Typst Vorlagen für die Schule

Vorlagen für Arbeitsblätter, Klausuren und andere Materialien für die Schule.

v0.1.0 2024-02-26

Jonas Neugebauer

https://github.com/jneug/schule-typst

schule-typst ist eine Sammlung von Typst Vorlagen zur Gestaltung von Arbeitsmaterialien (Arbeitsblätter, Klausuren, Wochenpläne ...) für die Schule. Das Paket ist eine Adaption meines LaTeX Pakets für Typst.

Table of contents

Part I.
About

Part II.
Verwendung der Vorlagen

II.1. Installation

Part III.
Vorlagen

Kernstück von SCHULE-TYPST sind die Dokumentvorlagen.

III.1. Arbeitsblatt (ab)

Die Vorlage `#arbeitsblatt()` ist die Basisvorlage für alle anderen Vorlagen und Grundlage für die Gestaltung von Arbeitsmaterialien. Alle Argumente, die von `#arbeitsblatt()` akzeptiert werden, werden daher auch von allen anderen Vorlagen akzeptiert.

```
#arbeitsblatt(
  ⟨autor⟩,
  ⟨kuerzel⟩,
  ⟨titel⟩,
  ⟨reihe⟩,
  ⟨nummer⟩,
  ⟨fach⟩,
  ⟨kurs⟩,
  ⟨version⟩,
  ⟨datum⟩,
  ⟨margin⟩,
  ⟨typ⟩: "Arbeitsblatt",
  ⟨loesungen⟩: "sofort",
  ⟨fontsize⟩: 13pt,
  ⟨paper⟩: "a4",
  ⟨flipped⟩: false,
  ⟨lochung⟩: false,
  ..⟨args⟩
)[⟨body⟩]
```

Alle zusätzlichen Argumente in `[⟨args⟩]` werden nach Prefix gefiltert und an die entsprechende Funktion weitergegeben. Beispielsweise wird `⟨par-justify⟩: false` als `⟨justify⟩: false` an `par` weitergegeben.

- `par-` an `par`
- `font-` an `text`

III.1.1. Basisvorlage für ein Arbeitsblatt

3.1.1 *Arbeitsblatt (ab)*

```
#import "@local/schule:0.1.0": ab
#import ab: *

#show: arbeitsblatt.with(
  titel: "Potenzmengenkonstruktion",
  reihe: "Endliche Automaten und formale Sprachen",
  nummer: "IV.11",
  kurs: "Q2-LK",

  autor: "J. Neugebauer",
  kuerzel: "Ngb",

  version: "2024-02-16",
  datum: datetime.today()
)

#abtitel()

#lorem(100)

#aufgabe[
  #lorem(50)
]
```

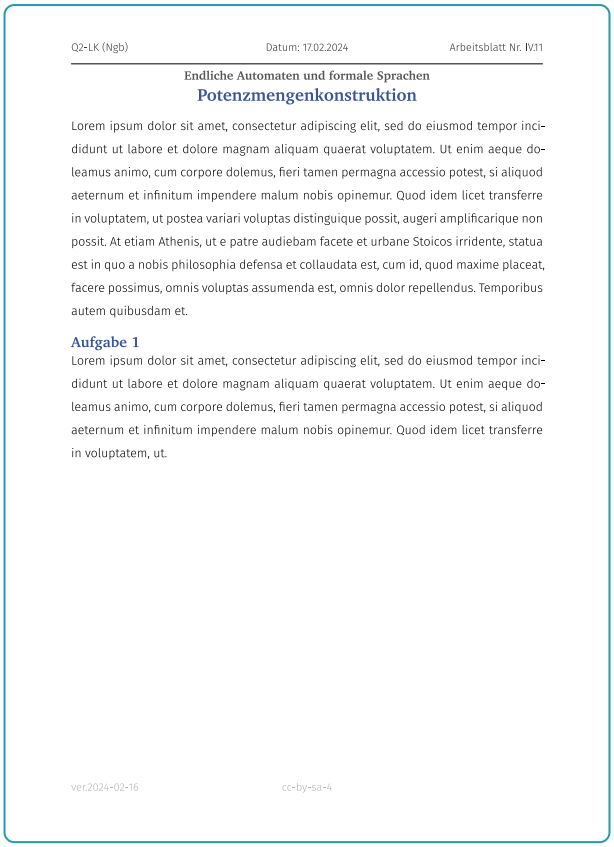

Figure 1: Ausgabe der Basisvorlage (`examples/ab.pdf`)

III.2. Klassenarbeit (ka)

III.3. Klausur (kl)

Part IV.
Allgemeine Kommandos

IV.1. Dokumentinformationen

IV.2. Auszeichnungen

IV.3. Layout

#fussLinks()	#kopfLinks()	#seitenzahl()
#fussMitte()	#kopfMitte()	#seitenzahl-format()
#fussRechts()	#kopfRechts()	#tablefill()
#fusszeile()	#kopfzeile()	#tabular()
#wrapfig()		

IV.3.1. Seitenzahlen

#seitenzahl-format(⟨current⟩, ⟨body⟩, ⟨total⟩) ⟶ **content**

Formatiert die Seitenzahl basierend auf der Anzahl Textseiten, die durch den Inhalt des Dokument erzeugt werden und der Gesamtanzahl Seiten, die im finalen Dokument vorhanden sind. Diese kann von der Anzahl Textseiten abweichen, wenn Inhalte automatisch generiert werden (beispielsweise Lösungen oder Erwartungshorizonte).

- `#seitenzahl-format(1, 1, 1)` ⟶
- `#seitenzahl-format(3, 2, 3)` ⟶ I
- `#seitenzahl-format(4, 2, 3)` ⟶ –
- `#seitenzahl-format(4, 3, 5)` ⟶ I
- `#seitenzahl-format(5, 3, 5)` ⟶ II

Argument

⟨current⟩ int

Aktuelle Seitenzahl.

Argument

⟨body⟩ int

Gesamtanzahl Textseiten im Dokument.

Argument

⟨total⟩ int

Gesamtanzahl Seiten im Dokument inklusive automatisch generierte Seiten wie Lösungen.

```
#seitenzahl(⟨format⟩: "seitenzahl-format")
```

Zeigt die aktuelle Seitenzahl an. Als Standard wird die Seitenzahl mit `#seitenzahl-format()` formatiert. Es kann aber auch eine eigene Formatfunktion angegeben werden.

- `#seitenzahl()` ⟶ 8 von 19
- `#seitenzahl(format: (c, b, t) => [#c/#b])` ⟶ 8/19

Argument

`⟨format⟩: "seitenzahl-format"` `function`

Eine Funktion `(int,int,int) => content`. Für eine Beschreibung der Paramter siehe `#seitenzahl-format()`.

IV.3.2. Kopf- und Fußzeilen

`#kopfLinks()`
Format für den linken Teil der Kopfzeile.

`#kopfMitte()`
Format für den mittleren Teil der Kopfzeile.

`#kopfRechts()`
Format für den rechten Teil der Kopfzeile.

`#kopfzeile(⟨links⟩: "kopfLinks", ⟨mitte⟩: "kopfMitte", ⟨rechts⟩: "kopfRechts")`
Formatierung der Kopfzeile in drei Teilen: `⟨links⟩`, `⟨mitte⟩`, `⟨rechts⟩`.

`#fussLinks()`
Format für den linken Teil der Fußzeile.

`#fussMitte()`
Format für den mittleren Teil der Fußzeile.

`#fussRechts()`
Format für den rechten Teil der Fußzeile.

`#fusszeile(⟨links⟩: "fussLinks", ⟨mitte⟩: "fussMitte", ⟨rechts⟩: "fussRechts")`
Formatierung der Fusszeile in drei Teilen: `⟨links⟩`, `⟨mitte⟩`, `⟨rechts⟩`.

