Zustandsmodelle

M. Jakob

Gymnasium Pegnitz

10. Dezember 2014

Inhaltsverzeichnis

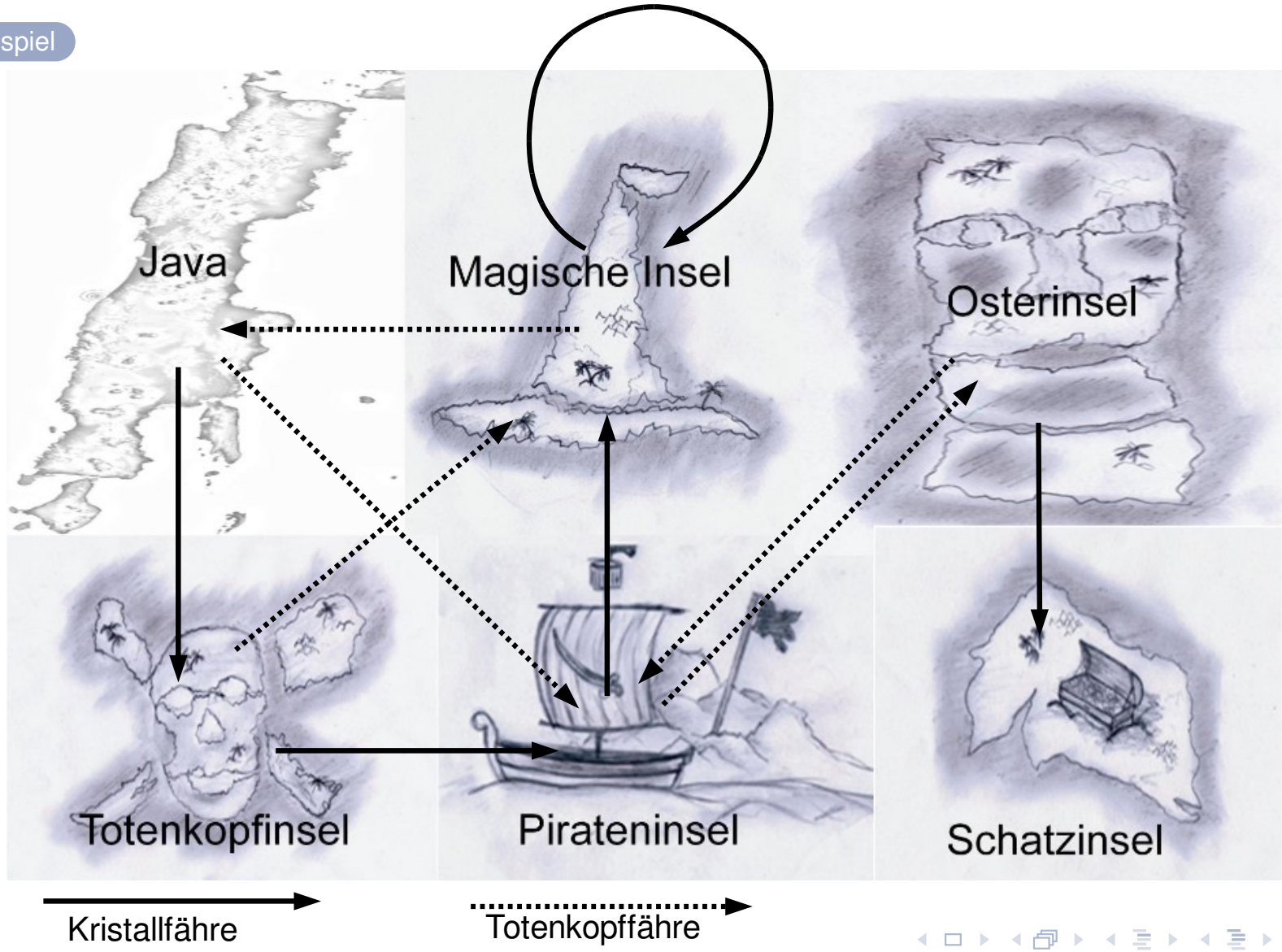
Gliederung

In diesem Abschnitt

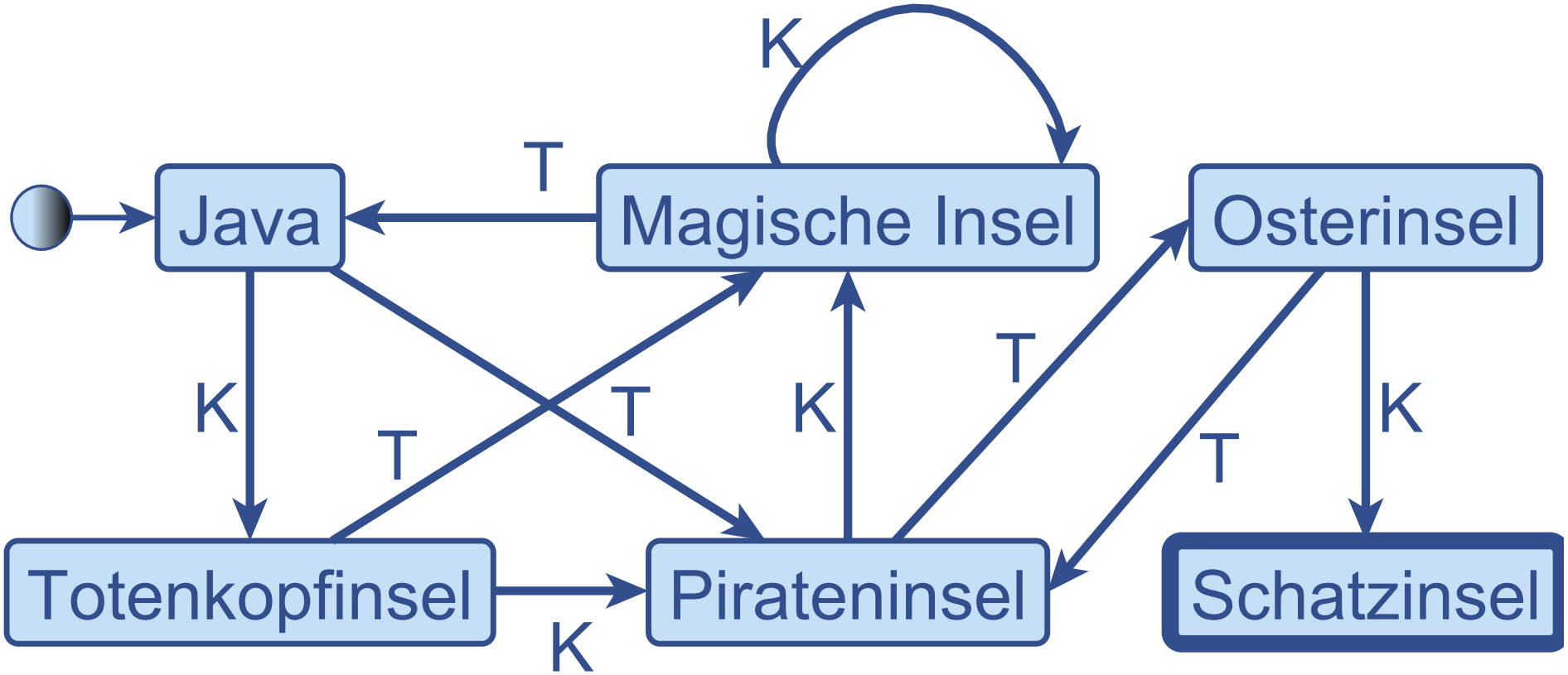
Das Inselspiel

Das Inselspiel

Inselspiel



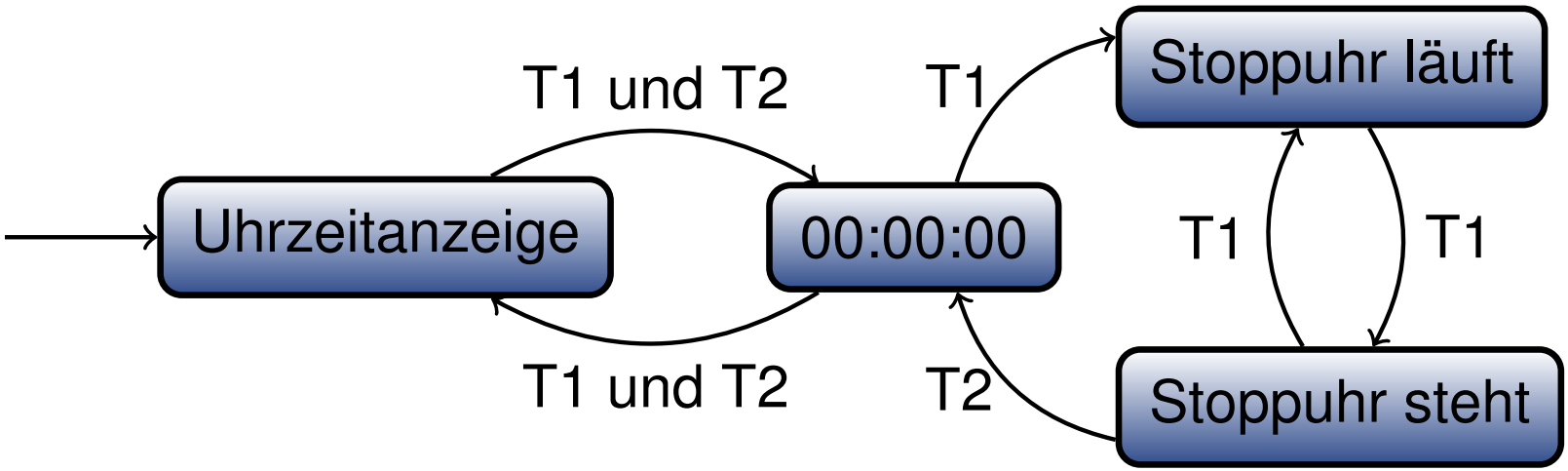
Das Inselspiel — Zustandsdiagramm



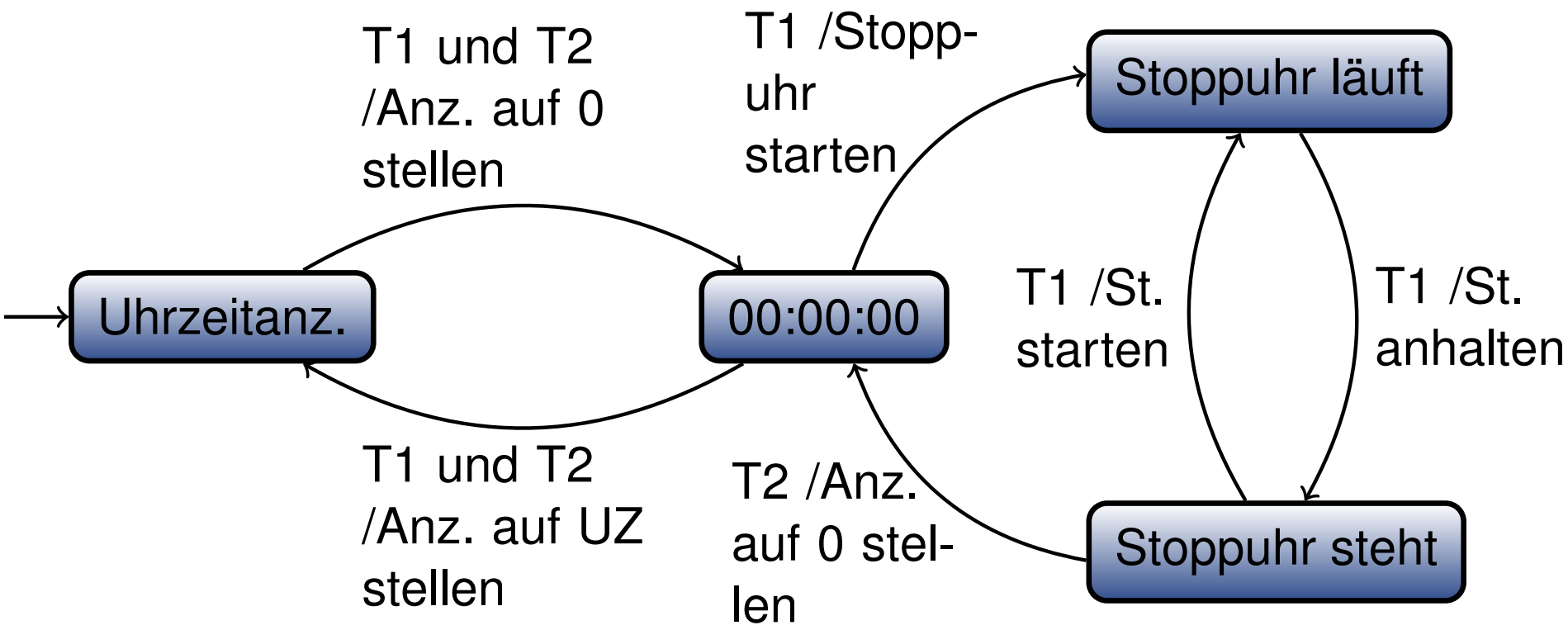
Mikrobetriebsysteme

- Elektronische Geräte (Handy, mp3-Player,...) brauchen zur Funktion eine kleines Betriebssystem, das festlegt wie sich das Gerät sich verhalten soll.
- Zur Entwicklung solcher Betriebssysteme werden zuerst Zustandsdiagramme entworfen.

Beispiel: Einfache Stoppuhr



Beispiel: Einfache Stoppuhr mit ausgelösten Aktionen



Übung

➥Ü 1.1: Zustandsübergänge mit ausgelösten Aktionen, Vorlage: ZUEDUbg.graphml

