ulm university universität ulm

**Prüfungsklausur zur Vorlesung**

# Programmierung von Systemen

**im Sommersemester 2020**

**Institut für Softwaretechnik und Programmiersprachen**
**Prof. Dr. Matthias Tichy, Stefan Götz 30. September 2020, 8 Uhr**

**Bearbeitungszeit: 90 Minuten**

| Nachname: | Vorname: | Matrikelnummer: |
|---|---|---|
| Studiengang und Abschluss: | | |
| TischNr: | | |

**Hiermit erkläre ich, dass ich prüfungsfähig bin.**
**Sollte ich nicht auf der Liste der angemeldeten Studierenden aufgeführt sein, dann nehme ich hiermit zur Kenntnis, dass diese Prüfung nicht gewertet werden wird.**

_____

**Datum, Unterschrift des Prüfungsteilnehmers**

## Zur allgemeinen Beachtung:

- Zur Bearbeitung der Klausur stehen **90 Minuten** zur Verfügung.
- Es sind keine zusätzlichen Hilfsmittel zugelassen.
- Sie können Ihre Lösungen **direkt auf die Aufgabenblätter** schreiben.
- Bei Erklärungsaufgaben werden ganze Sätze und nicht nur Stichpunkte erwartet.
- Falls Sie zusätzlich Papier benötigen:
  - Papier wird Ihnen zur Verfügung gestellt.
  - Schreiben Sie Ihren **Namen** und Ihre **Matrikelnummer** auf **jedes zusätzliche Blatt**!

## Viel Erfolg!

| Aufgabe | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Σ |
|---|---|---|---|---|---|---|---|---|
| Max. | 14 | 10 | 11 | 10 | 11 | 13 | 10 | 79 |
| Punkte | | | | | | | | |
| Hz. | | | | | | | | |

| **Note** | **Unterschrift Prüfer** |
|---|---|
| | |

## Aufgabe 1 - OOP (7 + 4 + 3 = 14 Punkte)

Gegeben sei das Klassendiagramm, welchen Sie auf dem beiliegenden Extrablatt finden können. Das Interface `Openable` definiert die Möglichkeit einen Ort zu öffnen (`open`) oder zu schließen (`close`).

**a)** Implementieren Sie die Methoden `move(Stable host)` der Klasse `Horse` und `checkAddable()` der Klasse Stable. Mittels der Methode `checkAddable()` in `Stable` kann geprüft werden ob der Stall geöffnet ist und ob noch Platz darin frei ist. Die Methode `move(Stable host)` soll ein Pferd in einen neuen Stall stellen insofern dieser offen ist und Platz hat. `move()` soll *true* bzw. *false* zurück geben abhängig davon ob das Pferd in einen neuen Stall bewegt werden konnte oder nicht. *Hinweis: Beachten Sie, dass Sie auch alle Referenzen updaten müssen wenn ein Pferd in einen neuen Stall bewegt wird.*

```
public abstract class Horse {
    private Stable pos;

    public boolean move(Stable host) {









    }
}
```

```
//continues on next page!

public class Stable {
    private Horse[] holds = new Horse[20];
    private boolean open;

    public boolean checkAddable() {









    }

    public void open() {open=true;}
    public void close() {open=false;}

    public void addHorse(Horse h) {
        for (int i = 0; i < holds.length; i++) {
            if (holds[i] == null) {
                holds[i] = h;
                break;
            }
        }
    }

    public void removeHorse(Horse h) {
        for (int i = 0; i < holds.length; i++) {
            if (holds[i] == h) {
                holds[i] = null;
                break;
            }
        }
    }
}
```

**b)** Erläutern Sie was das Singleton-Pattern ist, wofür es eingesetzt werden kann und Erläutern Sie (ohne Code) wie es in Java umgesetzt werden kann.

**c)** Welche der folgenden Codestücke sind korrekt und welche führen zu **Compile**- oder **Laufzeitfehlern**? Begründen Sie, warum ein Codestück nicht korrekt ist. Sie können davon ausgehen, dass die Aufrufe aus einer `main`-Methode in einem anderen *Package* gemacht werden welches die nötigen imports vorweist.