IV.3.3. Abbildungen und Tabellen

```
#wrapfig(
  ⟨align⟩,
  ⟨width⟩: auto,
  ⟨gutter⟩: 0.75em,
  ⟨element⟩,
  ⟨body⟩
)
```

```
#wrapfig(
  left + horizon,
   rect(fill:gradient.linear(..color.map.rainbow), width:2cm, height:
1.4cm, radius:4pt),
  lorem(100),
  gutter: 5mm
)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

```
#tablefill(
  ⟨fill⟩: luma(100%),
  ⟨headerfill⟩: "theme.table.header",
  ⟨footerfill⟩: "theme.table.header",
  ⟨oddfill⟩: "theme.bg.muted",
  ⟨striped⟩: true,
  ⟨headers⟩: 1,
  ⟨footers⟩: 0,
  ⟨colheaders⟩: 0,
  ⟨colfooters⟩: 0,
  ⟨columns⟩: auto,
  ⟨rows⟩: auto,
  ⟨fills⟩: "(rows: (:), cols: (:))"
)
```

Hilfesfunktion für die Formatierung von Füllfarben für Tabellen. Die Funktion wird mit der

```
#table(
  columns: 4,
  fill: tablefill(
    footerfill: gradient.linear(..color.map.vlag, angle:90deg),
    oddfill: color.map.vlag.first(),
    headers: 2,
    footers: 1,
    colheaders: 1,
    rows: 9
  ),
  ..range(36).map(str)
)
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35

```
#tabular(
  ⟨inset⟩: "5pt",
  ⟨fill⟩: none,
  ⟨line-height⟩: 1.5em,
  ⟨header⟩: none,
  ⟨footer⟩: none,
  ..⟨args⟩
)
```

```
#tabular(
  header: [Eine Tabelle],
  columns: 4,
  ..range(36).map(str)
)
```

Eine Tabelle			
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35

Part V.
Beispiele

V.1. Vorlage Arbeitsblatt

V.1.1. `examples/10Diff-AB.IV.09-Listen.typ`

```
#import "@local/schule:0.0.5": ab
#import ab: *

#show: arbeitsblatt.with(
  titel:    [Listen],
  reihe:    [Programmierung mit TigerJython],
  kurs:     [DI10-NI],
  fach:     "Informatik",
  nummer:   "IV.09",
  version:  "2023-11-15",
  fontsize: 10pt,
)

#abtitel()

Eine Variable kann benutzt werden, um einen Wert zu speichern. Möchstest du
mehrere Werte speichern, kannst du eine Liste benutzen. Eine Liste speichert in
einer Variablen beliebig viele Werte "hintereinander".

#aufgabe[
  Studiere das Programm und überlege, was es macht. Schau dir vor allem die neuen Befehle an.

  Übernimm dann das Programm in _TigerJython_ und probiere es aus.

  #sourcecode[```python
  from gturtle import *

  eingabe = 1
  seiten = []     #neu: eine leere Liste erstellen
  while eingabe > 0:
    eingabe = input("Gib eine Zahl ein. 0 um zu beenden.")
    seiten.append(eingabe) #neu: Wert von eingabe hinten an die Liste anhängen

  makeTurtle()

  for s in seiten:  #neu: s nimmt jeden Wert in seiten an
    fd(s)
    rt(90)
  ```]
]

#aufgabe[
  Lest den Kasten "Neue Konzepte und Befehle" auf Seite 101.

  Bearbeitet dann auf Seite 102 das Beispiel 3 und Aufg. 3 und 4.
]

#aufgabe[
  Bearbeitet die folgenden Aufgaben nach eigenem Ermessen:

  #columns(2)[
    - S.103, Aufg. 5, 6, 8
    - S.106, Aufg. 14
    - S.108, Aufg. 17, 19
    #colbreak()
    - S.109, Aufg. 21; S.110, Aufg. 23
    - S.112, Aufg. 27, 28
  ]
]

