

CS100J 3 April 2008
Exceptions in Java. Read chapter 10.

HUMOR FOR LEXOPHILES (LOVERS OF WORDS):

Police were called to a day care; a three-year-old was resisting a rest.
 Did you hear about the guy whose whole left side was cut off?
 He's all right now.

The butcher backed into the meat grinder and got a little behind in his work.

When fish are in schools they sometimes take debate.

A thief fell and broke his leg in wet cement. He became a hardened criminal.

Thieves who steal corn from a garden could be charged with stalking.

When the smog lifts in Los Angeles, U.C.L.A.

What happens when an error of some sort occurs?

```
// String s is supposed to contain an integer.
// Store that integer in variable b.
b= Integer.parseInt(s);
```

```
/** Parse s as a signed decimal integer and return
    the integer. If s does not contain a signed decimal
    integer, throw a NumberFormatException. */
public static int parseInt(String s)
```

Exceptions and Errors

In Java, there is a class Throwable:

```
class Throwable {
    detailMessage "/ by zero"
    getMessage()
}
```

When some kind of error occurs, an **exception** is "thrown" — you'll see what this means later.

An **exception** is an instance of class Throwable (or one of its subclasses)

Exceptions and Errors

So many different kind of exceptions that we have to organize them.

```
class Throwable {
    detailMessage "/ by zero"
    getMessage()
}
class Exception {
}
class RuntimeException {
}
class ArithmeticException {
}
```

Throwable
 ├── Exception
 └── Error (Do nothing with these)

Exception
 └── RuntimeException (You can "handle" these)
 └── ArithmeticException

```
/** Illustrate exception handling */
public class Ex {
    public static void first() {
        second();
    }
    public static void second() {
        third();
    }
    public static void third() {
        int x = 5 / 0;
    }
}
```

Call: Ex.first();

Output: ArithmeticException: / by zero
 at Ex.third(Ex.java:13)
 at Ex.second(Ex.java:9)
 at Ex.first(Ex.java:5)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(...)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(...)
 at java.lang.reflect.Method.invoke(Method.java:585)

```
/** Illustrate exception handling */
public class Ex {
    public static void first() {
        second();
    }
    public static void second() {
        third();
    }
    public static void third() {
        throw new ArithmeticException("I threw it");
    }
}
```

Call: Ex.first();

Output: ArithmeticException: I threw it
 at Ex.third(Ex.java:14)
 at Ex.second(Ex.java:9)
 at Ex.first(Ex.java:5)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(...)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(...)
 at java.lang.reflect.Method.invoke(Method.java:585)

Won't compile. Needs a "throws" clause, see next slide

Class: → `public class Ex {`

```

    public static void first() {
        second();
    }
    public static void second() {
        third();
    }
    public static void third() {
        throw new
            MyException("mine");
    }
}

```

Call
 Ex.first();
 ArithmeticException: mine
 at Ex.third(Ex.java:14)
 at Ex.second(Ex.java:9)
 at Ex.first(Ex.java:5)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(...)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(...)
 at java.lang.reflect.Method.invoke(Method.java:585)

The "throws" clause

```

/** Class to illustrate exception handling */
public class Ex {
    public static void first() throws MyException {
        second();
    }
    public static void second() throws MyException {
        third();
    }
    public static void third() throws MyException {
        throw new MyException("mine");
    }
}

```

```

public static void first() throws MyException{
    try {
        second();
    }
    catch (MyException ae) {
        System.out.println
            ("Caught MyException: " + ae);
    }
    System.out.println("procedure first is done");
}

public static void second() throws MyException {
    third();
}

public static void third() throws MyException {
    throw new MyException("yours");
}

```

Catching a thrown exception

Execute the try-block. If it finishes without throwing anything, fine.

If it throws a MyException object, catch it (execute the catch block); else throw it out further.