



Verinnerlichung einer Vorstellung zur Programmausführung im ersten Programmiersemester

Robin Goltermann, Frank Höppner

Ostfalia Hochschule, Wolfenbüttel

ABP'17, 06.10.2017



Motivation

Modellvorstellung der Programmausführung

Speicherprotokolle

Automatische, interaktive Protokolle

Erste Erfahrungen

Fazit



Motivation

Modellvorstellung der Programmausführung

Speicherprotokolle

Automatische, interaktive Protokolle

Erste Erfahrungen

Fazit



- ▶ Technische Unterstützung in Programmierkursen:
 - ▶ Verbreitet: Funktionstest (JUnit-Tests)
Automatische Bewertung von Programmieraufgaben
- ▶ Resonanz
 - ▶ obere Leistungshälfte kommt damit sehr gut klar
 - ▶ untere Leistungshälfte, Anfänger z.T. überfordert
- ▶ Fokus des Vortrags
 - ▶ eben diese untere Leistungshälfte, Anfänger
 - ▶ Grundlagen / Einstieg in Programmierung



Mögliches Szenario

- ▶ binäres Feedback (rot/grün) belohnt inkrementelle Fortschritte kaum
- ▶ es entsteht Frust, Termindruck
- ▶ zusätzliche Probleme durch Bedienung der Testumgebung (SVN)
- ▶ Betreuer verweisen auf Tests, die können die Studierenden aber nicht lesen
- ▶ beschäftigen sich viel mit Trial & Error statt Wissenslücken zu schließen, nicht konstruktiv
- ▶ es fehlt mglw. an Fähigkeit, sich den Stoff selbst noch mal zu erschließen
- ▶ an Ende, um keine Punkte zu verlieren, Lösung von Kommilitonen abgeben (plagiarism as a cry for help [Vogts, 2009], Gegenseite: Ruf nach Plagiatserkennung)



Mögliches Szenario

- ▶ binäres Feedback (rot/grün) belohnt inkrementelle Fortschritte kaum
- ▶ es entsteht Frust, Termindruck
- ▶ zusätzliche Probleme durch Bedienung der Testumgebung (SVN)
- ▶ Betreuer verweisen auf Tests, die können die Studierenden aber nicht lesen
- ▶ beschäftigen sich viel mit Trial & Error statt Wissenslücken zu schließen, nicht konstruktiv
- ▶ es fehlt mglw. an Fähigkeit, sich den Stoff selbst noch mal zu erschließen
- ▶ an Ende, um keine Punkte zu verlieren, Lösung von Kommilitonen abgeben (plagiarism as a cry for help [Vogts, 2009], Gegenseite: Ruf nach Plagiatserkennung)



Mögliches Szenario

- ▶ binäres Feedback (rot/grün) belohnt inkrementelle Fortschritte kaum
- ▶ es entsteht Frust, Termindruck
- ▶ zusätzliche Probleme durch Bedienung der Testumgebung (SVN)
- ▶ Betreuer verweisen auf Tests, die können die Studierenden aber nicht lesen
- ▶ beschäftigen sich viel mit Trial & Error statt Wissenslücken zu schließen, nicht konstruktiv
- ▶ es fehlt mglw. an Fähigkeit, sich den Stoff selbst noch mal zu erschließen
- ▶ an Ende, um keine Punkte zu verlieren, Lösung von Kommilitonen abgeben (plagiarism as a cry for help [Vogts, 2009], Gegenseite: Ruf nach Plagiatserkennung)



Mögliches Szenario

- ▶ binäres Feedback (rot/grün) belohnt inkrementelle Fortschritte kaum
- ▶ es entsteht Frust, Termindruck
- ▶ zusätzliche Probleme durch Bedienung der Testumgebung (SVN)
- ▶ Betreuer verweisen auf Tests, die können die Studierenden aber nicht lesen
- ▶ beschäftigen sich viel mit Trial & Error statt Wissenslücken zu schließen, nicht konstruktiv
- ▶ es fehlt mglw. an Fähigkeit, sich den Stoff selbst noch mal zu erschließen
- ▶ an Ende, um keine Punkte zu verlieren, Lösung von Kommilitonen abgeben (plagiarism as a cry for help [Vogts, 2009], Gegenseite: Ruf nach Plagiatserkennung)