#rahmen[
  #set text(.88em)
   Der `repeat`-Befehl ist kein richtiger Python-Befehl. Es gibt ihn nur in _TigerJython_. In Python erzeugt man stattdessen für eine n-fache
Wiederholung eine Liste mit den Zahlen `0` bis `n-1` und durchläuft sie mit `for`:
  #sourcecode(frame:none)[```python
  from gturtle import *

  repeat 4:   # funktioniert nur in TigerJython
    fd(100)
    rt(90)
```

Listing 1: Quelltextvorschau des Beispiels 10Diff-AB.IV.09-Listen
```

5.1.1 Vorlage Arbeitsblatt

Informatik DI10-NI Datum: _____ Arbeitsblatt Nr. IV.09

Programmierung mit TigerJython

Listen

Eine Variable kann benutzt werden, um einen Wert zu speichern. Möchstest du mehrere Werte speichern, kannst du eine Liste benutzen. Eine Liste speichert in einer Variablen beliebig viele Werte „hintereinander“.

Aufgabe 1

Studiere das Programm und überlege, was es macht. Schau dir vor allem die neuen Befehle an.

Übernimm dann das Programm in *TigerJython* und probiere es aus.

```
from gturtle import *

eingabe = 1
seiten = []     #neu: eine leere Liste erstellen
while eingabe > 0:
  eingabe = input("Gib eine Zahl ein. 0 um zu beenden.")
  seiten.append(eingabe) #neu: Wert von eingabe hinten an die Liste anhängen

makeTurtle()

for s in seiten:  #neu: s nimmt jeden Wert in seiten an
  fd(s)
  rt(90)
```

Aufgabe 2

Lest den Kasten „Neue Konzepte und Befehle“ auf Seite 101.

Bearbeitet dann auf Seite 102 das Beispiel 3 und Aufg. 3 und 4.

Aufgabe 3

Bearbeitet die folgenden Aufgaben nach eigenem Ermessen:

- S.103, Aufg. 5, 6, 8
- S.106, Aufg. 14
- S.108, Aufg. 17, 19
- S.109, Aufg. 21; S.110, Aufg. 23
- S.112, Aufg. 27, 28

Der `repeat`-Befehl ist kein richtiger Python-Befehl. Es gibt ihn nur in *TigerJython*. In Python erzeugt man stattdessen für eine n-fache Wiederholung eine Liste mit den Zahlen `0` bis `n-1` und durchläuft sie mit `for`:

```
from gturtle import *

repeat 4:     # funktioniert nur in TigerJython
  fd(100)
  rt(90)

for i in range(4):  # range ergeugt eine Liste mit vier Zahlen: [0,1,2,3]
  fd(100)
  rt(90)
```

