



Unweit von hier befand sich das Basislager von dem aus  
die wagemutigen BRÜDER FLORIAN UND MICHAEL BLOCH  
mit ihren unerschrockenen Sherpas am  
Sonntag den 28.6.2009 um 11:44:17 MEZ  
zur ERSTBESTEIGUNG DES PATSCHERKOFEL MIT SAUERSTOFFMASKEN aufbrachen.  
Um 13:35:02 MEZ konnte am Fuß des Gipfelkreuzes die Pfeifnudel-Flagge gehisst werden.

[www.pfeifnudel.de](http://www.pfeifnudel.de)

## 06-Input/Output-2

Objektorientierte Programmierung | Matthias Tichy

# Lernziele

- Weitere Aspekte von Dateien
- Java New I/O
- Einfaches Parsen

# Weitere Themen zu Input/Output

# Anmerkungen

- Anhängen an existierende Dateien:
    - `FileOutputStream(File file, boolean append)`
    - `FileWriter(File file, boolean append)`
  - Gar kein wahlfreier Zugriff?
    - `RandomAccessFile(File file, String mode)`
    - `RandomAccessFile(String name, String mode)`
    - `long getFilePointer()`
    - `void seek (long pos)`
    - `(mode = "r|rw|rws|rwd")`

# Anmerkungen

- Schließen vergessen?
- try-with-resources-Statement ab Java 7:

```
try (FileInputStream in = new FileInputStream("beispiel.txt");
      FileOutputStream out = new FileOutputStream("out.txt")) {
    int c;
    while ((c = in.read()) != -1) {
        out.write(c);
    }
}
```

- deklarierte Variablen nur im try-Block bekannt, nicht in catch- oder finally-Blöcken
- Voraussetzung: AutoCloseable-Interface

# IOExceptions



# Locale

3,14156

-123.456.789,987654

- Locales geben "Gebietsschemaparameter" an
  - Sprache
  - Zeichensatz
  - Tastaturlayout
  - Zahlenformate
  - Währungsformat
  - Datum- und Zeitformate
- Codes gebräuchlich:
  - de\_DE, de\_AT, en\_US, en\_GB, ...

# formatierte Textausgabe

```
int i = 5;
double r = Math.sqrt(i);
String s = "Alexander";
Calendar cal = Calendar.getInstance();
Date now = cal.getTime();

System.out.format("%15s wants to know the result at %tH:%<tM
                  The square root of %d is %.12f.%n", s, now, i, r);
```

- Formatstring
- unendliche Argumentliste (varargs)
- Benutzung in format() oder printf()

# formatierte Textausgabe

Allgemeine Syntax:

`%[argument_index$][flags][width][.precision]conversion`

[ ] bedeutet optional

- argument\_index: Position in der Argumentliste,  
Abkürzung "<" für vorhergehendes  
Argument
- flags: z.B. führende Nullen, Vorzeichen, Linksbündigkeit
- width: minimale Breite in der das Argument gesetzt wird
- precision: Anzahl der Nachkommastellen
- conversion: Formatierungshinweise

# Standard Streams

Bereits bekannt:

- System.out      PrintStream
- System.err      PrintStream
- System.in        InputStream (keine Character!)

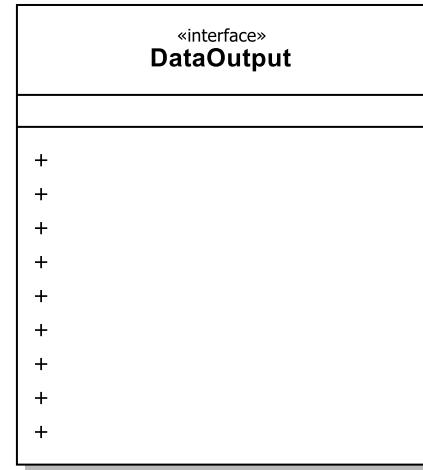
## ■ Vorteile des Decorator:

```
InputStreamReader cin = new InputStreamReader(System.in, "UTF-8");
int c = cin.read();
System.out.println(c);
```

