Writing your own Java I/O Decorator

Okay, you know the Decorator Pattern, you've seen the I/O class diagram. You should be ready to write your own input decorator.

How about this: write a decorator that converts all uppercase characters to lowercase in the input stream. In other words, if we read in "I know the Decorator Pattern therefore I RULE!" then your decorator converts this to "i know the decorator pattern therefore i rule!"

No problem. I just have to extend the FilterInputStream class and override the read() methods.

Don't forget to import java.io (not shown).

First, extend the FilterInputStream, the abstract decorator for all InputStreams.

```java
public class LowerCaseInputStream extends FilterInputStream {
    public LowerCaseInputStream(InputStream in) {
        super(in);
    }
    public int read() throws IOException {
        int c = super.read();
        return (c == -1 ? c : Character.toLowerCase((char) c));
    }
    public int read(byte[] b, int offset, int len) throws IOException {
        int result = super.read(b, offset, len);
        for (int i = offset; i < offset + result; i++) {
            b[i] = (byte) Character.toLowerCase((char) b[i]);
        }
        return result;
    }
}
```

REMEMBER: we don't provide import and package statements in the code listings. Get the complete source code from the headfirstlabs web site. You'll find the URL on page xxxiii in the intro.

Test your new Java I/O Decorator

Write some quick code to test the I/O decorator:

```java
public class InputTest {
    public static void main(String[] args) throws IOException {
        try {
            InputStream in =
                    new LowerCaseInputStream(
                        new BufferedInputStream(
                            new FileInputStream("test.txt")));
            while ((c = in.read()) >= 0) {
                System.out.print((char) c);
            }
            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Just use the stream to read characters until the end of file and print as we go.

Give it a spin: