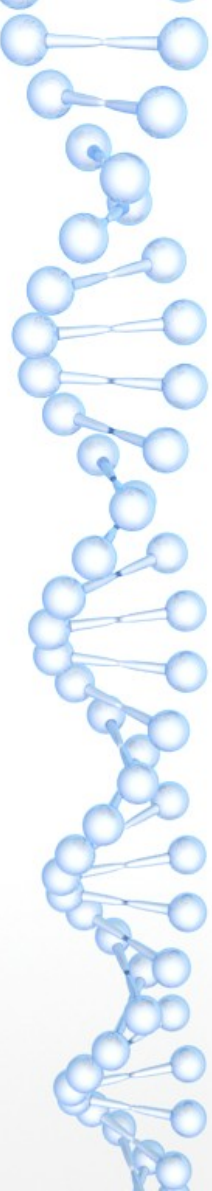
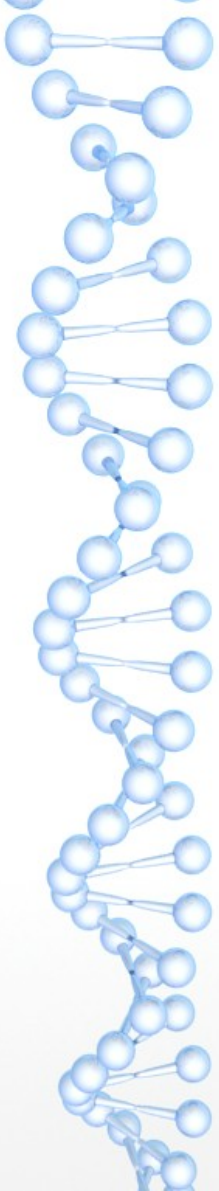




File, IOStream, Decorator, ZipFile

Gliederung

- 
- File
 - Exkurs URI
 - InputStream
 - Decorator
 - ZipFile



File



File-Allgemein

- 4 Konstruktoren
 - `File(String pathname)`, `File(URI uri)`,
`File (File parent, String child)`
`File(String parent, String child)`
- Methoden zum Umgang mit Dateien und Verzeichnissen



Exkurs URI

- URI = Uniform Resource Identifier
- URL = Uniform Resource Locator
- Anwendung:
`<scheme>:<authority></path><?query><#fragment>`



Exkurs URI

- `<scheme>`
- `<//authority>`
- `</path>`
- `<?query>`
- `<#fragment>`



Exkurs URI

- `<scheme>`
 - gibt Zugriffart an
 - Beispiele: ftp, http, file
- `<authority>`
- `</path>`
- `<?query>`
- `<#fragment>`



Exkurs URI

- `<scheme>`
- `<//authority>`
 - Optional
 - User, Passwort, Host, Port
 - Beispiel: `user:password@host:port`
- `</path>`
- `<?query>`
- `<#fragment>`



Exkurs URI

- `<scheme>`
- `<authority>`
- `</path>`
 - Hierarchischer Pfad
 - Beispiel: `/home/username/documents`
- `<?query>`
- `<#fragment>`



Exkurs URI

- `<scheme>`
- `<||authority>`
- `</path>`
- `<?query>`
 - Abfrage, falls Angabe des Pfads nicht reicht
 - Beispiel: `name=someName`
- `<#fragment>`



Exkurs URI

- `<scheme>`
- `<authority>`
- `<path>`
- `<?query>`
- `<#fragment>`
 - Referenziert eine Stelle in der Ressource



Exkurs URI

- Beispiele:
 - `file:///home/username/documents/sometext.txt`
 - `mailto:max.musterman@beispiel.de`
 - `ssh://admin:admin@192.10.0.9:22`
 - `https://de.wikipedia.org/wiki/Uniform_Resource_Identifier`



File-Methoden

- erstellen, löschen, verschieben (boolean)
- Abfrage ob Objekt ein Verzeichnis oder Datei (boolean)
- Existenz, Sichtbarkeit von File-Objekten (boolean)
- Zeit wann File-Objekt zuletzt geändert wurde (long)
- ändern des Namens (boolean)
- ausführbar, lesbar, schreibgeschützt machen (boolean)