Mögliches Szenario

- ▶ binäres Feedback (rot/grün) belohnt inkrementelle Fortschritte kaum
- ▶ es entsteht Frust, Termindruck
- ▶ zusätzliche Probleme durch Bedienung der Testumgebung (SVN)
- ▶ Betreuer verweisen auf Tests, die können die Studierenden aber nicht lesen
- ▶ beschäftigen sich viel mit Trial & Error statt Wissenslücken zu schließen, nicht konstruktiv
- ▶ es fehlt mglw. an Fähigkeit, sich den Stoff selbst noch mal zu erschließen
- ▶ an Ende, um keine Punkte zu verlieren, Lösung von Kommilitonen abgeben (plagiarism as a cry for help [Vogts, 2009], Gegenseite: Ruf nach Plagiatserkennung)



Mögliches Szenario

- ▶ binäres Feedback (rot/grün) belohnt inkrementelle Fortschritte kaum
- ▶ es entsteht Frust, Termindruck
- ▶ zusätzliche Probleme durch Bedienung der Testumgebung (SVN)
- ▶ Betreuer verweisen auf Tests, die können die Studierenden aber nicht lesen
- ▶ beschäftigen sich viel mit Trial & Error statt Wissenslücken zu schließen, nicht konstruktiv
- ▶ es fehlt mglw. an Fähigkeit, sich den Stoff selbst noch mal zu erschließen
- ▶ an Ende, um keine Punkte zu verlieren, Lösung von Kommilitonen abgeben (plagiarism as a cry for help [Vogts, 2009], Gegenseite: Ruf nach Plagiatserkennung)



Mögliches Szenario

- ▶ binäres Feedback (rot/grün) belohnt inkrementelle Fortschritte kaum
- ▶ es entsteht Frust, Termindruck
- ▶ zusätzliche Probleme durch Bedienung der Testumgebung (SVN)
- ▶ Betreuer verweisen auf Tests, die können die Studierenden aber nicht lesen
- ▶ beschäftigen sich viel mit Trial & Error statt Wissenslücken zu schließen, nicht konstruktiv
- ▶ es fehlt mglw. an Fähigkeit, sich den Stoff selbst noch mal zu erschließen
- ▶ an Ende, um keine Punkte zu verlieren, Lösung von Kommilitonen abgeben (plagiarism as a cry for help [Vogts, 2009], Gegenseite: Ruf nach Plagiatserkennung)



Was sind also die Ursachen für fehlgeschlagene Tests?

1. Lesekompetenz, Textverständnis, Oberflächlichkeit
2. unbeachtete Schwierigkeit der Aufgabe (Sonderfälle)
3. u.v.a.m.
4. falsches Verständnis der Sprachkonzepte

Dann ist eine richtige Lösung nur durch Zufall zu erreichen... !



Was sind also die Ursachen für fehlgeschlagene Tests?

1. Lesekompetenz, Textverständnis, Oberflächlichkeit
2. unbeachtete Schwierigkeit der Aufgabe (Sonderfälle)
3. u.v.a.m.
4. falsches Verständnis der Sprachkonzepte

Dann ist eine richtige Lösung nur durch Zufall zu erreichen... !



Was sind also die Ursachen für fehlgeschlagene Tests?

1. Lesekompetenz, Textverständnis, Oberflächlichkeit
2. unbeachtete Schwierigkeit der Aufgabe (Sonderfälle)
3. u.v.a.m.
4. falsches Verständnis der Sprachkonzepte

Dann ist eine richtige Lösung nur durch Zufall zu erreichen... !



Beispiel

```
int x = 3;
{
  int y = x%2;
  if (y==0) {
    x = x+1;
  } else {
    x = x-1;
  }
  System.out.println(x);
}
if (x>0) ...
```

“Zufallsentdeckung” bei Nachfrage eines Studierenden:
Erwartung einer Art Rollback-Semantik



Hypothese

Lernziel-Taxonomie (Bloom) beim Programmierens:

Erinnern	Kenntnis der Sprachkonstrukte.
Verstehen	Was macht dieses Programm? Wie funktionieren die Sprachkonstrukte?
Anwenden	Komposition der Sprachkonstrukte zu einer Lösung

Einstieg der Unterstützung in der Stufe **Anwenden**, keine Unterstützung für **Verstehen**.

Wir überspringen beim Feedback eine Stufe und **hoffen** nur, dass sie richtig verstanden wurde.

Hypothese: Wenn wir diesen Schritt besser kontrollieren, können wir größere Erfolge in der Stufe **Anwenden** erzielen.



Hypothese

Lernziel-Taxonomie (Bloom) beim Programmierens:

Erinnern	Kenntnis der Sprachkonstrukte.
Verstehen	Was macht dieses Programm? Wie funktionieren die Sprachkonstrukte?
Anwenden	Komposition der Sprachkonstrukte zu einer Lösung

Einstieg der Unterstützung in der Stufe **Anwenden**, keine Unterstützung für **Verstehen**.

Wir überspringen beim Feedback eine Stufe und **hoffen** nur, dass sie richtig verstanden wurde.

Hypothese: Wenn wir diesen Schritt besser kontrollieren, können wir größere Erfolge in der Stufe **Anwenden** erzielen.



Hypothese

Lernziel-Taxonomie (Bloom) beim Programmierens:

Erinnern	Kenntnis der Sprachkonstrukte.
Verstehen	Was macht dieses Programm? Wie funktionieren die Sprachkonstrukte?
Anwenden	Komposition der Sprachkonstrukte zu einer Lösung

Einstieg der Unterstützung in der Stufe **Anwenden**, keine Unterstützung für **Verstehen**.

Wir überspringen beim Feedback eine Stufe und **hoffen** nur, dass sie richtig verstanden wurde.

Hypothese: Wenn wir diesen Schritt besser kontrollieren, können wir größere Erfolge in der Stufe **Anwenden** erzielen.



Hypothese

Lernziel-Taxonomie (Bloom) beim Programmierens:

Erinnern	Kenntnis der Sprachkonstrukte.
Verstehen	Was macht dieses Programm? Wie funktionieren die Sprachkonstrukte?
Anwenden	Komposition der Sprachkonstrukte zu einer Lösung

Einstieg der Unterstützung in der Stufe **Anwenden**, keine Unterstützung für **Verstehen**.

Wir überspringen beim Feedback eine Stufe und **hoffen** nur, dass sie richtig verstanden wurde.

Hypothese: Wenn wir diesen Schritt besser kontrollieren, können wir größere Erfolge in der Stufe **Anwenden** erzielen.



Motivation

Modellvorstellung der Programmausführung

Speicherprotokolle

Automatische, interaktive Protokolle

Erste Erfahrungen

Fazit



Konstruktivismus (Lernpsychologie):

Lernende schaffen sich im Lernprozess eine individuelle Repräsentation der Welt.



Modellvorstellung der Programmausführung

Fokus auf Zuweisung, Bedingungen, Iteration, Rekursion, Array, Objekte etc.

Testfragen für den Fall von Zuweisungen [Bornat et al., 2008]:

<p>1. Read the following statements and tick the box next to the correct answer in the next column.</p> <pre>int a = 10; int b = 20; a = b;</pre>	<p>The new values of a and b are:</p> <table> <tbody> <tr><td><input type="checkbox"/></td><td>a = 10</td><td>b = 10</td></tr> <tr><td><input type="checkbox"/></td><td>a = 30</td><td>b = 20</td></tr> <tr><td><input type="checkbox"/></td><td>a = 0</td><td>b = 10</td></tr> <tr><td><input type="checkbox"/></td><td>a = 20</td><td>b = 20</td></tr> <tr><td><input type="checkbox"/></td><td>a = 0</td><td>b = 30</td></tr> <tr><td><input type="checkbox"/></td><td>a = 10</td><td>b = 20</td></tr> <tr><td><input type="checkbox"/></td><td>a = 20</td><td>b = 10</td></tr> <tr><td><input type="checkbox"/></td><td>a = 20</td><td>b = 0</td></tr> <tr><td><input type="checkbox"/></td><td>a = 10</td><td>b = 30</td></tr> <tr><td><input type="checkbox"/></td><td>a = 30</td><td>b = 0</td></tr> </tbody> </table> <p>Any other values for a and b:</p> <table> <tbody> <tr><td>a =</td><td>b =</td></tr> <tr><td>a =</td><td>b =</td></tr> <tr><td>a =</td><td>b =</td></tr> </tbody> </table>	<input type="checkbox"/>	a = 10	b = 10	<input type="checkbox"/>	a = 30	b = 20	<input type="checkbox"/>	a = 0	b = 10	<input type="checkbox"/>	a = 20	b = 20	<input type="checkbox"/>	a = 0	b = 30	<input type="checkbox"/>	a = 10	b = 20	<input type="checkbox"/>	a = 20	b = 10	<input type="checkbox"/>	a = 20	b = 0	<input type="checkbox"/>	a = 10	b = 30	<input type="checkbox"/>	a = 30	b = 0	a =	b =	a =	b =	a =	b =	<p>Use this column for your rough notes please</p>
<input type="checkbox"/>	a = 10	b = 10																																				
<input type="checkbox"/>	a = 30	b = 20																																				
<input type="checkbox"/>	a = 0	b = 10																																				
<input type="checkbox"/>	a = 20	b = 20																																				
<input type="checkbox"/>	a = 0	b = 30																																				
<input type="checkbox"/>	a = 10	b = 20																																				
<input type="checkbox"/>	a = 20	b = 10																																				
<input type="checkbox"/>	a = 20	b = 0																																				
<input type="checkbox"/>	a = 10	b = 30																																				
<input type="checkbox"/>	a = 30	b = 0																																				
a =	b =																																					
a =	b =																																					
a =	b =																																					



