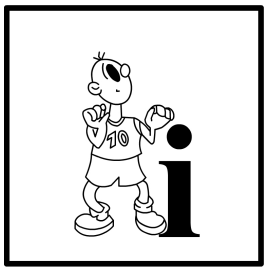
Idee, Zweck, Systembeschreibung:



In der Schule benötigen die *Personen (Lehrer, Schüler, Mitarbeiter)* ein sicheres *Passwort* für ihren Benutzeraccount. Eine Anwendung soll es ermöglichen, dass sich jede *Person* aus zwei *Worten (mit jeweils mindestens 4 Zeichen)* ein sicheres *Passwort generieren* kann. Die *Person* kann dazu auf einer *Benutzeroberfläche (Hauptfenster)* „*wort1*“ und „*wort2*“ *eingeben*. Wenn die *Person* danach die *Schaltfläche Passwort generieren* bedient, soll nach dem unten aufgeführten Regelwerk ein sicheres *Passwort erstellt* und *angezeigt* werden. Eine weitere Schaltfläche *Eingaben und Anzeige löschen* soll existieren, um die Felder auf der *Benutzeroberfläche (Hauptfenster)* zu *leeren*. Hinweis: Eingegebene Worte (wort1, wort2) müssen mindestens sechs Zeichen lang sein.

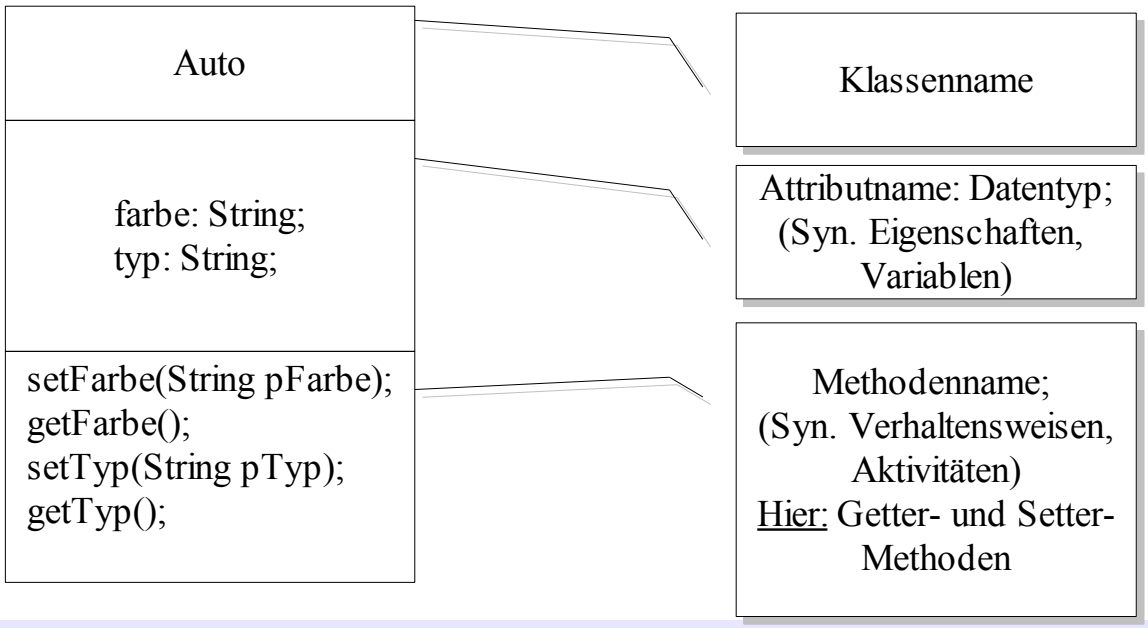
Anwendungsfälle:

		
Myra Bellamy ist *Mitarbeiterin der Schulverwaltung.* Sie gibt als *wort1* Sekretariat und als *wort2* Gymnastik ein und erhält das *Passwort* Gym4Sekr:	*Red Barklay* ist *Schüler der Berufsschule.* Er gibt als *wort1* Party und als *wort2* Wochenende ein und erhält das *Passwort* Woc4Part%	*Homer Simpson* ist *Lehrer der KSW.* Er gibt als *wort1* Hausaufgabe und als *wort2* Kontrolle ein und erhält das *Passwort* Kon4Haus!