Beispiel

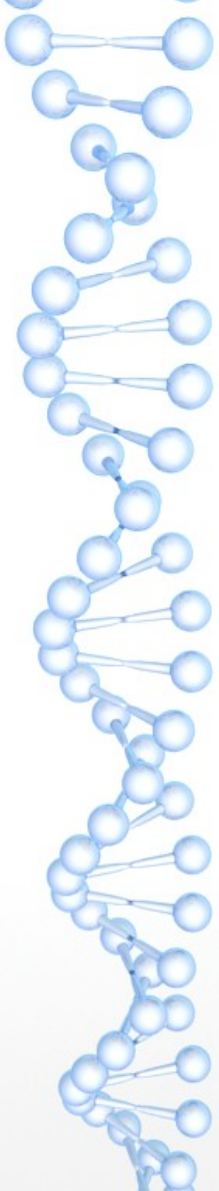
```
File file = new File(new URI(uri));

if (file.mkdirs()) {
    System.out.println("Ordner erstellt");
} else {
    System.out.println("Erstellen nicht möglich");
}

File testFile = new File(file, "test.txt");
if (testFile.createNewFile()) {
    System.out.println("Datei erstellt");
} else {
    System.out.println("Erstellen nicht möglich");
}
```



Input/Output Streams



Basisklasse (abstrakt)	Bytes/Byte- Arrays	Zeichen/ ZeichenArrays
Eingabe	InputStream	Reader
Ausgabe	OutputStream	Writer



OutputStream

- abstrakt: `write(int b)`
- `write(byte[] b)`, `write(byte[] b, int off, int len)`
- `close()`, `flush()`
- nur Bytes geschrieben



InputStream

- abstrakt: read()
- available(), read(byte[] b), read(byte[] b, int off, int len), skip(long n)
- markSupported(), mark(int readlimit), reset()
- close()



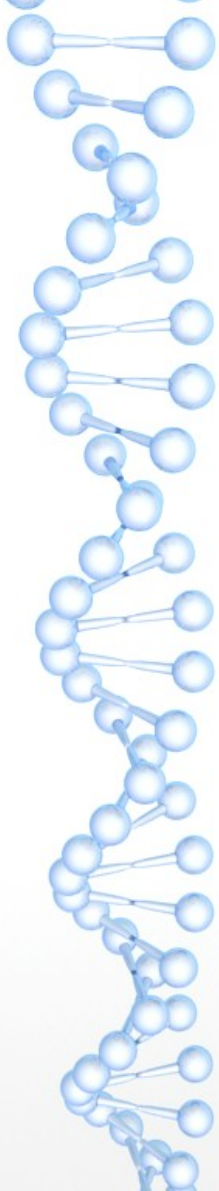
Writer

- abstrakt: `write(char[] cbuf, int off, int len)`, `flush()`, `close()`
- `write(int c)`, `write(String str)`,
`write(String str, int off, int len)`, `append(char c)`,
`append(CharSequence csq)`



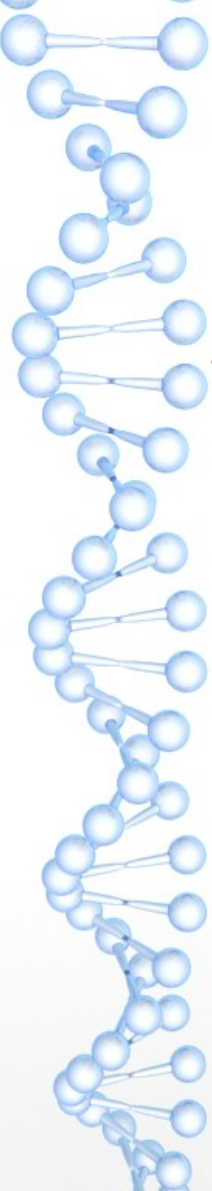
Reader

- abstrakt: `read(char[] cbuf, int off, int len)`, `close()`
- `read(CharBuffer target)`, `read()`, `read(char[] cbuf)`