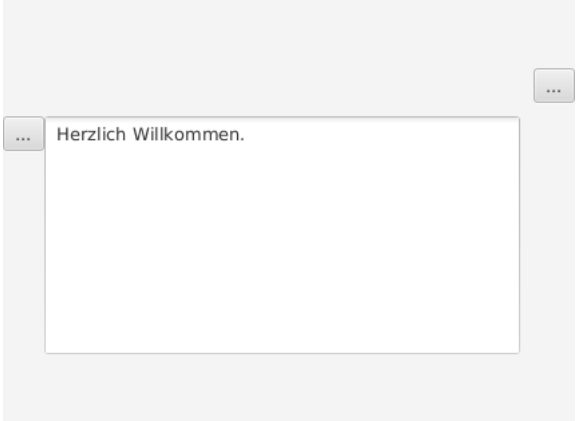
```
var stable = new Stable();
stable.addHorse(new Horse());
```

```
Horse h = new Pony();
Openable hos = new Stable();
h.move(hos);
```

```
Stable stable = new Stable();
Openable h  = stable;
h.close();
```

## Aufgabe 2 - JavaFX (6 + 3 + 1 = 10 Punkte)

Gegeben Sei die folgende Grafik:




**a)** Ergänzen Sie die `start`- Methode so, dass die oben gezeigte Grafik daraus resultiert. Beachten Sie hierfür auch den JavaFX Teil des **CheatSheets** am Ende der Klausur.

```
@Override
public void start(Stage stage) {

}
```

**b)** Schreiben Sie Code damit beim Drücken eines der zwei Buttons der gedrückte Button verschwindet.

**c)** Geben Sie CSS Code an um die Hintergrundfarbe aller Buttons **rot** zu färben.

## Aufgabe 3 - Java IO (7 + 4 = 11 Punkte)

**a)** Implementieren Sie eine Methode welche die ersten 20 Bytes von allen übergebenen Channels nacheinander in die Datei **out.data** schreibt. Verwenden Sie hierfür ausschließlich Methoden aus der Streams-API für Java Collections KEINE manuell implementierten Schleifen.

Achten Sie auf Behandlung der folgenden Exception mit einer passenden Meldung:

- `IOException`

Beachten Sie hierfür auch den JavaIO Teil des **CheatSheets** am Ende der Klausur.

```
public static void useBytes(FileChannel[] ics) {









}
```

**b)** Java bietet eine Vielzahl an Streams für die Verarbeitung von Daten an. Darunter auch gepufferte und nicht gepufferte Varianten. Erläutern Sie *textuell* anhand von `BufferedOutputStream` und `OutputStream` die Unterschiede zwischen gepufferten und nicht gepufferten Streams. Erläutern Sie zusätzlich mit Bezug auf `OutputStream` wie das Decorator-Pattern Anwendung findet.
*Hinweis: Beachten Sie auch den Dateiein- und -ausgabe Teil des* ***CheatSheets*** *am Ende der Klausur.*

## Aufgabe 4 - XML (2 + 8 = 10 Punkte)

Betrachten Sie das folgende XML-Dokument:

```
1  <xs:schema attributeFormDefault="unqualified"
2   elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3    <xs:element name="Horse">
4      <xs:complexType>
5        <xs:simpleContent>
6          <xs:extension base="xs:string">
7            <xs:attribute type="xs:byte" name="maxSpeed" use="optional"/>
8          </xs:extension>
9        </xs:simpleContent>
10     </xs:complexType>
11   </xs:element>
12   <xs:element name="Location" type="xs:string"/>
13   </xs:element name=Stable>
14     <xs:complexType>
15       <xs:sequence>
16         <xs:element ref="Horse" maxOccurs="unbounded" minOccurs="0"/>
17       </xs:Sequence>
18       <xs:attribute type="xs:byte" name=id/>
19     </xs:complexType>
20   </xs:element>
21   <xs:element name="Steadings">
22     <xs:complexType mixed="true">
23       <xs:sequence>
24         <xs:element ref="Location" minOccurs="1" maxOccurs="1">
25         <xs:element ref="Stable" minOccurs="0"/>
26       </xs:sequence>
27     </xs:complexType>
28   </xs:element>
29 </xs:schema>
```

**a)** Das XML Dokument ist nicht wohlgeformt. Geben Sie die Zeilennummern von **vier** der **fünf** vorkommenden Fehler an und erklären Sie wie diese korrigiert werden können. Konformität mit dem Schema für XSDs muss dabei nicht berücksichtigt werden.

**b)** Die obige XML-Datei beschreibt ein XML-Schema. Schreiben Sie eine DTD-Datei so, dass diese Dokumente von **selber Struktur** validiert. Markieren Sie in ihrer DTD die Stellen an denen dies nicht möglich ist und erklären Sie warum dies nicht möglich ist.