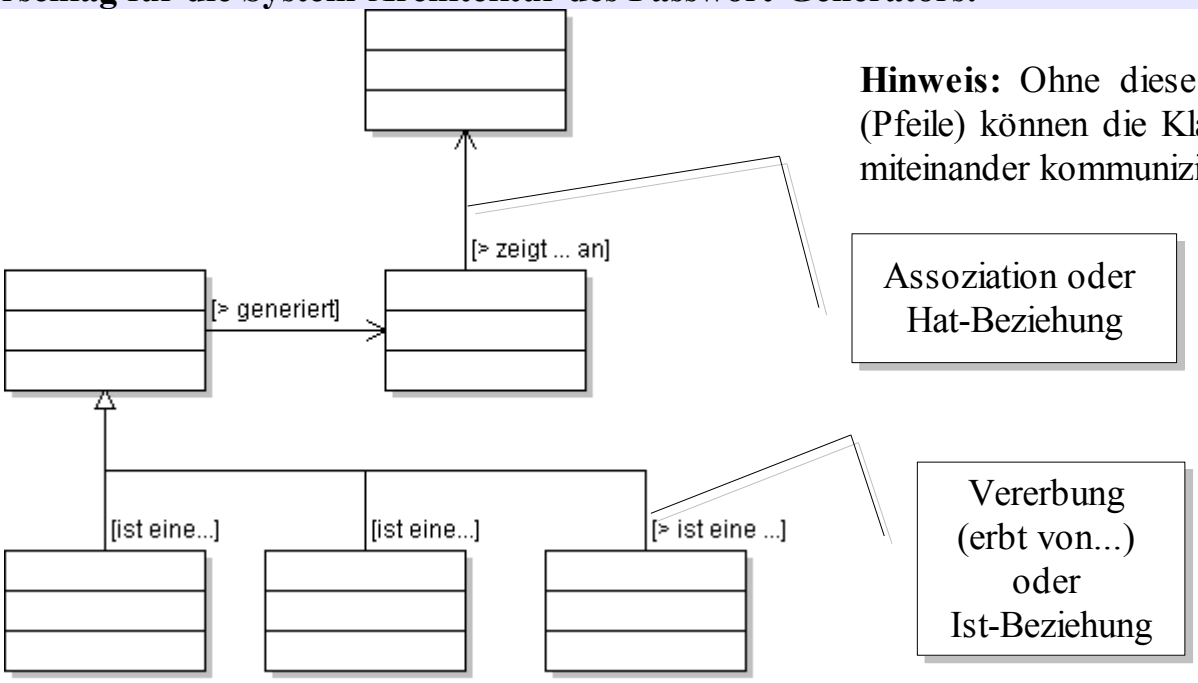
[Grafiken: The Simpsons TM, erstellt von CJ]

Merke: Sogenannte Getter- und Setter-Methoden[1] (Syn. Aktivität, Verhaltensweise) existieren für jede Eigenschaft. Z.B. Die Methoden setFarbe(String pFarbe) und getFarbe() für das Attribut (syn. Variable, Eigenschaft) String farbe der Klasse Auto. Diese Methoden dienen dazu, Eigenschaftswerte einzelner Objekte zu modifizieren (bearbeiten, ändern) bzw. erstmalig zu initialisieren (Wert setzen). Es sind quasi Teilhandlungen auf unterster Ebene (Hinweis: kleinschrittig denken).

Zur Erinnerung die UML-Notation einer Klasse:



Vorschlag für die System-Architektur des Passwort-Generators:



Hinweis: Ohne diese Verbinder (Pfeile) können die Klassen nicht miteinander kommunizieren.