Modellvorstellung der Programmausführung

- ▶ We “study the range and consistency of mental models held by students of both simple (value assignment) and more challenging (reference assignment) programming concepts towards the end of a first-year course” [Ma et al., 2007].
- ▶ “Jeder Studierende baut sein eigenes Konzept (z.B. einer Schleife) auf. Die Konstruktionsleistung, die dem Aufbau eines solchen Konzepts zu Grund liegt, ist individuell sehr unterschiedlich und aus dem Verhalten schlecht ableitbar.” [Bauer-Messmer et al., 2005]
- ▶ In diesem Sinne wird in [Ma et al., 2007] argumentiert, dass “the problem of helping students to convert this diverse and persistent range of non-viable mental models into viable ones presents a real challenge to computer science educators”.



Modellvorstellung der Programmausführung

- ▶ We “study the range and consistency of mental models held by students of both simple (value assignment) and more challenging (reference assignment) programming concepts towards the end of a first-year course” [Ma et al., 2007].
- ▶ “Jeder Studierende baut sein eigenes Konzept (z.B. einer Schleife) auf. Die Konstruktionsleistung, die dem Aufbau eines solchen Konzepts zu Grund liegt, ist individuell sehr unterschiedlich und aus dem Verhalten schlecht ableitbar.” [Bauer-Messmer et al., 2005]
- ▶ In diesem Sinne wird in [Ma et al., 2007] argumentiert, dass “the problem of helping students to convert this diverse and persistent range of non-viable mental models into viable ones presents a real challenge to computer science educators”.



Modellvorstellung der Programmausführung

- ▶ We “study the range and consistency of mental models held by students of both simple (value assignment) and more challenging (reference assignment) programming concepts towards the end of a first-year course” [Ma et al., 2007].
- ▶ “Jeder Studierende baut sein eigenes Konzept (z.B. einer Schleife) auf. Die Konstruktionsleistung, die dem Aufbau eines solchen Konzepts zu Grund liegt, ist individuell sehr unterschiedlich und aus dem Verhalten schlecht ableitbar.” [Bauer-Messmer et al., 2005]
- ▶ In diesem Sinne wird in [Ma et al., 2007] argumentiert, dass “the problem of helping students to convert this diverse and persistent range of non-viable mental models into viable ones presents a real challenge to computer science educators”.



Verschiedene Modellvorstellungen

- ▶ Jeder Teilnehmer kann eine andere Modellvorstellung haben...
- ▶ In anderen Disziplinen mag es sinnvoll sein, mehrere Modell parallel zu nutzen (Physik), weil es kein einziges, richtiges gibt.
- ! Aber Informatik ist anders, weil es sich um selbst geschaffene, nach strengen Regeln ablaufende Systeme handelt. Kein Grund für “alternative Erklärungen”, kein Grund für “Analogien”.
Allenfalls für unterschiedliche Abstraktionsgrade.



Verschiedene Modellvorstellungen

- ▶ Jeder Teilnehmer kann eine andere Modellvorstellung haben...
- ▶ In anderen Disziplinen mag es sinnvoll sein, mehrere Modell parallel zu nutzen (Physik), weil es kein einziges, richtiges gibt.

! Aber Informatik ist anders, weil es sich um selbst geschaffene, nach strengen Regeln ablaufende Systeme handelt. Kein Grund für "alternative Erklärungen", kein Grund für "Analogien".

Allenfalls für unterschiedliche Abstraktionsgrade.



Verschiedene Modellvorstellungen

- ▶ Jeder Teilnehmer kann eine andere Modellvorstellung haben...
- ▶ In anderen Disziplinen mag es sinnvoll sein, mehrere Modell parallel zu nutzen (Physik), weil es kein einziges, richtiges gibt.
- ! Aber Informatik ist anders, weil es sich um selbst geschaffene, nach strengen Regeln ablaufende Systeme handelt. Kein Grund für “alternative Erklärungen”, kein Grund für “Analogien”.

Allenfalls für unterschiedliche Abstraktionsgrade.



Verschiedene Modellvorstellungen

- ▶ Jeder Teilnehmer kann eine andere Modellvorstellung haben...
- ▶ In anderen Disziplinen mag es sinnvoll sein, mehrere Modell parallel zu nutzen (Physik), weil es kein einziges, richtiges gibt.
- ! Aber Informatik ist anders, weil es sich um selbst geschaffene, nach strengen Regeln ablaufende Systeme handelt. Kein Grund für “alternative Erklärungen”, kein Grund für “Analogien”.
Allenfalls für unterschiedliche Abstraktionsgrade.



Müssen wir die Teilnehmer studieren?

- ▶ Warum Zeit darauf verwenden, zu verstehen, wie die aktuelle Vorstellung des Modells *eines individuellen Studierenden* aussieht (wäre nötig, um es zu korrigieren)?
- ▶ Warum nicht stattdessen von Beginn an die Entwicklung eines gangbaren Modells forcieren?



Müssen wir die Teilnehmer studieren?

- ▶ Warum Zeit darauf verwenden, zu verstehen, wie die aktuelle Vorstellung des Modells *eines individuellen Studierenden* aussieht (wäre nötig, um es zu korrigieren)?
- ▶ Warum nicht stattdessen von Beginn an die Entwicklung eines gangbaren Modells forcieren?



Aufbau der Modellvorstellung

Instructor Guidelines [Lui et al., 2004]:

1. Show behavior, no analogy (e.g. do not use the analogy of boxes for variables).
2. Build concepts first, jargon last (e.g. terms alone may provoke wrong presumptions).
3. Constant repetition.
4. ...

Speziell der dritte Punkt ist wichtig, so soll ein “Rückfall” auf Analogien (die sich Studierende selbst schaffen) verhindert werden.



Aufbau der Modellvorstellung

- ▶ In [Lui et al., 2004] wird dabei nicht von technischer Unterstützung gesprochen.
- ▶ Es gibt einige Systeme, die mit Visualisierungen beim Aufbau der Modellvorstellung helfen können,



Abläufe visualisieren: Jeliot [Moreno et al., 2004]

The screenshot displays the Jeliot software interface, which is used for visualizing program execution. It is divided into several sections:

- Code Editor:** Contains the following Java code:


```
import jeliot.io.*;

public class Random {
    public static void main() {
        int n = 6;
        int[] array = new int[n];
        int i, j, tmp;

        // initialize array
        for (i = 0; i < n; ++i) {
            array[i] = 1;
        }

        // randomize array
        for (i = n-1; i > 0; --i) {
            j = (int)(Math.random() * i);
            tmp = array[i];
            array[i] = array[j];
            array[j] = tmp;
        }
    }
}
```
- Visual Environment:** A central workspace with a corkboard background. It features:
 - A pink box labeled "main" containing variables: `n` (6), `array`, `i` (1), `j` (0), and `tmp` (2).
 - A yellow box with the value `0.0692`.
 - A green box with a multiplication symbol `*` and the value `1`.
 - A yellow box with the value `0.0692`.
 - A vertical stack of yellow boxes representing an array with values: 2, 3, 5, 4, 1, 0.
 - An orange box labeled "CONSTANTS".
- Control Panel:** Located at the bottom, it includes buttons for "Edit", "Compile", "Step", "Play", "Pause", and "Reset". Below these buttons is a slider for "Animation speed" and the Jeliot logo.
- Output Window:** A white box at the bottom right for displaying the program's output.



Aber: Jeliot und Jive visualisieren.

Student bleibt passiv, konsumiert.

Können wir sicher sein, dass Abläufe verstanden wurden?



Motivation

Modellvorstellung der Programmausführung

Speicherprotokolle

Automatische, interaktive Protokolle

Erste Erfahrungen

Fazit



- ▶ Entwicklung des Speicherinhalts über die Zeit
- ▶ von Beginn an bei jedem neuen Sprachkonstrukt genutzt
- ▶ präzisiert Erklärungen des Dozenten



Eigenschaften und Detailgrad

- ▶ abstrahiert von konkreter Ausführung in JVM
- ▶ genau genug, um die typischen Probleme darzustellen
- ▶ erlaubt nachträgliches Nachvollziehen aller Schritte
kein Schwammeinsatz ;)
- ▶ spätestens im 2. Semester sind die Details wichtig
- ▶ folgt [Lui et al., 2004]: concepts first, jargon last (keine Pfeile für Zeiger, kein Jargon (“zeigt auf”), sondern Adressen)



Integration in Lehre

- ▶ Herausforderungen stellen, Clickerabfrage → Lösung nicht “verraten”, sondern durch Ausfüllen des Protokolls erarbeiten.
- ▶ kann auch in schriftlichen Testaten (mitten im Semester) genutzt werden, sind aber mühsam zu korrigieren – und Fehler erfordern eigentlich weitere Aufgaben, weiteres Feedback
- ▶ Aber: Viele Studierende sind (trotz vieler Wiederholungen meinerseits) nicht in der Lage, diese Protokolle alleine richtig auszufüllen