## Aufgabe 5 - ER-Modellierung (1.5 + 6 + 3.5 = 11 Punkte)

Ein Gestüt besteht aus einer Menge an Pferdeställen aber mindestens einem. Sowohl Pferdeställe als auch Gestüte befinden sich an einem bestimmten Ort, welcher eine PLZ hat. Pferde können nur genau einem Gestüt aber mehreren Ställen auf diesem Gestüt zugehören. Ein Stall kann maximal 30 Pferde halten.

**a)** Gibt es eine Anforderung aus dem obigen Sachverhalt die nicht dem **relationalen Datenbankschema** modelliert werden kann. Wenn ja, wie könnte diese stattdessen in der Software umgesetzt werden?

**b)** Modellieren Sie den beschriebenen Sachverhalt als **E-R-Diagramm**.

**c)** Erstellen Sie zu dem E-R-Diagramm das dazugehörige **relationale Datenbankschema**. Achten Sie hierbei auf sinnvoll gewählte **Primärschlüssel** und stellen Sie sicher, dass das Schema mindestens in 3. Normalform ist. Sie können bei Bedarf auch IDs einführen, um die Eindeutigkeit eines Primärschlüssels zu gewährleisten.

## Aufgabe 6 - SQL (1.5 + 3.5 + 4 + 4 = 13 Punkte)

Gegeben seien die folgenden Relationen-schemata (auch zu finden auf dem beiliegenden Extrablatt):

| Schiffe | | |
|---|---|---|
| SID | Name | **MID** |

| Rennen | | | |
|---|---|---|---|
| RID | RennName | Startzeit | Preisgeld |

| TeilnehmendeSchiffe | |
|---|---|
| **SID** | **RID** |

| Matrosen | |
|---|---|
| MID | MName |

| Besatzung | |
|---|---|
| **MID** | **SID** |

Primärschlüssel sind unterstrichen, Fremdschlüssel sind **fett** dargestellt. Formulieren Sie, insofern nicht anders spezifiziert, die **SQL-Statements** zur Lösung folgender Teilaufgaben:

**a)** Fügen Sie den **neuen** Matrosen ‘Matrose Stefan‘ mit der ID ‘42‘ in die Matrosentabelle ein.

**b)** Geben Sie an, wie viele Matrosen existieren, deren Namen an 2. Stelle ein **e** enthält.

**c)** Geben Sie die Namen aller Matrosen aus, die auf mehr als 3 Schiffen arbeiten. Besatzungen werden mittels der Tabelle Besatzungen die zwischen Schiffe und Matrosen Tabelle gestellt ist definiert.

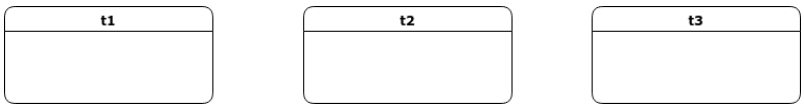
**d)** Geben Sie die Namen aller der Schiffe aus, deren Besitzer Teil der Besatzung der 'Titanic III' ist. *Hinweis: Besitzer sind als MID in den Schiffen vermerkt!*

## Aufgabe 7 - Threads (2 + 6 + 2 = 10 Punkte)

**a)** Betrachten Sie den folgenden Code:

```
1   public class StateManager {
2       public static void main(String... args) throws Exception{
3           Thread t1  = new S1();
4           Thread t2  = new S2();
5           Thread t3  = new S3();
6           t1.start();
7           t1.join();
8           t2.start();
9           t3.start();
10          t3.join();
11          t2.join();
12          System.out.println("end");
13      }
14  }
```

