Exceptions

Ahhh, Oberlin, where all the students are exceptional.....

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 things 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 object 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 method 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 );
    }
```

This choice goes out the window when you are implementing an abstract class or an interface. If an abstract method doesn't say it throws an exception, your concrete implementation of it can't throw the exception, so you need to catch it.