

Zeichenkettenverarbeitung mit Processing

Ein- und Ausgabe von Texten

Im Leitfaden für den Einstieg in die Programmierung mit Processing¹ haben wir bereits die Möglichkeit kennengelernt, vom Anwender einen Text über ein Eingabefeld zu erfragen. Variablen vom Datentyp *String* (deutsch: Zeichenkette) eignen sich, um einen solchen Text aufzunehmen. In einer Variablen vom Typ *String* liegt der Text als Zeichenkette vor. Das heißt, es handelt sich um eine Aneinanderreihung von Zeichen, auf die wir auch einzeln zugreifen können. Dazu später mehr.

Für die Ausgabe haben wir die Methode *text()* kennengelernt, mit der sich ein Text im Programmfenster ausgeben lässt. Eine weitere Möglichkeit zur Ausgabe eines Textes ist der Befehl *System.out.println()*. Der hier übergebene Text erscheint in der Konsole im unteren Abschnitt der Processing-Programmierungsumgebung. Diese Ausgabe ist zwar für den Anwender nicht so schön, aber für das schnelle Testen und einfache Übungen schnell und einfach umzusetzen.

Hinweis: Anders als in Java reicht in Processing auch der Aufruf *println()* ohne das Voranstellen von *System.out*.

Im folgenden Beispiel erscheint ein Eingabefeld, wenn der Anwender die Taste 'e' drückt (Zeile 10 und 11). Der Text, den der Anwender dann in das Eingabefeld eingibt, wird in der Konsole wieder ausgegeben (Zeile 12).

```
1 import static javax.swing.JOptionPane.*;
2 String eingabeText = "";
3 void setup(){
4     size(400, 300);
5     background(0, 0, 0);
6 }
7 void draw(){
8 }
9 void keyPressed(){
10     if(key == 'e'){
11         eingabeText = showInputDialog("Bitte gib einen Text ein!");
12         System.out.println(eingabeText);
13     }
14 }
```

Beispiel 1: Ein- und Ausgabe von Zeichenketten

Aufgabe 1: Ändern Sie das Programm so, dass der Text mithilfe der Methode *text()* im Programmfenster angezeigt wird.

¹ Die Programmierungsumgebung Processing wurde 2001 von Ben Fry und Casey Reas initiiert. Es wurde Processing in der Version 3.5.3 verwendet. Nähere Informationen finden Sie unter <https://processing.org/>.

Operatoren und Methoden zur Verarbeitung von Zeichenketten

In Beispiel 1 ist in Zeile 11 auch schon zu sehen, dass Zeichenketten innerhalb des Programmcodes immer in Anführungszeichen gesetzt werden. Zwei aufeinanderfolgende Anführungszeichen "" stellen dabei eine leere Zeichenkette dar (s. Zeile 2). Ein einzelnes Zeichen wird hingegen in einfache Anführungszeichen gesetzt. Das sehen wir z. B. in Zeile 10 am Beispiel des Zeichens 'e'.

Um Zeichenketten verarbeiten und ggf. verändern zu können, stehen uns einige hilfreiche Operatoren und Methoden zur Verfügung. Die folgende Tabelle zeigt zunächst nur die wichtigsten. Die Beispiele verwenden zwei Variablen `text1` und `text2` vom Typ *String*, in denen zuvor die Zeichenketten "Hallo" bzw. "du" gespeichert wurden.

| Operator /Methode | Beispiel | Bedeutung |
|--------------------------|---|--|
| | <code>String text1 = "Hallo";</code> <code>String text2 = "du";</code> | Erzeugen von Variablen des Typs <i>String</i> |
| <code>+</code> | <code>text1 + " " + text2;</code> wird zu "Hallo du" | Verbinden von Zeichenketten zu einer Zeichenkette |
| <code>length()</code> | <code>text1.length();</code> liefert den Wert 5 | Bestimmen der Länge einer Zeichenkette (also die Anzahl der Zeichen) |
| <code>charAt(...)</code> | <code>text1.charAt(1);</code> liefert das Zeichen 'a' | Auslesen eines Zeichens an einer bestimmten Position. Dabei ist zu beachten, dass die Nummerierung der Zeichen mit 0 beginnt. |
| <code>equals(...)</code> | <code>text1.equals(text2);</code> liefert den Wert <code>false</code> | Prüfen des Inhalts von zwei Zeichenketten auf Gleichheit. Bei gleichem Inhalt wird <code>true</code> zurückgegeben, ansonsten <code>false</code> . |

Tabelle 1: Operatoren und Methoden zur Verarbeitung von Zeichenketten

Schauen wir uns die Methoden etwas genauer an. Der Aufruf der Methode folgt immer dem Muster *ZeichenkettenvARIABLE.Methode*. Das liegt daran, dass es sich bei der Variablen vom Typ *String* um ein Objekt handelt, für das eine Methode ausgeführt werden kann. Den Methoden kann zum Teil ein Parameter übergeben werden. Im Falle von `charAt()` gibt der Parameter die Position des Zeichens an, das zurückgegeben werden soll. Dabei müssen wir beachten, dass die Nummerierung der Zeichen mit Null beginnt, so dass 1 hier eigentlich die Position des zweiten Zeichens angibt (s. Tabelle 2).

| | | | | |
|---|---|---|---|---|
| H | a | l | l | o |
| 0 | 1 | 2 | 3 | 4 |

Tabelle 2: Nummerierung der Zeichen in einer Zeichenkette

Beim Vergleichen mit der Methode `equals()` wird als Parameter ebenfalls eine Zeichenkette, hier `text2`, übergeben. Das heißt, der Inhalt der Zeichenkettenvariablen `text1` soll mit dem Inhalt der Zeichenkettenvariablen `text2` verglichen werden. Die Zeichenketten werden dabei Zeichen für

Zeichen verglichen, so dass genau dann `true` zurückgegeben wird, wenn die einzelnen Zeichen an jeder Position identisch und die Zeichenketten gleich lang sind.

