

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

this is where the error occurred

20 -10

Normal : 20

Abnormal : -10

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

throw NVE.

20 -10

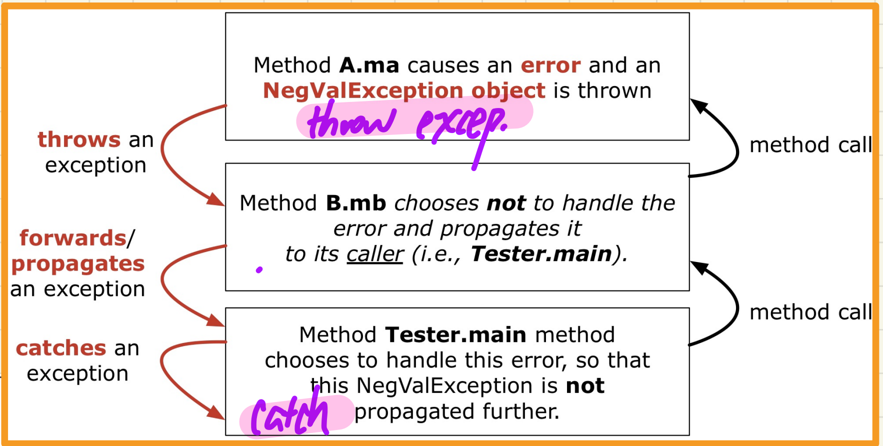
```
class Tester {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int i = input.nextInt();
    B ob = new B();
    ob.mb(i); /* Error, if any, would have been handled in B.mb. */
  } }
```

20 -10

20 -10

Version 2:

Handle the Exception in Tester.main



Method **A.ma** causes an **error** and an **NegValException object** is thrown

Method **B.mb** *chooses **not** to handle the error and propagates it to its caller (i.e., **Tester.main**).*

Method **Tester.main** method chooses to handle this error, so that this NegValException is **not** propagated further.

abnormal input: -10

specify

```
class A { -10
  ma(int i) throws NegValException {
    if (i < 0) { throw new NegValException("Error."); }
    else { /* Do something. */ }
} }
```

specify

```
class B { -10
  mb(int i) throws NegValException {
    A oa = new A();
    oa.ma(i); -> throws NVE
} }   -10
```

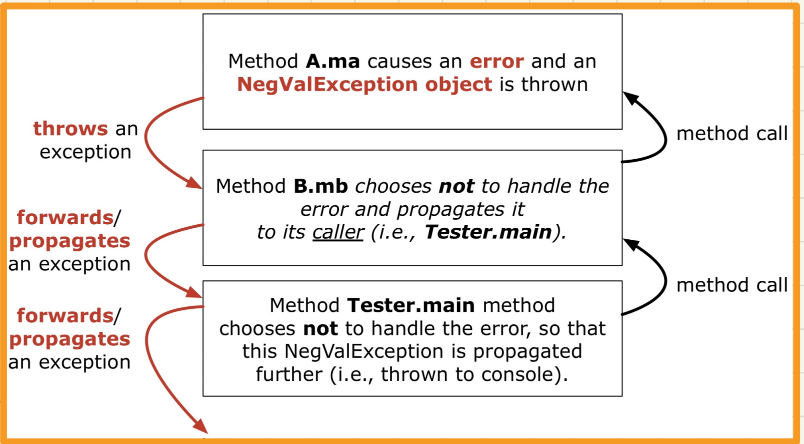
```
class Tester {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int i = input.nextInt(); -10
    B ob = new B();
    try { ob.mb(i); } -10
    catch(NegValException nve) { /* Do something. */ }
} }
```

this is where the exception gets handled.

exception handler

Version 3:

Handle in Neither Classes on Call Stack



throws an exception

Method **A.ma** causes an **error** and an **NegValException object** is thrown

method call

Method **B.mb** *chooses **not** to handle the error and propagates it to its caller (i.e., **Tester.main**).*

forwards/ **propagates** an exception

method call

Method **Tester.main** method chooses **not** to handle the error, so that this NegValException is propagated further (i.e., thrown to console).

forwards/ **propagates** an exception

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

```
class B {
 mb(int i) throws NegValException {
   A oa = new A();
   oa.ma(i);
 } }
```

```
class Tester {
 public static void main(String[] args) throws NegValException {
   Scanner input = new Scanner(System.in);
   int i = input.nextInt();
   B ob = new B();
   ob.mb(i);
 } }
```

-20 -20 -20 -20 -20 -20

specify

specify

prop. to terminal

abnormal input: -20.