Ressource	Zeichenorientiert	Byteorientiert
Datei	FileReader FileWriter	FileInputStream FileOutputStream
Hauptspeicher	CharArrayReader CharArrayWriter StringReader StringWriter	ByteArrayInputStream ByteArrayOutputStream
Pipe	PipeReader PipeWriter	PipeInputStream PipeOutputStream

Beispiel



```
try (Reader reader = new FileReader(filename)) {  
    while (reader.ready()) {  
        System.out.print( (char) reader.read());  
    }  
} catch (IOException e) {  
    e.printStackTrace();  
}
```



Streams filtern/verketteten

Eingabe	Ausgabe	Anwendung
BufferedInputStream	X	Daten puffern (Zwischenspeicher)
BufferedReader	X	
CheckedInputStream	X	Checksumme berechnen
DataInputStream	X	primitive Datentypen holen/schreiben
DigestInputStream	X	Digest (Checksumme) mitberechnen
InflaterInputStream	X	Kompression von Daten
LineNumberInputStream		Mitzählen von Zeilen
LineNumberReader		
PushbackInputStream		Daten in den Lesestrom zurückgeben
PushbackReader		
CipherInputStream	X	Daten verschlüsseln/entschlüsseln



Beispiel

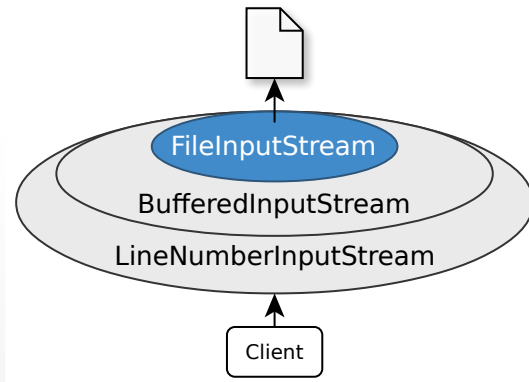
```
try (Reader reader = new FileReader(filename);  
    BufferedReader bufferedReader = new BufferedReader(reader)) {  
    String lyric;  
    while ((lyric = bufferedReader.readLine()) != null) {  
        System.out.println(lyric);  
    }  
} catch (IOException e) {  
    e.printStackTrace();  
}
```




Dekorator

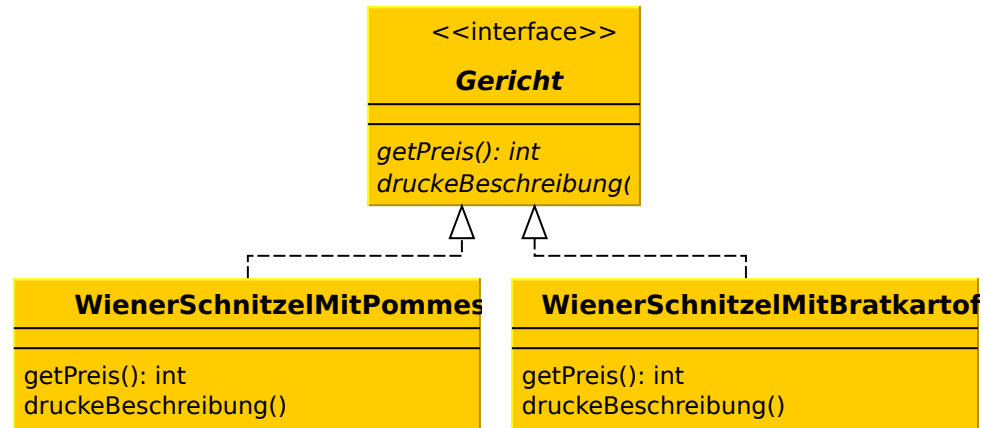
Dekorator

- component, decorator
- verleiht/entfernt zusätzliche Funktionalität
- Mögliche Nutzung mehrerer decorators
 - Beispiel:



Speisekarte Beispiel

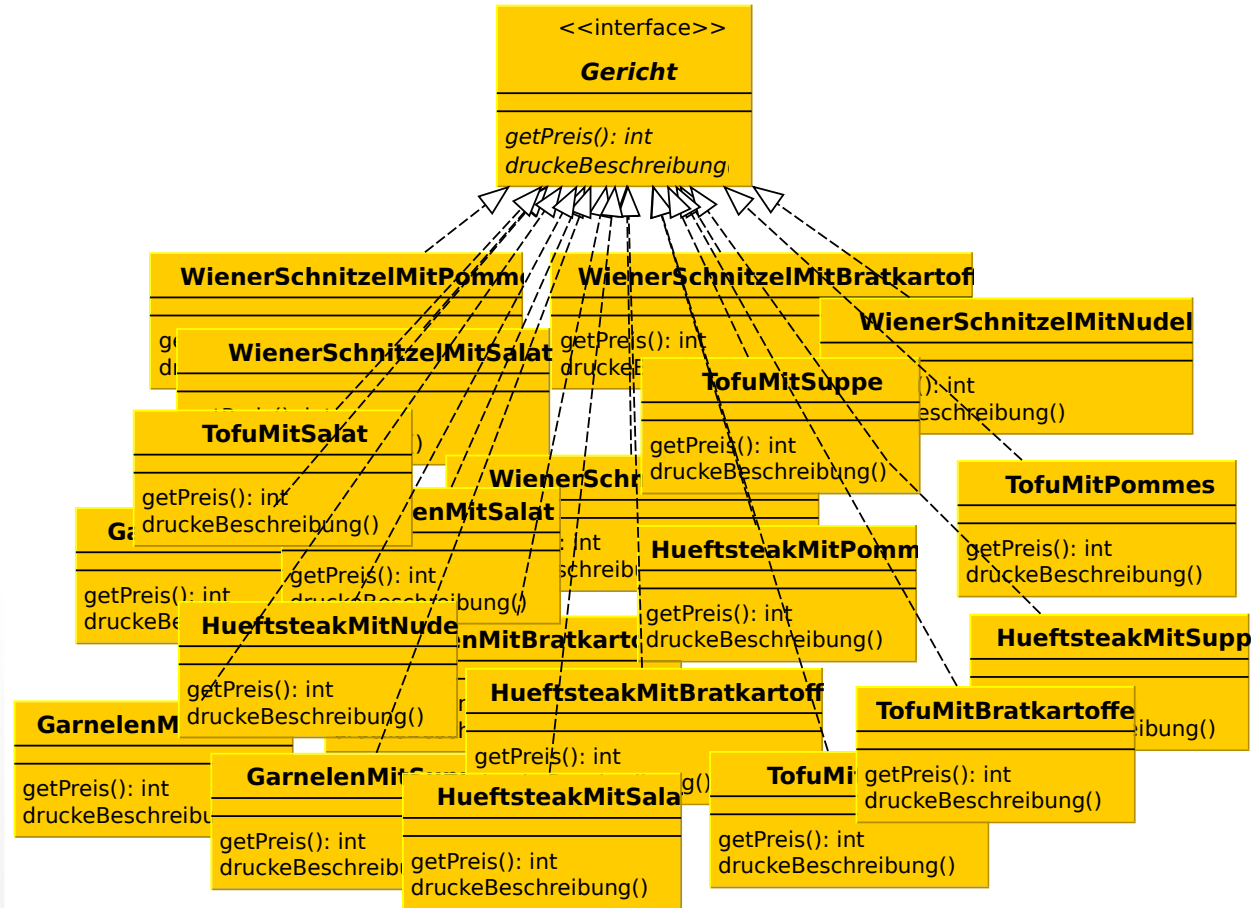
- Modellierung der zwei beliebtesten Gerichte





Speisekarte Beispiel

- Modellierung der zwei beliebtesten Gerichte
- Änderung der Anforderung:
 - Alle Gerichte modellieren
 - 4 Basisgerichte
 - 5 Beilagen
- 20 Gerichte/Klassen



