# Algorithmik

M. Jakob

Gymnasium Pegnitz

9. Februar 2015

# Inhaltsverzeichnis

# Begriffsbestimmung Algorithmus

# Begriffsbestimmung Algorithmus

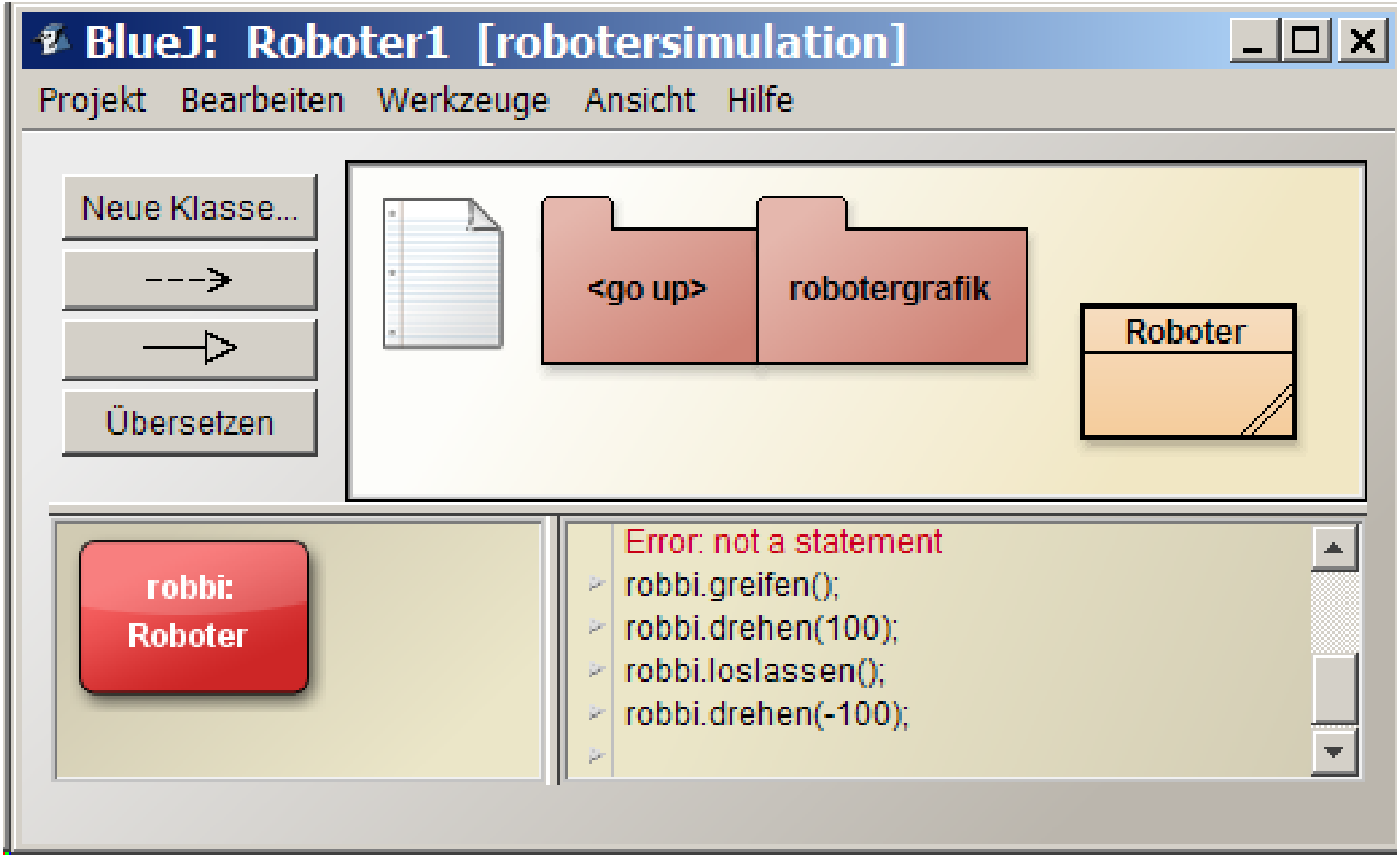
**Definition**

Ein Algorithmus ist eine Verarbeitungsvorschrift, die aus endlich vielen, eindeutig ausführbaren Anweisungen besteht.