# Data Streams

- siehe Beispielcode DataInOut.java
- Funktionen für primitive Datentypen (+String)
- Reihenfolge identisch beim Schreiben und beim Einlesen
- weiteres Beispiel für Decorator-Pattern:  

```
DataStream out =  
    new DataOutputStream(  
        new BufferedOutputStream(  
            new FileOutputStream("out.data")))
```
- Dateiendeerkennung über EOF-Exception



# Object Streams

- siehe Beispielcode SerializeExample.java
- Vereinfacht das Speichern auch von Kaskaden von Objekten
- kein Austauschformat zwischen Programmen.
- fragil bzgl. Klassenänderungen → nicht für längere Datenspeicherung
- Ursprüngliche Idee der Objektserialisierung:  
Objekt über Netzwerk schicken und auf anderer Seite wieder zusammenbauen

# Object Streams

- Serializable (Marker-Interface)
  - private static final long serialVersionUID = 42L;  
"Versionsnummer" einer Klasse
  - wichtigste Methoden:  
`writeObject(Object o)`  
`Object readObject()`
- danach tyecast auf eigentliche Klasse  
→ bei falscher Klasse: ClassNotFoundException

# Object Streams

- Was wird gesichert?
  - alle Felder (auch private!)
  - konkrete Angaben bzw. Ausnahmen möglich
- Warum überhaupt "Serializable-Interface"?
  - Threads sind z.B. nicht ?
- transient
  - nicht serialisiert, z.B. Passwörter!

# Object Streams

- Referenzen werden ebenfalls wieder hergestellt
  - Trotzdem viele Probleme bei verschiedenen Klassen z.B. SwingKomponenten nicht einmal von einer VM-Version zur anderen übertragbar
- mit Vorsicht genießen! Security-Probleme!
- <https://docs.oracle.com/en/java/javase/20/core/serialization-filtering1.html>
- Wie funktionierts intern? Reflection
  - Spezifikation:  
<https://docs.oracle.com/en/java/javase/20/docs/specs/serialization/serial-arch.html>

# Object Streams

- Vorteile gegenüber einer Textrepräsentation:
- Nachteile gegenüber einer Textrepräsentation:

# Dateisystem

# Dateisystem

Dieser PC > Lokaler Datenträger (C:) > Benutzer > mtt > svns > oop > folien					
	Name	Änderungsdatum	Typ	Größe	
>	mdse				
>	mentor				
>	mice				
>	mice-papers				
>	mtt				
>	oop				
>	code				
>	.gradle				
	.settings				
>	de.uni.ulm.sp.oop.arrays				
>	de.uni.ulm.sp.oop.collections				
>	de.uni.ulm.sp.oop.math				
>	de.uni.ulm.sp.oop.math_initial				
>	GitExample				
	TypeCompatibilityExamples				
>	de.uni.ulm.sp.oop.math-initial.zip				
>	exercises				
>	_shared				
>	exercises				
>	sose23				
>	folien				
	images				
>	orga				
	tafelbilder				
>	PP				
	images	21.05.2023 21:34	Dateiordner		
	orga	10.05.2023 19:40	Dateiordner		
	tafelbilder	17.05.2023 19:01	Dateiordner		
	.gitignore	25.04.2023 21:30	Textdokument	1 KB	
	Modulbeschreibung.md	27.10.2022 17:56	MD-Datei	4 KB	
	OOP-00-Willkommen.pptx	17.04.2023 20:23	Microsoft PowerP...	1.825 KB	
	OOP-01-Programmiersprachen-1.pptx	17.04.2023 20:23	Microsoft PowerP...	1.048 KB	
	OOP-02-Imperativ-1.pptx	28.04.2023 17:47	Microsoft PowerP...	1.423 KB	
	OOP-02-Imperativ-2.pdf	23.04.2023 21:26	Microsoft Edge P...	957 KB	
	OOP-02-Imperativ-2.pptx	21.04.2023 15:54	Microsoft PowerP...	1.407 KB	
	OOP-02-Imperativ-3.pptx	08.05.2023 21:11	Microsoft PowerP...	2.909 KB	
	OOP-03-DevelopmentTools-1.pdf	23.04.2023 21:31	Microsoft Edge P...	1.666 KB	
	OOP-03-DevelopmentTools-1.pptx	23.04.2023 20:29	Microsoft PowerP...	4.219 KB	
	OOP-03-DevelopmentTools-2.pdf	23.04.2023 21:47	Microsoft Edge P...	1.512 KB	
	OOP-03-DevelopmentTools-2.pptx	23.04.2023 21:47	Microsoft PowerP...	3.950 KB	
	OOP-03-DevelopmentTools-3.pptx	07.05.2023 20:35	Microsoft PowerP...	3.589 KB	
	OOP-03-DevelopmentTools-4.pptx	27.04.2023 10:09	Microsoft PowerP...	3.670 KB	
	OOP-04-Objekte-1.pptx	10.05.2023 19:40	Microsoft PowerP...	1.426 KB	
	OOP-04-Objekte-2.pptx	10.05.2023 19:40	Microsoft PowerP...	1.567 KB	
	OOP-04-Objekte-3.pptx	16.05.2023 20:40	Microsoft PowerP...	1.656 KB	
	OOP-05-Collections-1.pptx	17.05.2023 19:01	Microsoft PowerP...	4.814 KB	
	OOP-06-IO.pptx	21.05.2023 21:39	Microsoft PowerP...	5.041 KB	
	OOP-06-IO-2.pptx	23.05.2023 21:39	Microsoft PowerP...	4.802 KB	
	OOP-EarlyEval.pptx	17.05.2023 19:01	Microsoft PowerP...	1.218 KB	
	README.md	27.10.2022 17:56	MD-Datei	7 KB	

# Filemanipulation

- `java.io.File`
- komplexe API:

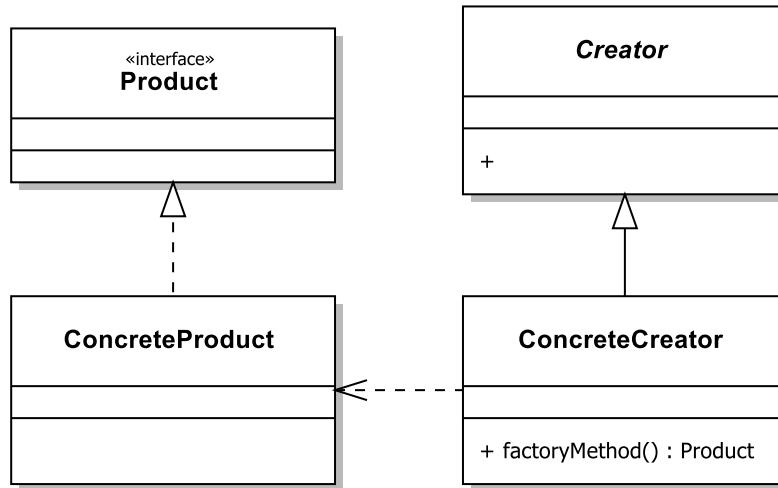
<code>boolean</code>	<code>canExecute()</code>	<code>boolean</code>	<code>isHidden()</code>
<code>boolean</code>	<code>canRead()</code>	<code>long</code>	<code>lastModified()</code>
<code>boolean</code>	<code>canWrite()</code>	<code>long</code>	<code>length()</code>
<code>boolean</code>	<code>createNewFile()</code>	<code>String[]</code>	<code>list()</code>
<code>static File</code>	<code>createTempFile(</code>	<code>String[]</code>	<code>list(FilenameFilter filter)</code>
<code>boolean</code>	<code>String prefix, String suffix)</code>	<code>File[]</code>	<code>listFiles()</code>
<code>boolean</code>	<code>delete()</code>	<code>File[]</code>	<code>listFiles(FileFilter filter)</code>
<code>void</code>	<code>deleteOnExit()</code>	<code>File[]</code>	<code>listFiles(FilenameFilter filter)</code>
<code>boolean</code>	<code>equals(Object obj)</code>	<code>static File[]</code>	<code>listRoots()</code>
<code>boolean</code>	<code>exists()</code>	<code>boolean</code>	<code>mkdir()</code>
<code>File</code>	<code>getAbsoluteFile()</code>	<code>boolean</code>	<code>mkdirs()</code>
<code>String</code>	<code>getAbsolutePath()</code>	<code>boolean</code>	<code>renameTo(File dest)</code>
<code>File</code>	<code>getCanonicalFile()</code>	<code>boolean</code>	<code>setExecutable(boolean executable)</code>
<code>String</code>	<code>getCanonicalPath()</code>	<code>boolean</code>	<code>setExecutable(boolean executable,</code>
<code>long</code>	<code>getFreeSpace()</code>		<code>                  boolean ownerOnly)</code>
<code>String</code>	<code>getName()</code>	<code>boolean</code>	<code>setLastModified(long time)</code>
<code>String</code>	<code>getParent()</code>	<code>boolean</code>	<code>setReadable(boolean readable)</code>
<code>File</code>	<code>getParentFile()</code>	<code>boolean</code>	<code>setReadable(boolean readable,</code>
<code>String</code>	<code>getPath()</code>		<code>                  boolean ownerOnly)</code>
<code>long</code>	<code>getTotalSpace()</code>	<code>boolean</code>	<code>setReadOnly()</code>
<code>long</code>	<code>getUsableSpace()</code>	<code>boolean</code>	<code>setWritable(boolean writable)</code>
<code>boolean</code>	<code>isAbsolute()</code>	<code>boolean</code>	<code>setWritable(boolean writable,</code>
<code>boolean</code>	<code>isDirectory()</code>		<code>                  boolean ownerOnly)</code>
<code>boolean</code>	<code>isFile()</code>	<code>Path</code>	<code>toPath()</code>
		<code>URI</code>	<code>toURI()</code>

# **Java New I/O**

# Java New I/O

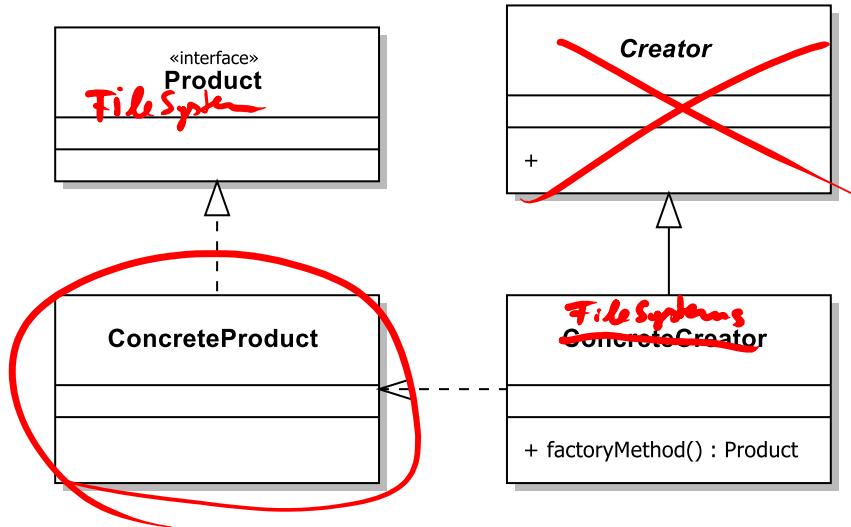
- Moderne API zur FileIO und File-Manipulation
- Wichtigste Klassen:
  - `FileSystem(s)`  
Wurzelverzeichnisse / Arten von Dateisystemen
  - `Path(s)`  
Pfade zu Verzeichnissen oder Dateien
  - `Files`  
`copy`, `create`, `move`, `delete`, `attributes`, `owner`, ...

