Die Datentypen char und String

Mit dem Datentyp `char` (character, engl. für Zeichen) kann ein einzelnes Zeichen gespeichert werden. Ein `char`-Wert wird in einfachen Anführungszeichen angegeben.
Ein Beispiel ist die vordefinierte Processing-Variable key. Sie speichert die zuletzt gedrückte Taste. Sie kann beispielsweise in Kombination mit dem Unterprogramm keyPressed verwendet werden: Dieses wird automatisch aufgerufen, wenn eine Taste gedrückt wird.

Sketch Kap5_Bsp_char

```
void keyPressed() {
  if (key == 'j' || key == 'y') {
    println("Ja/Yes!");
  } else {
    println("Nein/No!");
  }
}
```

Mit dem Datentyp `String` (engl. für Kette) kann eine Zeichenkette – also einen Text – gespeichert werden. Ein `String`-Wert wird mit doppelten Anführungszeichen angegeben.
Mehrere Strings und auch Werte von anderen Datentypen können mit einem „+" verbunden werden, um die Werte aneinanderzureihen.

Sketch Kap5_Bsp_String_A

```
String name = "Albert";
int geburtsjahr = 1879;

String text = name + " wurde " + geburtsjahr + " geboren.";
println(text);
```

Datentypen wie `String`, die mit einem Großbuchstaben beginnen kennzeichnen in Java „Klassen". Wir werden dieses Programmierkonzept noch nicht kennenlernen, müssen aber einige Dinge über die „Klasse" `String` wissen. Eine Klasse bringt zugehörige Unterprogramme („Methoden"), die mit dem Wert der Variable arbeiten, mit.

Wir werden diese Methoden der Klasse String benötigen:

- `length()`
  Gibt die Anzahl der Zeichen zurück.
- `charAt(int n)`
  Gib das n-te Zeichen im String zurück.
  (Die Zählung beginnt bei 0.)
- `equals(String s)`
  Prüft ob zwei Strings identische Werte haben.
  (Bei Strings kann man nicht == verwenden!)

Sketch Kap5_Bsp_String_A

```
String name = "Albert";
println(name);

// Beispiel für length
int anzahl = name.length();
println(anzahl + " Zeichen");

// Beispiel für charAt
char anfang = name.charAt(0);
println("Erster Buchstabe: " + anfang);
char ende = name.charAt(anzahl - 1);
println("Letzter Buchstabe: " + ende);

// Beispiel für equals
if (passwort == "z/b78!kl-P")) { // FALSCH!
  println("Login erfolgreich");
}
if (passwort.equals("z/b78!kl-P")) { // richtig
  println("Login erfolgreich");
}
```

1 James Bond

Du musst dem britischen Geheimdienst helfen verschiedene Verwaltungstätigkeiten mit ihren Geheimagenten zu automatisieren. Verwende dazu die Unterprogramme `aufgabeA, aufgabeB, …` in der Vorlage *Kap5A1_JamesBond_Vorlage*.

a) Das Programm soll diese Ausgaben erzeugen und an Vor- und Nachnamen anpassen:

```
James Bond
Mein Name ist Bond, James Bond!
Initialen: J.B.
```

b) Prüfe, ob als Name „Clark Kent" angeben wurde und gib dann „`Zutritt erlaubt`" aus, ansonsten „`Zutritt verboten`".

c) Prüfe nun nur die Initialen „JB" für den Zutritt.

d) Aus Datenschutzgründen dürfen bei machen Ausdrucken nur der Anfangs- und Endbuchstabe angegeben sein. Statt der anderen Buchstaben wird dann ein Sternchen ausgegeben: `J***s B**d`

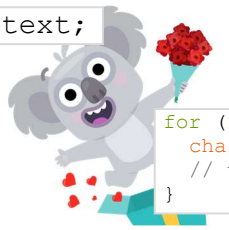
e) Gib den Namen rückwärts aus: `dnoB semaJ`

f) Tausche zwei Vokale (z.B. a und e) in der Ausgabe: `Jemas Bond`

Iteration über die Zeichen eines Strings

Eine häufige Strategie ist es jedes Zeichen eines Strings nacheinander mit einer Zählschleife abzuarbeiten. Man sagt, dass man über die Zeichen des Strings iteriert *(iterare = wiederholen)*.

```
String text;
```



```
for (int i=0; i<text.length(); i++) {
  char zeichen = text.charAt(i);
  // verarbeite das Zeichen
}
```

Manchmal muss das Vorgehen angepasst werden (wenn z.B. nur ein Teil des Strings durchlaufen werden soll, oder nur jedes zweite Zeichen verarbeitet werden muss).

