Vorbesprechung
Abstrakter Datentyp
Datenkapselung

Herzlich Willkommen zu

Einführung in die Programmierung 2

Das Team

Vortragende

M. Anton Ertl
Ulrich Neumerkel
Michael Pollak
Franz Puntigam
Michael Reiter
Dietmar Schreiner
Sebastian Zambanini

Tutor_innen

Sergey Andreenkov
Lachezar Damyanov
Lukas Dichtl
Svenja Einwögerer
Ivaylo Georgiev
Radina Grancharova
Gabriel Korherr
Oliver Kubicek
Simon Mader
Rikardo Minollari
Kalvin Ofenböck
Gloria Reinbacher
Sila Saricam
Simon Schachenhofer
Julian Schwendinger
Franz Uchatzi
Fridolin Ulbel
Konstantin Unterweger
Zeki Uzunbacak

Vorausgesetzte Kompetenzen

einfache Java-Programme: erstellen
tracen (Programmablauf nachzeichnen)
modifizieren
debuggen
dokumentieren

beschreiben: prozedurale Konzepte von Java,
Such- und Sortieralgorithmen auf Arrays,
häufige Fehlerquellen in Programmen

bezüglich Zusammenarbeit: selbstständig programmieren,
einfache Programmeigenschaften kommunizieren

Zu erwerbende Kompetenzen (Lernergebnisse)

umsetzen: natürlichsprachige Programmieraufgabe → ausführbares Programm

implementieren, verwenden: vorgegebenen abstr. Datentyp, Algorithmus, Datenstruktur

anwenden: Techniken für Datenabstraktion, Traversierung, Qualitätssicherung, ...

beschreiben: systematische Vorgehensweisen beim Programmieren,
Queue, Stack, Liste, Baum, Hashtabelle, Iterator, Mergesort, etc.,
Fehlerquellen und Techniken zur Qualitätssicherung

bezüglich Zusammenarbeit: im Zweierteam und selbständig programmieren,
Programmeigenschaften kommunizieren

Themengebiete

Java

Sprachkonstrukte

Berechnungsmodell

Software-Engineering

Algorithmen & Datenstrukturen

Programmorganisation

Qualitätssicherung

Aufgabe: Themengebiete und Vorerfahrung

Diskutieren Sie in Gruppen zu 2 bis 3 Personen:

1. Welche Themengebiete werden in EP2 behandelt?
2. Wie wichtig sind diese Themengebiete jeweils (eigenes Empfinden)?
3. Welche Vorerfahrungen haben Sie auf diesen Gebieten?

Zeit: 2 Minuten

Lernaktivitäten

Vorlesungen	inhaltliche Übersicht, Beispiele
Skriptum	inhaltliche Details, Aufgabensammlung
Aufgabenblätter	praktisches Üben bei freier Zeiteinteilung
geleitete Übungen	Lösungen präsentieren und diskutieren, Feedback, unter kontrollierten Bedingungen programmieren
Tutorium	zwangloser Erfahrungsaustausch
Tests	gezielte Vorbereitung, Leistungskontrolle
freies Üben	Üben ohne Vorgaben

Aufgabenblätter

7 Aufgabenblätter (AB1...AB7)

Termine siehe **Semestereinteilung** (TUWEL),
Aufgaben in IntelliJ-IDEA-Projekt auf Git-Server verfügbar,
Lösung vor Deadline **auf Git-Server hochladen** (AB1...AB7)

Aufgabenblätter ohne fremde Hilfe **selbständig lösen**

(weil das die beste Vorbereitung auf die Tests ist)

Zusammenarbeit in **Lerngruppen** erlaubt

jede_r Studierende muss selbst hochladen und **alleine** präsentieren

Geleitete Übungen

Teilnahme an 9 Übungseinheiten (ÜE1. . . ÜE9), je 45 Minuten

ausgewählte Studierende **präsentieren Lösungen** und vergleichen mit Musterlösung,
Auswahl durch freiwillige Meldung, Zufall, Leistungen (ÜE1. . . ÜE3, ÜE5. . . ÜE8)