Beispiel 2 zeigt ein Programm, das mithilfe der Methoden aus Tabelle 1 verschiedene Informationen zu einer Texteingabe ausgibt:

```
1 import static javax.swing.JOptionPane.*;
2 String eingabeText = "";
3 void setup(){
4     size(400, 300);
5     background(0, 0, 0);
6 }
7 void draw(){
8 }
9 void keyPressed(){
10     if(key == 'e'){
11         eingabeText = showInputDialog("Bitte gib einen Text ein!");
12         int laenge = eingabeText.length();
13         char zeichen = eingabeText.charAt(laenge-1);
14         String ausgabe = "Der Text \"" + eingabeText + "\" ist " + laenge
15             + " Zeichen lang. \nDas letzte Zeichen ist ein " + zeichen;
16         System.out.println(ausgabe);
17 }
```

Beispiel 2: Verarbeitung von Zeichenketten mithilfe der Methoden `length()` und `charAt()`

Wie in Beispiel 1 lassen wir den Anwender einen Text eingeben, wenn er die Taste 'e' drückt und speichern ihn in der Variablen `eingabeText` (s. Zeile 10 und 11). In Zeile 12 ermitteln wir zunächst die Länge des Textes und speichern den Wert in der Variablen `laenge`. In Zeile 13 rufen wir die Methode `charAt()` auf und übergeben als Parameter den Wert `laenge-1`. Da wir mit dem Zählen der Zeichen bei 0 beginnen, ist `laenge-1` die Position des letzten Zeichens. Dieses speichern wir in der Variablen `zeichen`. Die Variable `zeichen` ist vom Typ `char`. Dies ist der Datentyp für Variablen, die genau ein einzelnes Zeichen aufnehmen sollen. In Zeile 14 bauen wir die Zeichenkette für die Ausgabe zusammen. Dazu verknüpfen wir mithilfe des `+`-Operators Text, den wir als Programmierer fest eingeben, mit den Zeichenketten, die in den Variablen gespeichert sind. Variablen vom Datentyp `int` oder `char` werden dabei automatisch in die Zeichenketten mit den entsprechenden Zeichen umgewandelt. Da die Anführungszeichen im Quellcode so interpretiert werden, dass sie eine Zeichenkette beginnen oder beenden, muss ein umgekehrter Schrägstrich `\` vorangestellt werden, damit die Anführungszeichen als Zeichen mit in die Zeichenkette aufgenommen werden. Die Zeichenkombination `\n` steht für einen Zeilenumbruch. Gibt der Anwender z. B. den Text *Auf der grünen Wiese steht eine Kuh und lächelt* ein, sähe die Ausgabe wie in Abbildung 1 aus:

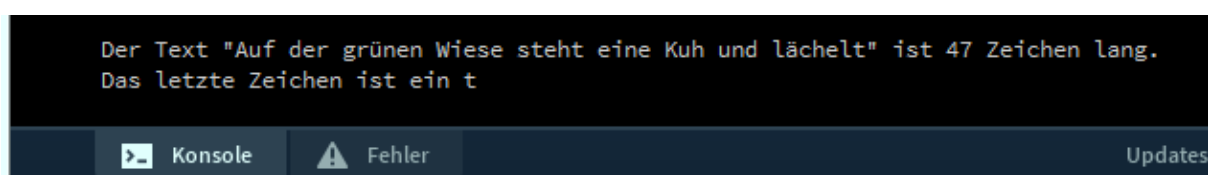


Abbildung 1: Ausgabe des Programms in Beispiel 2 für die Eingabe "Auf der grünen Wiese steht eine Kuh und lächelt"

Aufgabe 2:

- a) Ändern Sie das Programm aus Beispiel 2 so, dass das erste, das mittlere und das vorletzte Zeichen angezeigt werden.
- b) Legen Sie ein Codewort fest, das der Anwender eingeben muss, um eine Funktion des Programms zu nutzen.

Hinweis: Verwenden Sie eine globale Variable vom Typ Zeichenkette, um das Passwort zu speichern.

Verwenden Sie die Methode *equals*, um die Eingabe des Anwenders mit dem gespeicherten Passwort zu vergleichen.

- c) Erlauben Sie dem Anwender bei Drücken der Taste 'c' das Codewort zu ändern. Dazu muss er erst das alte Codewort eingeben und darf dann ein neues wählen. Das neue Codewort soll mindestens 8 Zeichen lang sein.

Aufgabe 3: Bei einer Kindersuchmaschine sollen bestimmte Suchwörter nicht erlaubt sein.