Nachfolgend sind Zustände, Ereignisse oder ausgelöste Aktionen angegeben. Ergänze die fehlenden Bausteine

(a) Telefonhörer abheben.

(b) Eine Waschmaschine hat die Waschtemperatur erreicht.

(c) Eine Heizung soll aufhören zu heizen.

(d) Bei einem Getränkeautomat soll ein Getränk im Ausgabefach liegen.

(e) Eine Braut soll heiraten.

(f) Peter Lustig soll eine E-Mail von seinem Provider abrufen können.

In diesem Abschnitt

Zustandsdiagrame in der **U**nified**M**odelling **L**anguage

UML (**U**nified **M**odelling **L**anguage) bei Zustandsdiagrammen

- Zustände werden durch abgerundete Rechtecke dargestellt.
- Ein Zustandsübergang wird durch ein (Übergangs-)Ereignis hervorgerufen und durch einen Pfeil dargestellt.
- Bei jedem Zustandsübergang kann eine Aktion ausgelöst werden.
- Jedes Zustandsdiagramm muss genau einen Anfangszustand haben (Symbol: ).
- Endzustände (Symbol: ) darf es beliebig viele geben.

Übung

Ü 1.2: Buch, Aufgabe 2.6

TA8(a) in yEd-Textfeld bearbeiten, yEd-Datei hochmoodeln

Ü 1.3: Buch, Aufgabe 2.13

Mit ausgelösten Aktionen ausführen. yEd-Datei hochmoodeln

Ü 1.4: Buch, Aufgabe 2.14

Texteditor eingemoodelt

Übung

Ü 1.5: Zahnbürste

Eine elektrische Zahnbürste lässt sich als Automat mit genau zwei Zuständen (`an` und `aus`) darstellen. Das Ereignis für einen Zustandsübergang ist das Drücken den An-/Ausknopfes, die ausgelöste Aktion das Laufen bzw. Stoppen des Motors.

(a) Zeichne das ZÜD.

(b) Ergänze das ZÜD um einen dritten Zustand `lädt`, im dem sich die Bürste beim Laden befindet. Trage auch die dazugehörigen Ereignisse und ausgelösten Aktionen ein.