V.1.2. Vorlage Klausur

V.1.3. examples/Q1-GK-Klausur_1.typ

```
#import "@local/schule:0.0.5": kl, informatik
#import kl: *
#import informatik: docs

#show: klausur.with(
  autor:   "J. Neugebauer",
  kuerzel: "Ngb",
  titel:   "1. Klausur",
  reihe:   "Objektorientierte Programmierung",
  nummer:  "1",
  fach:    "Informatik",
  kurs:    "Q1 GK",
  version: "2023-09-20",

  fontsize: 10pt,

  dauer: 135,
  datum: "20.09.2023",
  loesungen: "seite",
)

#let key(label) = box(stroke:.5pt + gray, inset:(x:2pt), outset:(y:2pt), radius:2pt, fill:theme.bg.muted, text(.88em, label))

#let arr( len, ..data ) = {
  let d = data.pos() + ([],) * calc.max(0, (len - data.pos().len()))
  table(
    columns: (auto,) + (8mm,) * len,
    fill: (c,r) => if r == 1 { theme.bg.muted },
    align: center,
    [*Inhalt*], ..d.map((v) => [#v]),
    [*Index*], ..range(len).map(str).map(raw)
  )
}

#kltitel()

#aufgabe(titel:"Variablen und Arrays")[
  #teilaufgabe[
    _#operator[Beschreibe], was der Gültigkeitsbereich einer Variablen in Java ist._

    _#operator[Gib] jeweils ein Beispiel an, bei dem der Gültigkeitsbereich einer Variablen eingehalten bzw. verletzt wird._

    #erwartung([Beschreibt das Konzept "Gültigkeitsbereich".], 2)
    #erwartung([Gibt ein Beispiel mit korrektem Gültigkeitsbereich an.], 2)
    #erwartung([Gibt ein Beispiel mit verletztem Gültigkeitsbereich an.], 2)

    #loesung[
      Der _Gültigkeitsbereich_ einer Variablen ist der Bereich einer Methode oder Klasse, in der die Variable _gültig_ ist, also benutzt werden kann. Der Gültigkeitsbereich beginnt ab der Deklaration der Variablen und besteht bis zum Ende des aktuellen Blocks. Dabei ist die Variable auch in verschachtelten Blöcken gültig.

      Eine Ausnahme sind Attribute, die immer in der gesamten Klasse gültig sind.

      *Beispiel:*
      ```java
      int zahl = 0;
      if( zahl > 0 ) {
        int wahr = true;
      }
      if( wahr ) {  // nicht korrekt, da wahr nicht mehr gültig
        zahl = 1;   // korrekt, da verschachtelter Block
      }
      ```
    ]
  ]

  #teilaufgabe[
    Arrays sind eine häufig genutzte lineare Datenstruktur in vielen Programmiersprachen.

    #operator[Erkläre] die Funktion von Arrays zur Datenspeicherung anhand passender Beispiele.

    #erwartung([Erklärt die Funktion von Arrays zur Datenspeicherung.], 2)
```

Listing 2: Quelltextvorschau des Beispiels Q1-GK-Klausur_1

5.1.3 Vorlage Arbeitsblatt

1. Klausur (135 Minuten)
Informatik Q1 GK (Ngb)

Aufgabe 1 Variablen und Arrays — 36 Punkte

a) *Beschreibe, was der Gültigkeitsbereich einer Variablen in Java ist.*
Gib jeweils ein Beispiel an, bei dem der Gültigkeitsbereich einer Variablen eingehalten bzw. verletzt wird.

(2 + 2 + 2)

b) Arrays sind eine häufig genutzte lineare Datenstruktur in vielen Programmiersprachen.
Erkläre die Funktion von Arrays zur Datenspeicherung anhand passender Beispiele.

(2 + 2)

c) *Analysiere den Quelltext links, indem du jeweils die Werte der Variablen* `a`, `b` *und* `c` *in der Tabelle notierst.*

ℹ **Hinweis:** Der `+`-Operator verknüpft zwei Strings miteinander zu einem neuen (Konkatenation):

```
"Hallo " + "Welt" → "Hallo Welt"
```

Algorithmus	a	b	c

`String a = "x";`
`int b = 10;`
`for( int c = 1; c < b; c++ ) {`
`  a = a + "x";`
`}`			

`int a = 0;`
`int c = 10;`
`for( ; c > 0;  ) {`
`  a += 1;`
`  int b = a-1;`
`  c = c-1;`
`}`			

`int[] arr = new int[4];`
`int a = 1;`
`while( a <= arr.length ) {`
`  arr[a-1] = a*a;`
`  a += 1;`
`}`
`int b = arr[2];`
`int c = arr[3];`			

(2 + 2 + 2)

d) *Analysiere den Quelltext in Listing 1 und beschreibe möglichst exakt seine Funktionsweise. Erläutere dabei insbesondere die Verwendung von Arrays im Algorithmus. Nimm bei deinen Erklärungen Bezug zu den jeweiligen Zeilennummern.*


```
1  public int[] wasMacheIch( int[] a ) {
2    int[] b = new int[a.length];
3
4    for( int n = a.length; n > 0; n -= 1 ) {
5      for( int i = 0; i < n; i += 1 ) {
6        b[n-1] += a[i];
7      }
8    }
9
10   return b;
11 }
```

Listing 1: Quelltext der Methode `wasMacheIch`.

