FernUniversität in Hagen

FAKULTÄT FÜR
MATHEMATIK UND INFORMATIK

Lehrgebiet Datenverarbeitungstechnik
Univ.-Prof. Dr.-Ing. Bernd Krämer

Kurs 20022: Objektorientiertes Programmieren

Errata: Fehler im Kursmaterial vom SS 2012 (Stand: 30. Juli 2012)

Anbei finden Sie eine Liste bekannter Fehler in den Kursunterlagen (die wir selbstverständlich bei der nächsten Auflage der Materialien beheben werden). Finden Sie weitere, hier noch nicht verzeichnete Fehler, senden Sie uns bitte eine Fehlerbeschreibung an oop.tutor@fernuni-hagen.de zu, so dass wir den Kurs verbesseren und auch Ihre Kommilitonen von Ihrer Beobachtung profitieren können. Vielen Dank für Ihre Unterstützung.

Kurseinheit 1, Bemerkung 2.5-1

Bemerkung 2.5-1: Schlüsselwörter und Stereotypen in der UML
Bei der Beschriftung `«include»` handelt es sich um reserviertes Schlüsselwort der UML. Diese werden, ebenso wie Stereotypen in spitzen Klammern dargestellt und kommen bei verschiedensten Elementen zum Einsatz.

Kurseinheit 1, Tabelle 2.6-1, S. 18

Im Abschnitt Fehlersituation muss die rechte Spalte folgendermaßen lauten:
Situationen, die bei diesem Schritt zu einer nicht erfolgreichen Ausführung führen. Beschreibung des Systemzustands nach einer nicht erfolgreichen Ausführung

Kurseinheit 1, Kapitel 2.5, S. 15

Auch `«extend»` ist ein Schlüsselwort und kein Stereotyp.

Kurseinheit 1, Lösung zu SA 3.2-2, S. 59

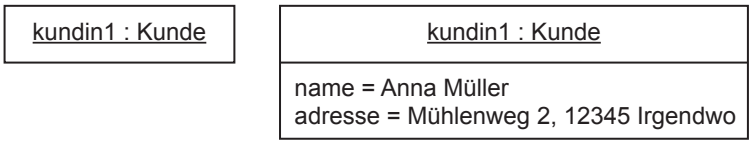
Bei Rechnungsnummer handelt es sich ebenfalls um ein Attribut.

Kurseinheit 1, Kapitel 3.6, S. 33 f.

In einem Objektdiagramm muss der optionale Name und der mit einem Doppelpunkt abgetrennte Typ unterstrichen werden. Der Satz aus S. 33 lautet folglich: *In einem UML-Objektdiagramm wird ein Objekt durch ein Rechteck dargestellt. In dem Rechteck stehen, durch einen Doppelpunkt getrennt und unterstrichen, der Name und die Klasse des Objekts.*

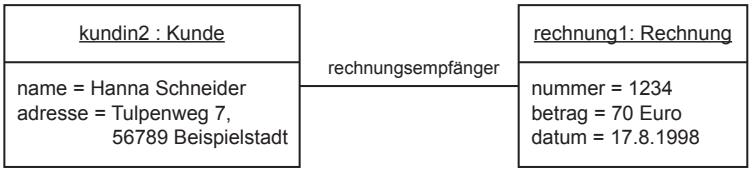
Kurseinheit 1, Abbildung 3.6-1

In Abbildung 3.6-1 müssen die Objektnamen unterstrichen werden:



Kurseinheit 1, Lösung zu SA 3.6-1, Abb. 3.6-2

In Abbildung 3.6-2 müssen die Objektnamen unterstrichen werden:



Kurseinheit 1, Bemerkung 6.2-2, S. 53

Neben syntaktischen Fehler gibt es auch noch andere Fehler, die bei der Übersetzung gefunden werden, wie zum Beispiel Typfehler. Die Bemerkung muss somit lauten:
Bemerkung 6.2-2: Übersetzungsfehler vs. Laufzeitfehler
Übersetzungsfehler, sind Fehler im Programm, die während der Übersetzung entdeckt werden. Ein Beispiel für Übersetzungsfehler sind syntaktische Fehler. Bei der Ausführung können sowohl Fehler auftreten, die das Programm zum Abbruch zwingen, als auch Fehler im Sinne von falschen Ergebnissen oder fehlerhaftem Verhalten.

Kurseinheit 2, Kapitel 9.3, S. 79 f.

Die Definition für Gleitkommaliterale ist fehlerhaft und muss korrekterweise folgendermaßen lauten:
Literale für Gleitkommazahlen (engl. floating-point number) werden durch eine Sequenz folgender Zeichen dargestellt:

- einer optionalen Folge von Dezimalziffern (0, 1, ..., 9),
- einem optionalen Punkt „.“,
- einer optionalen Folge von Dezimalziffern (0, 1, ..., 9),
- einem optionalen Exponentenpart, der den Exponenten auf der Basis 10 darstellt, bestehend aus dem Buchstaben e oder E sowie einer Folge aus einem optionalen Vorzeichen („+“oder „-“) und einer Dezimalzahl,
- einem optionalen Buchstaben f (F) oder d (D) zur Unterscheidung einer float- von einer double-Gleitkommazahl doppelter Präzision.