Übung

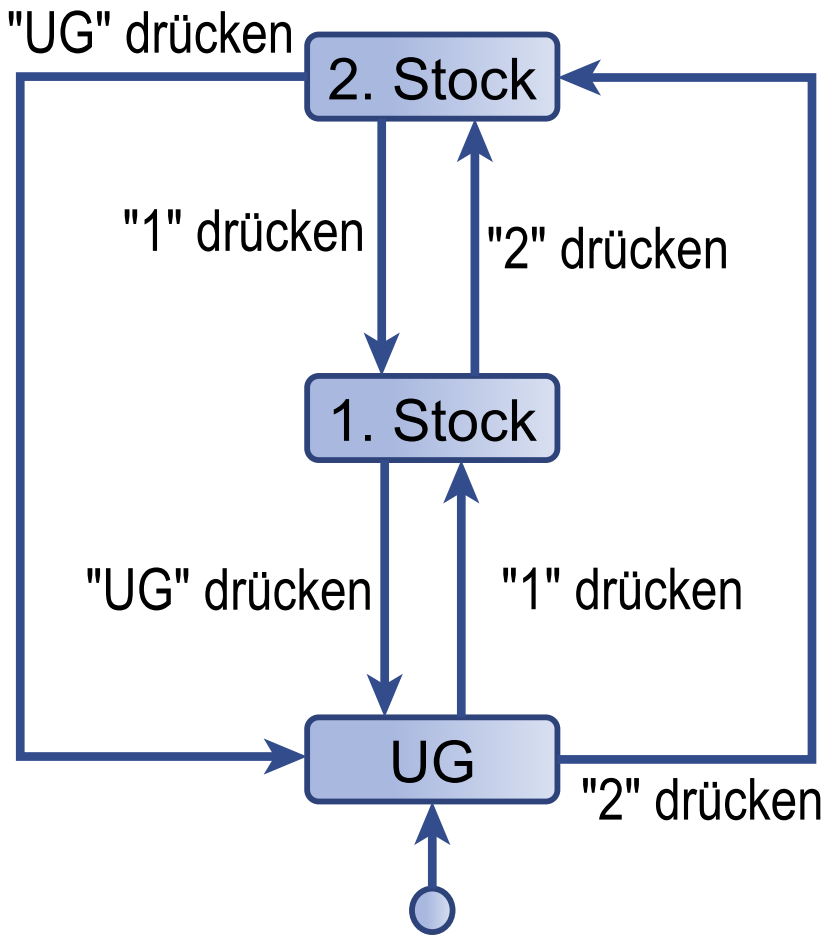
Ü 1.6: verliebt-verlobt-verheiratet-getrennt

Bekanntlich können zwei Personen verliebt, verlobt, verheiratet oder getrennt sein. Erstelle ein dazu passendes ZÜD mit Übergangsereignissen und ausgelösten Aktionen.

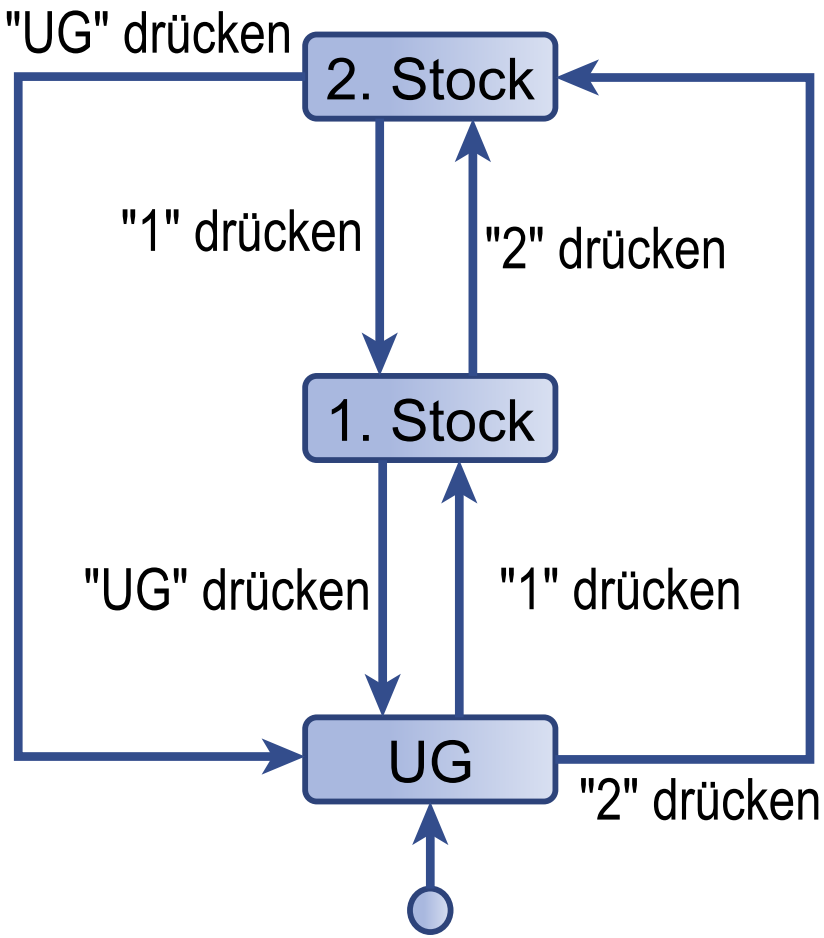
Gliederung

In diesem Abschnitt

Bespiel: Speiseaufzug



Bespiel: Speiseaufzug



Speiseaufzug
zustand: int
abschickenZu(stockwerk)

Die Zustände werden in der Variable `zustand` gespeichert, die Übergänge durch die Methode `abschickenZu` modelliert