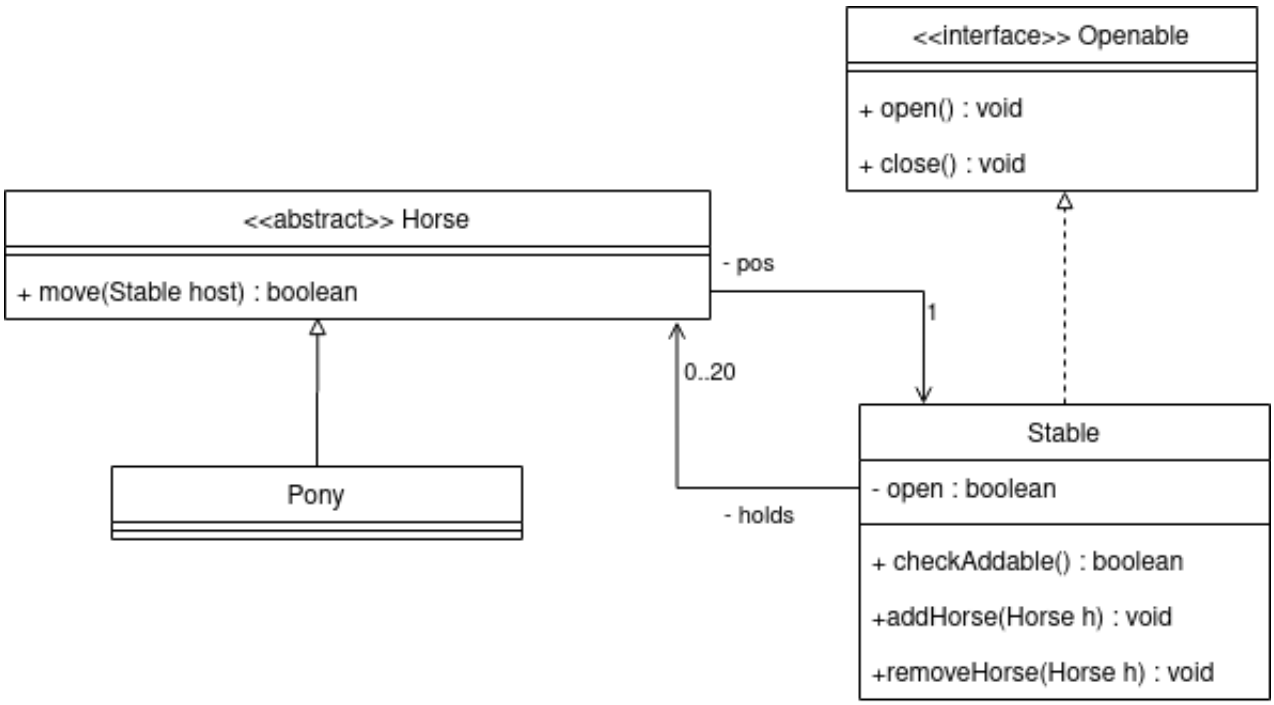
Ergänzen Sie den untenstehenden UML-Zustandsautomat mit sinnvollen Zustandsübergängen, Start- und Endmarkierungen so wie gegebenenfalls forks und joins um den im oben abgebildeten Code dargestellten Sachverhalt abzubilden.



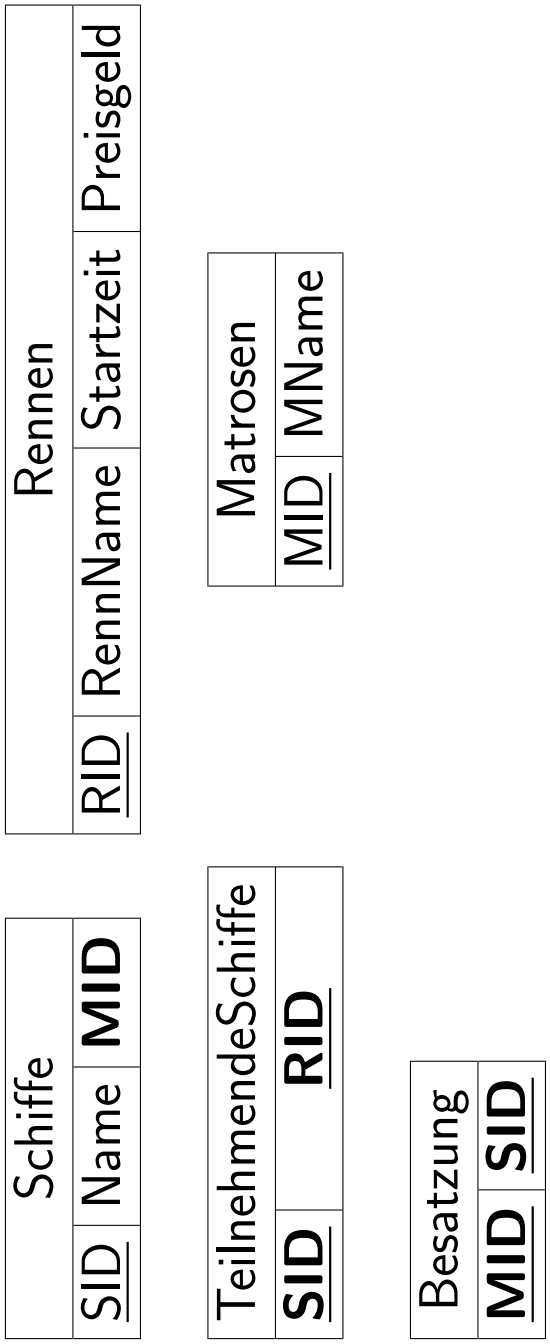
**b)** Erläutern Sie anhand eines selbst gewählten Code-Beispiels wie **Deadlocks** entstehen. Gehen Sie in Ihrer Erklärung auch darauf ein warum die Entstehung eines Deadlocks nicht zwingend deterministisch ist.