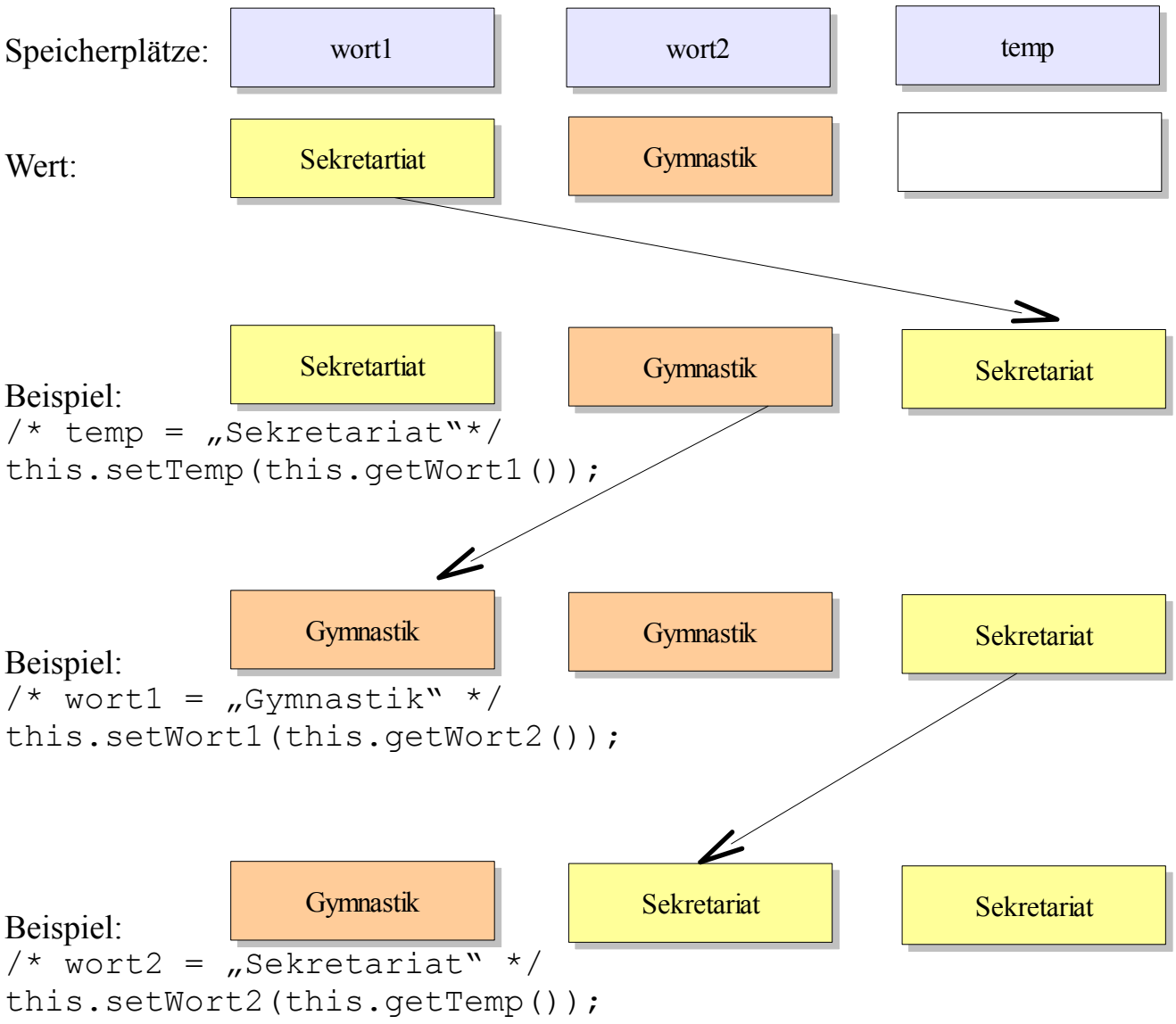
1 „get“ steht für „holen“, „set“ steht für „setzen“

[Grafiken: The Simpsons TM, erstellt von CJ]

Passwort-Regelwerk (Algorithmus):

%%Tausch: Wert in wort1 wird zu Wert in wort2 und umgekehrt.**%%**

Tausch-Algorithmus: public void tauschen(){...}



[Grafiken: The Simpsons TM, erstellt von CJ]

%%Substring von Wert in wort1: Reduziere wort1 auf die ersten drei Buchstaben.**%%**

Beispiel: aus wort1 = „Gymnastik“ wird „Gym“

```
public void reduziereWort1(){
     String temp = this.getWort1().substring(0,3);
     this.setWort1(this.getTemp());
}
```

include

exclude

index	0	1	2	3	4	5	6	7	8
wert	G	y	m	n	a	s	t	i	k

%%Substring von Wert in wort2: Reduziere wort2 auf die ersten vier Buchstaben.**%%**

Beispiel: wort2 = „Sekretariat“ wird „Sekr“

```
public void reduziereWort1(){
     String temp = this.getWort2().substring(0,4);
     this.setWort2(this.getTemp());
}
```

include

exclude

index	0	1	2	3	4	5	6	7	8	9	10
wert	S	e	k	r	e	t	a	r	i	a	t

%%Wähle per Zufall ein Sonderzeichen: wähle zufällig ein Zeichen**%%**

Beispiel:

```
//Import der API-Klasse Random
import java.util.Random;

//Attribut
private String sonderzeichen;

//Methode
public void waehleSonderzeichen(){
      String[] szliste;
      szliste = new String[] { "%", "!", ";", ":", "+"};
      Random rand = new Random();
      String zeichen = new String();

      for(int i =0; i < szliste.length;i++){
            zeichen = szliste[rand.nextInt(szliste.length)];
      }
      this.setSonderzeichen(zeichen);
}
```

Es existiert eine Liste szliste (Deklaration)

Die Liste szliste enthält folgende Sonderzeichen (Initialisierung)

[Grafiken: The Simpsons TM, erstellt von CJ]

Kontrollstrukturen: Loops mit der FOR-SCHLEIFE

Mit einer *For-Schleife* wird in unserem Fall eine Zeichenkette (Sonderzeichen, *szliste*) Zeichen für Zeichen durchlaufen. Das *i* steht für Index und bezeichnet die *Stelle in der Zeichenkette*.

Wir lesen diese Schleife wie am folgenden Beispiel erläutert:

```
for(int i = 0; i< szliste.length();i++) {
    System.out.println("mache irgendwas...");
}
```

Für den Anfang ist der Zähler 0 *(int i = 0)*. Dann durchlaufen wir die Zeichenkette, solange wir nicht am Ende angekommen sind *(solange i < als die Länge der Zeichenkette)* und zählen dann den Zähler um eins hoch *(rücken also um eine Stelle in der Zeichenkette weiter mit i++)*.

`szliste:`

index	0	1	2	3	4	...					
wert	%	!	;	:	+	...					

Nutzung finaler Attribute

Finale Attribute werden bei Werten genutzt, die sich (auch in Zukunft) nicht ändern werden. Sie benötigen keine Getter- und Setter-Methode, sondern werden direkt aufgerufen.

Beispiel:
Die Altersgrenze für die Nutzung eines Systems soll ab 18 Jahre festgelegt und gewährt werden.

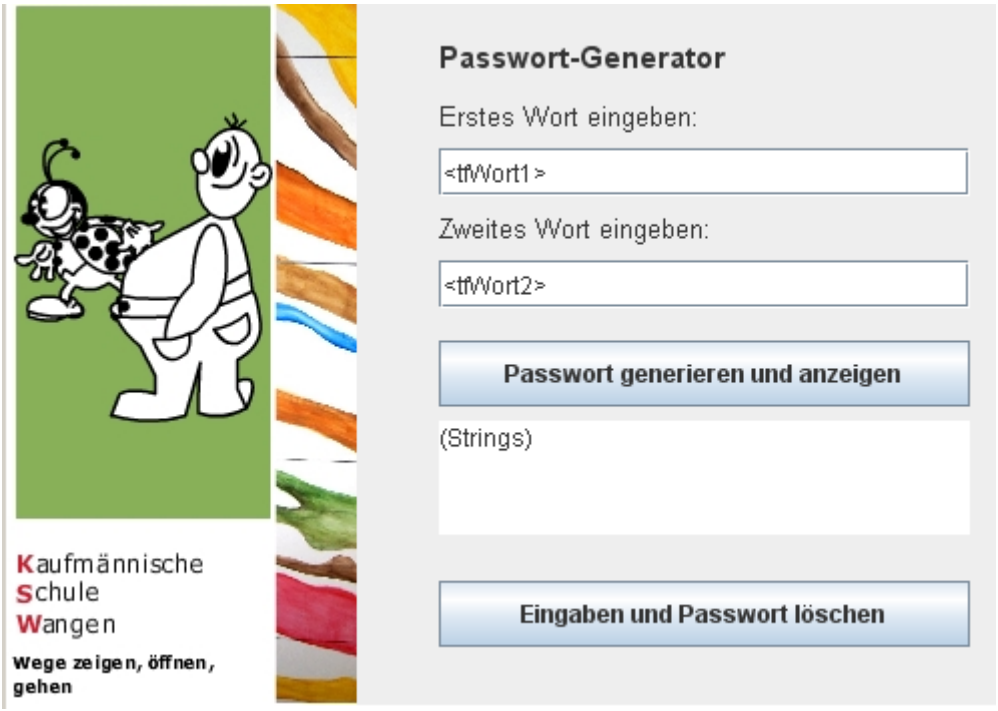
Fachklasse: Deklaration und Initialisierung

```
private static final int ALTERSGRENZE = 18;
```

Verwendung des Attributs:

```
public void pruefeAlter(pAlter){
mAl= pAlter;
     if(mAl > this.ALTERSGRENZE){
          System.out.println("Zugang gewährt!");
     }else{
          System.out.println("Zugang nicht gewährt!");
     }
}
```

So könnte die grafische Benutzeroberfläche aussehen:



[Grafiken: The Simpsons TM, erstellt von CJ]