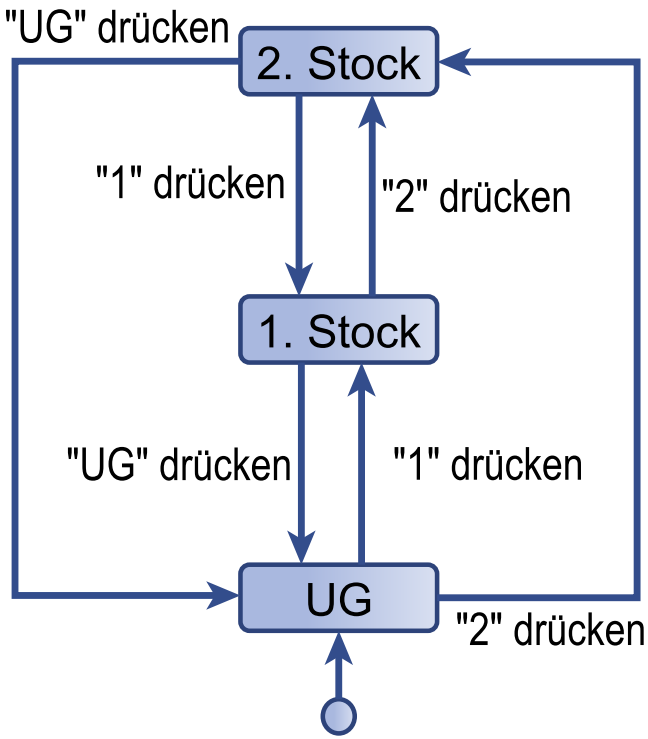
Speiseaufzug Attributdeklaration

Speiseaufzug
zustand: int
abschickenZu(stockwerk)

```
1 public class Speiseaufzug(){
2
3   private int zustand;
4
5   //Konstruktor
6   //Methoden
7 }
```

Speiseaufzug — Konstruktor

Speiseaufzug
zustand: int
abschickenZu(stockwerk)




```
1 public Speiseaufzug(){
2   zustand = 0;
3 }
```

Speiseaufzug — Methode abschicken(stockwerk)

```
1  public void abschickenZu(int stockwerk){
2
3    if ( zustand == 0 ){
4      if ( stockwerk==1 ){ zustand = 1; }
5      else {
6        if ( stockwerk==2 ){zustand = 2; }
7      }
8    }
9    if ( zustand == 1; ){
10     ...
11   }
12 }
```

Zusammenfassung

Implementation von Zustandsübergangsdiagrammen

ZÜD werden wie folgt implementiert:

- ein Attribut `zustand` deklarieren,
- im Konstruktor dem Attribut `zustand` den Anfangszustand zuweisen,
- Methode(n) für die Zustandsübergänge implementieren.
  Dabei müssen in einer Fallunterscheidung für alle Zustände alle Übergänge implementiert werden.

In diesem Abschnitt

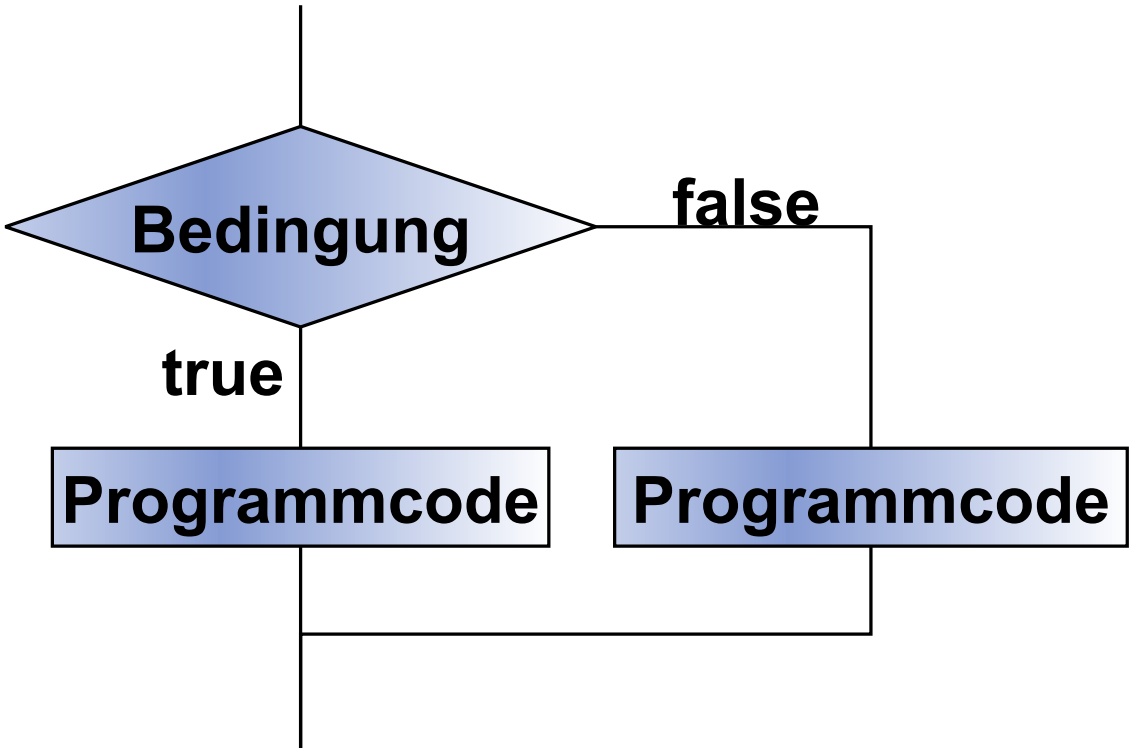
Verzweigungen

Verzweigungen

werden mit der `if (Bedingung){...} else {...}` -Kontrollstruktur umgesetzt. Der `if`-Teil wird ausgeführt, falls die Bedingung wahr ist, der `else`-Teil, falls die Bedingung falsch ist. `else`-Teil kann auch entfallen. Beispiel:

```
1 if ( zustand == 0 ){
2 System.out.println("Zustand ist Null");
3 }
4 else {
5     System.out.println("Zustand ist nicht Null")
6 }
```

Verzweigungen - Datenflussdiagramm



Fallstrick

Eine Wertzuweisung ist kein Vergleich

Wertzuweisung `A=B`

Speicherzelle A wird mit dem Wert von Speicherzelle B belegt. Speicherzelle A wird also verändert.

logischer Vergleich `A==B`

Es wird verglichen, ob die beiden Zelle A und B den gleichen Inhalt haben. Wenn „ja“ ist das Ergebnis von `A==B` gleich **`true`**, sonst **`false`**.

