Exceptions

In standard English being "exceptional" is a good thing. Your mother smiles when she talks about how exceptional you are.

In system terms, being exceptional is usually a bad thing. Exceptions are events that cause a system to crash if they are not handled internally.

Exceptions are handled in Java via try-catch statements. The basic structure looks like this:

```
try {
        <code>
}
catch (<exception class> e) {
        <code to handle the exception>
}
```

If <code> executes without a problem the catch code is never invoked. If an exception of the class in the catch phrase is thrown, the corresponding catch code is executed.

An alternative to a try-catch statement is to add to the metho
header a declaration that the method "throws" the exception

For example, when you construct a new File object the
construct possibly throws a FileNotFoundException.   You can
either note in the header

```
void myFilePrinter( String fname) throws FileNotFoundException {
      ....
}
```

or else handle the exception yourself:

```
void myFilePrinter( String fname) {
    try {
            Scanner reader = new Scanner(new File(fname));
            while (reader.hasNextLine()) {
                    String line = reader.nextLine();
                    System.out.println( line );
            }
    }
    catch (FileNotFoundException e ) {
            System.out.printf( "File %s not found.", fname );
    }
```

In Java there are two kinds of exceptions. Almost any method could conceivably throw an exception. Those are regular exceptions. There are a few exceptions that occur so often and are so damaging that the Java compiler checks to make sure you have handled them. These are called *checked* exceptions. When a procedure is capable of throwing a checked exception you must either put a try-catch block around it or note that the surrounding procedure throws the exception.

The primary checked exception we will deal with this term is FileNotFoundException, which is thrown by the File constructor. We call this constructor every time we read a text file. Here is a standard way to handle that, assuming that String *filename* is the name of the file to be opened:

```
Scanner scan = null;     // declares and initializes variable scan
try {
        scan = new Scanner( new File(  filename ) );
}
catch (FileNotFoundException e) {
        System.out.println(e);        // prints an error message
        System.exit(-1);              // halts the program
}
```