Entwickeln Sie ein Programm, das die Eingabe des Anwenders nur dann identisch wieder ausgibt, wenn es sich nicht um eines der verbotenen Wörter handelt. Welche Wörter nicht erlaubt sind, dürfen Sie selbst festlegen.

Aufgabe 4: Überlegen Sie sich fünf Fragen für ein kleines Quiz. Der Anwender kann das Quiz z. B. mit der Taste 'q' starten. Ihm werden dann nacheinander die fünf Fragen gestellt, zu denen er in einem Eingabefeld die Antwort eingeben muss. Am Ende erscheint im Programmfenster die Anzahl der richtigen Antworten.

Mögliche Erweiterungen:

- Bei einer falschen Antwort erhält man einen zweiten Versuch.
- Es wird zu jeder Frage angezeigt, ob die Antwort richtig oder falsch war.
- Es kann zwischen zwei Schwierigkeitsstufen gewählt werden.

Exkurs: Vergleich von Zeichenketten

Warum verwendet man für das Vergleichen nicht einfach den Operator `==`, den wir z. B. für das Vergleichen von Ganzzahlen und einzelnen Zeichen kennengelernt haben? Das liegt daran, dass es sich bei den Zeichenketten um Objekte handelt. Bei Objekten enthält die Variable zunächst nur eine Referenz auf einen Speicherplatz. Dort steht dann der eigentliche Wert. Bei dem Vergleich `text3 == text4`, würde daher nur geprüft, ob die beiden Variablen auf den gleichen Speicherplatz zeigen. In Abbildung 2 zeigt z. B. `text3` auf Speicherplatz 124 und `text4` auf Speicherplatz 128, so dass der Vergleich `false` ergibt, auch wenn beide Speicherplätze die Zeichenkette "Hokuspokus" enthalten. Beim Vergleich mit `equals` wird hingegen der Inhalt der Speicherplätze Zeichen für Zeichen verglichen, so dass `true` zurückgegeben wird, wenn in beiden Speicherplätzen der gleiche Text steht.²

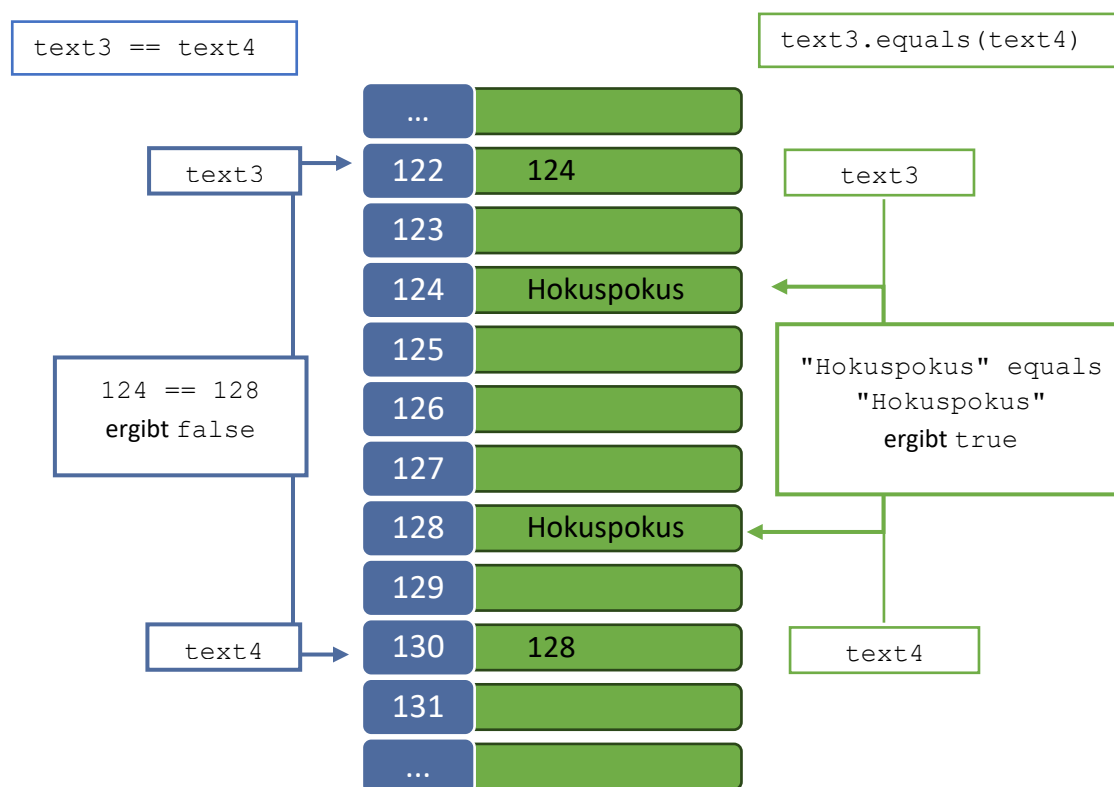


Abbildung 2: Der Unterschied zwischen dem Vergleich mittels `==` und `equals`

² Wer es noch ein wenig genauer wissen möchte, kann sich die Erläuterungen auf einer der folgenden Seiten anschauen:

Ullenboom, C. (2017). Java ist auch eine Insel. Kap. 4.4.7 Gut, dass wir verglichen haben.