Vergleichsoperatoren bei Zahlen und Einzelzeichen

Name	Term	**true** wenn
Gleichheit[a]	`A==B`	A und B den gleichen Wert haben
Ungleichheit[a]	`A!=B`	A und B verschiedene Werte haben
größer/kleiner	`A>B`, `A<B`	A größer/kleiner ist als B
größer/kleiner gleich	`A>=B`, `A<=B`	A größer/kleiner oder gleich B ist.

[a] Auch bei Strings

Übungen

Ü 2.1: Speiseaufzug

Material: Paket `Speiseaufzug2Vorlage` für TA c und d.

(a) Deklariere die Klasse `Speiseaufzug` (Attribute und Konstruktor) wie oben beschrieben.

(b) Ergänze die Methode `positionGeben()`, die dem Benutzer auf der Konsole angibt, in welchem Stockwerk sich der Aufzug befindet. Dabei soll folgender Text in der Konsole erscheinen:
`Der Aufzug befindet sich im UG` bzw. `ersten Stock` bzw. `zweiten Stock`.

(c) Implementiere die Methode `abschickenZu(int stockwerk)` nach obiger Vorgabe und zeichne das Datenflussdiagramm mit yEd oder per Hand ins Heft.

Übungen

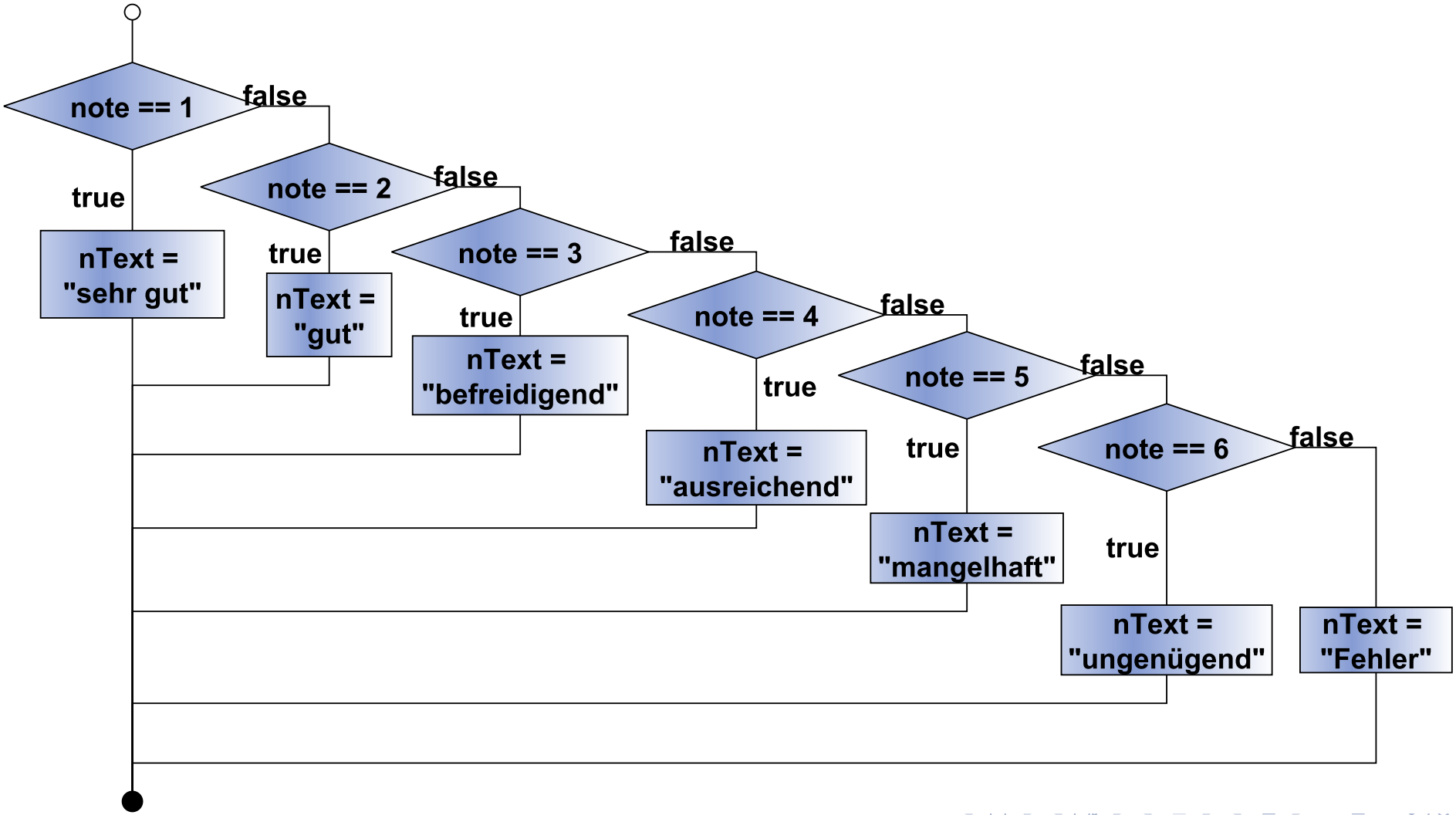
Ü 2.2: Kühlschrank

In diesem Abschnitt

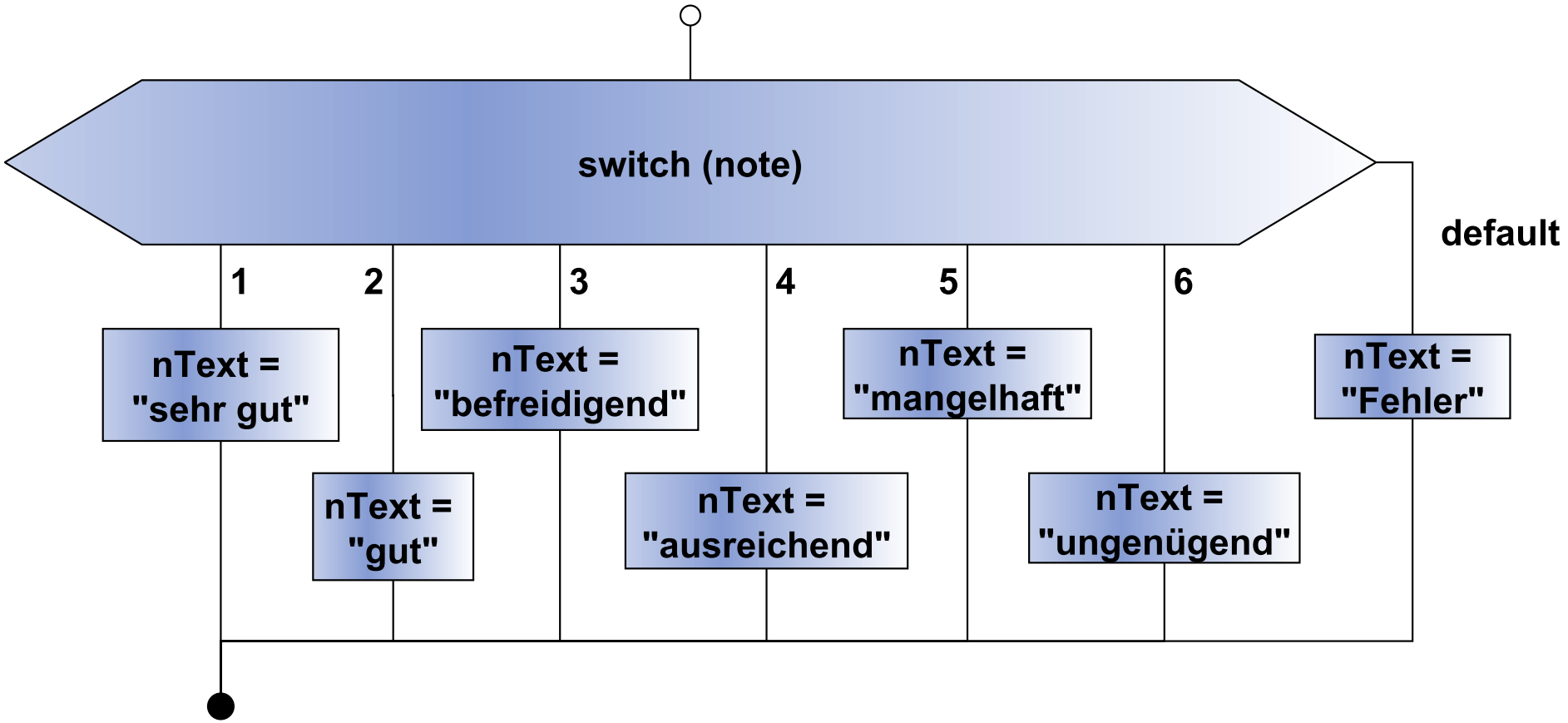
Unübersichtliche und verschachtetelte `if`-Anweisungen

```
1  public void abschickenZu(int stockwerk){
2
3    if ( zustand == 0 ){
4      if ( stockwerk==1 ){ zustand = 1; }
5      else {
6        if ( stockwerk==2 ){zustand = 2; }
7      }
8    }
9    if ( zustand == 1; ){
10     ...
11   }
12 }
```

Unübersichtliche und verschachtetelte `if`-Anweisungen



Mehr übersicht durch `switch`-Anweisung



Implementation `switch`-Anweisungen

```
1  switch ( note ) {
2      case 1: nText = "sehr gut";
3              break;
4      case 2: nText = "gut";
5              break;
6      ... //andere Fälle hier ausgelassen
7      case 6: nText = "ungenügend";
8              break;
9      default: nText = "Fehler";
10             break;
11 }
```