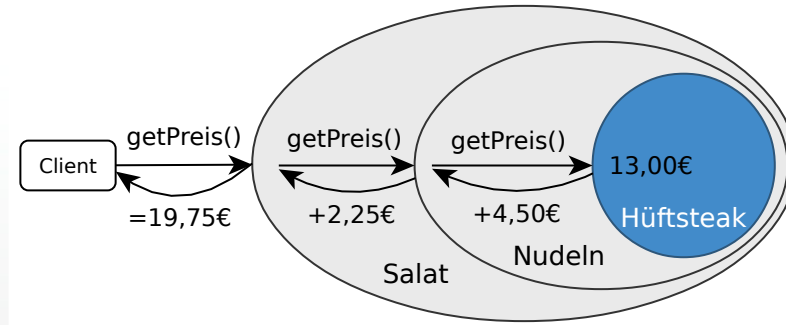


Speisekarte Beispiel

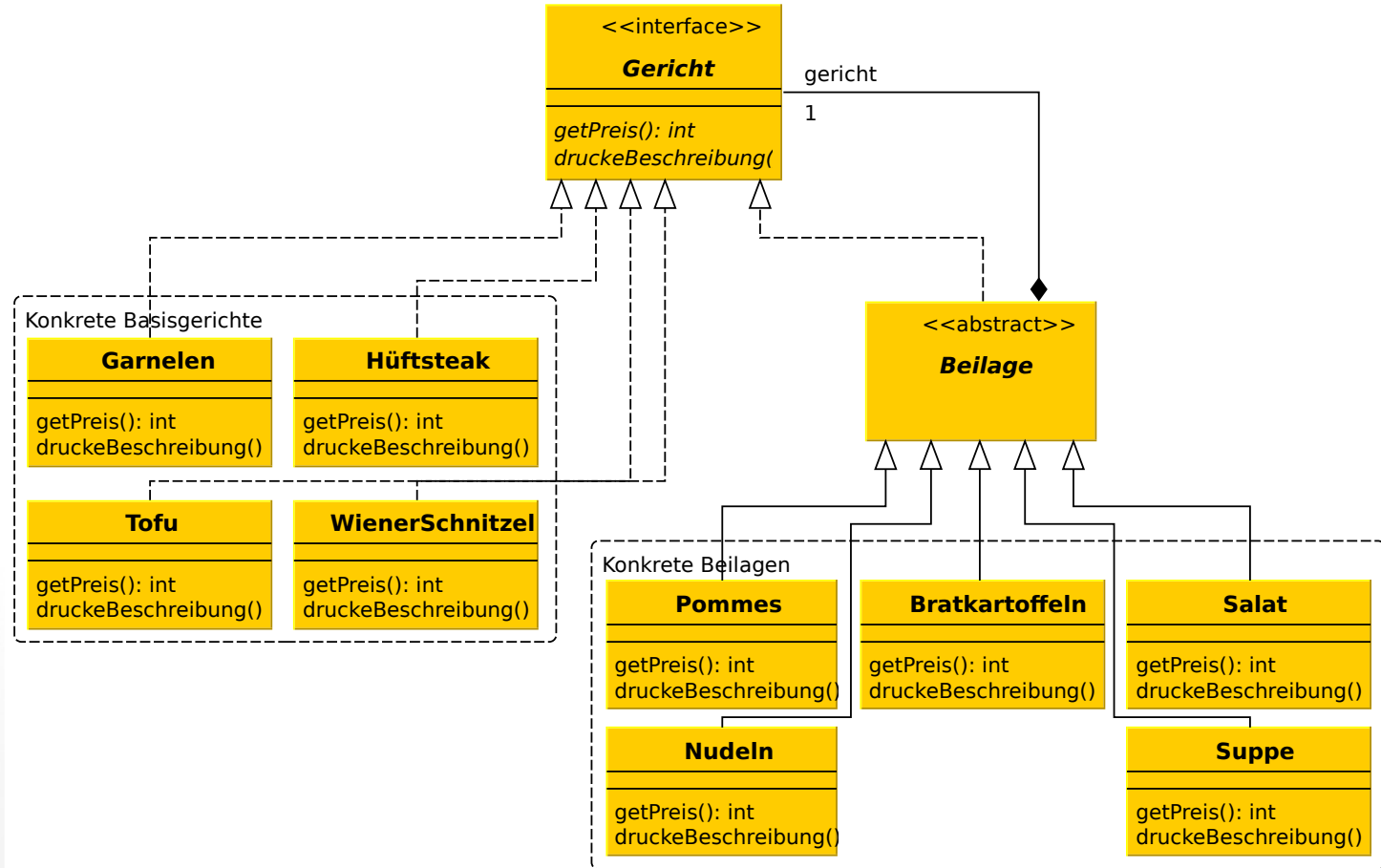
- Was ist konstant?
 - Basisgericht
 - Beilage
- Was ist variabel?
 - Kombination aus Basisgericht und Beilage