http://openbook.rheinwerk-verlag.de/javainsel/04_004.html#u4.4.7 [Datum des Zugriffs: 21.06.2019]

javabeginners.de (2016). Was hat es mit String-Objekten in Java auf sich und wie kann man sie vergleichen?

https://javabeginners.de/String/Strings_vergleichen.php [Datum des Zugriffs: 21.06.2019]

Zeichenweise Verarbeitung einer Zeichenkette

Für viele algorithmische Probleme, bei denen Zeichenketten überprüft oder verändert werden, ist es notwendig, eine Zeichenkette Zeichen für Zeichen zu betrachten. Das grundsätzliche Vorgehen dazu schauen wir uns an einem Beispiel an. Dieses Grundgerüst kann dann auf andere Probleme übertragen und entsprechend angepasst werden.

```
import static javax.swing.JOptionPane.*;

1  String eingabeText = "";

2  void setup(){
3      size(400, 300);
4      background(0, 0, 0);
5  }

6  void draw(){
7  }

8  void keyPressed(){
9      if(key == 'e'){
10         eingabeText = showInputDialog("Bitte gib einen
                                     Text ein!");

11         String neuerText = "";
12         for(int i = 0; i < eingabeText.length(); i++){
13             char zeichen = eingabeText.charAt(i);
14             if(zeichen == 'ä'){
15                 neuerText = neuerText + "ae";
16             }else{
17                 neuerText = neuerText + zeichen;
18             }
19             System.out.println(neuerText);
20         }
21     }
22 }
```

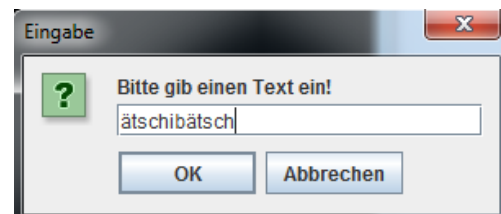


Abbildung 3: Exemplarische Eingabe für Beispiel 3

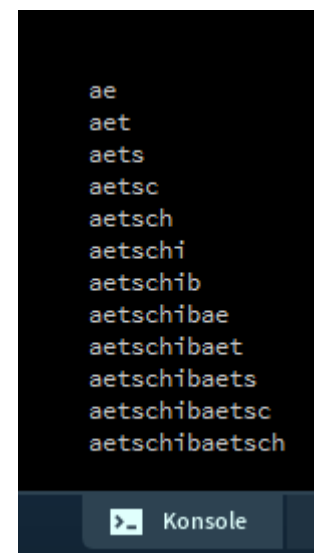


Abbildung 4: Ausgabe für die Eingabe in Abbildung 3

Beispiel 3: Zeichenweises verarbeiten einer Zeichenkette

Bevor Sie weiterlesen, sollten Sie sich den Quelltext und die Ausgabe in Abbildung 4, die für den Eingabetext *ätschibätsch* erzeugt wurde, einmal anschauen. Vielleicht können Sie den Ablauf des Programms selbst erläutern.

Wenn wir die Eingabe mit der Ausgabe vergleichen, sehen wir, dass in der eingegebenen Zeichenkette alle 'ä' durch 'ae' ersetzt wurden. Da die Umlaute 'ä', 'ö' und 'ü' nicht in allen Sprachen bekannt sind, ist das manchmal notwendig. Wie wir an der Ausgabe ebenfalls sehen, wurden die Zeichen aber nicht einfach ersetzt, sondern die neue Zeichenkette wurde Zeichen für Zeichen aufgebaut. Dort wo vorher ein 'ä' stand, wurde ein 'ae' angefügt und ansonsten wurde das jeweilige Zeichen aus der alten Zeichenkette übernommen. Dazu wird in Zeile 12 vor Beginn der *for*-Schleife eine neue Variable *neuerText* vom Typ *String* erzeugt, die zunächst leer ist. Die *for*-Schleife startet an Position 0 (*i* = 0) und zählt die Variable *i* solange hoch, bis sie die Länge des Textes erreicht. Da *Textlänge-1* die Position des letzten Zeichens ist, muss die Schleife an dieser Stelle abbrechen. Für jede Position *i* von 0 bis *Textlänge-1* wird nun das Zeichen an dieser Position ermittelt und in der Variablen *zeichen* vom Typ *char* zwischengespeichert. Für dieses Zeichen wird nun überprüft, ob es sich um ein 'ä' handelt. Wenn dies der Fall ist, wird ein 'ae' an den neuen Text gehängt, ansonsten

das Zeichen selbst. Bei der Zuweisung an die Variable `neuerText` ist wichtig, dass der bisherige Inhalt von `neuerText` und das neue Zeichen zunächst mit dem `+`-Operator verbunden werden und dann in die Variable `neuerText` geschrieben werden, da der bisherige Inhalt sonst jedes Mal verloren ginge.

Es ist sehr ratsam, den ursprünglichen Eingabetext innerhalb der Schleife nicht zu verändern. Denn wenn wir ein 'ä' durch ein ae ersetzen oder andersherum, verändert sich die Positionszahl aller nachfolgenden Zeichen und die Länge der Zeichenkette. Es könnten dadurch Zeichen vergessen oder doppelt betrachtet werden. Wenn wir Zeichen löschen und in der Schleifenabbruchbedingung die Länge der ursprünglichen Zeichenkette verwenden, könnte das Programm sogar Fehlermeldungen erzeugen, weil eine Position angesprochen wird, die plötzlich gar nicht mehr existiert. Deshalb ist man auf der sicheren Seite, wenn man den neuen Text in einer neuen Variablen aufbaut.