Motivation

Modellvorstellung der Programmausführung

Speicherprotokolle

Automatische, interaktive Protokolle

Erste Erfahrungen

Fazit



- ▶ Web-Applikation zum Ausfüllen der Protokolle
- ▶ erlaubt (demnächst) spontane Code-Schnipsel anzubieten
- ▶ Lösung wird automatisch erstellt (über JDI)



Herausforderungen bei Umsetzung

- ▶ Problematik: Speicherinhalt *nach* Ausführung einer Zeile
- ▶ Debugger stoppt am Anfang einer Zeile
- ▶ Debugger zeigt keine uninitialisierten Variablen

```
int s = 4;
{
    int j = s*3;
    s+=++j;
}
System.out.println(s);
```

Abbildung: Änderung in letzter Zeile eines Blocks.

```
int i = 2;
int a = f(i);
if (a>0) {
    int j=3*(i%7); a+=j;
}
System.out.println(a);
```

Abbildung: Lokale Variable in Einzeiler-Block.

```
int y;
if (x>2) {
    y=0;
} else {
    y=2;
}
```

Abbildung: Späte Initialisierung.



```
1 |
2 |
3 | public class Array2 {
4 |
5 |     public static void main(String[] args) {
6 |         int[] a = new int[3];
7 |         int[] b = new int[3];
8 |         int barr = 0;
9 |         for(int i = 0; i < a.length; ++i){
10 |             a[i] = i+1;
11 |             if(a[i]%2 != 0 && barr < b.length){
12 |                 b[barr] = a[i];
13 |                 a[i] = 0;
14 |                 ++barr;
15 |             }
16 |             System.out.println();
17 |         }
18 |     }
19 | }
```

Line	Heap (0x0 -> 0x8)										Stack					
1	6		3	0	0	0					-	+	a=0x1			
2	7		3	0	0	0	3	0	0	0	-	+	a=0x1	b=0x5		
3	8		3	0	0	0	3	0	0	0	-	+	a=0x1	b=0x5	barr=i	
4	9		3	0	0	0	3	0	0	0	-	+	a=0x1	b=0x5	barr=i	i=0
5	10		3	1	0	0	3	0	0	0	-	+	a=0x1	b=0x5	barr=i	i=0
6	11		3	1	0	0	3	0	0	0	-	+	a=0x1	b=0x5	barr=i	i=0

Nicht genug Zeilen ausgefüllt!

- +

Evaluate



[Auswertung](#) ▾
 [Fehler](#) ▾
 [Steuerung](#) ▾
 [Aufgaben](#) ▾
 [Startseite](#)

```

1 package de.ostfalia.tracer;
2
3 public class Expr1 {
4
5     public static void main(String[] args) {
6         int x = 4;
7         int y = 2;
8     {
9         int yi = 2;
10        int z = 4 + yi;
11        y++;
12        z = x * z - 34;
13        x++;
14    }
15    {
16        int z = x * y;
17        z++;
18        x = z;
19    }
20    System.out.println(x);
21 }
22
23
    
```

Heap					Line	Stack				#			
					6	x=4	TS(1)						7
					1 LP(2)								
					7	x=4	y=2						7
					1 LP(2)		MS(1)						
					9	x=4	y=2	yi=2					7
					1 LP(2)		MS(1)	yi NV(1)					
					10	x=4	y=2	yi=2	z=6				7
					1 LP(2)		MS(1)	yi NV(1)	z NV(1)				
					11	x=4	y=3	yi=2	z=6				7
					10 LP(1)			yi NV(1)	z NV(1)				
					12	x=4	y=3	yi=2	z=-10				7
					11 LP(1)			yi NV(1)	z NV(1)				
					13	x=5	y=3	yi=2	z=-10				7
					12 LP(1)			yi NV(1)	z NV(1)				
					16	x=5	y=3	z=15	TS(2)				7
					14 LP(1)			z NV(1)					
					17	x=5	y=3	z=16	TS(1)				7
					15 LP(1)			z NV(1)					
					18	x=16	y=3	z=16	TS(1)				7
					16 LP(1)			z NV(1)					
					20	x=16	y=3		TS(2)				6
					19 LP(1)								



- ▶ Erklären, erstes Beispiel an der Tafel, zweites Beispiel jeder für sich, direkte Zusammenfassung des Stands im Anschluss
- ▶ Zeitlich unabhängiges Üben mit Feedback jederzeit möglich.
- ▶ Teil von Hausaufgaben: Entwicklung des Verständnisses ablesbar
- ▶ Gute Vorbereitung für den Einsatz des Debuggers