**c)** Erläutern Sie wie mittels einer `BlockingQueue` mehrere Threads gemeinsam an einer Aufgabe arbeiten können. Erklären Sie zusätzlich wie dies mittels `Futures` ebenfalls erreicht werden kann.

# Aufgabe 1 - OOP



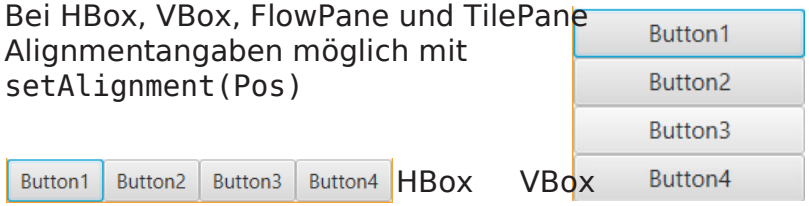
# Aufgabe 6 - SQL

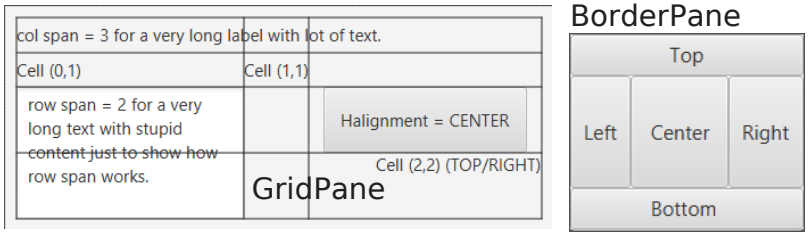


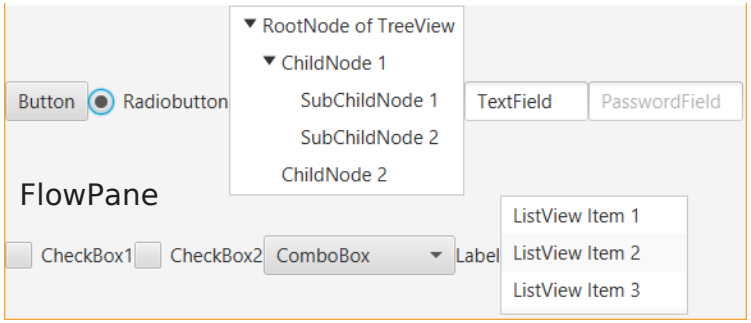
# Aufgabe2 - JavaFX

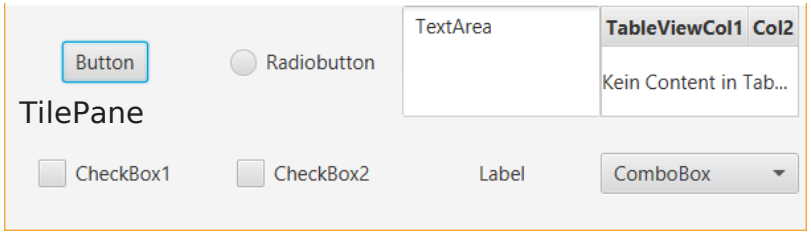
## einige LayoutManager

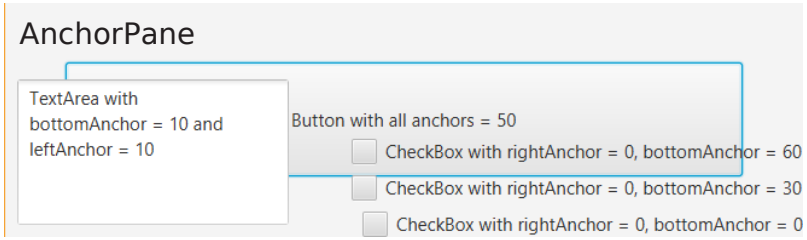
Bei HBox, VBox, FlowPane und TilePane Alignmentangaben möglich mit `setAlignment(Pos)`