Aufgabe 5:

- a) Erweitern Sie Beispiel 3 so, dass auch alle 'ö' und 'ü' sowie alle 'Ä', 'Ö' und 'Ü' durch 'oe', 'ue', 'Ae', 'Oe' bzw. 'Ue' ersetzt werden.

Tipp: Es bietet sich an, diese Aufgabe mit einer *switch-case*-Verzweigung zu lösen. Mit dem Schlüsselwort *default* kann dabei ein letzter Fall angegeben werden, der eintritt, wenn alle anderen Fälle nicht gelten. Zur Erinnerung:

```
switch(Variable)
case wert1: Anweisungen
case wert2: Anweisungen
...
default: Anweisungen
```

- b) In Beispiel 3 wurde erläutert, wie eine Zeichenkette, die in einer Variablen gespeichert ist, verändert werden kann, indem sie Zeichen für Zeichen durchläuft und das Ergebnis in einer neuen Variablen aufgebaut wird. Dazu wurden die folgenden drei Codefragmente verwendet. Identifizieren Sie diese Fragmente in dem Programmcode aus Beispiel 3. Welche Variablen entsprechen den Variablen `alt` und `neu`, welcher Teil des Programms aus Beispiel 3 ersetzt den TODO Block?

```
> for (int i = 0; i < alt.length(); i++) {
    //TODO
}
> alt.charAt(i);
> neu = neu + Zeichen; //Zeichen steht für ein oder mehrere beliebige Zeichen
```

- c) Ändern Sie das Programm so, dass statt der *for*-Schleife eine *while* - Schleife mit einer Zählvariablen verwendet wird.
- d) Verändern Sie das Programm so, dass der eingegebene Text rückwärts ausgegeben wird.

Aufgabe 6: Es gibt die Empfehlung, einstellige Zahlen in Texten auszuschreiben. Erstellen Sie ein Programm, das die Zahlen 0 bis 9 in einem Eingabetext durch die entsprechenden Zahlwörter ersetzt. Sie können hier zunächst davon ausgehen, dass der Eingabetext keine mehrstelligen Zahlen enthält. Eine komplexere Variante, die auch der Empfehlung nachkommt, die Zahlen 10, 11 und 12 auszuschreiben und Ziffern nicht ersetzt, wenn sie Teil einer größeren Zahl sind, betrachten wir später in Aufgabe 12. Sie können aber schon einmal überlegen, weshalb eine Umsetzung mit unserem bisherigen Wissen schwierig ist.

Aufgabe 7: Erweitern Sie Ihre Lösung zu Aufgabe 2 so, dass der Anwender bei der Wahl seines Codeworts mindestens eines der Sonderzeichen #, *, +, ! oder ? verwenden muss. Ansonsten wird er zur Wahl eines anderen Codeworts aufgefordert.

Aufgabe 8: Eine Möglichkeit, einen Text unleserlich zu machen, ist, die Reihenfolge der Buchstaben zu vertauschen.

- a) Erstellen Sie ein Programm, das die Buchstaben so vertauscht, dass erst alle Zeichen an einer geraden Position (0, 2, 4, ...) und dann alle Zeichen an einer ungeraden Position (1, 3, 5, ...) ausgegeben werden. Aus dem Wort *Apfelmus* würde so z. B. *Aflupems*.
- b) Erweitern Sie anschließend Ihr Programm aus a) um die Option, die Vertauschung wieder rückgängig zu machen.
- c) Überlegen Sie sich weitere Regeln für die Vertauschung der Buchstaben und erstellen Sie ein entsprechendes Programm.

Aufgabe 9: Beim Knacken von Geheimschriften kann es hilfreich sein, die Anzahl eines bestimmten Zeichens in einem Text zu bestimmen. Dabei müssen Klein- und Großbuchstaben nicht unterschieden werden. Deshalb ist es für die folgenden Aufgaben sinnvoll, die Eingabe mithilfe der Methode `toLowerCase()` in Kleinbuchstaben zu verwandeln. Für die Variable `eingabeText` vom Typ `String`, sähe das so aus:

```
eingabeText = eingabeText.toLowerCase();
```

- a) Erstellen Sie ein Programm, das ausgibt, wie oft der Buchstabe 'e' in einem Text, den der Anwender eingibt, vorkommt.
- b) Erstellen Sie ein Programm, bei dem der Anwender zunächst eine der Tasten a bis z drückt, um den Buchstaben zu wählen, der gezählt werden soll. Anschließend gibt er über ein Eingabefeld den Text ein, in dem der entsprechende Buchstabe gezählt werden soll.
- c) Erstellen Sie ein Programm, das für den Text, den der Anwender eingibt, ermittelt, welcher Vokal am häufigsten vorkommt.

Ausblick: Ein Programm, das allgemein den häufigsten Buchstaben in einem Text ermittelt, erstellen Sie in Aufgabe 14. Dazu ist es hilfreich sich im Folgenden zunächst etwas genauer mit der internen Codierung von Zeichen zu beschäftigen.