# FactoryMethod-Pattern



- leichter austauschbare Implementierung
- sprechendere Namen für FactoryMethoden

# FactoryMethod-Pattern



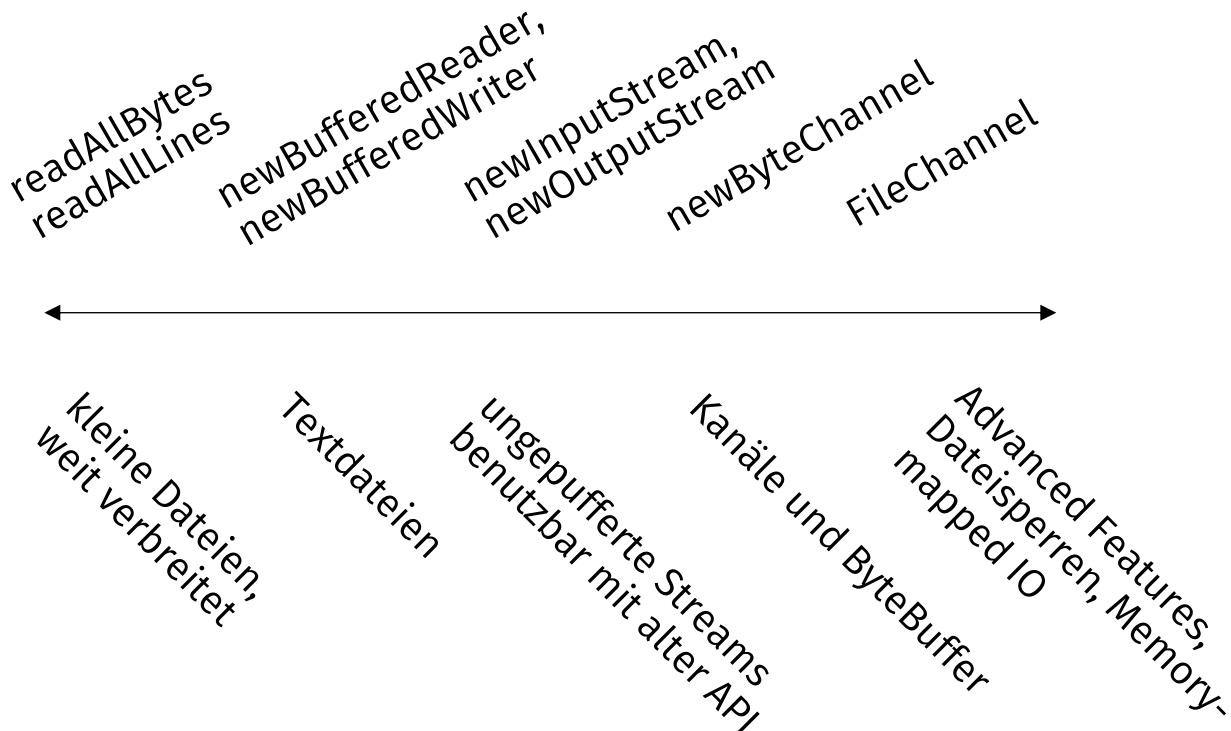
- leichter austauschbare Implementierung
- sprechendere Namen für FactoryMethoden

# Funktionales Java ab 1.8

Mehr später

- Jede Methode liefert das Objekt selbst als Ergebnis zurück → method-Chaining
  - kompakte Schreibweise ohne Zwischenvariablen
- Lambda-Expression als Parameter für filter und foreach-Funktion

# Java NIO

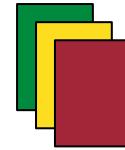


# Java IO vs. Java NIO

Java IO	Java NIO
Streams	Buffers
Blockierendes I/O	Nicht-blockierendes I/O
	Selektoren (warten auf mehrere Kanäle gleichzeitig)