Übungstest = zur Übung zählender kurzer Programmiertest,
paarweise (ÜE1, ÜE2, ÜE5) oder alleine (ÜE3, ÜE6. . . ÜE8),
Aufgaben bauen auf Musterlösung auf, hängen von Lösung des Aufgabenblatts ab

Testvorbereitung = selbständiges Lösen von Aufgaben wie bei Tests (ÜE4, ÜE9)
aber ohne Beurteilung der Leistung

Terminauswahl bei **Anmeldung zu Übungsgruppe**
Anmeldungen in TISS ab heute, 17:00 Uhr bis spätestens Freitag (7. 3.), 10:00 Uhr,
Änderung der Übungstermine nach Ende der Anmeldefrist nicht möglich

Beurteilung der geleiteten Übungen

max. 49 Punkte **Aufgabenblätter** (⌀ 7 pro Aufgabenblatt, nach Schwierigkeitsgrad)
max. 2 Punkte **Anwesenheit** (1 Punkt pro Vorbereitungstest in ÜE4, ÜE9)
max. 49 Punkte **Übungstests** (7 pro Übungstest in ÜE1...ÜE3, ÜE5...ÜE8)

0 Punkte für Aufgabenblatt (AB) ohne Teilnahme an Übungseinheit (Besprechung),
Punktereduktion für AB bei Weigerung, Nichtverstehen eigener Lösung oder Störung,
Mitarbeitspunkte bei besonders guter Mitarbeit,
Mitarbeitspunkte werden Punkten für Übungstest der ÜE zugeschlagen (max. 7 pro ÜE)

Beurteilung pro Aufgabenblatt:

- Phase 1: automatisierte Beurteilung über Testfälle (unzuverlässig)
- Phase 2: händische Beurteilung durch Tutor_in mit Feedback
  (nachträglich, nicht immer, überschreibt automatisierte Beurteilung)

Aufgabe: Arbeitsaufwand pro Übungspunkt

Für die Lösung eines Aufgabenblatts sind 3 bis 9 Stunden veranschlagt.

Diskutieren Sie in Gruppen zu 2 bis 3 Personen:

1. Wie hoch ist der Aufwand pro Punkt wenn man Übungstests und Punkte für Mitarbeit nicht betrachtet?
2. Wie hoch ist der Aufwand pro Punkt wenn man berücksichtigt, dass Aufgabenblätter und deren Besprechungen auf das Lösen von Übungstests vorbereiten?
3. Wie hoch ist der Aufwand in Relation zum Ergebnis wenn man berücksichtigt, dass Aufgabenblätter und Übungstests auf die großen Tests vorbereiten?

Zeit: 2 Minuten

Tests

2 reguläre Tests zu je 75 Minuten: T1 am 30. 4. und T2 am 25. 6.

praktische Programmieraufgaben und Multiple-Choice-Aufgaben,
ohne Hilfsmittel an Computern in mehreren Laboren

zu jedem regulären Test ein Nachtragstest:

T1n am 23. 6. für T1, T2n Ende September für T2,

Teilnahme nur wenn entsprechender regulärer Test negativ ($< 50\%$) oder versäumt
und Übungsbeurteilung positiv ($\geq 50\%$)

Gesamtbeurteilung

Gesamtbeurteilung positiv wenn jede dieser Bedingungen zutrifft:

T1 oder Nachtragstest T1n $\geq 50\%$
T2 oder Nachtragstest T2n $\geq 50\%$
Beurteilung der geleiteten Übungen $\geq 50\%$

Wenn Beurteilung positiv ergibt sich Note aus Durchschnitt dieser 3 Leistungen in %:

sehr gut:	87.5% – 100.0%
gut:	75.0% – 87,5%
befriedigend:	62.5% – 75.0%
genügend:	50.0% – 62.5%

Tutorium und Kommunikation

Tutorium = entspanntes Treffen mit Studierenden und Tutor_innen:

Mittwoch (ab 12. 3.), 14:30 – 16:30 Uhr, Pong-Raum im InfLab

Informationsaustausch über **TUWEL-Diskussionsforum**

Mail an EP2-Team (Vortragende): `ep2@complang.tuwien.ac.at`

Mail an Tutor_in der Übungsgruppe: ⟨Gruppenname⟩`@complang.tuwien.ac.at`

Nächste Schritte

Vorlesungen besuchen	Audi.Max.	4. 3., 11. 3., . . . , 14:00 – 16:00 (c.t.)
Anmeldung zu Übungsgruppe	TISS	4. 3., 17:00 bis 7. 3., 10:00 (s.t.)
Informationen zu Git erhalten	eMail	spätestens am 11. 3.
Projekt mit AB1 herunterladen	Git	ab 11. 3., ca. 16:00
Tutorium (optional)	Pong	12. 3., 19. 3., . . . , 14:30 – 16:30
Projekt mit Lösung AB1 hochladen	Git	bis spätestens 18. 3., 13:00 (s.t.)
an Übung ÜE1 teilnehmen	InfLab	ab 19. 3. (je nach Gruppe)

Aufgabe: Abstraktion in der Programmierung

Methoden sind ein wichtiges Werkzeug zur Abstraktion.

Diskutieren Sie in Gruppen zu 2 bis 3 Personen:

1. Aufgrund welcher Eigenschaften eignen sich Methoden für die Abstraktion?
2. Kennen Sie weitere Werkzeuge zur Abstraktion in der Programmierung?

Zeit: 2 Minuten

Klassifizierung und Modularisierung

```
public class TestFactorial {
    public static void main(String[] args) {
        System.out.println(Factorial.fact(3));
    }
}
```

Verweise auf andere Klassen

```
public class Factorial {
    public static long fact(int n) {
        return n < 2 ? (long)n : n * fact(n - 1);
    }
}
```

Verweis auf Methode innerhalb der Klasse
entspricht `Factorial.fact(n - 1)`

Verwendungsbeispiele abstrakter Datentypen

```
private static void printFifthChar(String s) {
    if (s != null && s.length() > 4) {
        System.out.print(s.charAt(4));
    }
}

private static void echo() {
    Scanner scanner = new Scanner(System.in);
    while (scanner.hasNextLine()) {
        System.out.println(scanner.nextLine());
    }
}
```

String

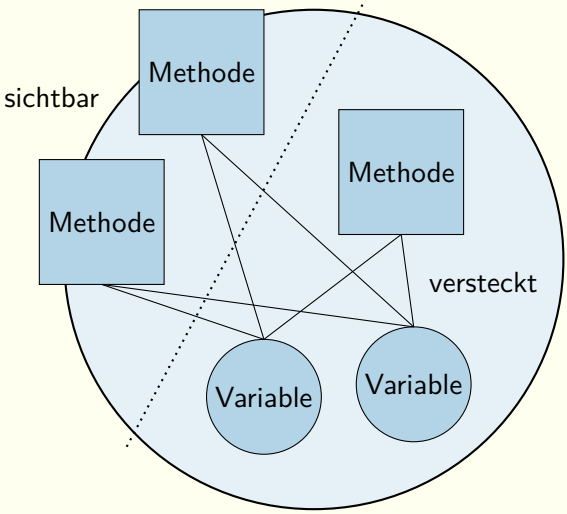
Referenz auf Objekt

Scanner

Datenabstraktion = Datenkapselung + Data-Hiding

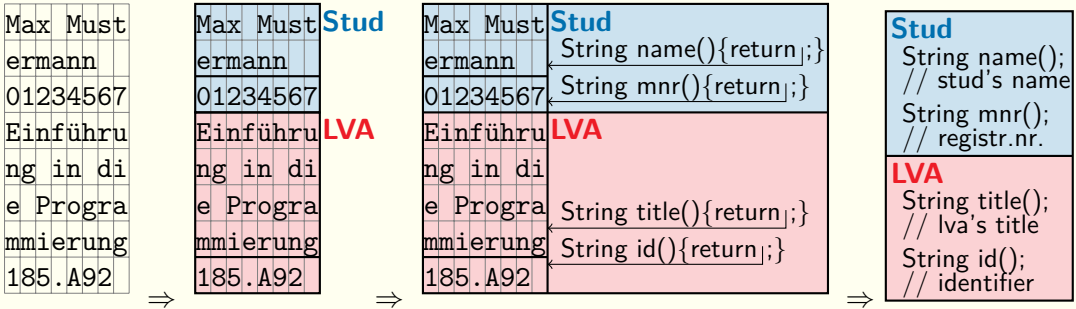
Datenkapselung = Zusammenfassen
von Methoden und Variablen zu Einheit