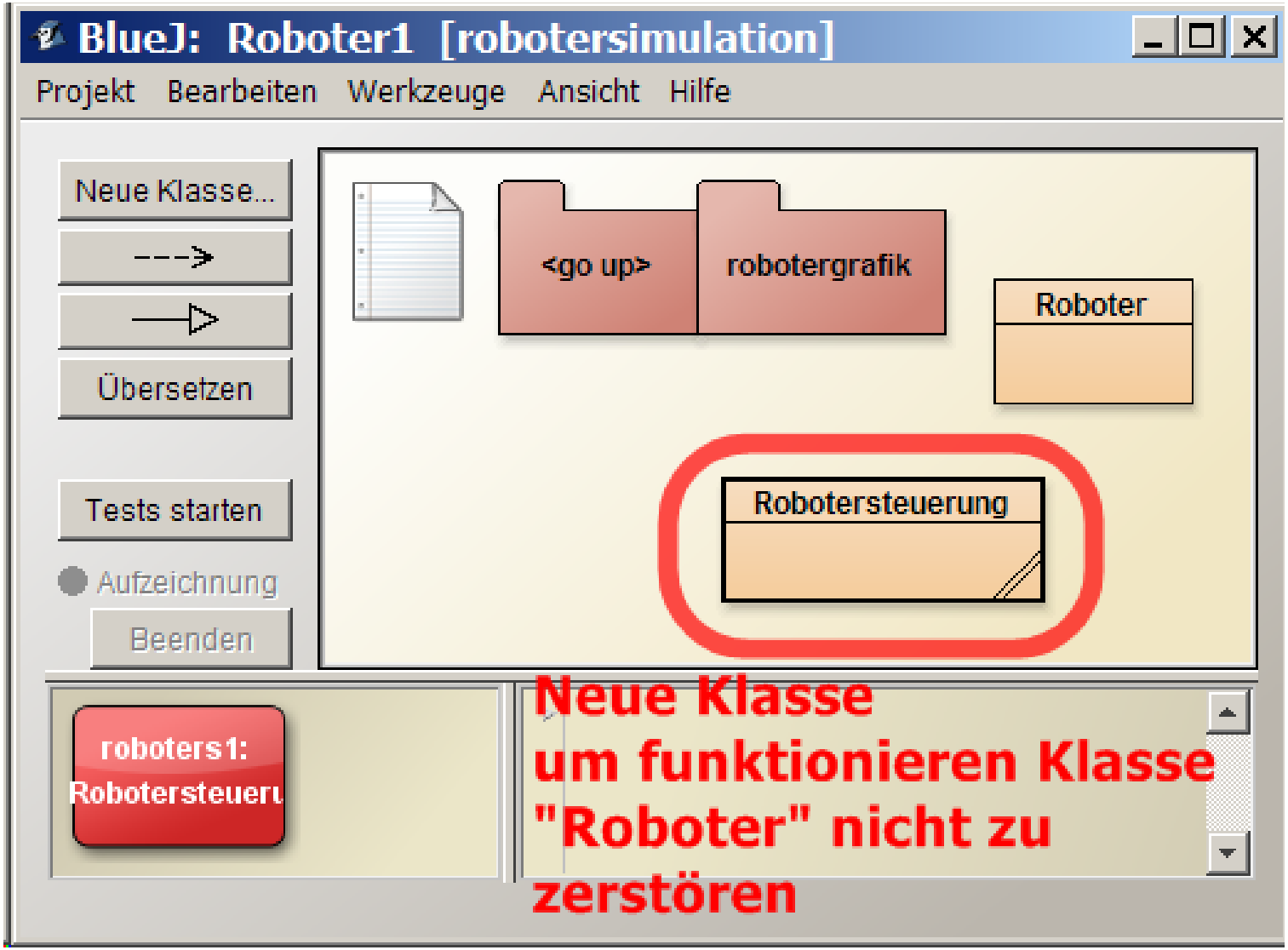
Beispiel:

- schriftlicher Multiplikations-Algorithmus
- Computerprogramme
- Kochrezepte und Gebrauchsanweisungen (mit Einschränkungen)

# Beispiel: Roboter Kugel in Topf A



# Und jetzt als Methode



# Never change an running system

Ein bestehendes und nicht selbst programmiertes Programm wird niemals verändert, weil

- die Gefahr viel zu groß ist, dass man vorhandene Programmteile, nicht genau versteht und deshalb zerstört,
- weil man den Überblick verliert.