Motivation

Modellvorstellung der Programmausführung

Speicherprotokolle

Automatische, interaktive Protokolle

Erste Erfahrungen

Fazit



- ▶ sammeln erste Erfahrungen
- ▶ Versuch in einem Erstsemester
 - ▶ Aufteilung in 2 Gruppen (gleich stark nach Vorkenntnissen)
 - ▶ Hälfte übt 30 Min. mit Applikation, Hälfte auf Papier
 - ▶ danach Testat und Umfrage



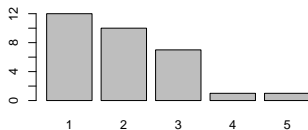
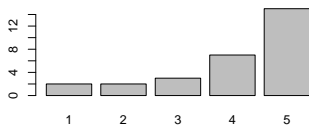
Ergebnis: Lebenszeit von Variablen

	Vorbereitung		
	Papier	Werkzeug	
Variablen-Lebenszeit falsch	11	3	14
Variablen-Lebenszeit richtig	12	26	38
	23	29	

$$\chi^2 = 7.353, p < 1\%$$



- ▶ “Das Werkzeug war hilfreich für meine Vorbereitung” (links)
- ▶ “Die Papierversion wäre ausreichend gewesen” (rechts)



Likert-Skala (1='strongly disagree', 5='strongly agree')



Motivation

Modellvorstellung der Programmausführung

Speicherprotokolle

Automatische, interaktive Protokolle

Erste Erfahrungen

Fazit



Verständnis der Sprachkonstrukte

Wie bekommt ein Anfänger die Sprachkonstrukte vermittelt?

- ▶ *Dozent* erklärt Semantik von Anweisungen/Konzepten
- ▶ *Dozent* gibt ein oder mehrere Beispiele
- ▶ Vereinzelt erklären *freiwillige Studierende* auch mal ein Beispiel
- ▶ Nächster Schritt ist das Lösen eigener Aufgaben
- ▶ Kontrolle des Verständnisses erst bei der Erklärung ihrer Lösungen (wenn sie denn soweit gekommen sind)



Verständnis der Sprachkonstrukte

Wie bekommt ein Anfänger die Sprachkonstrukte vermittelt?

- ▶ *Dozent* erklärt Semantik von Anweisungen/Konzepten
- ▶ *Dozent* gibt ein oder mehrere Beispiele
- ▶ Vereinzelt erklären *freiwillige Studierende* auch mal ein Beispiel
- ▶ Nächster Schritt ist das Lösen eigener Aufgaben
- ▶ Kontrolle des Verständnisses erst bei der Erklärung ihrer Lösungen (wenn sie denn soweit gekommen sind)



Verständnis der Sprachkonstrukte

Wie bekommt ein Anfänger die Sprachkonstrukte vermittelt?

- ▶ *Dozent* erklärt Semantik von Anweisungen/Konzepten
- ▶ *Dozent* gibt ein oder mehrere Beispiele
- ▶ Vereinzelt erklären *freiwillige Studierende* auch mal ein Beispiel
- ▶ Nächster Schritt ist das Lösen eigener Aufgaben
- ▶ Kontrolle des Verständnisses erst bei der Erklärung ihrer Lösungen (wenn sie denn soweit gekommen sind)



Verständnis der Sprachkonstrukte

Wie bekommt ein Anfänger die Sprachkonstrukte vermittelt?

- ▶ *Dozent* erklärt Semantik von Anweisungen/Konzepten
- ▶ *Dozent* gibt ein oder mehrere Beispiele
- ▶ Vereinzelt erklären *freiwillige Studierende* auch mal ein Beispiel
- ▶ Nächster Schritt ist das Lösen eigener Aufgaben

- ▶ Kontrolle des Verständnisses erst bei der Erklärung ihrer Lösungen (wenn sie denn soweit gekommen sind)



Verständnis der Sprachkonstrukte

Wie bekommt ein Anfänger die Sprachkonstrukte vermittelt?

- ▶ *Dozent* erklärt Semantik von Anweisungen/Konzepten
- ▶ *Dozent* gibt ein oder mehrere Beispiele
- ▶ Vereinzelt erklären *freiwillige Studierende* auch mal ein Beispiel
- ▶ Nächster Schritt ist das Lösen eigener Aufgaben

- ▶ Kontrolle des Verständnisses erst bei der Erklärung ihrer Lösungen (wenn sie denn soweit gekommen sind)



- ▶ Anfänger haben oft viele Baustellen
(Lesekompetenz, Problemlösekompetenz, Ablaufmodell / Sprachkonstrukte, Abstraktionsvermögen, Kreativität, Selbstorganisation, Diagnosefähigkeit, Kommunikation)
 - ▶ Automatisierte Tests zielen auf das Endprodukt.
 - ▶ Einige Schritte sind sehr schwer zu unterstützen (z.B. Abstraktionsvermögen)
 - ▶ Andere Probleme lassen sich mglw. leichter aus dem Spiel nehmen.
- Einsatz von automatisch kontrollierbarem Verständnis des Ablaufs kann so ein Problem sein.



