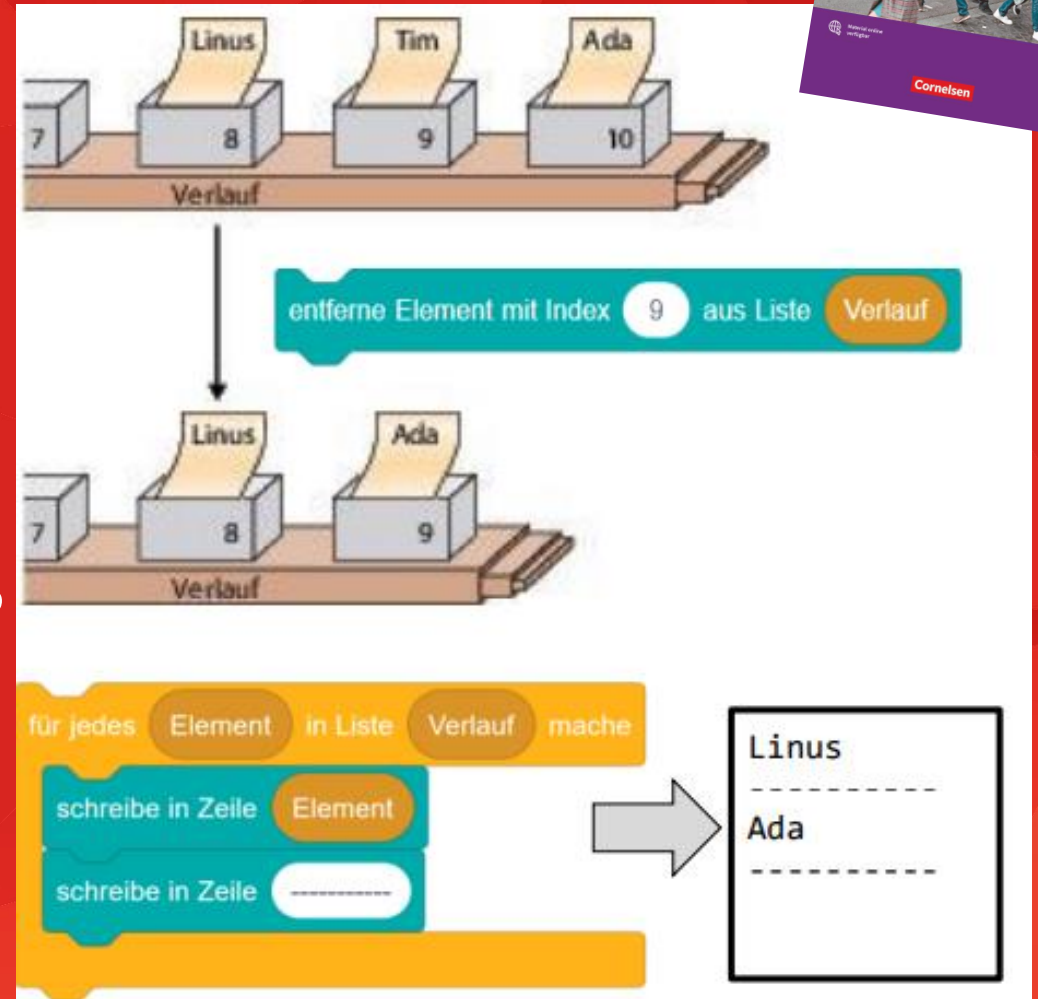
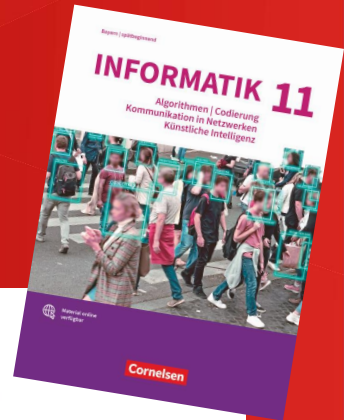


Informatik spät beginnend – Algorithmik praxisorientiert

Peter Brichzin

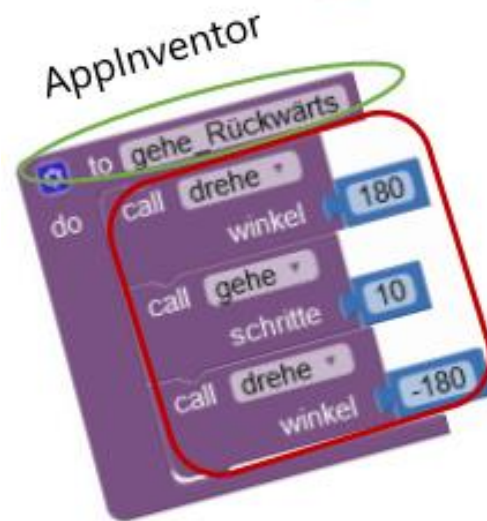
- didaktische Ansätze und Überlegungen
 - Werkzeuge (blockbasiert / textbasiert)
 - Aufgaben aus motivierenden Kontexten
- Das alles passend zum neuen LehrplanPLUS



Didaktische Überlegungen – selbstdefinierte Methoden

- Strukturierung von Information
Modularisierung als zentrales informatisches Konzept
- grundlegende Begriffe der Objektorientierung
Methode versus Methodenaufruf

Programmiersprachen
unabhängig Strukturen
aufzeigen und
Visualisieren



Python

```
def RückwaertsGehen():  
    Drehen(180)  
    Gehen(10)  
    Drehen(-180)
```

Java

```
void RückwaertsGehen(){  
    Drehen(180);  
    Gehen(10);  
    Drehen(-180);  
}
```

grün: Methodenkopf
rot: Methodenrumpf mit einer Sequenz aus drei Anweisungen.

Didaktische Überlegungen – selbstdefinierte Methoden

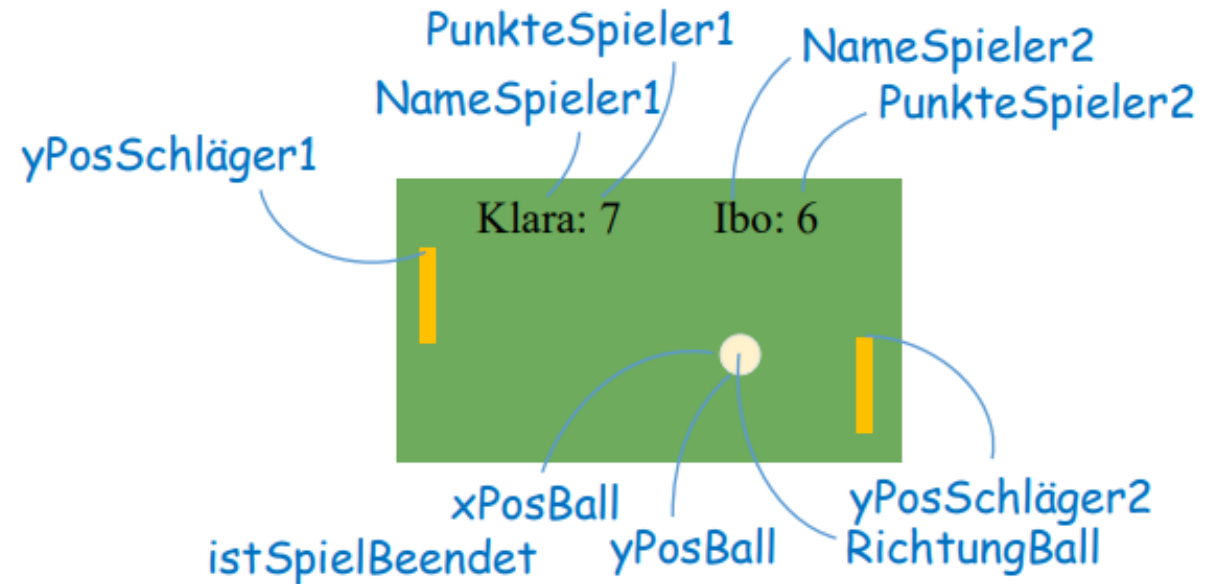
- Strukturierung von Information
Modularisierung als zentrales informatisches Konzept
- grundlegende Begriffe der Objektorientierung
Methode versus Methodenaufruf
- **Frühzeitig**
Mehrwert (Lesbarkeit; Wiederverwendung) herausheben



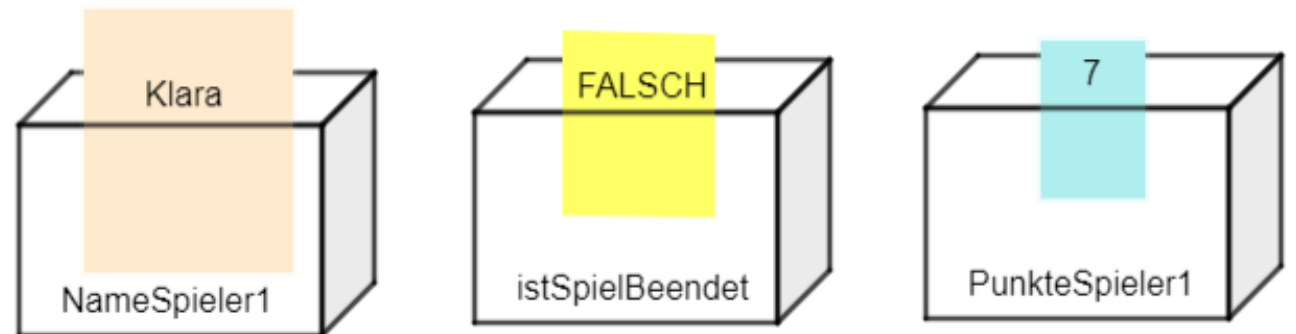
Didaktische Überlegungen – Variablen

Bits und Bytes sind nicht sichtbar.

Gebt den Lernenden Bilder, z. B. das Schachtelmodell für Variablen (Attribute)

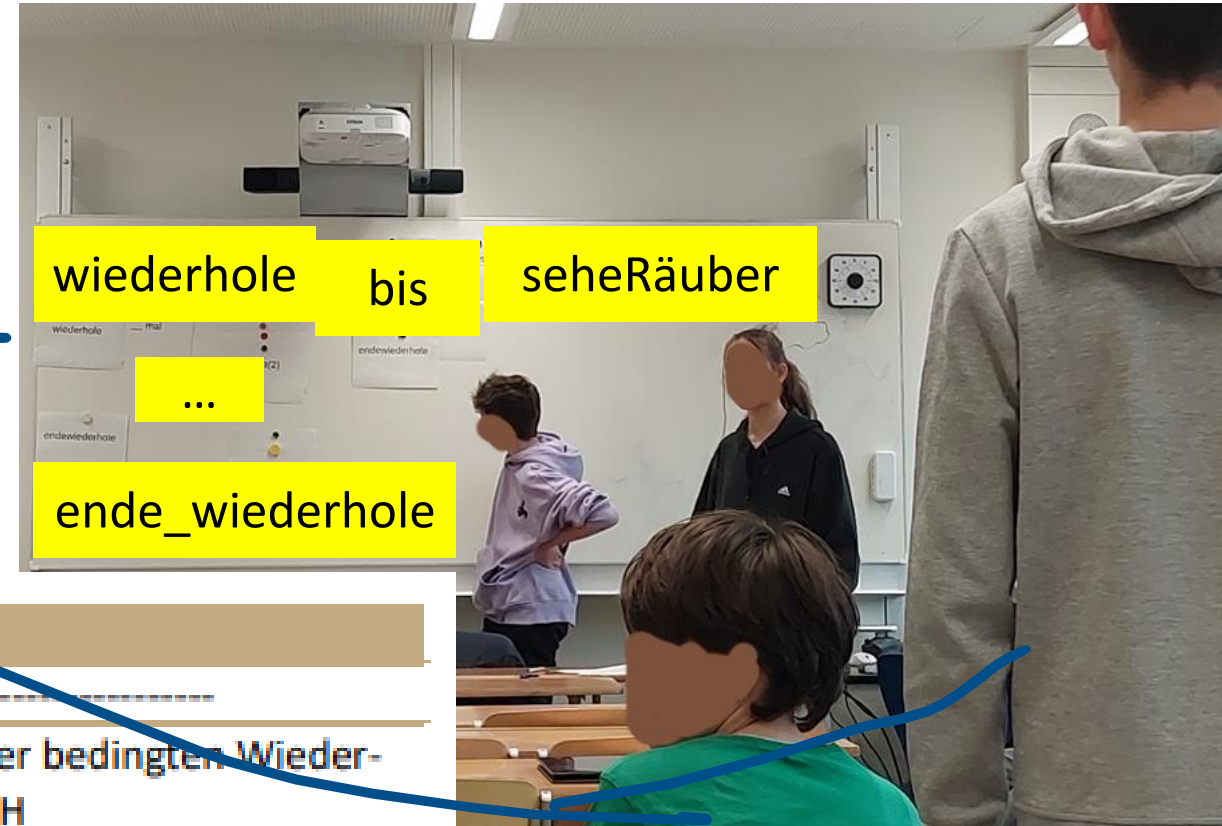


Visualisierungen



Didaktische Überlegungen – Kontrollstrukturen/Algorithmik

- Abläufe sichtbar machen:*
- *Schrittweiser Ablauf an der Tafel*
 - *Methodenaufrufe steuern lebende Figuren*
 - *Schreibtischtest*



Zeile in Ausführung	Zahl	Erläuterung
1	5	-----
2	5	Auswertung der Bedingung in der bedingten Wiederholung: (Zahl = 30) ergibt FALSCH
3	5	Auswertung der Bedingung in der bedingten Anweisung: istGerade(Zahl) ergibt FALSCH
5	5	sonst-Fall (der dann-Fall wurde übersprungen)
6	5	erster Durchlauf der Wiederholung mit fester Anzahl
7	10	-----
8	10	Ende der Wiederholungsanweisung

Teste die selbst:
Kompetenzorientierte
Lernzielkontrolle

Kontrollstrukturen/Algorithmik – Darstellung von Quelltext

Kontrollstrukturen legen die Reihenfolge der Abarbeitung von Anweisungen fest.



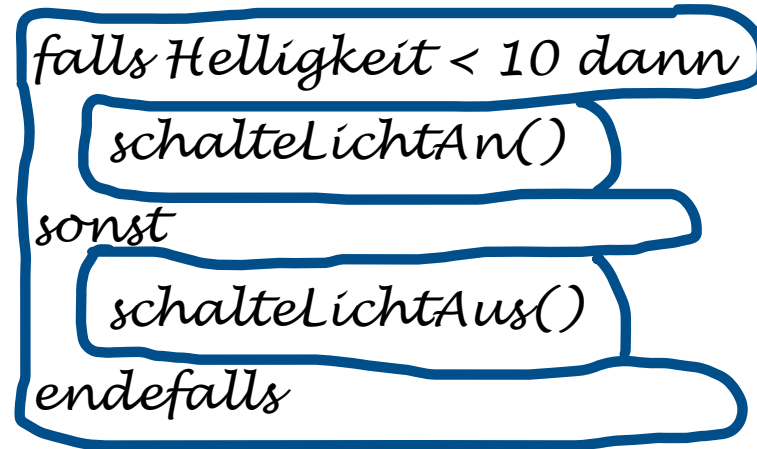
Blöcke in Snap!



Text mit Einrückung

```
falls Helligkeit < 10 dann
    schalteLichtAn()
sonst
    schalteLichtAus()
endefalls
```

grafische Darstellung



Informatische Konzepte vermitteln, kein Werkzeug

Struktogramm (verstehen)

Helligkeit < 10	
wahr	falsch
SchalteLichtAn ()	SchalteLichtAus ()

Kontrollstrukturen/Algorithmik – Darstellung von Quelltext

Im Beispiel rechts spricht man von einer Schachteltiefe 2, weil es neben der übergeordneten bedingten Anweisung noch eine untergeordnete Ebene gibt.



geschachtelte Prüfung auf eine Schülerermäßigung für den Kinotag

geschachtelte Prüfung auf eine Schülerermäßigung ohne Kinotag

```
setze Preis auf (9.90)
falls istKinotag? dann
    falls istSchüler? dann
        reduziere Preis um 4
    sonst
        reduziere Preis um 2
    endeFalls
sonst
    falls istSchüler? dann
        reduziere Preis um 2
    endeFalls
endeFalls
```

Didaktische Überlegung - Eindimensional indizierte Datenstruktur

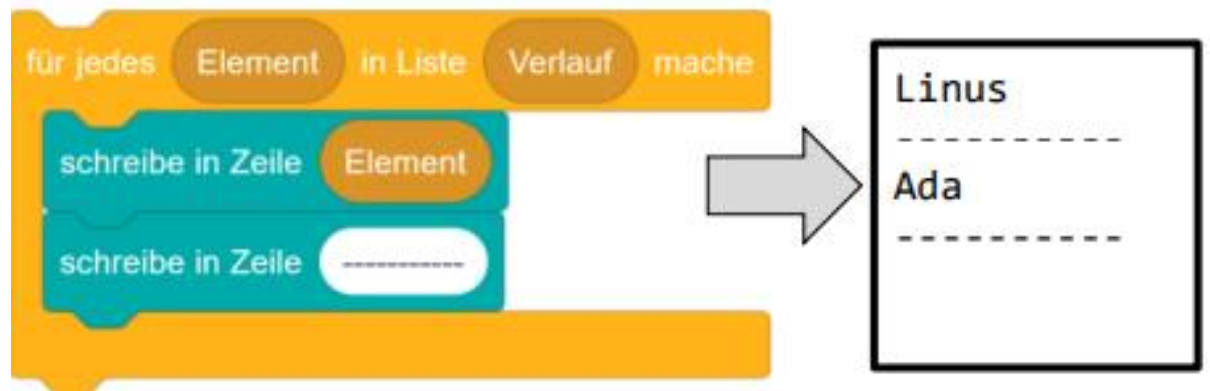
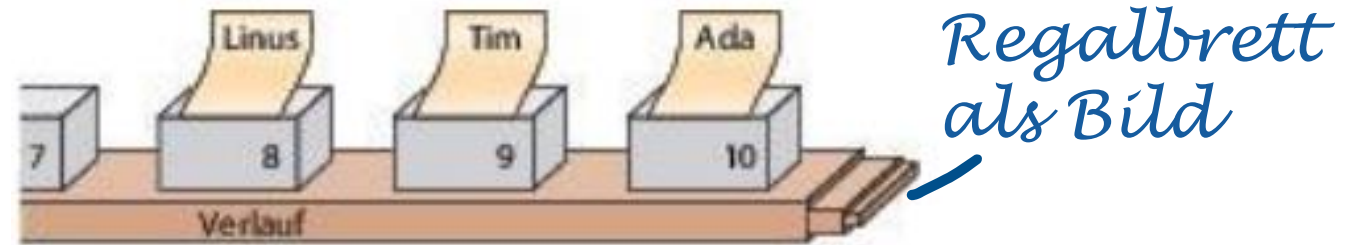
Hintergrundwissen für Lehrkräfte
Liste / Feld

(Jgst. 11 spb/ Jgst. 10 NTG)

- Zugriffskomplexität $O(1)$
- Blackboxsicht
- Gerne dynamisch

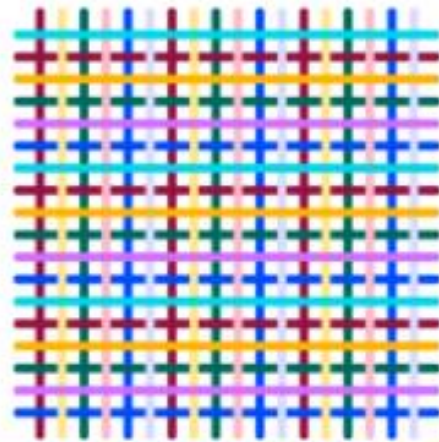
Verkettete Liste (Jgst. 12)

- Zugriffskomplexität $O(n)$
- Whiteboxsicht (Rekursion)



Didaktische Überlegung – motivierende Aufgaben trotz kleiner Portionen

Digitale Kunst:
Vasa Mihich
Painting #207



Variantenreiche
Aufgabenkontexte



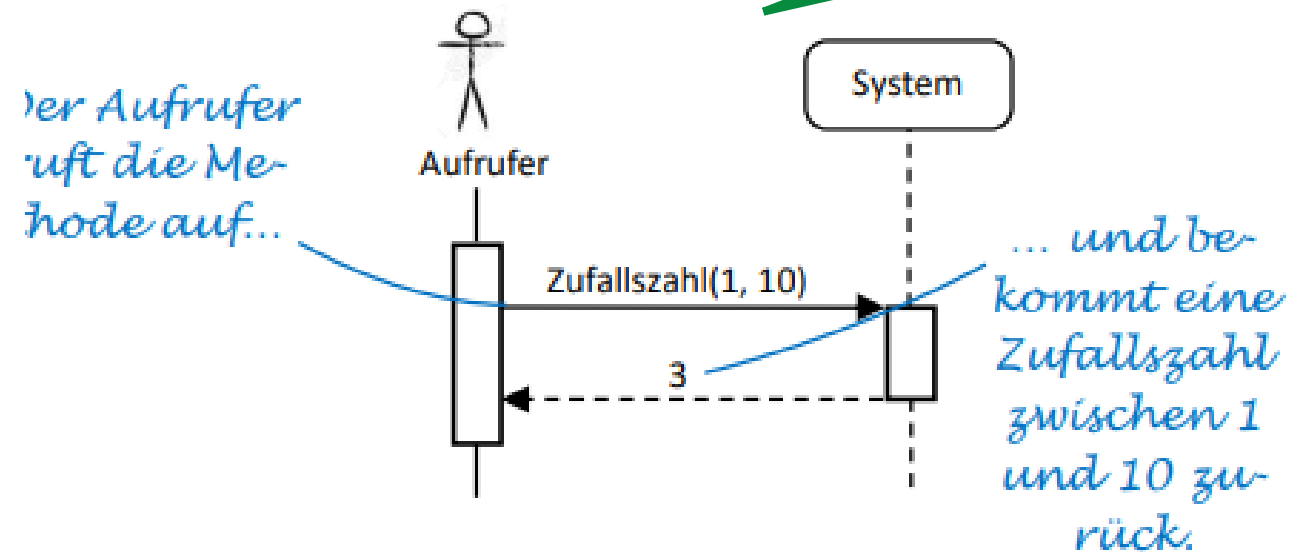
zeichneRechteck(100, 60)



```

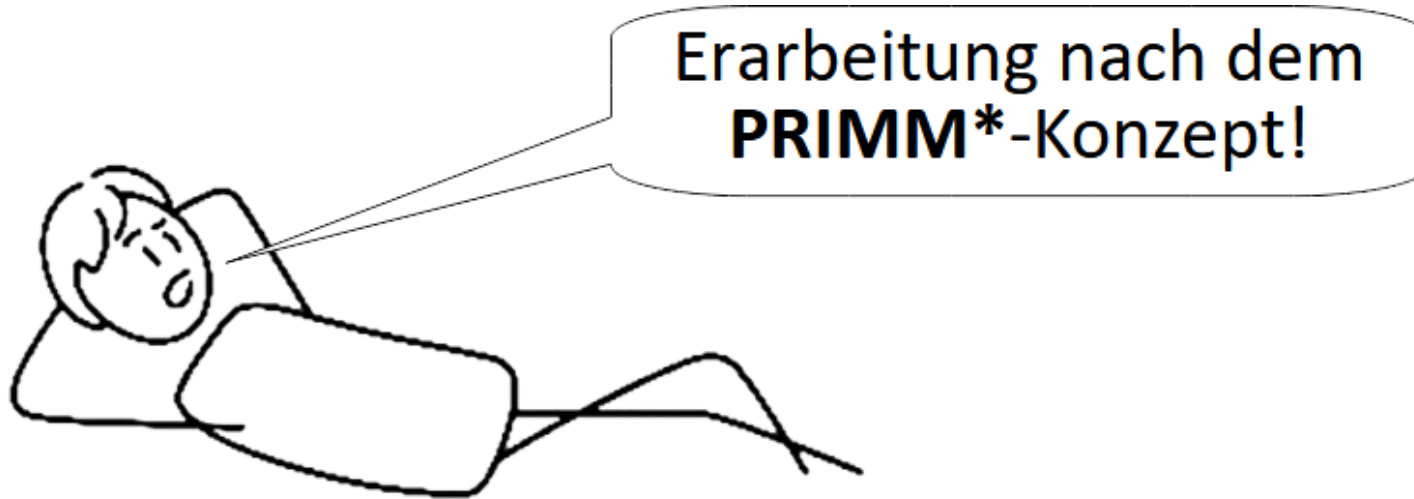
methode zeichneRechteck(GANZZAHL Breite, GANZZAHL Höhe)
  wiederhole 2 mal
    gehe (Breite) => gehe(100)
    drehe (90)
    gehe (Höhe) => gehe(60)
    drehe (90)
  endwiederhole
endemethode
  
```

Visualisierungen



Didaktische Überlegungen

Jedes Programm mit Zeile 1 beginnen??



* Sentance, S. & Waite, J. (2017) PRIMM: Exploring pedagogical approaches for teaching text-based programming in school, In: Barendsen, E. and Hubwieser, P. (eds.) Proceedings of the 12th Workshop on Primary and Secondary Computing Education, 8–10 November 2017, Nijmegen, The Netherlands. New York, ACM. pp. 113–114.

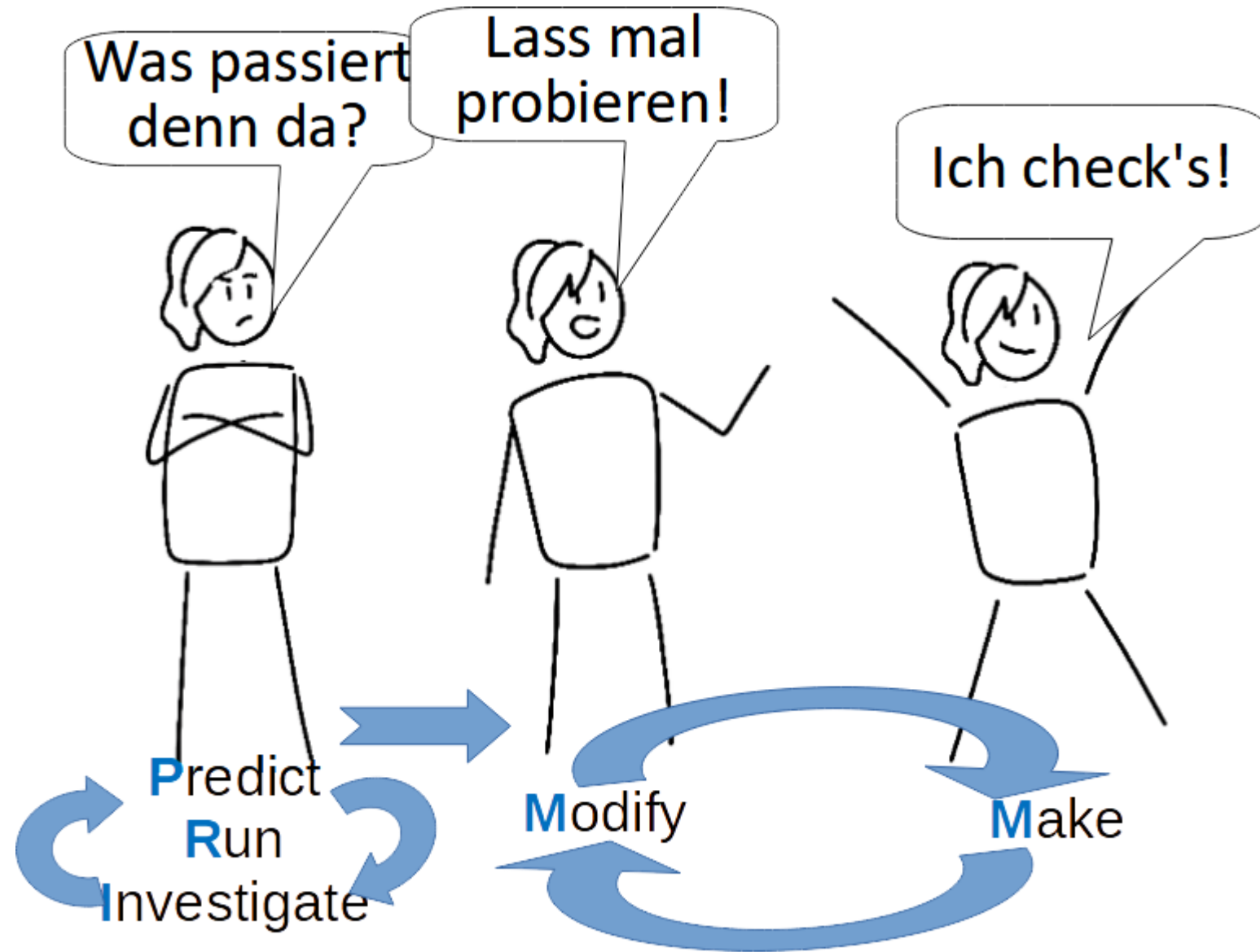
Predict
(Vorhersagen)

Run
(Ausführen)

Investigate
(Erkunden)

Modify
(Anpassen)

Make
(Erstellen)



Didaktische Überlegungen – handlungsorientierte Einstiegsaufgaben (angelehnt an das PRIMM-Konzept)



1 Algorithmik

1.2 Speichern von Daten: Variablen



Eine Timer-App ist eine praktische Sache beim Sporttraining, beim Kochen und in vielen weiteren Bereichen des Alltags: Man stellt eine gewünschte Zeitdauer ein, startet den Timer und erhält ein Signal, wenn die Zeit vorüber ist.

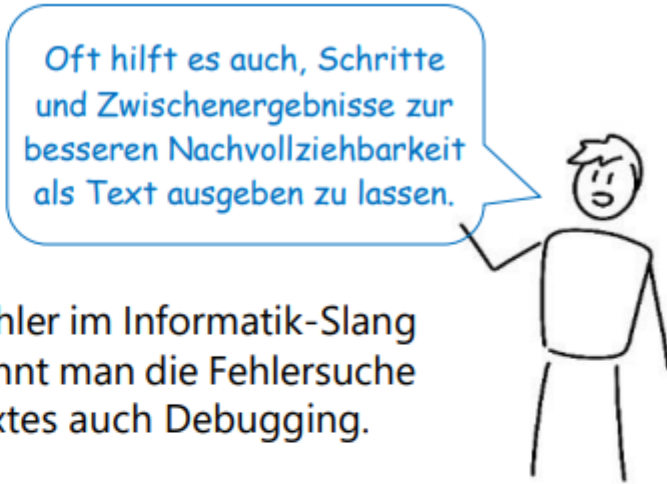
- a** Testen und erkunden Sie das vorgegebene Projekt: Finden Sie heraus, wie die verbleibende Zeit gespeichert ist und verändert wird.
- b** Bauen Sie den Timer um zu einer Stoppuhr, die hochzählt.

Methodik:



Debugging

Das schrittweise Durchlaufen des statischen Quelltexts Zeile für Zeile ermöglicht das Beobachten der Werte von Variablen und des dynamischen Ablaufs. Dadurch können einerseits funktionale Fehler gefunden werden, andererseits wird auch ein größeres Verständnis für Programmabläufe erzielt. Da Fehler im Informatik-Slang Bugs (engl. für Ungeziefer) genannt werden, nennt man die Fehlersuche über ein schrittweises Durchlaufen des Quelltextes auch Debugging.



Besonderes Aufgaben-Icon im Bereich Algorithmik



Pair-Programming

Eine Möglichkeit, die Qualität von Programmen zu erhöhen, ist das Pair-Programming: Man programmiert im Zweierteam mit verteilten Rollen (s. rechts). Dabei unterstützt der Navigator den Driver, um Fehler schon während der Entstehung zu vermeiden. Selbstverständlich werden die Rollen regelmäßig gewechselt.

- Der Driver ...**
 - verwendet Tastatur und Maus und verfasst den Quelltext.
 - „denkt laut“, d.h. teilt seine Absichten dem Navigator mit.
- Der Navigator ...**
 - stellt regelmäßig Fragen.
 - achtet auf Lesbarkeit des Quelltextes, z.B. durch aussagekräftige Variablennamen.

Test aus Nutzersicht

Getestet werden muss die Funktionalität des Programms und einzelner Methoden. Aus Nutzersicht wird hier ohne Wissen über den Quelltext (Blackbox-Sicht) überprüft, ob alle Anforderungen erfüllt sind.

Informatik Reihe des Cornelsen Verlags bietet:

Ein klares didaktisches Konzept abgestimmt auf den Lehrplan

→ Wissen wird mit anschaulichen Visualisierungen Schritt für Schritt aufgebaut

→ Methodische Vielfalt

→ **eigene** Bände zu **NTG** und **spätbeginnend** (in Jgst. 12 drei Bände : **spätbeginnend, grundlegend, vertieft**)

→ Wir sind garantiert in 12/13 am Start

→ Sie erhalten das Lehrwerk pünktlich zu Schuljahresbeginn (Manuskripte Jgst. 12 sind bereits beim Verlag)

→ Mannigfaltiges Material unter

informatikschulbuch.de

Flexibel und sicher im Umgang

Anwendungsprogrammen

→ Sprachunabhängiger Ansatz - Block- und Textsprachen (Snap!, MIT AppInventor, Java, Python)

→ Vorlagen zu den Aufgaben für

14 verschiedene Anwendungsprogramme

The screenshot shows the website interface for 'Algorithmen (Kapitel 1, Informatik 11)'. At the top, there are navigation menus for 'Jahrgangsstufe' (6, 7, 9, 10, 11) and 'Ausblick Für Lehrkräfte'. Below the navigation, there are links for 'Impressum und Datenschutzerklärung' and 'Autorenteam'. The main title 'Algorithmen (Kapitel 1, Informatik 11)' is prominently displayed. A search bar is located on the right side. The central content area features a traffic light icon on the left. To its right, there is a code block for 'textuelle Sprache' (textual language) with the following code:

```
ampel.schalteRotPhase ()
uhr.warte (5)
ampel.schalteRotGelbPhase ()
uhr.warte (2)
ampel.schalteGrünPhase ()
uhr.warte (7)
ampel.schalteGelbPhase ()
uhr.warte (2)
```

Further to the right, there is a block-based programming diagram for 'Blocksprache' (block language) showing a sequence of blocks: 'schalte zu RotPhase', 'warte 5 Sekunden', 'schalte zu RotGelbPhase', 'warte 2 Sekunden', 'schalte zu GrünPhase', 'warte 7 Sekunden', 'schalte zu GelbPhase', and 'warte 2 Sekunden'. At the bottom of the page, there are links for 'Vorlagen', 'Startklar', 'Videos', 'Werkzeuge', and '(Tipp)fehler'.

Informatik SII

Ein klares didaktisches Konzept bietet große Flexibilität.

Wissen aufbauen:


Gut portionierte Stoffpakete (Unterkapitel) mit ausführlicher und anschaulicher Stoffdarstellung sowie passenden Aufgaben

Wissen sichern:

Testseiten mit Aufgaben zur Selbstüberprüfung
Kompakte Wissensdarstellung zum Nachlesen

Wissen vertiefen:

Angebot zur Erarbeitung zusätzlicher Themen

	1 Algorithmen	7
1.1	Erste Schritte: Arbeiten mit der Entwicklungsumgebung	8
1.2	Speichern von Daten: Variablen.....	14
	Qualität von Software erhöhen: Testen.....	20
1.3	Alternativen im Programmablauf: Bedingte Anweisungen.....	22
1.4	Bedingungen verknüpfen: Logische Operatoren.....	28
1.5	Gleiches öfter ausführen: Wiederholungen.....	30
1.6	Viele Daten effizient verwalten: die Liste.....	38
1.7	Beim Methodenaufruf Informationen mitgeben: Parameter	46
1.8	Eine Antwort erhalten: Methoden mit Rückgabewert.....	50
1.9	Schau genau: Da steckt Informatik drin!	54
	Teste dich selbst	63
	Zusammenfassung	65
	Zum Weiterlesen	
L1	Kleine Fehler, große Wirkung.....	69
L2	Informatik als interdisziplinäre Wissenschaft	71

Informatik Reihe des Cornelsen Verlags bietet:

Ein klares didaktisches Konzept abgestimmt auf den Lehrplan

→ Wissen wird mit anschaulichen Visualisierungen Schritt für Schritt aufgebaut

→ Methodische Vielfalt

→ **eigene** Bände zu **NTG** und **spätbeginnend** (in Jgst. 12 drei Bände : **spätbeginnend, grundlegend, vertieft**)

→ Wir sind garantiert in 12/13 am Start

→ Sie erhalten das Lehrwerk pünktlich zu Schuljahresbeginn (Manuskripte Jgst. 12 sind bereits beim Verlag)

→ Mannigfaltiges Material unter

informatikschulbuch.de

Flexibel und sicher im Umgang

Anwendungsprogrammen

→ Sprachunabhängiger Ansatz - Block- und Textsprachen (Snap!, MIT AppInventor, Java, Python)

→ Vorlagen zu den Aufgaben für

16 verschiedene Anwendungsprogramme

The screenshot shows the website interface for 'Algorithmen (Kapitel 1, Informatik 11)'. At the top, there are navigation menus for 'Jahrgangsstufe 6' through '11', 'Ausblick', and 'Für Lehrkräfte'. Below this is a search bar with the text 'Suche...'. The main content area features a traffic light icon on the left. To its right, there are two columns of code. The first column, labeled 'textuelle Sprache', contains the following code:

```
ampel.schalteRotPhase()  
uhr.warte(5)  
ampel.schalteRotGelbPhase()  
uhr.warte(2)  
ampel.schalteGrünPhase()  
uhr.warte(7)  
ampel.schalteGelbPhase()  
uhr.warte(2)
```

The second column, labeled 'Blocksprache', shows a visual representation of the same code using colored blocks (red, yellow, green) and arrows, with a 'warte' block indicating a 2-second delay. The page footer includes links for 'Vorlagen', 'Startklar', 'Videos', 'Werkzeuge', and '(Tipp)fehler'.

Grüße vom Autorenteam

- **Albert Wiedemann** (Seminarlehrer und Fachberater Informatik a. D.)
- **Florian Janus** (Informatiklehrer und „Admin“ von dbiu.de)
- **Franz Jetzinger** (Informatiklehrer, Abordnung an die DDI der TUM)
- **Johannes Neumeyer** (Informatiklehrer)
- **Klaus Reinold** (Seminarlehrer und Fachberater Informatik)
- **Matthias Haupt** (Informatiklehrer, Teilabordnung an die DDI der LMU)
- **Peter Brichzin** (Seminarlehrer Informatik, engagiert in den Lehrerverbänden)
- **Stefan Seegerer** (promovierter Didaktiker, aktiv im Quantencomputing, einer der Köpfe von AI unplugged)






**Qualität,
Engagement und
Konstanz!**

MUSIKER	spielt in > n	BAND
name	m < besteht aus	name

Informatik SII

Aufgaben, Aufgaben, Aufgaben bieten methodische Vielfalt und technologische Flexibilität

kleine Aufgaben für mehr Abwechslung

   **2 Methoden mit Rückgabe in der Entwicklungsumgebung**
In Ihrer Programmiersprache existieren zahlreiche Methoden mit Rückgabe, welche Sie bereits eingesetzt haben.

- a** Stellen Sie die Stoppuhr auf zwei Minuten und versuchen Sie in dieser Zeit mehr dieser Methoden mit Rückgabe zu finden als Ihr Nachbar oder Ihre Nachbarin.
- b** Vergleichen Sie das Gefundene und testen Sie die Methoden, welche Sie bisher noch nicht verwendet haben. Erklären Sie sich gegenseitig deren Funktionsweise.

Vorlagen zu gängigen Entwicklungsumgebungen und Anwendungsprogrammen



Snap!

```
biosiegel? oder regional?  
siehtGutAus? und riechtGut?  
nicht istEinseitig?
```

App Inventor

```
get global biosiegel? or get global regional?  
get global siehtGutAus? and get global riechtGut?  
not get global istEinseitig?
```

Python

```
biosiegel and regional  
siehtGutAus or riechtGut  
not istEinseitig
```

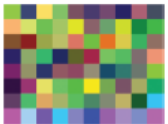
Java

```
biosiegel || regional  
siehtGutAus && riechtGut  
!istEinseitig
```

größere themenzentrierte Aufgabenblöcke

10 Steganografie

Haben Sie als Kind einmal mit Zitronensaft schreibend zwischen den Zeilen eines normalen Briefes eine geheime Nachricht versteckt? Die adressierte Person musste nur das Papier erwärmen und Ihre Mitteilung wurde sichtbar. Das Verstecken von Informationen in unauffälligen Trägerdaten nennt man → Steganografie. Digitale Text-, Ton- oder Bilddaten eignen sich sehr gut für dieses Verfahren. Moderne Anwendungen sind neben einer Nachrichtenübermittlung auch Wasserzeichen als Kopierschutz. Verwendet man als Träger ein Bild wie in der Abbildung, kann man beispielsweise den Blauanteil des RGB-Farbwerts (Kap. 1.2 Aufgabe 9) der Quadrate nutzen, um eine versteckte Nachricht einzubetten. Folgende Schritte geben eine Hilfestellung das Verfahren zu implementieren:



→ griech. Steganós graphem: bedeckt schreiben

- a** Zeichnen eines Trägerbildes bestehend aus farbigen Quadraten:
Erstellen Sie ein Programm, das ausgefüllte Quadrate gleicher Größe (z. B. Seitenlänge 40) aneinanderreihet. Die Quadrate müssen zeilenweise gezeichnet werden. Erstellen Sie als Test ein Bild mit unterschiedlichen Blautönen – beginnend links oben ist das dunkelste, schrittweise wird die Farbe heller, rechts unten ist das hellste.
- b** Einbetten einer Nachricht: Speichern Sie die Nachricht als eine Liste aus Buchstaben. Wandeln Sie jeden Buchstaben eindeutig in eine Zahl um. Suchen Sie dazu eine Methode Ihres Werkzeuges, das den Unicode zu dem entsprechenden Buchstaben zurückgibt. Erweitern Sie das Programm aus a) so, dass (zeilenweise) der blaue Farbanteil des Quadrats den Zahlenwert des Buchstaben hat.
Hinweis: Sollte die Anzahl der Zeichen der Nachricht geringer als die Anzahl der Quadrate sein, müssen die übrigen Quadrate einen Blauwert erhalten, der sich von Buchstaben zugeordneten Zahlen deutlich unterscheidet, z. B. 255.
- c** Extrahieren einer Nachricht:
Zeilenweise muss das Trägerbild mit der Nachricht durchlaufen, bei jedem Quadrat der blaue Farbanteil ausgelesen und in einer Liste gespeichert werden. Implementieren Sie eine entsprechende Methode.
Hinweis: Es sollen nur Zahlen gespeichert werden, die zur Nachricht gehören (vgl. den Hinweis bei b)).
- d** Ausgabe der Nachricht:
Wandeln Sie nun die Liste mit Zahlen aus c) in Buchstaben um und geben Sie die Nachricht aus.
- e** Qualitätsüberprüfung:
Testen Sie ihr Programm, indem Sie zu zweit sich gegenseitig eine versteckte Nachricht schreiben und extrahieren.
Lesen Sie auch die Nachricht aus dem oben gegebenen Bild aus. (Das Bild ist als Vorlage gegeben. Die Quadrate dort haben eine Größe von 40x40 Pixeln.)
- f** Für Schnelle: Erweitern Sie Ihr Programm, z. B. durch das zufällige Setzen von roten und grünen Farbwerten – dadurch ist der Informationskanal im Trägerbild noch unscheinbarer – oder eine nutzerfreundliche Eingabe von Nachrichten.

Buchstabe	Unicode
G	71
e	101
h	104
e	101
i	105
m	109

Informatik SII

Kompetenzorientierung und methodische Vielfalt

Wissen aufbauen

Die **Symbole** an den Aufgaben geben eine Orientierung zur **Kompetenzerwartung** und zu **methodischem Einsatz**.



Recherchieren, lesen, damit du selbst Verantwortung für den Lernfortschritt übernehmen kannst.



Vernetzen, damit du neue Inhalte mit anderen Bereichen und bereits Gelerntem in Beziehung setzen kannst.



Kommunizieren, weil es wichtig ist, anderen etwas erklären zu können und sich in der Gruppe zu besprechen.



Kooperieren, damit eure Produkte vielseitiger, hochwertiger und umfangreicher werden.



Kreativ arbeiten, damit du originell und mit persönlicher Note arbeiten kannst.



Analysieren, um Strukturen, Verfahren und Zusammenhänge zu erkennen und zu erfassen.



Modellieren, um Probleme zu verstehen und Lösungen zu planen.



Mit Rechnereinsatz lösen/implementieren, um mithilfe des Computers praktische Umsetzungen zu schaffen.



Handlungsorientiert lösen, damit du auch ohne Rechner aktiv werden kannst, z. B. bei Rollenspielen.



Reflektieren, begründen, damit du dir Lösungswege bewusst machst und ähnliche Aufgaben künftig schneller lösen kannst.



Offene Aufgabenstellung individuell ausgestalten, damit du dir selbst Ziele setzen und deinen eigenen Lösungsweg suchen kannst.



Diese Inhalte und Aufgaben gehen über den Lehrplan hinaus.

Die **Symbole** beschreiben für jede Aufgabe Arbeitsweisen bzw. Zielsetzung.



Das **Debugging-Symbol** markiert Aufgaben und Tipps zur Fehlersuche.



Informatik SII

Ein klares didaktisches Konzept bietet große Flexibilität.

Wissen sichern

... mit Testaufgaben zur Selbstüberprüfung

... mit kompakter Zusammenfassung der zentralen Inhalte

Teste dich selbst

T1 Richtig oder falsch?

Beurteilen Sie, ob folgende Aussagen richtig oder falsch sind. Begründen Sie Ihre Meinung bei falschen Aussagen und geben Sie eine berichtigte Aussage an:

- a Jedes Programm ist ein Algorithmus und umgekehrt.
- b Eine Initialisierung ist die erstmalige Zuweisung eines Werts an eine Variable.
- c Jede bedingte Anweisung ergibt ausgewertet WAHR oder FALSCH.
- d Jede einseitige bedingte Anweisung lässt sich in eine zweiseitige bedingte Anweisung umwandeln.
- e Der Ausdruck rechts ist ausgewertet WAHR.
- f In eine Liste kann ich neue Elemente nur am Anfang oder Ende einfügen bzw. entfernen.
- g In einer Verlaufsliste werden neue Einträge immer hinten eingefügt (z. B. Seitenaufrufe in der Verlaufsliste eines Browsers).
- f Durch die abgebildete Wiederholungsanweisung erhält man eine nach Datum sortierte Ausgabe beginnend mit dem aktuellsten Eintrag.



T2 Quellcode analysieren – Abläufe verstehen

- a Übertragen Sie den rechts abgebildeten Pseudocode als strukturierten Quelltext mit Einrückungen in Ihr Heft. Markieren Sie in einer Farbe alle Bedingungen, in einer anderen alle Methodenaufrufe. Begründen Sie anhand Ihrer Einrückungen, warum eine Schachtelung von Kontrollstrukturen vorliegt und wie hoch die Schachtelungstiefe ist.
- b Programme werden zeilenweise ausgeführt, dabei werden u. A. Werte gesetzt und auch entschieden, welches die nächste Zeile im Ablauf ist. In der folgenden Tabelle sind die Reihenfolge der ausgeführten Programmzeilen, der jeweilige Wert der Variablen Zahl und entsprechende Erläuterungen eingetragen. Setzen Sie diese Ablaufbeschreibung in Ihrem Heft für weitere 15 Zeilen fort.



Zeile in Ausführung	Zahl	Erläuterung
1	5	-----
2	5	Auswertung der Bedingung in der bedingten Wiederholung: (Zahl = 30) ergibt FALSCH
3	5	Auswertung der Bedingung in der bedingten Anweisung: IstGerade(Zahl) ergibt FALSCH
5	5	sonst-Fall (der dann-Fall wurde übersprungen)
6	5	erster Durchlauf der Wiederholung mit fester Anzahl
7	10	-----
8	10	Ende der Wiederholungsanweisung
6	10	zweiter Durchlauf der Wiederholung mit fester Anzahl
...

Zusammenfassung

Programm, Anweisung, selbstdefinierte Methoden

Ein Algorithmus zur Lösung einer Aufgabenstellung, der in einer konkreten Programmiersprache formuliert wurde, heißt Programm. Es besteht aus Anweisungen – eindeutig formulierten Einzelschritten wie Methodenaufrufen. Eine Folge nacheinander auszuführender Anweisungen heißt Sequenz.

Selbstdefinierte Methoden helfen, ein Programm zu strukturieren. Zur Festlegung müssen im Methodenkopf der Name der Methode und im Methodentrumpf die Sequenz der auszuführenden Anweisungen notiert werden.

Snap!

Python

```
def rückwärtsGehen ():
    Drehe(180)
    Gehe(10)
    Drehe(-180)
```

Java

```
void rückwärtsGehen () {
    Drehe(180);
    Gehe(10);
    Drehe(-180);
}
```

AppInventor

grün: Methodenkopf
rot: Methodentrumpf mit einer Sequenz aus drei Anweisungen.

Daten speichern in Variablen

Variablen dienen in Programmen zum Speichern von Werten. Für jede Variable muss ein Name, in vielen Sprachen auch ein Datentyp festgelegt werden (Deklaration). Das Speichern eines Startwerts nennt man Initialisierung. Durch eine Zuweisung kann eine Variable einen neuen Wert erhalten. Der Datentyp legt fest, welche Art von Werten gespeichert und welche Operationen mit einer Variablen ausgeführt werden können.

Snap!

AppInventor

Java

```
int punkteSpieler1;
punkteSpieler1 = 6;
```

Python

```
punkteSpieler1 = 6
```

In Snap und Java notiert man Deklaration und Initialisierung getrennt, im AppInventor und in Python ist dies ein Schritt

Informatik SII

Ein klares didaktisches Konzept bietet große Flexibilität.

Wissen sichern

... mit Testaufgaben zur Selbstüberprüfung

... mit kompakter Zusammenfassung der zentralen Inhalte

Teste dich selbst

T1 Richtig oder falsch?

Beurteilen Sie, ob folgende Aussagen richtig oder falsch sind. Begründen Sie Ihre Meinung bei falschen Aussagen und geben Sie eine berichtigte Aussage an:

- a Jedes Programm ist ein Algorithmus und umgekehrt.
- b Eine Initialisierung ist die erstmalige Zuweisung eines Werts an eine Variable.
- c Jede bedingte Anweisung ergibt ausgewertet WAHR oder FALSCH.
- d Jede einseitige bedingte Anweisung lässt sich in eine zweiseitige bedingte Anweisung umwandeln.
- e Der Ausdruck rechts ist ausgewertet WAHR.
- f In eine Liste kann ich neue Elemente nur am Anfang oder Ende einfügen bzw. entfernen.
- g In einer Verlaufsliste werden neue Einträge immer hinten eingefügt (z. B. Seitenaufrufe in der Verlaufsliste eines Browsers).
- f Durch die abgebildete Wiederholungsanweisung erhält man eine nach Datum sortierte Ausgabe beginnend mit dem aktuellsten Eintrag.



T2 Quellcode analysieren – Abläufe verstehen

- a Übertragen Sie den rechts abgebildeten Pseudocode als strukturierten Quelltext mit Einrückungen in Ihr Heft. Markieren Sie in einer Farbe alle Bedingungen, in einer anderen alle Methodenaufrufe. Begründen Sie anhand Ihrer Einrückungen, warum eine Schachtelung von Kontrollstrukturen vorliegt und wie hoch die Schachtelungstiefe ist.
- b Programme werden zeilenweise ausgeführt, dabei werden u. A. Werte gesetzt und auch entschieden, welches die nächste Zeile im Ablauf ist. In der folgenden Tabelle sind die Reihenfolge der ausgeführten Programmzeilen, der jeweilige Wert der Variablen Zahl und entsprechende Erläuterungen eingetragen. Setzen Sie diese Ablaufbeschreibung in Ihrem Heft für weitere 15 Zeilen fort.



Zeile in Ausführung	Zahl	Erläuterung
1	5	-----
2	5	Auswertung der Bedingung in der bedingten Wiederholung: (Zahl = 30) ergibt FALSCH
3	5	Auswertung der Bedingung in der bedingten Anweisung: IstGerade(Zahl) ergibt FALSCH
5	5	sonst-Fall (der dann-Fall wurde übersprungen)
6	5	erster Durchlauf der Wiederholung mit fester Anzahl
7	10	-----
8	10	Ende der Wiederholungsanweisung
6	10	zweiter Durchlauf der Wiederholung mit fester Anzahl
...

Zusammenfassung

Programm, Anweisung, selbstdefinierte Methoden

Ein Algorithmus zur Lösung einer Aufgabenstellung, der in einer konkreten Programmiersprache formuliert wurde, heißt Programm. Es besteht aus Anweisungen – eindeutig formulierten Einzelschritten wie Methodenaufrufen. Eine Folge nacheinander auszuführender Anweisungen heißt Sequenz.

Selbstdefinierte Methoden helfen, ein Programm zu strukturieren. Zur Festlegung müssen im Methodenkopf der Name der Methode und im Methodentrumpf die Sequenz der auszuführenden Anweisungen notiert werden.

Snap!

Python

```
def rückwärtsGehen():
    Drehe(180)
    Gehe(10)
    Drehe(-180)
```

Java

```
void rückwärtsGehen() {
    Drehe(180);
    Gehe(10);
    Drehe(-180);
}
```

AppInventor

grün: Methodenkopf
rot: Methodentrumpf mit einer Sequenz aus drei Anweisungen.

Daten speichern in Variablen

Variablen dienen in Programmen zum Speichern von Werten. Für jede Variable muss ein Name, in vielen Sprachen auch ein Datentyp festgelegt werden (Deklaration). Das Speichern eines Startwerts nennt man Initialisierung. Durch eine Zuweisung kann eine Variable einen neuen Wert erhalten. Der Datentyp legt fest, welche Art von Werten gespeichert und welche Operationen mit einer Variablen ausgeführt werden können.

Snap!

AppInventor

Java

```
int punkteSpieler1;
punkteSpieler1 = 6;
```

Python

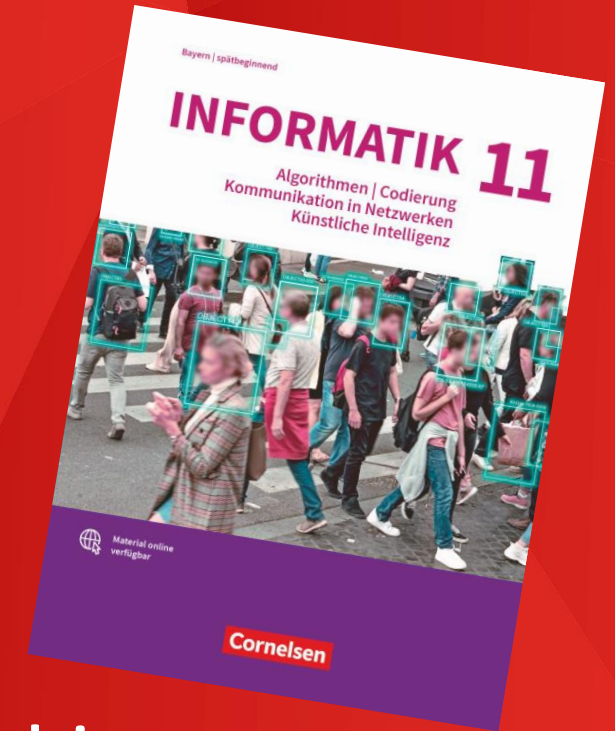
```
punkteSpieler1 = 6
```

In Snap und Java notiert man Deklaration und Initialisierung getrennt, im AppInventor und in Python ist dies ein Schritt

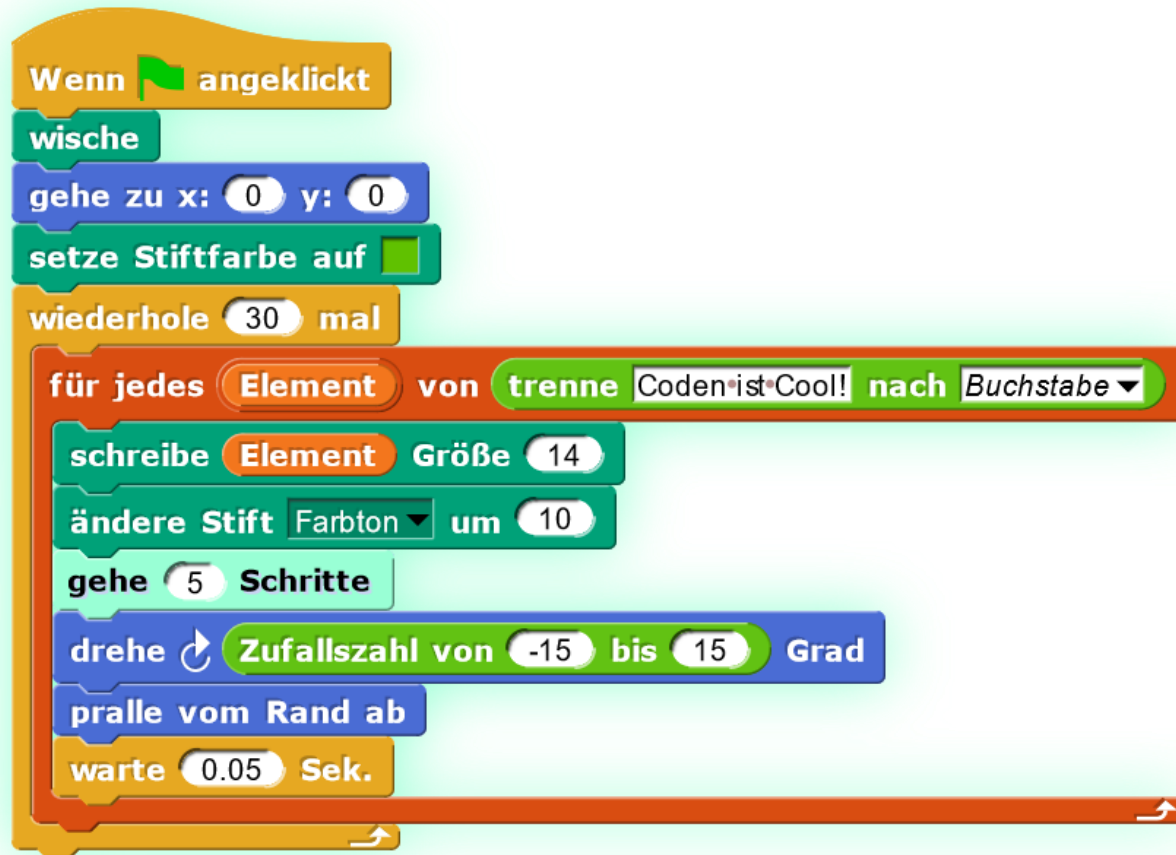
Ihr seid dran! 40 Minuten Coden

Ein Austausch ist wichtig. Deshalb

- Coden in 3er-Gruppen (Break-out-Rooms)
- Gemeinsame Auswahl der Aufgaben
- Eine/-r teilt den Bildschirm (Wechsel zur Halbzeit erwünscht)
- Austausch über mehrere Schulen:
zufällige Teilnehmerauswahl
- Aufgabenvorschläge auf den folgenden Folien
- Vollständige Aufgabentexte im Prüfexemplar als print bzw. „Blick ins Buch“ unter <https://www.cornelsen.de/produkte/informatik-oldenbourg-algorithmen-codierung-kommunikation-in-netzwerken-kuenstliche-intelligenz-band-fuer-spaetbeginnende-informatik-schulbuch-11-jahrgangsstufe-9783637029590>
->Klick auf das Cover
- Vorlagen unter [Informatikschulbuch.de](https://www.informatikschulbuch.de)

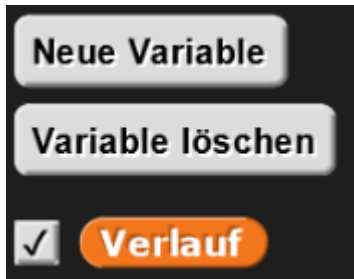


Einstieg Textanimation zum Nachprogrammieren



Einstieg Listenoperationen – eine Auswahl

Deklaration einer Liste
(dynamische Typisierung)



Initialisierung einer Liste



Hinzufügen/ Einfügen von Listenelementen



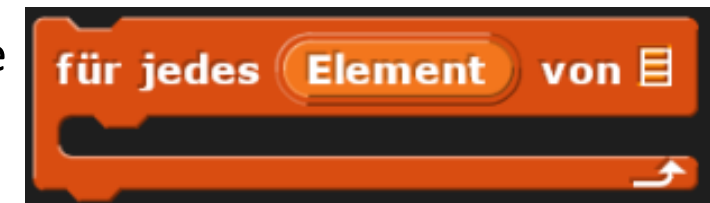
Eigenschaften von Listen



Zugriff auf Elemente

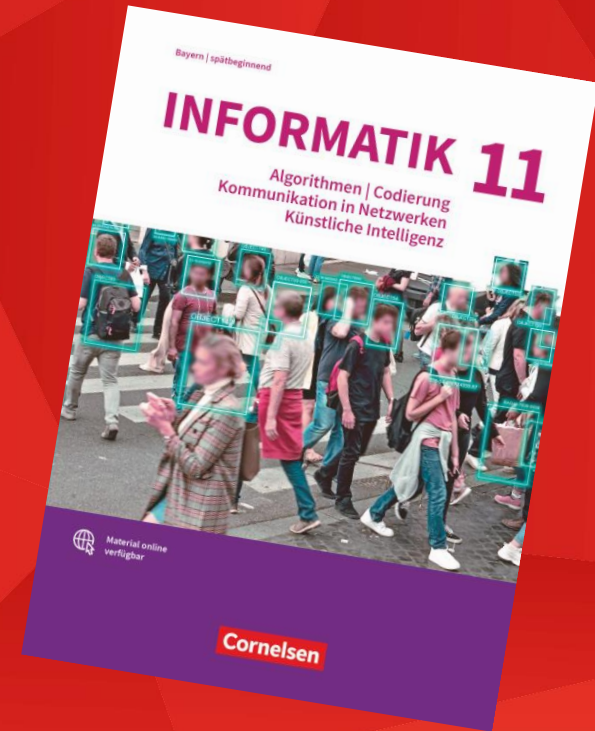


Durchlaufen einer Liste
(Wiederholung über
alle Elemente)



Informatik SII Bayern

Ihr seid dran!
40 Minuten Coden
Aufgabenbeispiele aus



Aufgabenvorschläge

- Eigene Methoden: Textanimation S. 13/10
- Speichern von Daten: Einstiegsaufgabe S. 14
- Bedingte Anweisung: Einstiegsaufgabe S. 22
- Wiederholungsanweisungen: Digitale Kunst „Flowers“ S. 36/7
(ohne Vorlage)
- Liste/Feld: Einstiegsaufgabe S. 38
- Umfangreiche Aufgabe zur Vertiefung: Steganografie S. 61
(ohne Vorlage)

Ihre Erfahrungen? Ihre Fragen?