Deshalb erzeugen wir eine neue Klasse `Robotersteuerung`, die die Dienste der Klasse `Roboter` nutzt.

# Deklaration und Konstruktor

```
1  package robotersimulation;
2  public class Robotersteuerung
3  {    Roboter robbi;
4
5      public Robotersteuerung()
6      {
7          robbi=new Roboter();
8      }
```

# . . . und die erste Methode

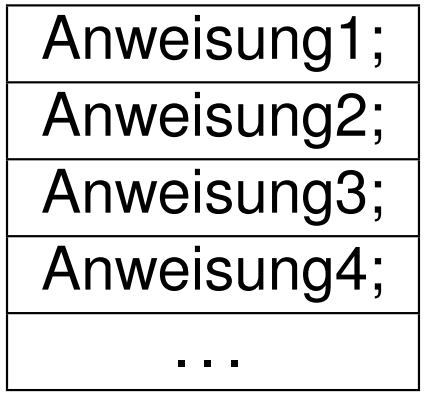
```
1    public void kugelInTopfATransportieren(){
2        robbi.greifen();
3        robbi.drehen(100);
4        robbi.loslassen();
5        robbi.drehen(-100);
6    }
```

# In diesem Abschnitt

# Sequenzen

## Definition

Eine Folge von Anweisungen, die nacheinander abgearbeitet werden, heißt Sequenz.

| Java-Beispiel | allgemein | Blockdiagramm |
|---|---|---|
| `robbi.greifen();`<br>`robbi.drehen(winkel);`<br>`robbi.loslassen();`<br>`robbi.drehen(-winkel);` | `Anweisung1;`<br>`Anweisung2;`<br>`Anweisung3;`<br>`...;` |  |

# Übungen I

## ➥Ü 2.1: Roboter1Sequenzen

(a) Implementiere die Klasse `Robotersteuerung` und die Methode `kugelInTopfATransportieren()` wie oben beschrieben.

(b) Implementiere entsprechend die Methode `kugelInTopfBTransportieren()`, die eine Kugel in Topf B transportiert.

(c) Der Benutzer legt einen Drehwinkel fest. Der Roboterarm soll eine Kugel holen und nach Drehung um den gewünschten Winkel fallenlassen (`kugelTransportieren(int winkel)`).

# Übungen II

## Ü 2.2: Fortsetzung Roboter1Sequenzen

Implementiere in der Klasse `Robotersteuerung` Methoden, die folgende Dienste erfüllen(lege die Methodensignatur selbst fest). Hinweis: Nutze die in der letzten Übung definierte Methoden.

(a) Zehn Kugeln hintereinander in den Topf A ablegen.

(b) Zehn Kugeln hintereinander in den Topf B ablegen.

(c) Je Zehn Kugeln abwechseln in Topf A und Topf B ablegen.

(d) Der Benutzer legt einen Drehwinkel fest. Der Roboterarm soll zehn Kugel holen und nach Drehung um den gewünschten Winkel fallenlassen.

(e) Der Benutzer legt zwei Drehwinkel fest. Der Roboterarm soll zwanzig Kugeln holen, abwechseln um die gewünschten Winkel drehen und fallenlassen.

# In diesem Abschnitt

# Mängel der bisherigen Robotersteuerung

- Die Mehrfachbewegung einer Kugel ist umständlich/unmöglich zu programmieren.
- Die Zieltöpfe tauchen im Methodennamen auf (z.B. `kugelInTopfB()`). Die Topfbezeichnung wäre besser als Parameter, also `kugelInTopf(char zielTopf)`

# Mängel der bisherigen Robotersteuerung

- Die Mehrfachbewegung einer Kugel ist umständlich/unmöglich zu programmieren.
- Die Zieltöpfe tauchen im Methodennamen auf (z.B. `kugelInTopfB()`). Die Topfbezeichnung wäre besser als Parameter, also `kugelInTopf(char zielTopf)`

## Kontrollstrukturen

Neben Sequenzen kann ein Programm auch Verzweigungen, auch bedingte Anweisungen genannt, und Wiederholgungen enthalten.