# Einfaches Parsen von Dateien

# Scanner



- Werkzeuge der lexikalischen Analyse (Tokenizer, Lexer)
- Zerlegung von Fließtext in logische Einheiten (Tokens)

# formatierter Text

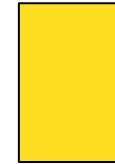
Klasse Scanner kann primitive Typen scannen:

```
Scanner(File source)
Scanner(File source, String charsetName)
Scanner(InputStream source)
Scanner(InputStream source, String charsetName)
Scanner(String source)
boolean hasNext()           //Standardtrenner: Whitespaces
boolean hasNextDouble()
boolean hasNextInt()
boolean hasNextBoolean()

...
double nextDouble(), int nextInt(), boolean nextBoolean()...
void useDelimiter(String regExp);
```

# reguläre Ausdrücke?

- ... habe ich schon 'mal gehört
- ... kenne ich so grob
- ... kenne ich gut



# reguläre Ausdrücke

# reguläre Ausdrücke

## Kurzübersicht

- a b c A @ Ø konkrete Zeichen
- . beliebiges Zeichen
- [a-zA-Z0-9-+\_ ] Zeichenauswahl
- (a)? 0- oder 1-mal pattern a
- (a)\* beliebig oft pattern a
- (a)+ mind. 1-mal
- \. . (Escape durch \)
- \\ \
- ^ Beginn
- \$ Ende

<https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/util/regex/Pattern.html>

Java Playground auf

<https://www.online-java.com/n4mNZ9sRMF>

- Murmelgruppe – 5 Minuten
- Bearbeiten Sie mit Ihrem Nachbarn (~2-3 Personen) folgende Aufgabe:
  - Definieren Sie einen regulären Ausdruck zur Erkennung einer gültigen E-Mail-Adresse.
- Im Plenum sammeln wir Ihre Vorschläge.

# reguläre Ausdrücke

weiterführende Aspekte:

- Capture groups (0 = gesamter Ausdruck)  
 $(\underset{1}{(}) \underset{2}{(} (\underset{3}{(}) \underset{4}{(}) \underset{5}{)}) \underset{6}{)}$
- Named capture groups  
 $(?<\text{name}>X)$  (name ist Name der Gruppe, X ist regulärer Ausdruck)
- Zugriff über  $\backslash n$  oder  $\backslash k<\text{name}>$  im gleichen Ausdruck
- genaue Anzahl von Zeichen:  
 $\{n\}$   $\{n,\}$   $\{n,k\}$

Java Playground auf

<https://www.online-java.com/n4mNZ9sRMF>

- Murmelgruppe – 5 Minuten
- Bearbeiten Sie mit Ihrem Nachbarn (~2-3 Personen) folgende Aufgabe:
  - Ändern Sie den Ausdruck der vorherigen Aufgabe so, dass auf den gematchten Hostnamen direkt zugegriffen werden kann und nur kurze (zwischen 2 und 6 Zeichen) top level domains erlaubt sind.
  - Im Plenum sammeln wir Ihre Vorschläge.

# Ausflug: Injection Attacks

- Prüfen Sie **\*immer\*** bei jeder Eingabemöglichkeit, ob die Daten ihrem erlaubten Format entsprechen (whitelisting → RegExp)!
- Wenden Sie Output-Sanitization+Escaping bei der Ausgabe an.
- Nutzen Sie wenn möglich bereits existierende Funktionalität, das für Sie zu tun.
- Mehr z.B.:
  - [https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Injection\\_Prevention\\_in\\_Java\\_Cheat\\_Sheet.md](https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Injection_Prevention_in_Java_Cheat_Sheet.md)
  - [https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input\\_Validation\\_Cheat\\_Sheet.md](https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md)
  - Vorlesung von Prof. Kargl: Sicherheit in IT-Systemen

# Lernziele

- Weitere Aspekte von Dateien
- Java New I/O
- Einfaches Parsen