Für die Aufgaben auf dieser Seite kannst du dieselbe Vorlage

Kap5_Vorlage_Texte

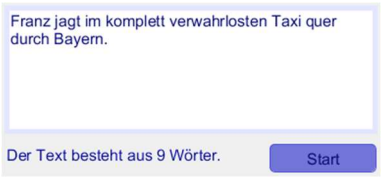
verwenden. Sie besteht aus einem Eingabefeld und einem Button der das Unterprogramm

```
String verarbeite(
                String text)
```

aufruft. Der Rückgabewert dieses Unterprogramms wird dann wieder im Programmfenster angezeigt.

Du kannst im setup-Block beim Aufruf des Unterprogramms `setzeEingabe` angeben, welche Eingabe zu Beginn im Programm erscheint, um es sinnvoll zu testen.

2 Wörter Zählen



Zähle die Wörter in der Eingabe, indem du die Leerzeichen zählst. Erstelle als Rückgabewert einen String mit Antworttext wie im Beispiel.

Bonus: Zähle auch bei einem Zeilenumbruch ein neues Wort. Da man ein manche Zeichen nicht in einfachen Anführungszeichen angeben kann gibt es in Java „Escape-Sequenzen" mit einem Backslash: So steht `'\n'` *für ein Zeilenumbruch,* `'\''` *für das einfache Anführungszeichen und* `'\\'` *für einen Backslash selbst.*

3 Vokale entfernen

Entferne alle kleingeschriebenen Vokale in der Eingabe.
Tipp: Entferne zuerst einen Vokal, z.B. alle `'a'`.

4 Palindrome

Palindrome sind Worte oder Sätze, die vorwärts und rückwärts gelesen gleich sind. Beispiele sind „ANNA", „LAGERREGAL" oder „RELIEFPFEILER". Überprüfe, ob es sich bei der Eingabe um ein Palindrom handelt.

Bonus: Palindromsätze kann man auch vorwärts und rückwärts lesen. Man muss aber oft Leerzeichen umordnen.
Bsp.: „DIE LIEBE IST SIEGER, REGE IST SIE BEI LEID."
Passe dein Programm an, dass Leerzeichen ignoriert werden und so auch der Palindromsatz erkannt wird.

Beim Datentyp `String` werden die Daten als Text interpretiert:

```
String a = "5";
String b = "7";
println(a+b); //57
```

Möchte man rechnen, muss man die Strings zuerst in einen passenden Datentyp umwandeln:

```
int x = int(a);
int y = int(b);
println(x+y); //12
```

Dieses vereinfachte Vorgehen zur Konvertierung stellt Processing zur Verfügung.

In „reinem" Java gibt es andere Vorgehen für diese Konvertierung. Zum Beispiel:

```
int x =
  Integer.parseInt(a);
```

5 Umwandlung ins Binärsystem

Die Tabelle rechts zeigt die bekannte Strategie zur Umwandlung von Zahlen ins Binärsystem.

a) Formuliere den Algorithmus zur Umwandlung ins Binärsystem in Worten.

b) Begründe ob in Java eine `for`- oder `while`-Schleife hier besser geeignet ist.

Beispiel: zahl = 26
Schleifenablauftabelle:

Durchlauf	zahl	zahl : 2	ergebnis
Nr.1	26	13 R0	"0"
Nr.2	13	6 R1	"10"
Nr.3	6	3 R0	"010"
Nr.4	3	1 R1	"1010"
Nr.5	1	0 R1	"11010"

c) Begründe, dass für das ergebnis der Datentyp `String` besser geeignet ist als `int`.

d) Implementiere den Algorithmus in Java.

```
String verarbeite(String text) {
  int zahl = int(text);
  String ergebnis = "";
  // hier Algorithmus ergänzen
  return ergebnis;
}
```

Bonus: Programmiere eine Umwandlung ins 8-er-System und ins 16-er-System (Hexadezimalsystem).

Autor: Jakob AEG RT 2021/22