# Bedingte Anweisung — Pseudocode

```
1  public class Robotersteuerung {
2      Roboter robbi;
3      ...
4      public void kugelTransportieren(int winkel){
5          ...
6      }
7      public void kugelTransportierenZuTopf(char
           topf){
8          wenn (Kugel zum Topf A soll){
9              kugelTransportieren(100);
10         }
11         sonst{
12             kugelTransportieren(160);
13         }
14     }
15 }
```

# Bedingte Anweisung — Pseudocode

```
1  public class Robotersteuerung {
2          ...
3      public void kugelTransportierenZuTopf(char
           topf){
4          wenn (Kugel zum Topf A soll){
5              kugelTransportieren(100);
6          }
7          sonst{
8              kugelTransportieren(160);
9          }
10     }
11 }
```

# Bedingte Anweisung — Pseudocode

```
1    public void kugelTransportierenZuTopf(char
         topf){
2        if (Kugel zum Topf A soll){
```

wird zu

```
1    public void kugelTransportierenZuTopf(char
         topf){
2        if (topf == 'A'){
```

# Roboter — Methode kugelTransportierenZuTopf

```
1  public class Robotersteuerung {
2      Roboter robbi;
3      ...
4      public void kugelTransportieren(int winkel){
5          ...
6      }
7     public void kugelTransportierenZuTopf(char topf
          ){
8          if (topf == 'A'){
9              kugelTransportieren(100);
10         }
11         else{
12             kugelTransportieren(160);
13         }
14     }
15 }
```

# Verzweigungen — Zusammenfassung

Eine Programmverzweigung entscheidet aufgrund eines logischen Ausdrucks, welcher Teil des Programms abgearbeitet wird.

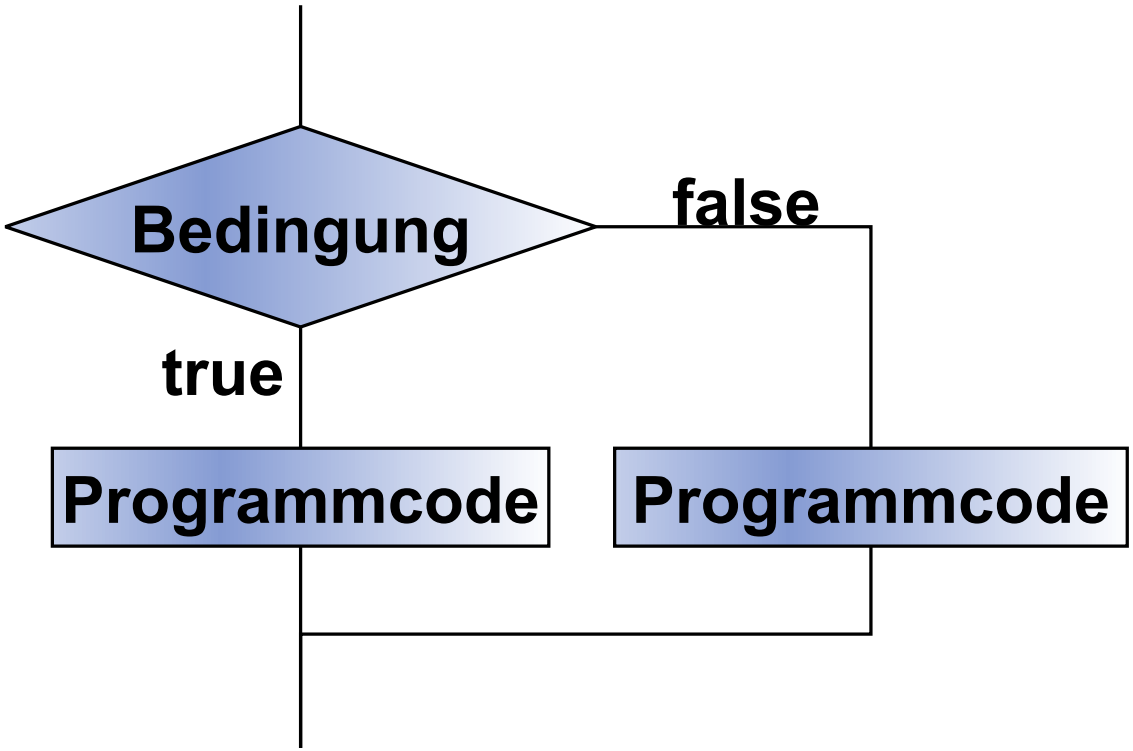
```
1 wenn <logischer Ausdruck>
2     dann auszuführende Anweisungen falls
                  <logischer Ausdruck> == wahr
3     sonst auszuführende Anweisungen falls
                 <logischer Ausdruck> == falsch
4 endewenn
```