Verarbeitung einzelner Zeichen mithilfe des ASCII-Codes

Da der Computer intern nur mit Binärzahlen arbeiten kann, müssen alle Zeichen binär codiert werden. Dazu wird jedem Zeichen eine Zahl zugeordnet. Einer der ersten Codes, der für die Codierung von Zeichen entwickelt wurde, ist der ASCII-Code. Dieser ist auch heute noch in modernen Codierungen wie dem Unicode enthalten und wird auch von Processing verwendet. Tabelle 3 zeigt einen Ausschnitt der Zuordnung von Zeichen zu Zahlen gemäß des ASCII-Codes. Zur besseren Lesbarkeit sind die Zahlen hier im Dezimalsystem dargestellt. Es fällt auf, dass sowohl die Ziffern von 0 bis 9 als auch die Klein- und Großbuchstaben jeweils fortlaufend durchnummeriert sind. Wir können uns den ASCII-Code daher zu Nutze machen, um die Zeichen des Alphabets systematisch zu durchlaufen, Zeichen lexikographisch zu vergleichen oder mit Zeichen "zu rechnen". Das schauen wir uns im Folgenden genauer an.

| Zeichen | ASCII | Zeichen | ASCII | Zeichen | ASCII |
|---------|-------|---------|-------|---------|-------|
| ! | 33 | 2 | 50 | D | 68 |
| # | 35 | ... | ... | ... | ... |
| * | 42 | 9 | 57 | Z | 90 |
| + | 43 | A | 65 | a | 97 |
| 0 | 48 | B | 66 | ... | ... |
| 1 | 49 | C | 67 | z | 122 |

Tabelle 3: Ausschnitt des ASCII-Codes

Das Programm in Beispiel 4 gibt für jedes Zeichen den entsprechenden ASCII-Code aus.

```

1 void setup(){
2   size(400, 300);
3   background(0, 0, 0);
4 }

5 void draw(){
6 }

7 void keyTyped(){
8   char zeichen = key;
9   int asciiCode = (int)zeichen;
10  System.out.println("Das Zeichen " + zeichen + " hat den ASCII-Code " +
                        asciiCode);
11 }

```

Beispiel 4: ASCII-Code eines Zeichens ausgeben

Um zu verstehen, wie das Programm den ASCII-Code zu einem Zeichen ermittelt, schauen wir uns die Zeilen 8 und 9 etwas genauer an. Wir wissen bereits, dass das Zeichen, das der Anwender zuletzt über die Tastatur eingegeben hat, in der Systemvariablen `key` gespeichert wird. Dies ist eine Variable vom Typ `char` und sie kann somit genau ein Zeichen aufnehmen. Den Wert der Variablen `key` können wir somit in Zeile 8 auch in eine andere Variable vom Typ `char` übertragen. Um den ASCII-Code des Zeichens zu erhalten, genügt es das Zeichen in eine Ganzzahl zu konvertieren, indem `(int)` vor die entsprechende Variable vom Typ `char` geschrieben wird (s. Zeile 9). Processing verwendet bei dieser Typkonvertierung automatisch den ASCII-Code. Den entsprechenden Wert speichern wir in der Variablen `asciiCode` vom Typ `int`. Sowohl das Zeichen als auch den ASCII-

Code können wir dann mithilfe der beiden Variablen `zeichen` und `asciiCode` in Zeile 10 in der Konsole ausgeben.

Tasten wie die Shifttaste benötigen wir nur, um z.B. einen Großbuchstaben einzugeben. Damit das Drücken der Shifttaste nicht als separate Eingabe gewertet wird, verwenden wir diesmal die Methode `keyTyped` (s. Zeile 7). Diese Methode ignoriert Sondertasten wie z. B. Shift, Alt oder Strg und wird nur ausgeführt, wenn ein Zeichen eingegeben wird.

Aufgabe 10:

- Stellen Sie anhand der Tabelle 3, die einen Ausschnitt des ASCII-Codes zeigt, eine Vermutung auf, welche Zahl, jeweils die folgenden Zeichen codiert: 3, 6, E, M, e, m
Überprüfen Sie Ihre Vermutung anschließend mithilfe des Programms aus Beispiel 4.
- Verwenden Sie das Programm aus Beispiel 4, um auch für die folgenden Zeichen den ASCII-Code zu ermitteln: @ = ! ; (
- Anstatt das Zeichen aus der Systemvariablen `key` erst in die Variable `zeichen` zu übertragen, können wir auch direkt mit der Systemvariablen `key` arbeiten und diese in den entsprechenden ASCII-Code umwandeln. Verändern Sie das Programm aus Beispiel 4 entsprechend, so dass auf die Variable `zeichen` verzichtet werden kann.

