Exceptions

Java Exceptions
Catch or Specify
Java syntax
Exceptions

Exceptions are a control flow construct for error management.

• Some similarity to event handling (lecture topic X2)
  – Both disrupt the normal flow of execution, transferring to event handler or exception handler
  – However: exceptions are exceptional situations (events are expected)
    • A file is not found or is inaccessible
    • An array is accessed incorrectly (out of bounds)
    • Division by zero
    • A null pointer is dereferenced, etc.
Java Exceptions

Exceptions are *thrown* either:

- Implicitly (via a program error) or
- Explicitly (by executing the `throw` statement).

Exceptions are *caught* with a `catch` block.

Exceptions are propagated from callee to caller until a matching handler is found. Methods throwing uncaught exceptions must have the `throws` clause in their declaration.
Java’s **Catch** or Specify Requirement

Three kinds of exception:

- **error** (*Error* and its subclasses),
- **runtime exception** (*RuntimeException* and its subclasses),
- **checked** (everything else, must comply with Catch or Specify)

Java requires that code that may throw a checked exception must be enclosed by either:

- a **try** statement with a suitable handler, or
- a method that declares that it **throws** the exception
Java `try/catch` Block Syntax

```
try {
    // do something that may generate an exception
} catch (ArithmeticException e1) {
    // first catch
    // this is an arithmetic exception handler
    // handle the error and/or throw an exception
} catch (Exception e2) {
    // may have many catch blocks
    // this an generic exception handler
    // handle the error and/or throw an exception
} finally {
    // this code is guaranteed to run
    // if you need to clean up, put the code here
}
```