# Verzweigungen — Java-Syntax

```
1 if (<logischer Ausdruck>)
2     {
3         auszuführende Anweisungen falls  <logischer
              Ausdruck> == true
4     }
5     else
6     {
7         auszuführende Anweisungen falls <logischer
              Ausdruck> == false
8     }
9 }
```

Bemerkung: Der `else`-Block kann weggelassen werden.

# Verzweigungen - Datenflussdiagramm



# Übung

## ➥Ü 2.3: Roboter2Verzweigungen

(a) Implemeniere die Methode `kugelTransportierenZuTopf`

(b) Bei welchen Eingaben landet eine Kugel im Topf 'B'?

(c) Schreibe die Methode `kugelTransportierenZuTopf` so um, dass Kugeln nur bei der Eingabe von 'A' bzw. 'B' in dem entsprechenden Topf landen.

(d) Ergänze die Methode `kugelTransportierenZuTopf` so, dass außerdem eine Benutzerinformation ausgegeben wird, wenn ein falscher Buchstabe für den Topf eingegeben wurde.

## ➥Ü 2.4: Programmcode analysieren

# Logische Ausdrücke . . .

sind Ausdrücke, deren Ergebnis vom Datentyp `boolean` ist.
Beispiele:

| Test auf | Bespiel-Term | Ergebnis |
|---|---|---|
| Gleichheit | `5==7` | **`false`** |
| Ungleichheit | `5!=6` | **`true`** |
| größer/kleiner | `4>3` / `4<3` | **`true`** / **`false`** |
| größer/kleiner gleich | `4>=3` / `4<=3` | **`true`** / **`true`** |

Als logische Ausdrücke sind auch Methoden erlaubt, deren Rückgabewert vom Datentyp `boolean` ist.
Beispiel aus der Roboterklasse: `hatOffeneHand()`

# 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`**.

# Zusammenfassung Vergleichsoperatoren bei einfachen Datentypen und `String`

| Name | Term | **`true`** wenn |
|---|---|---|
| Gleichheit | `A==B` | A und B den gleichen Wert haben |
| Ungleichheit | `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. |

# Zusammengesetzte logische Ausdrücke

## Zusammengesetzte logische Ausdrücke

| Name | Term | **true** wenn | Beispiel |
|---|---|---|---|
| Und | `L && M` | L und M wahr sind | `(x>1)&& (x<5)` |
| Oder | `L \|\| M` | L oder M[1] wahr | `(x>1)\|\| (x<-5)` |
| Nicht | `!L` | L falsch ist. | `!hatOffeneHand()` |

[1] oder beide

# Beispiel: Benutzerfreundlicher Transport

```
1  public void kugelTransportierenZuTopf(char topf){
2      if ( (topf == 'A') || (topf =='a') ){
3          kugelTransportieren(100);
4      }
5      else{
6          if ( (topf == 'B') || (topf =='b') ){
7              kugelTransportieren(160);
8          }
9      }
10 }
```

# Beispiel: Eine bestimmte Farbe in 'A' geben

```
1  public void kugelAussortieren(String gewFarbe){
2      String hatFarbe;
3      robbi.greifen();
4      hatFarbe=robbi.kugelFarbeGeben();
5      if (hatFarbe == gewFarbe){
6          kugelTransportierenZuTopf('A');
7      }
8      else {
9          kugelTransportierenZuTopf('B');
10     }
11 }
```

# Übungen

➥Ü 2.5: Roboter3Verzweigungen: Weitere Robotermethoden

(a) Verbessere die Methode `kugelTransportierenZuTopf(char topf)` wie in obigem Beispiel „Benutzerfreundlicher Transport“ und implementiere obige Methode `kugelAussortieren(String gewFarbe)`.

(b) Implementiere eine Methode, die drei Kugelfarben aussortieren kann.