Die Umwandlung vom ASCII-Code in das entsprechende Zeichen funktioniert ähnlich. Das Programm in Beispiel 5 erfragt in Zeile 10 vom Anwender über ein Eingabefeld zunächst einen ASCII-Code. Da es sich bei der Eingabe in ein Eingabefeld immer um eine Zeichenkette handelt, muss die Zeichenkette zunächst in die entsprechende Ganzzahl umgewandelt werden. Dies geschieht in Zeile 11 mithilfe der Methode `Integer.parseInt()`. Die Umwandlung des ASCII-Codes in das entsprechende Zeichen erfolgt diesmal durch Voranstellen von (`char`). Denn das Konvertieren des Datentyps `int` in den Datentyp `char` erfolgt wieder anhand der ASCII-Tabelle, so dass in Zeile 12 in der Variablen `zeichen` das passende Zeichen gespeichert wird. Voraussetzung ist natürlich, dass eine Zahl eingegeben wurde, die im ASCII-Code verwendet wird. Die darstellbaren Zeichen liegen dabei zwischen 33 und 126. In Zeile 13 wird die Zuordnung schließlich wieder mithilfe der Variablen `asciiCode` und `zeichen` ausgegeben.

```
1  import static javax.swing.JOptionPane.*;
2  String eingabeText = "";
3  void setup(){
4      size(400, 300);
5      background(0, 0, 0);
6  }
7  void draw(){
8  }
9  void keyPressed(){
10     eingabeText = showInputDialog("Bitte gib einen ASCII-Code ein!");
11     int asciiCode = Integer.parseInt(eingabeText);
12     char zeichen = (char) asciiCode;
13     System.out.println("Der ASCII-Code " + asciiCode + " steht für das
                           Zeichen " + zeichen);
14 }
```

Beispiel 5: Zeichen zu einem ASCII-Code ausgeben

Aufgabe 11:

- a) Verwenden Sie das Programm aus Beispiel 5, um zu verschiedenen Zahlen des ASCII-Codes die entsprechenden Zeichen zu ermitteln.
- b) Wenn der Anwender in Beispiel 5 keine Zahl, sondern Buchstaben oder andere Zeichen eingibt, bricht das Programm mit einer Fehlermeldung ab. Verbessern Sie das Programm, indem Sie vor der Typkonvertierung der eingegebenen Zeichenkette in eine Zahl in Zeile 11 überprüfen, ob es sich bei den eingegebenen Zeichen nur um Ziffern handelt. Prüfen Sie nach der Konvertierung zusätzlich, ob es sich bei der Eingabe um einen gültigen ASCII-Code zwischen 33 und 126 handelt. Geben Sie ansonsten entsprechende Fehlermeldungen aus.

Aufgabe 12*: Erweitern Sie Aufgabe 6 so, dass in einem Eingabetext die Zahlen 0 bis 12 durch das entsprechende Zahlwort ersetzt werden. Sind die Zahlen Teil einer größeren Zahl, z. B. 39 oder 120 soll keine Ersetzung stattfinden.

Tipp: Es kann hier hilfreich sein, mithilfe des ASCII-Codes zu prüfen, ob es sich bei einem Zeichen um eine Ziffer handelt.

Aufgabe 13:

- a) Erstellen Sie ein Programm, das für den Text, den der Anwender eingibt, die Codierung mit dem ASCII-Code ausgibt. Setzen Sie zwischen die Zahlen des ASCII-Code jeweils ein Leerzeichen.
Beispiel: Für die Eingabe „Hallo“ ist die Ausgabe 72 97 108 108 111
- b) Überlegen Sie, ob es sich hierbei um eine Verschlüsselung handelt. Begründen Sie Ihre Antwort.
- c) Erweitern Sie Ihr Programm so, dass auch eine Decodierung von der Zahlendarstellung im ASCII-Code in Buchstaben möglich ist. Gehen Sie davon aus, dass nur gültige ASCII -Werte getrennt durch Leerzeichen eingegeben werden, z. B. 73 110 102 111 114 109 97 116 105 107

Hinweis zu Aufgabe 14 und 15: Wandeln Sie die Eingabe zur Vereinfachung in Kleinbuchstaben um.

Aufgabe 14: Erstellen Sie ein Programm, das für einen einzugebenden Text ermittelt, welcher Buchstabe am häufigsten vorkommt. Gibt es mehrere häufigste Buchstaben, reicht die Ausgabe eines häufigsten Buchstabens.

Aufgabe 15:

Bei der Caesar-Verschlüsselung werden die Zeichen des Klartextes um eine bestimmte Anzahl an Stellen im Alphabet verschoben, um den Geheimtext zu erhalten. Um die Ver- und Entschlüsselung per Hand durchzuführen, kann z. B. eine Caesar-Scheibe verwendet werden. Im Folgenden soll die Verschiebung algorithmisch umgesetzt werden.

- a) Erstellen Sie als Vorübung ein Programm, das den zehnten Buchstaben im Alphabet nach dem eingegebenen Buchstaben ausgibt. Wenn der Buchstabe 'z' erreicht wurde, soll wieder bei 'a' begonnen werden.
- b) Erstellen Sie ein Programm, das als Eingabe den Klartext und einen Schlüssel, der die Verschiebung angibt, erhält und dazu den Geheimtext ausgibt. Überlegen Sie sich geeignete Eingaben zum Testen. Überlegen Sie auch, wie sie mit Satzzeichen und Leerzeichen verfahren.
- c) Erweitern Sie Ihr Programm, um die Möglichkeit der Entschlüsselung. Als Eingaben benötigen Sie dazu einen Geheimtext und den Schlüssel.
- d) Verwenden Sie Ihr Programm aus Aufgabe 14, um einen Geheimtext, der nach dem Caesar-Verfahren verschlüsselt wurde, zu knacken. Das heißt, das Programm soll zu einem Geheimtext ohne Kenntnis des Schlüssels einen wahrscheinlichen Vorschlag für den Klartext machen.