(6)

e) Gegegeben ist die Methode `void fuelleArray()` in Listing 2, die ein Array mit Zahlen füllt.

```
1  public void fuelleArray( int[] zahlen ) {
2    Scanner input = new Scanner(System.in);
3
4    zahlen[0] = 1;
5    for( int i = 1; i < zahlen.length; i += 1 ) {
6      int neueZahl = input.nextInt();
7      zahlen[i] = zahlen[i-1] * neueZahl + i;
8    }
9  }
```

Listing 2: Quelltext der Methode `fuelleArray`.

In Zeile 6 werden mittels eines Objektes der Klasse `Scanner` Zahlen vom Benutzer abgefragt. Es wird folgende Reihe von Zahlen eingegeben und jeweils mit ENTER bestätigt:

1, 8, 0, 10, 4

Die Methode wird mit einem leeren Array der Länge sechs aufgerufen:

Inhalt							
Index	0	1	2	3	4	5	6

ℹ **Hinweis:** Die Methode `nextInt()` der Klasse `Scanner` liest eine vom Benutzer eingegebene Zahl von der Kommandozeile ein.
Gib den Zustand des Arrays nach Zeile 4 und nach jeder eingegeben Zahl in der oben gezeigten Tabellenform an. (Insgesammt also sechs Tabellen.)

(6)

f) Die Methode `boolean and( boolean[] pArray )` erhält ein Array mit Wahrheitswerten als Parameter und prüft, ob *alle* Werte im Array `true` sind. Ist dies der Fall, wird `true` zurückgeleifert, ansonsten `false`. Ist das Array leer, dann ist das Ergebnis auch `false`.
Implementiere die Methode `and` *vollständig.*

(4)

g) Die Methode `shiftArray` in Listing 3 soll die Elemente des Arrays `pArray` um `pShift` Stellen innerhalb des Arrays verschieben. Wird die Methode beispielsweise mit `3` für `pShift` aufgerufen, dann soll das Element an Index `1` an den Index `1 + 3 = 4` verschoben werden.
Beurteile, ob es bei der Ausführung der Methode `shiftArray` *zu einer* `ArrayIndexOutOfBoundsException` *kommen kann. Begründe deine Antwort und erläutere dabei auch, was diese Exception bedeutet.*


```
1  public String[] shiftArray( String[] pArray, int pShift ) {
2    for( int i = 0; i < pArray.length; i += 1 ) {
3      pArray[i + pShift] = pArray[i];
4    }
5    return pArray;
6  }
```

Listing 3: Quelltext der Methode `shiftArray`.

(2 + 2)

Aufgabe 2 Heldenspiel — 24 Punkte

Eine kleines Startup möchte ein einfaches Rollenspiel mit Helden und Monstern entwickeln. Dazu wurde nach folgender Beschreibung vorgegangen:

> Im Spiel gibt es Helden und Monster, die bekämpft werden müssen. Helden und Monster können eine Waffe tragen. Eine Waffe hat eine Qualität und einen Waffenbonus. Der Angriffswert einer Waffe berechnet sich aus seiner Qualität multipliziert mit seinem Bonus. Der Angriffswert eines Monsters berechnet sich aus dem Angriffswert des Monsters plus dem Angriffswert seiner Waffe.
> Ein Kampf im Spiel läuft nach einer Kampfregel in Runden ab. Zuerst werden in einer Runde die Angriffswerte der Gegner berechnet und verglichen. Der höhere Angriffswert gewinnt und der Verlierer bekommt 5 Lebenspunkte abgezogen. Ist der Angriffswert des Verlierers aber mindestens halb so groß wie der des Gewinners, dann bekommt auch der Gewinner 2 Lebenspunkte abgezogen.
> Nach einem erfolgreichen Angriff (also wenn dem jeweiligen Gegner Lebenspunkte abgezogen wurden), verliert die benutzte Waffe an Qualität. Diese wird um eins reduziert. Haben beide Gegner noch Lebenspunkte übrig beginnt nun die nächste Kampfrunde.
> Wenn zum Beispiel ein Monster mit einem Angriffswert von $4,5$ und einer Waffe mit dem Bonus $2,5$ und einer Qualität von 8 gegen einen Helden mit dem berechneten Angriffswert 32 antritt, dann berechnet sich der Angriffswert des Monsters durch $4,5 + 2,5 * 8 = 24,5$. Der Held hat einen Höheren berechneten Angriffswert und gewinnt diese Kampfrunde. Das Monster verliert 5 Lebenspunkte. Da $24,5$ größer als $32 : 2 = 16$ ist, verliert der Held auch 2 Punkte. Beide Waffen reduzieren ihre Qualität um eins, da beide Gegner getroffen haben.