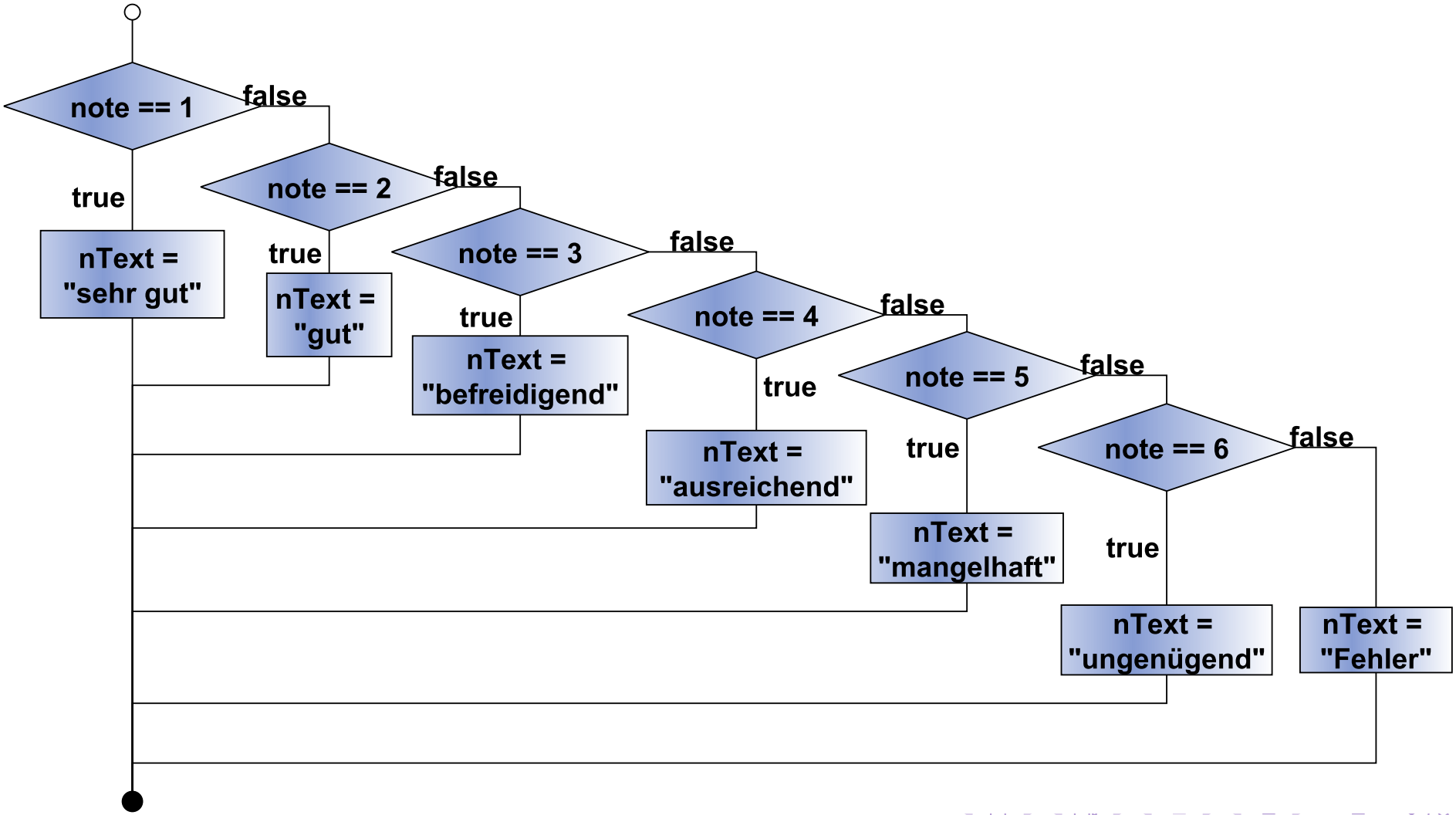
(c) Schreibe eine Methode, die zwei Kugelfarben aussortiert und dabei die Methode zum Aussortieren von drei Kugeln benutzt.

