

CS212

Exceptions

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

1

Motivation

- Some design considerations:
 - Why do programs crash?
 - Can a programmer account for every problem in the universe?
 - Should a program gracefully warn a user about a problem?
- So, what do you do?

2

Mistakes

- When/how mistakes can occur:
 - editing
 - compiling
 - runtime
 - logic
- How to handle each of these?
Want ways to deal with reality of errors:
 - smarter editors
 - smarter compilers
 - clearer runtime environment alerts

3

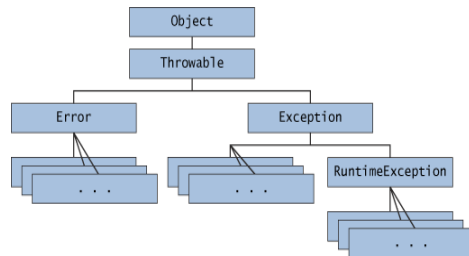
Handling

- Common concepts:
 - program must find the mistake: *catch*
 - program must alert something/someone: *throw*
- Java uses **Objects**:
 - **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)

4

Throwable Hierarchy

From Java's on-line tutorial:



Error: serious problem...crash!

Exception: something you can handle...catch!

5

Exceptions

- Java **Exception**:
 - Compile time
 - Runtime
- Compile time
 - **checked exception**
 - you must handle somehow in your code
 - example: see File I/O code
- Runtime:
 - **unchecked exception**
 - you do not need to handle in your code
 - eg) array out of bounds, null pointers, divide by zero...
- An aside...your compiler:
 - When you write your compiler, what kind of exceptions (for the most part) are you dealing with?
 - When we grade your compiler, what kind of exceptions are we looking for?

6

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

7

Catching Exceptions

- catching exceptions in body of method:

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

8