Projekt:

Einzelarbeit: Informieren Sie sich über verschiedene Verschlüsselungsverfahren. Entscheiden Sie sich für ein Verfahren, das Sie algorithmisch umsetzen. Beispiele wären das Vigenère-Verfahren oder die Skytale. Sie können sich natürlich auch selbst ein Verfahren ausdenken!

Gruppenarbeit: Informieren Sie sich über verschiedene Verschlüsselungsverfahren oder entwerfen Sie selbst verschiedene Verfahren. Setzen Sie die Verschlüsselungsverfahren algorithmisch um. Erstellen Sie in der Gruppe ein umfangreicheres Programm, das mehrere Verschlüsselungsverfahren anbietet. Teilen Sie die Implementierung der ausgewählten Verfahren unter sich auf.

Ausblick: Weitere Operationen für Zeichenketten

Für Zeichenketten stehen eine Reihe weiterer Operationen zur Verfügung, die bei der Verarbeitung von Zeichenketten hilfreich sein können. Exemplarisch betrachten wir hier die Operationen *substring* und *contains*. Tabelle 4 gibt einen Überblick. Die Beispiele verwenden eine Variable `text` vom Typ *String*, in der die Zeichenkette "Hallo Siri" gespeichert ist.

| Methode | Beispiele | Bedeutung |
|-----------------------------|--|--|
| | <code>String text = "Hallo Siri";</code> | Erzeugen von Variablen des Typs <i>String</i> |
| <code>substring(...)</code> | <code>text.substring(6);</code> liefert die Zeichenkette "Siri"; <code>text.substring(6,8);</code> liefert die Zeichenkette "Si"; | Auslesen einer Teilzeichenkette in einem bestimmten Bereich. Wird nur ein Parameter übergeben wird die Teilzeichenkette ab dieser Position bis zum Ende der ursprünglichen Zeichenkette ausgegeben. Wird ein zweiter Parameter übergeben, gibt dieser an bis zu welcher Position die Teilzeichenkette entnommen wird. Das Zeichen an der Position selbst ist nicht mehr in der Teilzeichenkette enthalten. |
| <code>contains(...)</code> | <code>text.contains("Siri");</code> liefert den Wert <code>true</code> <code>text.contains("Alexa");</code> liefert den Wert <code>false</code> <code>text.contains("a");</code> liefert den Wert <code>true</code> | Prüfen einer Zeichenkette auf eine Teilzeichenkette. Der Rückgabewert ist <code>true</code> , wenn die Zeichenkette, die als Parameter übergebene Zeichenfolge enthält, sonst <code>false</code> . Hinweis: Wenn geprüft werden soll, ob ein einzelnes Zeichen enthalten ist, muss dieses ebenfalls als Zeichenkette übergeben werden. |

Tabelle 4: Die Methoden *substring* und *contains*

Aufgabe 16: Eine IBAN ist nach einem festen Schema aufgebaut:

2-stellige 10-stellige
Prüfziffer Kontonummer

DE22100100500123456789

2-stellige 8-stellige
Länderkennung Bankleitzahl

Abbildung 5: Aufbau einer IBAN

Erstellen Sie ein Programm, das eine IBAN, die der Anwender eingibt, in die einzelnen Bestandteile zerlegt. Die Ausgabe zu der IBAN in Abbildung 5 könnte z. B. wie in Abbildung 6 aussehen.



Abbildung 6: mögliche Ausgabe

Aufgabe 17: Manche Spamfilter ordnen Mails als Spam ein, wenn sie bestimmte auffällige Wörter enthalten. Überlegen Sie sich vier Wörter, die auf eine Spam-Mail hinweisen könnten. Erstellen Sie anschließend ein Programm, das einen Text darauf prüft, ob eines dieser auffälligen Wörter enthalten ist und eine entsprechende Rückmeldung gibt, ob es sich um eine Spam-Mail handeln könnte.

Aufgabe 18: Plattformen, auf denen Bilder hochgeladen werden können, schränken manchmal die erlaubten Dateiformate ein. Erlaubt sein könnten z. B. die Formate "jpg" und "png".

Erstellen Sie ein Programm, das prüft, ob ein Dateiname, der eingegeben wird, mit einem gültigen Dateiformat endet und dem Anwender eine entsprechende Rückmeldung gibt.

Aufgabe 19:

- Betrachten Sie noch einmal die Aufgaben 3 und 4 am Beginn des Leitfadens. Diskutieren Sie, ob die Verwendung der Methode `contains()` statt der Methode `equals()` die Implementierung eines entsprechenden Programms erleichtert.
- In Aufgabe 7 sollte ein vom Anwender gewähltes Codewort darauf überprüft werden, ob es eines der Sonderzeichen #, *, +, ! oder ? enthält. Erläutern Sie, wie sich die Implementierung unter Verwendung der Methode `contains()` vereinfacht.

Lizenz

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#).

Für die korrekte Ausführbarkeit der Quelltexte in diesem Leitfaden und der beiliegenden Quelltexte zu den Beispielen wird keine Garantie übernommen. Auch für Folgeschäden, die sich aus der Anwendung der Quelltexte oder durch eventuelle fehlerhafte Angaben ergeben, wird keine Haftung oder juristische Verantwortung übernommen.