➥Ü 2.6: Notenberechnung

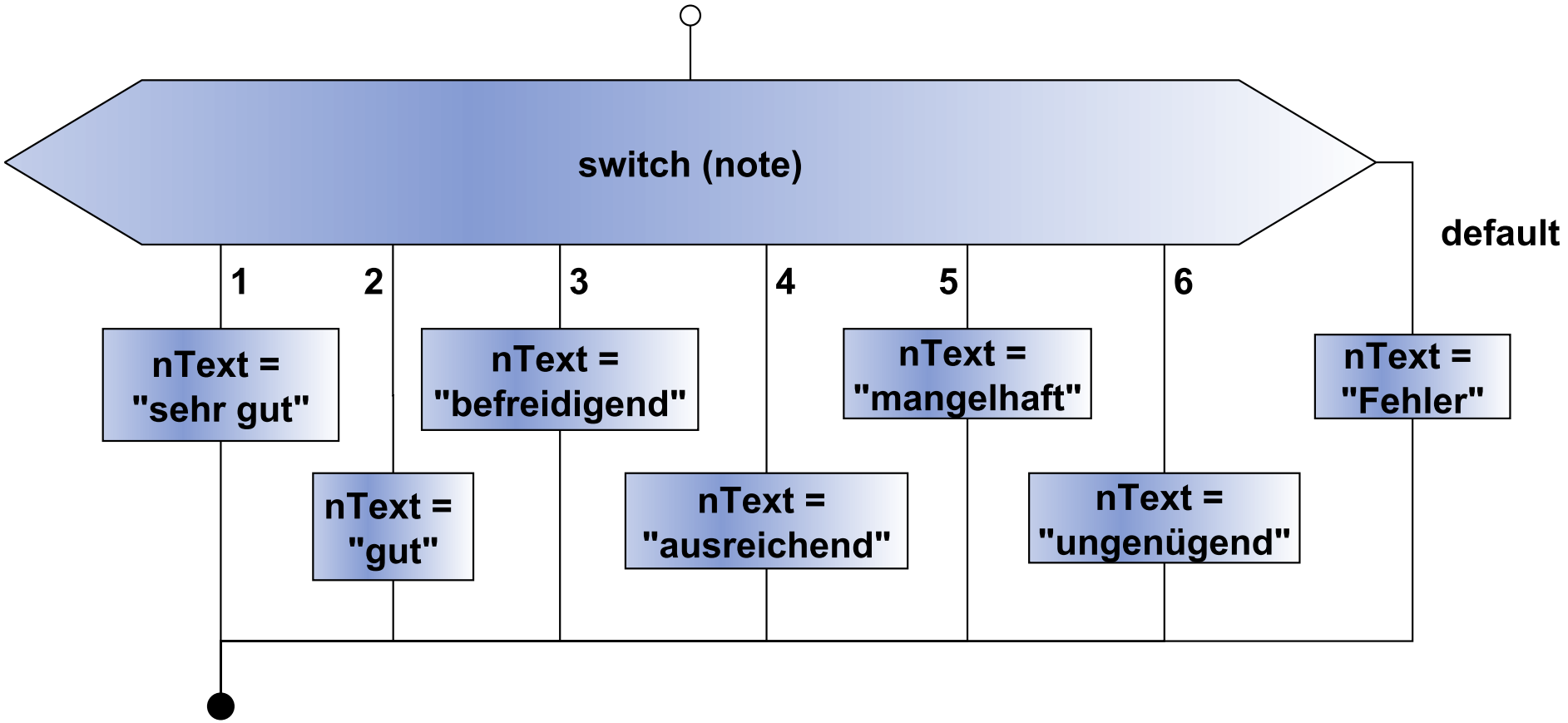
# Unübersichtliche und verschachtetelte `if`-Anweisungen

```
1  if ( note == 1 ){
2    System.out.println("sehr gut");
3    }
4  else {
5    if ( note == 2 ){
6      System.out.println("gut");
7      }
8    else {
9      if ( note == 3 ){
10       system.out.println("befriedigend");
11     }
12   ...
13   }
14 }
```

