CS 211: Exceptions

Chris Kauffman

Week 9
Front Matter

Today

- Exceptions
- Maybe Generics

Lab 9: Quiz

- Enumerations, Abstract Classes, Interfaces
- Practice problems up

P4: GateSim

- Due ~ 2 weeks
- Minor updates for clarification
- Field Questions
Exceptions

- Generally allow nonlocal control flow,
- Thrown when execution must change location
- Done most often when an error occurs
- Exceptions Move up and out
  - Up the call stack
  - Out to a catch
  - Crash program if not caught
Java exceptions

▶ Are objects: `Exception e = new Exception("FAIL!!");`
▶ Subclass of Throwable, Error is as well
▶ Can be thrown: `throw e;`
▶ Can be caught: `try{...} catch(Exception e){...}
▶ Throwing method must declare uncaught throws... sometimes; **Catch or Declare**
▶ Allow cleanup with finally
General Flow

always;
try {
    may cause exceptions;
    may also cause exceptions;
}
catch(SomeException e){
    handle this kind of exception;
    be graceful;
}
catch(OtherException o){
    handle a different kind;
}
finally{
    always do this;
    even if no exception thrown;
    even if exception thrown;
    even if uncaught is thrown;
    use it close files and flush output;
}
do this after no/handled exceptions;

Exercise

- SimpleExceptions.java
- Trace execution path
- What gets printed?

catch and Types

- catch figures types of exceptions identically to instanceof
- Ordering problems can arise
- See TieredExceptions.java
Errors Happen

Handling them is always painful
▶ Old style: return an error code - C
▶ Newish style: change the flow of control

Compare: Reading first Word of a File
▶ ReadFirstWord.c
▶ ReadFirstWord.java
▶ ReadFirstNonlocal.java
▶ ReadFirstManyCatch.java

Compare: Reading 3-column Input
▶ CrashOnErrors.java
▶ HandleErrors.java
▶ HandleErrors2.java
Catch or Declare... sometimes

How many times have you written

... throws NullPointerException

- RuntimeExceptions and Errors aren't declared
- Reserved for bugs rather than anticipated conditions

Unchecked runtime exceptions represent conditions that, generally speaking, reflect errors in your program’s logic and cannot be reasonably recovered from at run time.
– The Java Programming Language, by Gosling, Arnold, and Holmes

See also
- Overview of Checked v. Unchecked by Hirondelle Systems
- Chua Hock-Chuan’s Lecture Notes on Java Exceptions
- I am a great fan of RuntimeException during development
Checked versus Unchecked

Checked Exceptions
(MUST be caught or declared to be thrown)

Unchecked Exceptions

Source: Chua Hock-Chuan: Java Programming Exception Handling & Assertion