Reading for today: 10. Next lecture: Ch 9.3

A7: remember, "Don't look at any other group's code, in any form; don't show any other students' code".
(The similarity checker turned up a few problems on A6, which we are about to start the Academic Integrity violation process for. Note that the checker essentially performs variable-name substitutions, etc., so syntactic modification of the same original program is potentially flagged.)

No labs this week, no TA office hours Wed-Fri, see consultant calendar for the updated schedule.

There are "labs next week, but they will serve as office hours plus an optional exercise on exceptions (covered on final).

Final: Friday Dec 10th, 9-11:30am, Statler Auditorium.
Register conflicts (same time, or 3 finals in 24 hours) on CMS assignment "final exam conflicts" by Monday November 30th.

Please check that your grades on CMS match what you think they are. [For lab-grade issues, contact your lab TA, not the instructors.]

A7: remember, "Don't look at any other group's code, in any form; don't show any other students' code".
(The similarity checker turned up a few problems on A6, which we are about to start the Academic Integrity violation process for. Note that the checker essentially performs variable-name substitutions, etc., so syntactic modification of the same original program is potentially flagged.)

Errors (little e) cause Java to throw a Throwable object as a “distress signal”

Errors (little e) cause Java to throw a Throwable object as a “distress signal”

Q1: What happens when an error causes the system to abort?
(NullPointerException, ArrayIndexOutOfBoundsException, …)
Understanding this helps you debug.

Q2: Can we make something other than termination happen?
It is sometimes better to warn and re-prompt the user than to have the program crash (even if the user didn’t follow your exquisitely clear directions or preconditions).

Understanding this helps you write more flexible code.

Today’s (and next lab’s) topic: when things go wrong (in Java)

Q1: What happens when an error causes the system to abort?
(NullPointerException, ArrayIndexOutOfBoundsException, …)
Understanding this helps you debug.

Q2: Can we make something other than termination happen?
It is sometimes better to warn and re-prompt the user than to have the program crash (even if the user didn’t follow your exquisitely clear directions or preconditions).

System prints the call-stack trace on catching exception:
ArithmeticException: / by zero
at Ex.third(Ex.java:13)
at Ex.second(Ex.java:9)
at Ex.first(Ex.java:5)

Errors are signals that things can be "handled".

Exceptions are signals that intervention may still be possible; they can be "handled".

/** Illustrate exception handling */
public class Ex {  
  public static void first() {  
    second();  
  }  
  public static void second() {  
    third();  
  }  
  public static void third() {  
    int x = 5 / 0;  
    try {  
      int y = x + 1;  
    } catch (ArithmeticException e) {  
      System.out.println(e);  
    }  
  }  
}
How can we catch/handle Throwables? With Try/catch blocks.

```java
/** = reciprocal of x. Throws an ArithmeticException if x is 0. (Assume this is third-party code that you can’t change.)*/
public static double reciprocal(int x) {
    // ...;
}
/** = reciprocal(x), or -1 if x is 0.
Assume you can’t change this spec.*/
public static double ourReciprocal(int x) {
    return reciprocal(x);
}
```