# 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 }
```

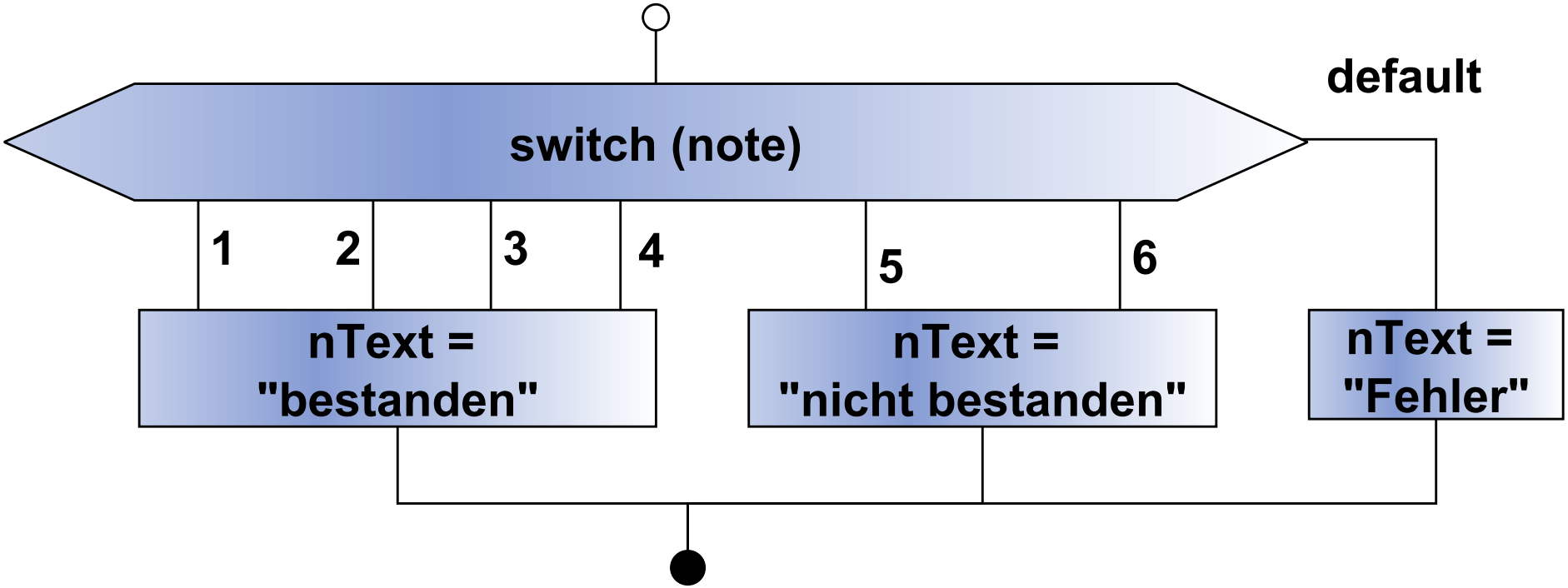
# Bemerkungen zur `switch`-Anweisung

- 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 2.7: `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).

# 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.

# 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 != 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 2.8: Roboter4Schleifen: Weitere Robotermethoden

(a) Implementiere die oben genannten Beispiele der Robotersimulation.

(b) Implementiere eine Methode `nKugelnEinerFarbeInTopfA(int anzahl, String farbe)` die so lang arbeitet, bis `anzahl` Kugeln der Farbe `farbe` in Topf `'A'` sind und alle anderen in Topf `'B'`.

(c) Implementiere eine Methode `stopBeiZweimalGleicheFarbe()`, die so lange Kugeln in Topf 'A' abgelegt, bis zweimal hintereinander die gleiche Farbe kommt.
Hinweis: Du musst obige Methode `kugelSuchen(String gewFarbe)` abwandeln. Lösung mit Variablentabelle erklären

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

# Beispiel: Benutzer wählt Anzahl — Version 1

```
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   }
```

Die Anzahl der Durchläufe, durch die Schleife steht fest. Für solche Situationen gibt es auch eine andere Syntax:

# Beispiel: Benutzer wählt Anzahl — Version 2

```
1        int zaehler;
2        zaehler=0;
3        while (zaehler<anzahl){
4            kugelTransportierenZuTopf(topf);
5            zaehler=zaehler+1;
```

wird zu

```
1    for(int zaehler=0;zaehler<anzahl;zaehler=
         zaehler+1){
2        kugelTransportierenZuTopf(topf)
3    }
```

# Wiederholungen mit fester Anzahl

## Pseudocode

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

## Java-Syntax

```
1 for (lokale Zaehlvariable;
      Bedingung;updateVar) {
2     Anweisungen
3 }
```

# Übungen

➥Ü 2.9: SchleifentrainerVorlage: Paket kennenlernen: Käsekästchen zeichnen
➥Ü 2.10: SchleifentrainerVorlage: Methoden nach Vorgabe implementieren