

# Not Business As Usual

CSC207H Summer 2009

May 26<sup>th</sup>, 2009

# What Are They Good For?

- Exceptions report exceptional conditions: unusual, strange, disturbing.
- These conditions deserve exceptional treatment: not the usual go-to-the-next-step, plod-onwards approach.
- They break the normal flow of execution to signal something exceptional.

# Before Exceptions. . .

- . . . we used special return values for functions to signify an error.
- . . . we used flags to signal a condition.

# The Big Picture

- To signal a special condition, a piece of code *throws* an exception.
- To handle a special condition, a piece of code *tries* to execute a block of code and *catches* any exceptions thrown inside that block.
- We put clean up code that runs *finally*, regardless of whether an exception was encountered or not.



# An Analogy

- throw = "I'm in trouble, so I throw a rock through a window, with a message tied to it."
- try = "Someone in the following block of code might throw rocks of various kinds. All you catchers line up ready for them."
- catch = "If a rock of my kind comes by, I'll catch it and deal with it."

## Two Kinds of Exceptions

- *Runtime exceptions* “can be thrown during the normal execution of the Java Virtual Machine”<sup>1</sup>. E.g., `IndexOutOfBoundsException`, `NullPointerException`, and `UnsupportedOperationException`.
- Every other kind of exception (*checked exceptions*).

---

<sup>1</sup><http://java.sun.com/j2se/1.4.2/docs/api/java/lang/RuntimeException.html>

# Can't Catch This (or at least generally shouldn't)

- `catch (Throwable thr) { ... }` – this is too broad, unless you just want to play ostrich.
- `catch (Exception e) { ... }` – this is too broad, unless you just want to play ostrich.
- While not an exception, we *can* also catch Errors, but an Error “indicates serious problems that a reasonable application should not try to catch.”<sup>2</sup>

---

<sup>2</sup><http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Error.html>

# Declaring “throws”

Consider `public void x() throws Y, Z {...}`.

- Any uncaught exceptions that belong to the classes (or subclasses of) `Y` or `Z` will be propagated (“re-thrown”) automatically.
- Any uncaught checked exceptions that are not instances of `Y` or `Z` (or their subclasses) will trigger a compiler error.
- `x()` reserves the right to throw a `Y` or a `Z`, so anything that calls `x()` needs to be prepared.

# Alternatives to Exceptions

- Exceptions are relatively heavyweight solutions in terms of memory usage.
- What if we may not have memory to create a new exception object?
- One solution employed in Symbian OS is the use of LEAVES and TRAPs.

# Alternatives to Exceptions

- Exceptions are relatively heavyweight solutions in terms of memory usage.
- What if we may not have memory to create a new exception object?
- One solution employed in Symbian OS is the use of LEAVEs and TRAPs.