Abbildung 1 zeigt einen Ausschnitt der Modellierung für das Heldenspiel.
Die Methode `reduziereQualitaet` in der Klasse `Waffe` reduziert die Qualität um eins. Die Methode `addiereLebenspunkte` in `Monster` und `Held` addiert die angegebene Zahl zu den Lebenspunkten und gibt die neuen Lebenspunkte zurück.

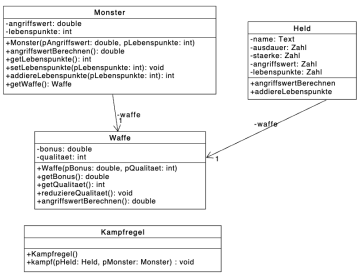

Abbildung 1: Implementationsdiagramm zum Heldenspiel.

a) Die Klasse `Held` wurde bisher nicht korrekt ins Implementationsdiagramm überführt.
Überführe die Klasse in ein Implementationsdiagramm, indem Du Dich anhand der Beschreibung oben für passende Datentypen für die Attribute entscheidest. Entscheide für jedes Attribut auch, ob ein Getter und / oder Setter sinnvoll ist. Begründe Deine Entscheidungen kurz.

(4 + 4)

b) Implementiere die Methode `double angriffswertBerechnen()` in der Klasse `Monster` passend zur Beschreibung oben.

(6)

c) Implementiere die Methode `void kampf(Held pHeld, Monster pMonster)` der Klasse `Kampfregel` passend zur Beschreibung eines Kampfes. Der Kampf soll dabei immer bis zum Ende durchlaufen (also bis einer der Gegner keine Lebenspunkte mehr hat).

(10)

ℹ **Hinweis:** Du kannst bei jeder Teilaufgabe die Methoden der anderen Klassen nutzen, als wären sie vollständig implementiert.

5.1.3 Vorlage Arbeitsblatt

Lösungen

Aufgabe 1 — 36 Punkte

a) Der *Gültigkeitsbereich* einer Variablen ist der Bereich einer Methode oder Klasse, in der die Variable *gültig* ist, also benutzt werden kann. Der Gültigkeitsbereich beginnt ab der Deklaration der Variablen und besteht bis zum Ende des aktuellen Blocks. Dabei ist die Variable auch in verschachtelten Blöcken gültig.
Eine Ausnahme sind Attribute, die immer in der gesamten Klasse gültig sind.

Beispiel:

```
int zahl = 0;
if( zahl > 0 ) {
  int wahr = true;
}
if( wahr ) {  // nicht korrekt, da wahr nicht mehr gültig
  zahl = 1;   // korrekt, da verschachtelter Block
}
```

b) Arrays speichern eine Reihe Variablen gleichen Typs in einer linearen Anordnung im Speicher. Die einzelnen Elemente des Arrays können dann über einen Index angesprochen werden, der bei 0 beginnt.
Ein Array hat eine feste Größe, die später nicht mehr geändert werden kann. Bei der Initialiierung wird der nötige Speicher für das Array reserviert.

c)

a	b	c
"xxxxxxxxx"	10	10
10	9	0
4	9	16

d)

e) 1)

Inhalt	1	0	0	0	0	0	0
Index	0	1	2	3	4	5	6

2)

Inhalt	1	2	0	0	0	0	0
Index	0	1	2	3	4	5	6

3)