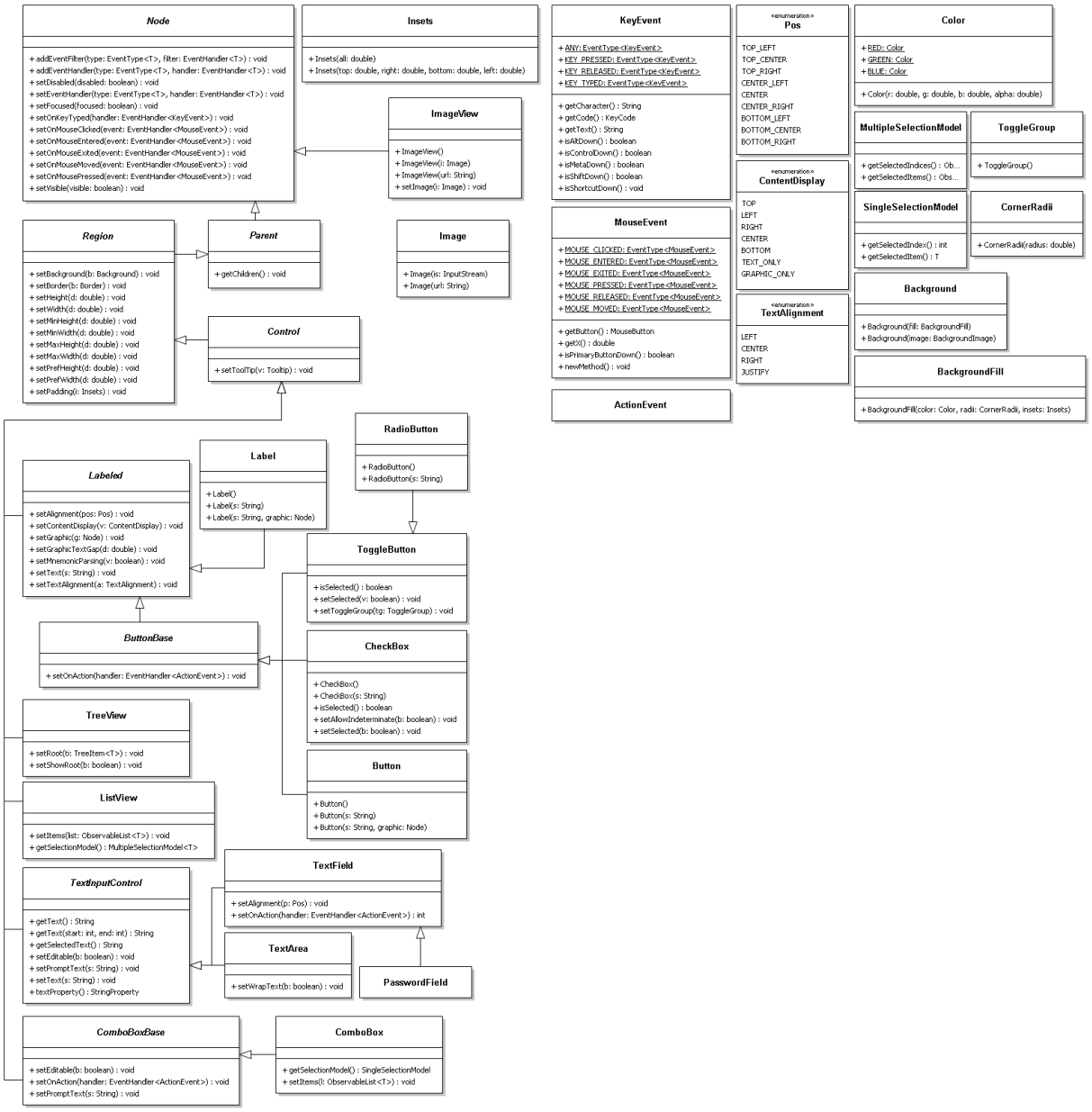




## Auszug aus der Klassenhierarchie



# Aufgabe 3 - JavaIO

## Java.io.FileOutputStream

| Modifier and Type | Method and Description |
| --- | --- |
| void | **close**()<br>Closes this file output stream and releases any system resources associated with this stream. |
| protected void | **finalize**()<br>Cleans up the connection to the file, and ensures that the `close` method of this file output stream is called when there are no more references to this stream. |
| **FileChannel** | **getChannel**()<br>Returns the unique **FileChannel** object associated with this file output stream. |
| **FileDescriptor** | **getFD**()<br>Returns the file descriptor associated with this stream. |
| void | **write**(byte[] b)<br>Writes `b.length` bytes from the specified byte array to this file output stream. |
| void | **write**(byte[] b, int off, int len)<br>Writes `len` bytes from the specified byte array starting at offset `off` to this file output stream. |
| void | **write**(int b)<br>Writes the specified byte to this file output stream. |

# Aufgabe 3 - JavaIO

## Java.nio.channels.FileChannel

| Modifier and Type | Method and Description |
|---|---|
| abstract MappedByteBuffer | map(FileChannel.MapMode mode, long position, long size)<br>Maps a region of this channel's file directly into memory. |
| static FileChannel | open(Path path, OpenOption... options)<br>Opens or creates a file, returning a file channel to access the file. |
| static FileChannel | open(Path path, Set<? extends OpenOption> options, FileAttribute<?>... attrs)<br>Opens or creates a file, returning a file channel to access the file. |
| abstract long | position()<br>Returns this channel's file position. |
| abstract FileChannel | position(long newPosition)<br>Sets this channel's file position. |