Data-Hiding = Verstecken
vor Zugriffen von außen



Datenabstraktion

unstrukt. Daten ⇒ Records oder Structs ⇒ Objekte von Klassen, Klasse = Implementierung abstr. Datentyps ⇒ abstrakte Datentypen



⇒ Klassifizierung ⇒ Datenkapselung ⇒ Data-Hiding

Aufgabe: Typische Eigenschaften von `String` und `Scanner`

In Gruppen zu 2 bis 3 Personen:

Beschreiben Sie typische Eigenschaften von `String` und `Scanner`.

Zeit: 2 Minuten

Abstrakten Datentyp entwerfen

```
/*******************************************************
class BoxedText: Rectangular text within border lines.
public methods:
  void newDimensions(int width, int height);
  void setLine(int index, String txt);
  void print();
  String toString();
*******************************************************/

private static void testBoxedText() {
    BoxedText t = new BoxedText();
    t.newDimensions(10, 3);
    t.setLine(1, "Das ist ein Text");
    t.print();
}
```

BoxedText als abstrakter Inhalt einer Objektreferenz

Implementierung eines abstrakten Datentyps

```
public class BoxedText {
    private int textWidth = 0;
    private int textHeight = 0;
    private char[][] text = new char[0][];

    public void newDimensions(int width, int height) {
        textWidth = width;
        textHeight = height;
        text = new char[height][width];
        for (char[] line : text) { fill(line, 0); }
    }
    private void fill(char[] line, int i) {
        for (; i < textWidth; i++) { line[i] = ' '; }
    }
    ...
```

Deklarationen von Objektvariablen

Objektmethode

Zugriffe auf Objektvariablen

private Objektmethode

Aufgabe: Objektvariablen erkennen

In Gruppen zu 2 bis 3 Personen:

Woran können wir erkennen, ob eine zugegriffene Variable eine Objektvariable ist?

Zeit: 2 Minuten

Arten von Variablen

Variablendeklaration in		
	Parameterliste	→ Parameter
	Methodenrumpf	→ lokale Variable
	Klasse (ohne Modifier `static`)	→ **Objektvariable**
	Klasse (mit Modifier `static`)	→ **Klassenvariable**

Objektvariable
existiert einmal pro erzeugtem Objekt
in Objektmethoden derselben Klasse direkt zugreifbar
wird automatisch mit `0`, `0.0` oder `null` vorinitialisiert

Klassenvariable
existiert insgesamt nur einmal — vermeiden
in Objekt- und Klassenmethoden zugreifbar
wird automatisch mit `0`, `0.0` oder `null` vorinitialisiert

Arten von Methoden

Methodendefinition	ohne Modifier `static` → **Objektmethode** mit Modifier `static` → **Klassenmethode**

Objektmethode

kann auf Objektvariablen derselben Klasse direkt zugreifen
greift häufig auf Objektvariablen zu – enge Zusammenarbeit
ermöglicht abstrakte Datentypen — ist wichtig
Aufruf z.B. `x.newDimensions(10, 3)` (`x` vom Typ `BoxedText`)
vereinfachter Aufruf auf aktuellem Objekt z.B. `fill(line, 0)`

Klassenmethode

kein direkter Zugriff auf Objektvariablen
nicht für abstrakte Datentypen (im engen Sinn) verwendbar — meiden
Aufruf z.B. `Factorial.fact(n - 1)`
vereinfachter Aufruf in eigener Klasse z.B. `fact(n - 1)`

Verwendung mehrerer Objekte

```
private static void testBoxedText2() {
    BoxedText x = new BoxedText();
    BoxedText y = new BoxedText();

    x.newDimensions(10, 3);
    y.newDimensions(20, 1);
    x.setLine(1, "Das ist ein Text");
    y.setLine(0, "Anderer Text");
    x.print();
    y.print();
}
```

unterschiedliche Objekte
→ unterschiedliche Objektvariablen

```
------------
|          |
|Das ist ei|
|          |
------------

----------------------
|Anderer Text        |
----------------------
```