Werkzeugwahl Beachte Jgst. 11 **KEINE** Objektorientierung

- **Snap! - blockbasiert** (Scratch)
 - ✓ Keine Installation
 - ✓ Motivierende Möglichkeiten (Media Computation, Arbeiten mit Daten, ...), die weit über die Jgst. 11 hinaus gehen
 - ✓ Didaktisch (Visualisierung, ...) und Professionell (Einzelschritt, ...)
- **AppInventor – blockbasiert**
 - ✓ Eigene Apps am Handy: Hohe Motivation; Informatik in der Hosentasche immer dabei
 - ✓ Kompetenzen auch im Bereich GUI-Entwicklung
- **Java BlueJ – textbasiert**
 - ✓ **Methodenbox** didaktisch reduzierte Umgebung
 - ✓ Leichter Übergang zur Kursphase
- **Python – textbasiert**
 - ✓ **Methodenbox** didaktisch reduzierte Umgebung
 - ✓ Leichter Übergang zur Kursphase

Grundlegende Konzepte und Begeisterung für Schüler*innen die Info ablegen werden versus Schüler*innen, die tiefer in das Programmieren einsteigen wollen

➤ **Differenzierung im Werkzeug bei gleichen Aufgaben**

Werkzeugwahl Beachte Jgst. 11 **KEINE** Objektorientierung

- **Snap! - blockbasiert** (Scratch)
 - ✓ Keine Installation
 - ✓ Motivierende Möglichkeiten (Media Computation, Arbeiten mit Daten, ...), die weit über die Jgst. 11 hinaus gehen
 - ✓ Didaktisch (Visualisierung, ...) und Professionell (Einzelschritt, ...)
- **AppInventor – blockbasiert**
 - ✓ Eigene Apps am Handy: Hohe Motivation; Informatik in der Hosentasche immer dabei
 - ✓ Kompetenzen auch im Bereich GUI-Entwicklung
- **Java BlueJ – textbasiert**
 - ✓ **Methodenbox** didaktisch reduzierte Umgebung
 - ✓ Leichterer Übergang zur Kursphase
- **Python – textbasiert**
 - ✓ **Methodenbox** didaktisch reduzierte Umgebung
 - ✓ Leichterer Übergang zur Kursphase

Grundlegende Konzepte und Begeisterung für Schüler*innen die Info ablegen werden
versus
Schüler*innen, die tiefer in das Programmieren einsteigen wollen

➤ **Differenzierung im Werkzeug bei gleichen Aufgaben**

Methodenbox: didaktisch reduzierte Umgebung für textbasierte Programmierung (hier Java/BlueJ analog für Python)

Zeichenfenster

(h) Erhöhen (r) Reduzieren (s) Start/Stop
Noch 2 Minuten 47 Sekunden

Differenzierung möglich, da Aufgaben in Informatik 11 werkzeugunabhängig gestellt sind. Hier z. B. die Einstiegsaufgabe zu Kapitel 1.2 – ein digitaler Timer.

Start Stop

0 100 200 300 400 500 600 700 800 900 1000

Blue: 1_2_Einstieg_Vorlage - Kopie

Projekt Bearbeiten Werkzeuge Ansicht Hilfe

Neue Klasse...
→
Übersetzen

Teamwork
Share...

Testing
Tests starten
Aufzeichnung

Methodenbox
(no source)

Timer

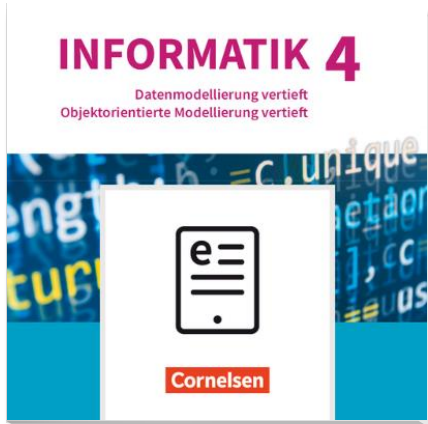
Methodenbox

```
textfielder : ArrayList<Text>  
Methodenbox()  
Anhalten() : void  
MausGeklickt(x : int, y : int, anzahl : int) : void  
SonderTasteGedrückt(taste : int) : void  
Starten() : void  
TaktImpulsAusführen() : void  
TaktdauerSetzen(dauer : int) : void  
TasteGedrückt(taste : char) : void  
erzeugeText(bezeichner : String, text : String, x : int, y : int) : void  
setzeText(bezeichner : String, neuerText : String) : void
```

timer1:
Timer

Wir bieten exzellente Unterstützung

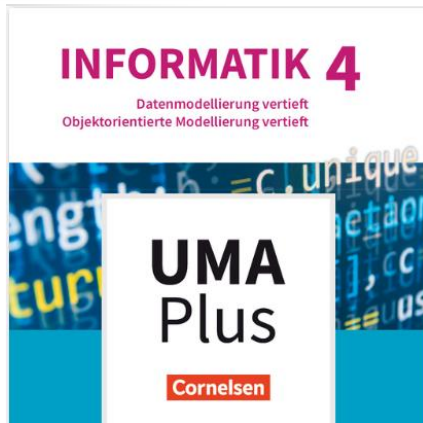
E-Book mit Medien und Unterrichtsmanager Plus



E-Book
mit Medien

E-Book mit Medien enthält:

- Schulbuchinhalte
- Erklärvideos zu verwendeten Programmen
- Vorlagen für Anwendungs- und Entwicklungsumgebungen



Unterrichtsmanager
Plus

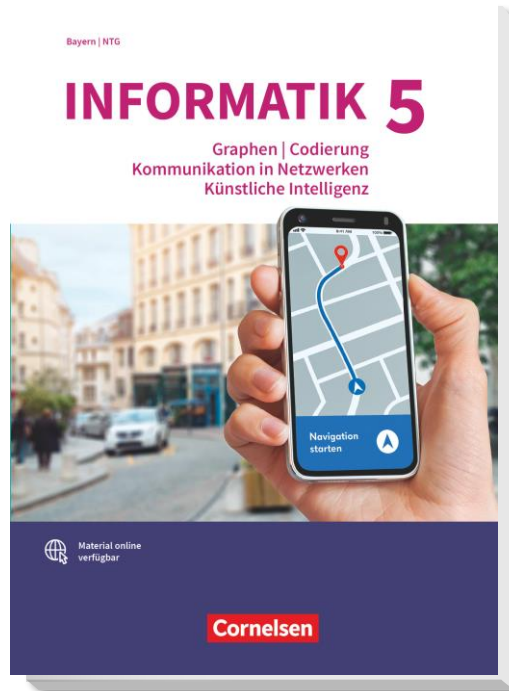
Unterrichtsmanager Plus enthält:

- E-Book
- Schriftlich Lösungen zu allen Aufgaben
- Lösungsdateien zu Programmieraufgaben
- Handreichung, editierbare Kopiervorlagen, Bilder, Grafiken, Audios und Videos , Folien

Informatik SII

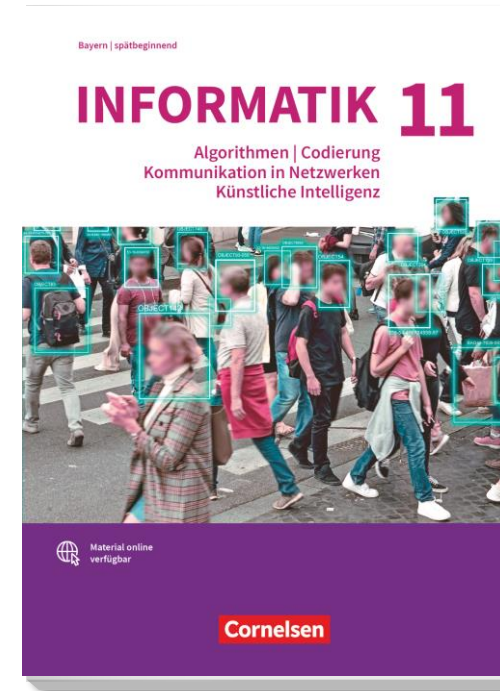
100% passend für Bayern - alle Lehrplaninhalten speziell für alle Anforderungen

NTG



Band 5 (Fortsetzung zu Band 4)

SWG | WWG | HG | SG | MuG



Band 11 | spätbeginnend

Vielen Dank für eure Aufmerksamkeit!

Cornelsen