- ▶ Anfänger haben oft viele Baustellen
(Lesekompetenz, Problemlösekompetenz, Ablaufmodell / Sprachkonstrukte, Abstraktionsvermögen, Kreativität, Selbstorganisation, Diagnosefähigkeit, Kommunikation)
 - ▶ Automatisierte Tests zielen auf das Endprodukt.
 - ▶ Einige Schritte sind sehr schwer zu unterstützen (z.B. Abstraktionsvermögen)
 - ▶ Andere Probleme lassen sich mglw. leichter aus dem Spiel nehmen.
- Einsatz von automatisch kontrollierbarem Verständnis des Ablaufs kann so ein Problem sein.



- ▶ Anfänger haben oft viele Baustellen
(Lesekompetenz, Problemlösekompetenz, Ablaufmodell / Sprachkonstrukte, Abstraktionsvermögen, Kreativität, Selbstorganisation, Diagnosefähigkeit, Kommunikation)
 - ▶ Automatisierte Tests zielen auf das Endprodukt.
 - ▶ Einige Schritte sind sehr schwer zu unterstützen (z.B. Abstraktionsvermögen)
 - ▶ Andere Probleme lassen sich mglw. leichter aus dem Spiel nehmen.
- Einsatz von automatisch kontrollierbarem Verständnis des Ablaufs kann so ein Problem sein.







- ▶ Anfänger haben oft viele Baustellen
(Lesekompetenz, Problemlösekompetenz, Ablaufmodell / Sprachkonstrukte, Abstraktionsvermögen, Kreativität, Selbstorganisation, Diagnosefähigkeit, Kommunikation)
 - ▶ Automatisierte Tests zielen auf das Endprodukt.
 - ▶ Einige Schritte sind sehr schwer zu unterstützen (z.B. Abstraktionsvermögen)
 - ▶ Andere Probleme lassen sich mglw. leichter aus dem Spiel nehmen.
- Einsatz von automatisch kontrollierbarem Verständnis des Ablaufs kann so ein Problem sein.






- ▶ Anfänger haben oft viele Baustellen
(Lesekompetenz, Problemlösekompetenz, Ablaufmodell / Sprachkonstrukte, Abstraktionsvermögen, Kreativität, Selbstorganisation, Diagnosefähigkeit, Kommunikation)
 - ▶ Automatisierte Tests zielen auf das Endprodukt.
 - ▶ Einige Schritte sind sehr schwer zu unterstützen (z.B. Abstraktionsvermögen)
 - ▶ Andere Probleme lassen sich mglw. leichter aus dem Spiel nehmen.
- Einsatz von automatisch kontrollierbarem Verständnis des Ablaufs kann so ein Problem sein.



-  Bauer-Messmer, B., Fässler, L., and Wyss, M. (2005).
Einführungskurse ins Programmieren - eine didaktische Herausforderung.
Technical report, ETH Zürich.
-  Bornat, R., Dehnadi, S., and Simon (2008).
Mental models, consistency and programming aptitude.
Conferences in Research and Practice in Information Technology Series, 78(1986):53–61.
-  Lessa, D., Jayaraman, B., and Czyz, J. K. (2010).
JIVE: A pedagogic tool for visualizing the execution of Java programs.
Technical report, University of New York at Buffalo.
-  Lui, A. K., Kwan, R., Poon, M., and Cheung, Y. H. Y. (2004).
Saving weak programming students.
ACM SIGCSE Bulletin, 36(2):72.



-  Ma, L., Ferguson, J., Roper, M., and Wood, M. (2007). Investigating the viability of mental models held by novice programmers.
Proceedings of the 38th SIGCSE technical symposium on Computer science education - SIGCSE '07, page 499.
-  Moreno, A., Myller, N., Sutinen, E., and Ben-Ari, M. (2004). Visualizing programs with Jeliot 3.
Proceedings of the working conference on Advanced visual interfaces - AVI '04, page 373.
-  Vogts, D. (2009). Plagiarising of source code by novice programmers a 'cry for help'?
Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, SAICSIT 2009, (October):141–149.