| abstract int | read(ByteBuffer dst)<br>Reads a sequence of bytes from this channel into the given buffer. |
| long | read(ByteBuffer[] dsts)<br>Reads a sequence of bytes from this channel into the given buffers. |
| abstract long | read(ByteBuffer[] dsts, int offset, int length)<br>Reads a sequence of bytes from this channel into a subsequence of the given buffers. |
| abstract int | read(ByteBuffer dst, long position)<br>Reads a sequence of bytes from this channel into the given buffer, starting at the given file position. |
| abstract long | size()<br>Returns the current size of this channel's file. |
| abstract long | transferFrom(ReadableByteChannel src, long position, long count)<br>Transfers bytes into this channel's file from the given readable byte channel. |
| abstract long | transferTo(long position, long count, WritableByteChannel target)<br>Transfers bytes from this channel's file to the given writable byte channel. |
| abstract FileChannel | truncate(long size)<br>Truncates this channel's file to the given size. |
| FileLock | tryLock()<br>Attempts to acquire an exclusive lock on this channel's file. |
| abstract FileLock | tryLock(long position, long size, boolean shared)<br>Attempts to acquire a lock on the given region of this channel's file. |
| abstract int | write(ByteBuffer src)<br>Writes a sequence of bytes to this channel from the given buffer. |
| long | write(ByteBuffer[] srcs)<br>Writes a sequence of bytes to this channel from the given buffers. |
| abstract long | write(ByteBuffer[] srcs, int offset, int length)<br>Writes a sequence of bytes to this channel from a subsequence of the given buffers. |
| abstract int | write(ByteBuffer src, long position)<br>Writes a sequence of bytes to this channel from the given buffer, starting at the given file position. |