Speisekarte Beispiel

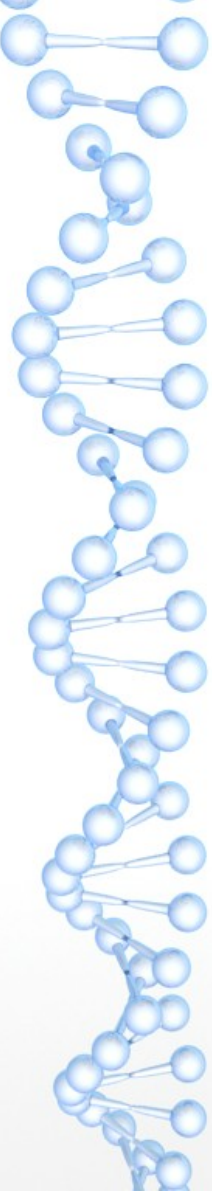
- Basisgerichte, Beilagen als Klassen
- Klassen dynamisch kombinieren
- Beilagen als “Wrapper”



Speisekarte Beispiel



Speisekarte Beispiel



```
import Decorator.Tofu;
import Decorator.Salat;
import Decorator.Spaetzle;
import Decorator.Gericht;

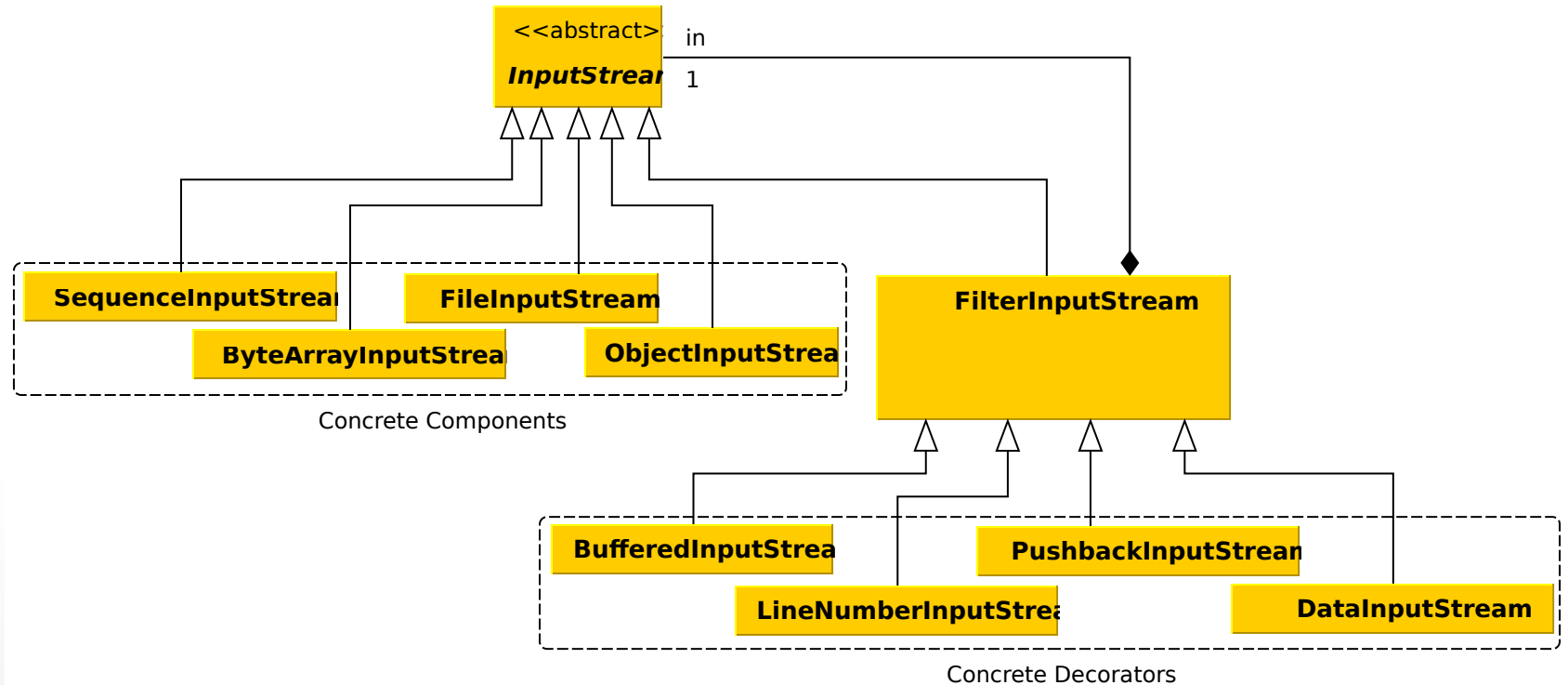
public class SpeisekarteMain {

    public static void main(String[] args) {
        Gericht gericht = new Salat(new Tofu());
        gericht.druckeBeschreibung();
        System.out.printf(" für %.2f Euro", gericht.getPreis());
        System.out.println();
        gericht = new Spaetzle(gericht);
        gericht.druckeBeschreibung();
        System.out.printf(" für %.2f Euro", gericht.getPreis());
    }
}
```