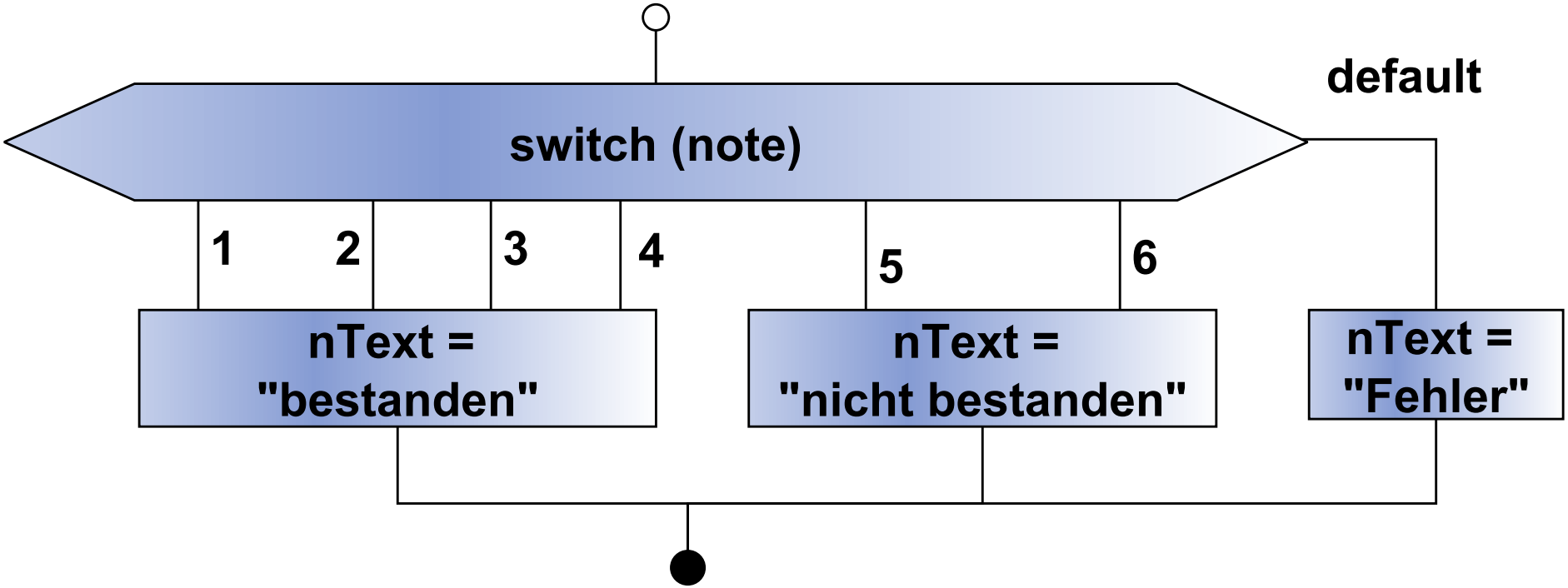
Bemerkungen zur `switch`-Anweisung

- Eine `switch`-Anweisung kann beliebig viele `case`-Klauseln enthalten.
- Die `break`-Anweisung nach einer `case`-Klausel sorgt dafür, dass nicht(!) die nächste `case`-Klausel bearbeitet, sondern ans Ende der `switch`-Anweisung gesprungen wird.
- Die `default`-Klausel ist optional und sammelt alle Fälle, die vorher nicht aufgeführt sind.

Fälle zusammenfassen

```
1  switch ( note ) {
2      case 1:
3      case 2:
4      case 3:
5      case 4: nText = "bestanden";
6              break;
7      case 5:
8      case 6: nText = "nicht bestanden";
9              break;
10     default: nText = "Fehler";
11             break;
12 }
```

Fälle zusammenfassen



Ü 2.3: `switch`-Anweisungen

(a) Implementiere die beiden oben aufgeführten Beispiele, die Noten in Prädikate bzw. Noten in (nicht) bestanden umsetzen.
Erzeuge dazu in BlueJ eine Klasse `Switchen` und darin die beiden Methoden `public String noteToPraedikat(int note)` und `public String noteToBestanden(int note)`, in die du die `switch`-Anweisungen überträgst.

(b) Erstelle mit Hilfe von `switch`-Anweisungen die Methode `public String monatToMonatsname(int monat)`, die zur Nummer des Monats seine Bezeichnung ausgibt. `monatToMonatsname(9)` soll also `September` zurückgeben. Implementiere auch die Umkehrmethode `monatsnameToMonat`.

(c) Erstelle mit Hilfe von `switch`-Anweisungen die Methode `public String anzahlTageBestimmen(String monat)`, die für jeden Monat die Anzahl seiner Tage ausgibt (Schaltjahre nicht vergessen).

Gliederung

In diesem Abschnitt

Begriffsbestimmung

Wiederholungen

Prgogrammabschnitte können durch Wiederholungsbefehle mehrfach ausgeführt werden. Man nennt Programmbereiche, die wiederholt werder auch Schleifen. Dabei unterscheidet man:

Bedingte Wiederholungen: Die Schleife wird nur ausgeführt, wenn eine bestimmte Bedingung erfüllt ist.

Wiederholungen mit fester Anzahl: Ein vorgegebener Wert legt fest, wie oft eine Schleife ausgeführt wird.

In diesem Abschnitt

Wiederholung mit fester Anzahl

Pseudocode

```
1 wiederhole n mal
2     Anweisungen
3 endewiederhole
```

Java-Syntax

```
1 for (int i=0;i<n;i=i+1) {
2     Anweisungen
3 }
```

In diesem Abschnitt

Bedingte Wiederholung

Pseudocode

```
1 wiederhole wenn (
      Bedingung erfüllt)
2     Anweisungen
3 endewiederhole
```

Java-Syntax

```
1 while (logischer
      Ausdruck) {
2     Anweisungen
3 }
```

Beispiel: Roboter im Dauerbetrieb

```
1   public void alleKugelnAussortieren(String
        gewFarbe){
2       while (true) {
3           kugelAussortieren(gewFarbe);
4       }
5   }
```

Beispiel: Roboter sucht Kugel

```
1   public void kugelSuchen(String gewFarbe){
2       String hatFarbe;
3       robbi.greifen();
4       hatFarbe=robbi.kugelFarbeGeben();
5       //Schleife
6       while (!(hatFarbe.equals(gewFarbe))){
7           kugelTransportierenZuTopf('A');
8           robbi.greifen();
9           hatFarbe=robbi.kugelFarbeGeben();
10      }
11  }
```

Beispiel: Benutzer wählt Anzahl

```
1   public void nKugelnTransportieren(int anzahl,
        char topf){
2       int zaehler;
3       zaehler=0;
4       while (zaehler<anzahl){
5           kugelTransportierenZuTopf(topf);
6           zaehler=zaehler+1;
7       }
8   }
```

Übungen

➥Ü 3.1: Roboter4Schleifen: Weitere Robotermethoden

(a) Implementiere die oben genannten Beispiele der Robotersimulation.

(b) Implementiere eine Methode, die so lange Kugeln in Topf 'A' abgelegt, bis zweimal hintereinander die gleiche Farbe kommt.

Lösung mit Variablentabelle erklären

(c) Implementiere eine Methode, die den Topf wechselt, wenn zweimal hintereinander die gleiche Farbe kommt.