# Aufgabe 3 - JavaIO

## Java.util.stream

| Modifier and Type | Method and Description |
|---|---|
| boolean | **allMatch**(**Predicate**<? super **T**> predicate)<br>Returns whether all elements of this stream match the provided predicate. |
| boolean | **anyMatch**(**Predicate**<? super **T**> predicate)<br>Returns whether any elements of this stream match the provided predicate. |
| static <T> **Stream.Builder**<T> | **builder**()<br>Returns a builder for a Stream. |
| <R,A> R | **collect**(**Collector**<? super **T**,A,R> collector)<br>Performs a **mutable reduction** operation on the elements of this stream using a Collector. |
| <R> R | **collect**(**Supplier**<R> supplier, **BiConsumer**<R,? super **T**> accumulator, **BiConsumer**<R,R> combiner)<br>Performs a **mutable reduction** operation on the elements of this stream. |
| static <T> **Stream**<T> | **concat**(**Stream**<? extends T> a, **Stream**<? extends T> b)<br>Creates a lazily concatenated stream whose elements are all the elements of the first stream followed by all the elements of the second stream. |
| long | **count**()<br>Returns the count of elements in this stream. |
| **Stream**<T> | **distinct**()<br>Returns a stream consisting of the distinct elements (according to **Object.equals(Object)**) of this stream. |
| static <T> **Stream**<T> | **empty**()<br>Returns an empty sequential Stream. |
| **Stream**<T> | **filter**(**Predicate**<? super **T**> predicate)<br>Returns a stream consisting of the elements of this stream that match the given predicate. |
| **Optional**<T> | **findAny**()<br>Returns an **Optional** describing some element of the stream, or an empty Optional if the stream is empty. |
| **Optional**<T> | **findFirst**()<br>Returns an **Optional** describing the first element of this stream, or an empty Optional if the stream is empty. |
| <R> **Stream**<R> | **flatMap**(**Function**<? super **T**,? extends **Stream**<? extends R>> mapper)<br>Returns a stream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying the provided mapping function to each element. |
| **DoubleStream** | **flatMapToDouble**(**Function**<? super **T**,? extends **DoubleStream**> mapper)<br>Returns an DoubleStream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying the provided mapping function to each element. |
| **IntStream** | **flatMapToInt**(**Function**<? super **T**,? extends **IntStream**> mapper)<br>Returns an IntStream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying the provided mapping function to each element. |
| **LongStream** | **flatMapToLong**(**Function**<? super **T**,? extends **LongStream**> mapper)<br>Returns an LongStream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying the provided mapping function to each element. |

# Aufgabe 3 - JavaIO

## Java.util.stream

| Modifier and Type | Method and Description |
|---|---|
| void | **forEach**(**Consumer**<? super **T**> action)<br>Performs an action for each element of this stream. |
| void | **forEachOrdered**(**Consumer**<? super **T**> action)<br>Performs an action for each element of this stream, in the encounter order of the stream if the stream has a defined encounter order. |
| static <T> **Stream**<T> | **generate**(**Supplier**<T> s)<br>Returns an infinite sequential unordered stream where each element is generated by the provided `Supplier`. |
| static <T> **Stream**<T> | **iterate**(T seed, **UnaryOperator**<T> f)<br>Returns an infinite sequential ordered `Stream` produced by iterative application of a function `f` to an initial element `seed`, producing a `Stream` consisting of `seed`, `f(seed)`, `f(f(seed))`, etc. |
| **Stream**<T> | **limit**(long maxSize)<br>Returns a stream consisting of the elements of this stream, truncated to be no longer than `maxSize` in length. |
| <R> **Stream**<R> | **map**(**Function**<? super **T**,? extends R> mapper)<br>Returns a stream consisting of the results of applying the given function to the elements of this stream. |
| **DoubleStream** | **mapToDouble**(**ToDoubleFunction**<? super **T**> mapper)<br>Returns a `DoubleStream` consisting of the results of applying the given function to the elements of this stream. |
| **IntStream** | **mapToInt**(**ToIntFunction**<? super **T**> mapper)<br>Returns an `IntStream` consisting of the results of applying the given function to the elements of this stream. |
| **LongStream** | **mapToLong**(**ToLongFunction**<? super **T**> mapper)<br>Returns a `LongStream` consisting of the results of applying the given function to the elements of this stream. |
| **Optional**<T> | **max**(**Comparator**<? super **T**> comparator)<br>Returns the maximum element of this stream according to the provided `Comparator`. |
| **Optional**<T> | **min**(**Comparator**<? super **T**> comparator)<br>Returns the minimum element of this stream according to the provided `Comparator`. |
| boolean | **noneMatch**(**Predicate**<? super **T**> predicate)<br>Returns whether no elements of this stream match the provided predicate. |
| static <T> **Stream**<T> | **of**(T... values)<br>Returns a sequential ordered stream whose elements are the specified values. |
| static <T> **Stream**<T> | **of**(T t)<br>Returns a sequential `Stream` containing a single element. |
| **Stream**<T> | **peek**(**Consumer**<? super **T**> action)<br>Returns a stream consisting of the elements of this stream, additionally performing the provided action on each element as elements are consumed from the resulting stream. |
| **Optional**<T> | **reduce**(**BinaryOperator**<T> accumulator)<br>Performs a **reduction** on the elements of this stream, using an **associative** accumulation function, and returns an `Optional` describing the reduced value, if any. |
| **T** | **reduce**(**T** identity, **BinaryOperator**<T> accumulator)<br>Performs a **reduction** on the elements of this stream, using the provided identity value and an **associative** accumulation function, and returns the reduced value. |

# Dateiein- und -ausgabe