Inhalt	1	2	18	0	0	0	0
Index	0	1	2	3	4	5	6

4)

Inhalt	1	2	18	3	0	0	0
Index	0	1	2	3	4	5	6

5)

Inhalt	1	2	18	3	34	0	0
Index	0	1	2	3	4	5	6

6)

Inhalt	1	2	18	3	34	141	0
Index	0	1	2	3	4	5	6

f)

```
public boolean and(boolean[] pArray) {
  if( pArray.length == 0 ) {
    return false;
  }
  for( int i = 0; i < pArray.length; i += 1 ) {
    if( !pArray[i] ) {
      return false;
    }
  }
  return true;
}
```

g) Es kommt zu einer `ArrayIndexOutOfBoundsException` sobald der Index `i` den Wert `pArray.length - shift` erreicht. Dann wird versucht auf den Index `i + shift` zuzugreifen, welcher nicht im Array existiert, da er gleich `pArray.length` ist.
Die Exception tritt auf, wenn ein Index im Array nicht existiert.

Aufgabe 2 — 24 Punkte

a)

b)

```
public double angriffswertBerechnen() {
  return waffe.angriffswertBerechnen() + angriffswert;
}
```

c)

```
public void kampf( Held pHeld, Monster pMonster ) {
  while( pHeld.getLebenspunkte() > 0 && pMonster.getLebenspunkte() > 0 ) {
    double aHeld    = pHeld.angriffswertBerechnen();
    double aMonster = pMonster.angriffswertBerechnen();
    if( aHeld >= aMonster ) {
      pMonster.addiereLebenspunkte(-5);
      pHeld.getWaffe().reduziereQualitaet();
      if( aMonster > aHeld/2 ) {
        pHeld.addiereLebenspunkte(-2);
        pMonster.getWaffe().reduziereQualitaet();
      }
    } else {
      pHeld.addiereLebenspunkte(-5);
      pMonster.getWaffe().reduziereQualitaet();
      if( aHeld > aMonster/2 ) {
        pMonster.addiereLebenspunkte(-2);
        pHeld.getWaffe().reduziereQualitaet();
      }
    }
  }
}
```


Erwartungshorizont

Name: _____

Aufg.	Die Schülerin / Der Schüler ...	mögl. Punkte	erreicht
1	*Variablen und Arrays*	**36**	
a)	... Beschreibt das Konzept „Gültigkeitsbereich". ... Gibt ein Beispiel mit korrektem Gültigkeitsbereich an. ... Gibt ein Beispiel mit verletztem Gültigkeitsbereich an.	6	
b)	... Erklärt die Funktion von Arrays zur Datenspeicherung. ... Gibt passende Beispiele an.	4	
c)	... Gibt die Werte der Variablen im ersten Code korrekt an. ... Gibt die Werte der Variablen im zweiten Code korrekt an. ... Gibt die Werte der Variablen im dritten Code korrekt an.	6	
d)	... Analysiert die Methode und beschreibt ihre Funktion.	6	
e)	... Gibt die sechs Zustände des Arrays an.	6	
f)	... Implementiert die Methode entsprechend der Vorgabe.	4	
g)	... Begründet, dass es zu einer Exception kommen kann. ... Erläutert die Bedeutung der Exception.	4	
2	*Heldenspiel*	**24**	
a)	... Gibt Datentypen für alle Attribute an und begründet die Entscheidungen. ... Entscheidet sich für Getter und Setter und begründet die Entscheidungen.	8	
b)	... Implementiert die Methode `angriffswertBerechnen` passend zur Beschreibung.	6	
c)	... Implementiert die Methode `kampf` passend zur Beschreibung.	10	
	Insgesamt:	**60**	

Note: _____

Note	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Prozent	95%	90%	85%	80%	75%	70%	65%	60%	55%	50%	45%	40%	33%	27%	20%	0%
Schwelle	57	54	51	48	45	42	39	36	33	30	27	24	19	16	12	0

Part VI.
Index