Error Handling with Exceptions

http://java.sun.com/docs/books/tutorial/essential/exceptions/index.html

Errors

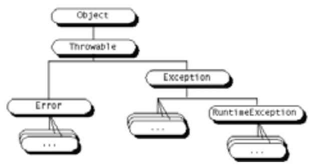
- Error (in general programming sense):
  - you've made a mistake!
- Categories:
  - language: syntax and semantic
  - runtime
  - logic
- When do those errors occur?
- How to handle?
  - write perfect language
  - all data is always legal
  - algorithms are precisely mapped
  - see http://java.sun.com/docs/books/tutorial/essential/exceptions/definition.html

Error Handling

- Want ways to deal with reality of errors
  - smart editors
  - smart compilers find errors
  - runtime environment alerts
- Common concept:
  - find error: ***catch***
  - alert something/something: ***throw***
- Java uses **`Object`**s:
  - **`Throwable`** object: special object that can be "sent" by code to other code (the "alert")
  - **`Error`**: special kind of **`Throwable`** for grievous error; not meant to recover
  - **`Exception`**: all other errors that can be dealt with (catch and throw)

Throwable Hierarchy

From Java tutorial:



Exceptions

- Java `Exception`:
  - Compile time (our focus)
  - Runtime
- Compile time
  - ***checked exception***
  - you must handle somehow in your code (why?)
  - example: see File I/O code
- Runtime:
  - ***unchecked exception***
  - you do not need to handle in your code (why?)
  - examples: array out of bounds, null pointers, divide by zero...

Handling Exceptions

- Three operations:
  - claiming exception
    - method informs compiler about its possible exceptions
    - sometimes you see method headers like this:
      `public void myMethod() throws IOException`
  - throwing exception
    - statement can cause (compile time) or does cause (runtime) an error
    - method creates an exception object, which has info about program state, and passes to JVM
    - you can also deliberately throw an exception with `throw`
  - catching exception
    - Java looks for code that handles the exception
    - starts in current method and then works backward in chain of method calls

Catching Exceptions

- catching exceptions in body of method

```
try {
   statements
}
catch (Exception1 e) {
   statements
}
catch (Exception2 e) {
   statements
}
.
.
.
finally {
   statements
}
```

- examples...