Eine Einführung in den Kontrollfluss durch ein JavaKara-Programm:
- Aufruf von Methoden (ohne Parameter, ohne Rückgabewerte)
- Programmausführung mit Call Stack

JavaKara programmieren: Methoden

Ein ganz einfaches Programm: Kara dreht sich um

```
public void myProgram() {
  kara.turnRight();
  kara.turnRight();
}
```

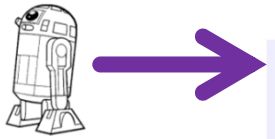
Ein ganz einfaches Programm: Einen neuen «Befehl» einführen

```
public void myProgram() {
  umdrehen();
}

void umdrehen() {
  kara.turnRight();
  kara.turnRight();
}
```

Ein ganz einfaches Programm: Methoden definieren und aufrufen

```
public void myProgram() {          Methode definieren
  umdrehen();                      Methode aufrufen
}

void umdrehen() {                  Methode definieren
  kara.turnRight();                Methode aufrufen
  kara.turnRight();                Methode aufrufen
}
```

Programmausführung in Zeitlupe: Ein einfaches Programm



```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Der Prozessor
Der Arbeitsknecht, der das Programm zum Leben erweckt. Er weiss, welche Programmzeile er als nächstes ausführen muss.

Bei JavaKara ist der Start immer in myProgram().

Das Programm
Der Programmtext, der ausgeführt werden soll.

Die Welt («Daten»)
Die Welt, in der das Programm ausgeführt werden soll. Sie enthält den Käfer, den der Prozessor steuert.

Programmausführung in Zeitlupe: Anleitung für ein Rollenspiel

```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Der Prozessor
Eine Schülerin spielt den Prozessor. Sie hat die Hauptrolle in dieser Aufführung.

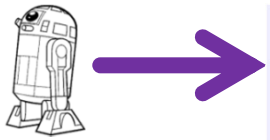
Das Programm
Ein Schüler erhält das auszuführende Programm.

Die Welt («Daten»)
Eine Schülerin steuert zB an einem Computer mit Beamer den Marienkäfer.

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

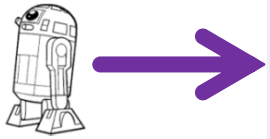
```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



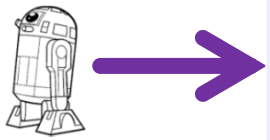
```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



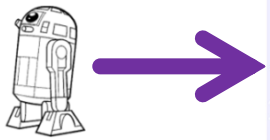
```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



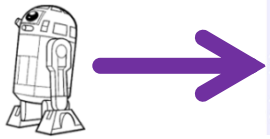
```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



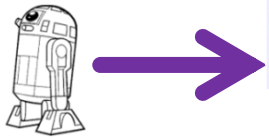
```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



```
5   public void myProgram() {
6     kara.turnLeft();
7     kara.move();
8     kara.turnRight();
9     kara.move();
10    kara.turnRight();
11  }
```

Der Prozessor
Bei JavaKara ist der Start immer in myProgram().

Wenn diese Methode abgearbeitet ist, hat der Prozessor seine Arbeit erledigt.

Das Programm
Der Programmtext, der ausgeführt werden soll.

Die Welt («Daten»)
Die Welt, in der das Programm ausgeführt werden soll. Sie enthält den Käfer, den der Prozessor steuert.

Programmausführung in Zeitlupe: Ein komplizierteres Programm

```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
```


Der Prozessor
Er weiss, welche Programmzeile er als nächstes ausführen muss.

Er muss neu beim Aufruf einer Methode merken, von welcher Programmzeile aus er eine Methode aufgerufen hat.

Das Programm
Es enthält jetzt eigene Methoden, die eigene Methoden aufrufen.

Die Notizen des Prozessors («Call Stack»)
Mit einem Stapel von Notizkarten kann sich der Prozessor merken, von welcher Programmzeile aus er eine Methode aufgerufen hat.

Alternativ als Tabelle dargestellt.

Programmausführung in Zeitlupe: Anleitung für ein Rollenspiel

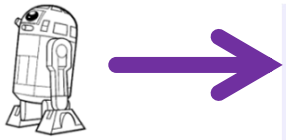
```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
```


Der Prozessor
Eine Schülerin spielt den Prozessor. Sie hat die Hauptrolle in dieser Aufführung.

Das Programm
Je ein Schüler erhält eine Methode des auszu-führenden Programms. Damit muss der Prozessor rumlaufen, das soll den Methodenaufruf verdeutlichen.

Die Notizen des Prozessors («Call Stack»)
Die Schülerin, welche den Prozesser spielt , erhält einen Stapel Notizzettel. Bei jedem Methodenaufruf schreibt sie auf einen neuen Zettel, aus welcher Methode und Zeile der Aufruf kommt.

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```


Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
myProgram

Zeile
6

myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
ganze Drehung
Zeile
10

ganzeDrehung: 10
myProgramm: 6

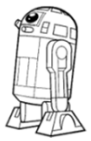
Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
dreiViertel
Drehung
Zeile
15

dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
halbe
Drehung
Zeile
28

halbeDrehung: 28
dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
halbe
Drehung
Zeile
28

halbeDrehung: 28
dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
halbe
Drehung
Zeile
28

halbeDrehung: 28
dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
halbe
Drehung
Zeile
28

halbeDrehung: 28
dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
halbe
Drehung
Zeile
28

halbeDrehung: 28
dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

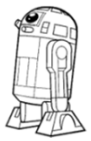
Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
dreiViertel
Drehung
Zeile
15

dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile



```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
halbe
Drehung
Zeile
29

halbeDrehung: 29
dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
dreiViertel
Drehung
Zeile
15

dreiViertelDrehung: 15
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
ganze Drehung
Zeile
10

ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
dreiViertel
Drehung
Zeile
16

dreiViertelDrehung: 16
ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
ganze Drehung
Zeile
10

ganzeDrehung: 10
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
myProgram

Zeile
6

myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
ganze Drehung
Zeile
11

ganzeDrehung: 11
myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

```
5   public void myProgram() {
6     ganzeDrehung();
7   }
8
9   void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```

Methode
myProgram

Zeile
6

myProgramm: 6

Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

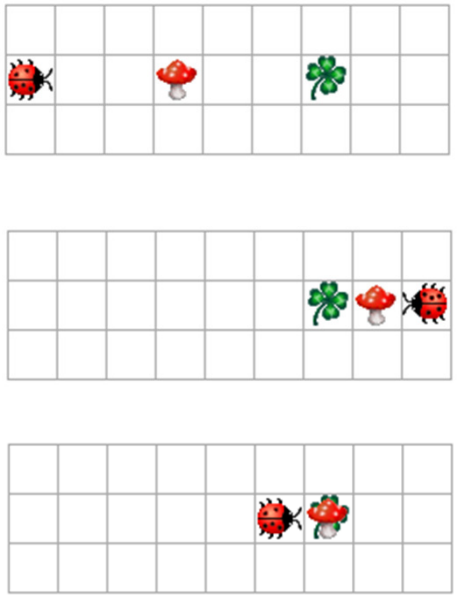
```
 5  public void myProgram() {
 6    ganzeDrehung();
 7  }
 8
 9  void ganzeDrehung() {
10    dreiViertelDrehung();
11    viertelDrehung();
12  }
13
14  void dreiViertelDrehung() {
15    halbeDrehung();
16    viertelDrehung();
17  }
18
19  void viertelDrehung() {
20    kara.turnLeft();
21    kara.move();
22    kara.turnRight();
23    kara.move();
24    kara.turnRight();
25  }
26
27  void halbeDrehung() {
28    viertelDrehung();
29    viertelDrehung();
30  }
```


Programmausführung in Zeitlupe: Zeile für Zeile für Zeile

Der Kontrollfluss bei der Ausführung eines Programms (noch ohne Daten; Methoden haben weder Input noch Output):

- Der Prozessor führt ein Programm Zeile für Zeile für Zeile aus.
- Ein Programm kann viele Methoden haben.
- Der Prozessor merkt sich mit Hilfe eines Stapel Notizkarten, woher ein Methodenaufruf kommt, damit er dorthin zurückkehren kann.
- Der Stapel kann gross werden, aber nicht unendlich wachsen (was passiert sonst?).

Programmlesbarkeit: Kara soll Pilz auf Blatt schieben

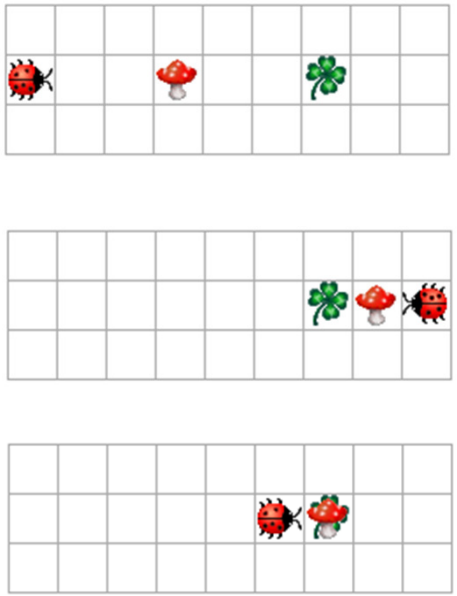


```
public void myProgram() {
  while (!kara.onLeaf()) {
    kara.move();
  }
  kara.turnLeft();
  kara.move();
  kara.turnRight();
  kara.move();
  kara.move();
  kara.turnRight();
  kara.move();
```

```
// Fortsetzung
  kara.turnRight();
  kara.move();
  kara.turnLeft();
  kara.move();
  kara.turnRight();
  kara.move();
  kara.move();
  kara.turnRight();
  kara.move();
  kara.turnRight();
}
```

Das Hauptprogramm ist unübersichtlich: Es ist unklar, welche Befehle inhaltlich zusammengehören und welche nicht. Und es hat viel doppelten Code!

Programmlesbarkeit: Kara soll Pilz auf Blatt schieben



```
public void myProgram() {
  suchePilz();
  umPilzHerum();
  kara.move();
  umPilzHerum();
}

void suchePilz() {
  while (!kara.onLeaf()) {
    kara.move();
  }
}
```

```
void umPilzHerum() {
  kara.turnLeft();
  kara.move();
  kara.turnRight();
  kara.move();
  kara.move();
  kara.turnRight();
  kara.move();
  kara.turnRight();
}
```

Das Hauptprogramm ist so schon fast natürlich-sprachig formuliert.
Die Absicht der einzelnen Methoden wird durch deren Namen klar ausgedrückt.

Java: Keine Methoden in Methoden

```
public void myProgram() {
  umdrehen();

  void umdrehen() {
    kara.turnRight();
    kara.turnRight();
  }
}
```

Java: Keine Methoden in Methoden

```
public void myProgram() {
  umdrehen();
}

void umdrehen() {
  kara.turnRight();
  kara.turnRight();
}
```