



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN



 mobile and
distributed systems group



Javakurs für Fortgeschrittene

Einheit 02: Streams filtern

Lorenz Schauer

Lehrstuhl für Mobile und Verteilte Systeme



1. Teil: Datenströme (Streams)

- Filtern
 - FilterWriter, - Reader

Praxis:

- Log-File verschlüsseln
- UI für Log-File

Lernziele

- Kennenlernen und Nutzen von Filter für Datenströme
- Weitere Details und Übungen zu Streams

Mit den abstrakten Klassen `FilterReader` bzw. `FilterWriter` können wir Datenströme vor dem eigentlichen Schreibvorgang filtern.

- Um nur bestimmte Informationen durchzulassen
 - Bsp.: Alle Wörter mit „a“
- Um Informationen zu verändern
 - Bsp.: Verschlüsselung

Abstrakte Klassen `FilterReader` und `FilterWriter` arbeiten mit dem entsprechenden `Reader` bzw. `Writer` zusammen

- Muss im Konstruktor übergeben werden!
- `protected FilterReader(Reader in)`
- `protected FilterWriter(Writer out)`

//Beispiel:

```
BufferedWriter writer = new BufferedWriter(new FileWriter(logFile,true));  
MyFilterWriter encode = new MyFilterWriter(writer); // Eigene Klasse...  
encode.write("Einzahlung auf Konto..");  
    // Schreiben geht jetzt über den eigenen FilterWriter
```

Methoden von `FilterWriter`:

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
void	close() Closes the stream, flushing it first.	
void	flush() Flushes the stream.	
void	write(char[] cbuf, int off, int len) Writes a portion of an array of characters.	
void	write(int c) Writes a single character.	
void	write(String str, int off, int len) Writes a portion of a string.	

- Eigenen `FilterWriter` schreiben (ableiten von `FilterWriter`)
 - Sollte mindestens die `write(int c)` Methode überschreiben!

Die Methode `write(int c)` benötigt einen Integer-Wert als Parameter

- `Int c` wird als 8-Bit Unicode Zeichen interpretiert und in den Stream geschrieben
- Dazu werden die beiden niederwertigen Bytes des 4-Byte Integer verwendet
- Für die Umwandlung in ein bestimmtes Zeichen siehe Unicode-Tabelle:
 - Bsp.: A = 65 = 0100 0001
 - Bsp.: Z = 90 = 01011010
 - Bsp.: a = 97 = 0110 0001
 - Bsp.: z = 122 = 0111 1010
- Zum Umwandeln von Groß auf Kleinbuchstaben: +32 rechnen

0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0010	DLE	DC1	DC2	DC3	NAK	SYN	ETB	CAN	EMU	SUB	ESC	FS	GS	RS	US	
0020		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	p	q	r	s	t	u	v	w	x	y	z	{		}	~	¸
0080																
0090																
00A0																
00B0																
00C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
00D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß

Quelle: <http://unicode-table.com/de/>

```
public class MyFilterWriter extends FilterWriter {
```

```
    protected MyFilterWriter(Writer out) {  
        super(out);  
    }
```

Konstruktor erwartet
einen **Writer**

```
    @Override  
    public void write(int c) throws IOException {
```

Entscheidende write-
Methode überschreiben

```
        if (c >= 65 && c <= 90)  
            super.write(c+32);  
        else  
            super.write(c);  
    }
```

Hier: Wandle alle Großbuchstaben
in Kleinbuchstaben um

```
    @Override  
    public void write(char[] cbuf, int off, int len) throws IOException {  
        for(int i=0; i< len; i++)  
            write(cbuf[off+i]);  
    }
```

Überschreibe die anderen write-
Methoden unter Verwendung der
`write(int c)`

```
    @Override  
    public void write(String str, int off, int len) throws IOException {  
        for(int i=0; i< len; i++)  
            write(str.charAt(off+i));  
    }
```

```
}
```

Methoden von FilterReader:

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
void	close()	Closes the stream and releases any system resources associated with it.
void	mark(int readAheadLimit)	Marks the present position in the stream.
boolean	markSupported()	Tells whether this stream supports the mark() operation.
int	read()	Reads a single character.
int	read(char[] cbuf, int off, int len)	Reads characters into a portion of an array.
boolean	ready()	Tells whether this stream is ready to be read.
void	reset()	Resets the stream.
long	skip(long n)	Skips characters.

Liest Zeichen aus dem Stream und gibt es als **int** (Unicod-Zeichen) zurück.

-1, falls das Ende erreicht wird

Liest **len** Zeichen aus dem Stream und legt diese im Char-Array **cbuf** ab. Es wird die Anzahl der gelesenen Zeichen zurückgegeben
-1, falls das Ende erreicht wird

- Eigenen **FilterReader** schreiben (ableiten von **FilterReader**)
 - Sollte mindestens die **read()** Methode überschreiben!

```
public class MyFilterReader extends FilterReader{
```

```
    protected MyFilterReader(Reader in) {  
        super(in);  
    }
```

```
    @Override  
    public int read() throws IOException {  
        return super.read() - 32;  
    }
```

```
    @Override  
    public int read(char[] cbuf, int off, int len) throws IOException {  
        int result = super.read(cbuf, off, len);  
  
        for(int i=0; i<result; i++)  
            cbuf[off+i] = (char) (cbuf[off+i]-32);  
  
        return result;  
    }
```

```
}
```

Konstruktor erwartet
einen Reader

read()-Methode
überschreiben

Hier: Alle Zeichen um 32
Unicodezeichen verschieben

Ggf. überschreibe die anderen
read-Methoden

Sie wollen nun Einträge in Ihre Log-Datei von letzter Stunde verschlüsseln, so dass kein anderer Nutzer die Datei lesen kann.

- Erzeugen Sie ein neues Eclipse-Package „uebung02“ und kopieren Sie Ihr Programm von letzter Stunde in das Package.
- Schreiben Sie nun einen eigenen **FilterWriter**, der den Ausgabestrom so manipuliert, dass die Daten nicht mehr einfach von Menschen lesbar sind
 - Bspw.: ändern Sie jedes zu übertragene Zeichen in ein anderes.
- Tätigen Sie wieder ein paar Ein- und Auszahlungen und schauen Sie sich die Einträge in Ihrer Log-Datei an.
- Schreiben Sie nun einen eigenen **FilterReader**, welcher die Verschlüsselung beim Lesen des Eingabestroms rückgängig macht.
- Schreiben Sie eine neue Methode **decodeLogFile()**, welche den dekodierten Inhalt der Datei auf der Konsole ausgibt.
- Rufen Sie die Methode **decodeLogFile()** auf und kontrollieren Sie das Ergebnis.