Tofu, Salat für 8,00 Euro

Tofu, Salat, Spätzle für 11,50 Euro

Decorator InputStream





Zip-File



ZipFile

- zum lesen von zip-Archiven
- Dateien und Ordner werden durch ZipEntry dargestellt
- Methoden für Datei-/Ordnerattributen
- Vorsicht vor Zip Slip Attack



ZipFile

- 6 Konstruktoren

- ZipFile(File file), ZipFile(File file, int mode), ZipFile(File file, Charset charset), ZipFile(File file, int mode, Charset charset)
- ZipFile(String name), ZipFile(String name, Charset charset)
- Default charset = UTF-8



ZipFile

- Methoden:

- close() schließt ZipFile
- entries() gibt die entahltenen Dateien als Enumeration zurück
- getComment() gibt Zip File comment zurück
- getEntry(String name) gibt Datei "name" oder "null" zurück
- getInputStream(ZipEntry entry) gibt Datenstrom zum lesen der Inhalte von Entry zurück
- getName() gibt den Pfd vom ZipFile zurück
- size() Anzahl der Inhalte
- stream()

ZipFile-Beispiele

1. Beispiel

```
ZipFile zipFile = new ZipFile(filepath);  
for (Enumeration<? extends ZipEntry> e = zipFile.entries(); e.hasMoreElements(); ) {  
    ZipEntry zipEntry = e.nextElement();  
    System.out.println(zipEntry.getName());  
}
```

2. Beispiel:

```
ZipEntry lyrics = zipFile.getEntry("IO Streams/lyrics.txt");  
InputStream zis = zipFile.getInputStream(lyrics);  
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(zis));  
String line;  
while ((line=bufferedReader.readLine()) != null){  
    System.out.println(line);  
}
```



Quellen

- https://openbook.rheinwerk-verlag.de/javainsel/18_005.html#u18.5.3
- <https://www.philippbauer.de/study/se/design-pattern/decorator.php>
- <https://docs.oracle.com/javase/7/docs/api/java/io/File.html>
- <https://docs.oracle.com/javase/8/docs/api/index.html?java/util/zip/ZipFile.html>
- <https://docs.oracle.com/javase/7/docs/api/java/net/URI.html>
- <https://java-tutorial.org/file.html>
- <https://www.geeksforgeeks.org/>