Sie unterliegen den folgenden Gestaltungsregeln:

- Es muss immer mindestens eine Dezimalziffer außerhalb des Exponentenparts enthalten sein.
- Es muss entweder der Punkt, der Exponentenpart oder der Buchstabe am Ende enthalten sein, damit das Literal als Gleitkommazahl interpretiert wird. *[JLS: § 3.10.2]*

Kurseinheit 2, Kapitel 10, S. 93

Variablen müssen vor dem ersten lesenden Zugriff initialisiert, d. h. erstmalig mit einem Wert belegt werden. Dies kann zum Beispiel bei ihrer Deklaration geschehen, wenn diese mit einer Zuweisung verbunden wird. Eine Initialisierung bei der Deklaration hat die Form: `type varName = expression;`

Kurseinheit 2, Kapitel 14.1, S. 120

Im letzten Quelltextbeispiel auf dieser Seite hat sich ein Syntaxfehler eingeschlichen, hinter condition4 fehlt eine schließende Klammer. Die korrigierte Fassung lautet:

```
while ((condition1 && condition2)
        || (condition3 && condition4)) {
    statement;
}
```

Kurseinheit 2, Lösung zu Selbsttestaufgabe 14.1-3

In der Lösung wird die `for`-Schleife verwendet, die zu diesem Zeitpunkt noch nicht bekannt ist. Allerdings ist dies eine typische Stelle an der man eine `for`-Schleife verwenden würde. Allerdings lässt sich die Aufgabe natürlich auch mit Hilfe einer `while`-Schleife formulieren.

```
int breite = ...;
int hoehe = ...;
int h = 0;
while (h < hoehe) {
    int b = 0;
    while (b < breite) {
        System.out.print("*");
        b++;
    }
    System.out.println();
    h++;
}
```

Kurseinheit 2, Kapitel 15, S. 132

Im Absatz unter Programm 15-1 muss der letzte Satz lauten: Hätten wir aber in Block F1 anstelle von `temp` den Namen `i`, `treffer`, `summe` oder `gefunden` gewählt, wäre das Programm illegal geworden.

Kurseinheit 3, Kapitel 17.2, S. 161

Auch bei «`create`» handelt es sich um ein Schlüsselwort und nicht um einen Stereotyp.

Kurseinheit 3, Selbsttestaufgabe 18.3-8

Hier muss es in der Bedingung in der Lösung `neuerRabatt` lauten. Die gesamte Methode lautet somit:

```
void legeRabattFest(final double neuerRabatt) {
    // neuer Rabatt ist nur von 0% bis 50% gültig
    if (neuerRabatt >= 0 && neuerRabatt <= 0.5) {
        this.rabatt = neuerRabatt;
    }
}
```

Kurseinheit 3, Selbsttestaufgabe 18.3-9

Hier kann in der Lösung für die Methode `legeRabattFest` natürlich auch die Lösung aus SA 18.3-8 verwendet werden.

Kurseinheit 3, Kapitel 20.2, S. 200

Hier muss die Deklaration der Felder `chessboard` und `dreieck` jeweils mit einem Semikolon abgeschlossen werden:

```
int[][] chessboard = {{0, 1, 2, 3, 4, 5, 6, 7},
                      {1, 2, 3, 4, 5, 6, 7, 8},
                      {2, 3, 4, 5, 6, 7, 8, 9},
                      {3, 4, 5, 6, 7, 8, 9, 10},
                      {4, 5, 6, 7, 8, 9, 10, 11},
                      {5, 6, 7, 8, 9, 10, 11, 12},
                      {6, 7, 8, 9, 10, 11, 12, 13},
                      {7, 8, 9, 10, 11, 12, 13, 14}};

int[][] dreieck = {{0},
                   {1, 2},
                   {2, 3, 4},
                   {3, 4, 5, 6},
                   {4, 5, 6, 7, 8},
                   {5, 6, 7, 8, 9, 10},
                   {6, 7, 8, 9, 10, 11, 12},
                   {7, 8, 9, 10, 11, 12, 13, 14}};
```

Kurseinheit 3, Lösung zu Selbsttestaufgabe 20.2-2

In der Aufgabenstellung wird von einer Matrix vom Typ `double` gesprochen, deshalb sollte die Lösung folgendermaßen aussehen:

```
double[][] einheitsmatrix = new double[6][6];
for (int i = 0; i < einheitsmatrix.length; i++) {
    einheitsmatrix[i][i] = 1;
}
```

Kurseinheit 3, Kapitel 21, S. 205

Hier muss es lauten:
Die Klasse `String` dient zur Darstellung von Zeichenketten.

Einsendeaufgabe zu Kurseinheit 3, Aufgabe 1 a

Da `new` ein Schlüsselwort und kein Operator ist, muss Antwort F lauten:
Mit Hilfe des Schlüsselwortes `new` und eines Konstruktors werden neue Exemplare erzeugt.

Kurseinheit 4, Kapitel 22.2, S. 237

Hier muss es lauten:
Ein durch eine Klasse `U` festgelegter Typ ist genau dann ein Subtyp einer durch die Klasse `O` festgelegten Typs, wenn `U` und `O` identisch sind oder `U` eine Unterklasse von `O` ist.

Kurseinheit 4, Selbsttestaufgabe 22.2-1

Hier hat sich in den Attributzugriffen und Methodenaufrufen ein Tippfehler eingeschlichen. Statt

```
new Lehrgebiet()().inhaber.gebeRaumnummerAus()
```

sollte es

```
new Lehrgebiet().inhaber.gebeRaumnummerAus()
```

lauten.

Kurseinheit 4, Definition 23.1-1

Hier haben sich zwei verschiedene Paketnamen eingeschlichen, die Definition sollte natürlich lauten:
Ein Paket (engl. package) wird wie folgt eingeführt:

```
package paketName;
```

Diese Anweisung muss am Anfang einer Übersetzungseinheit (z. B. einer einzelnen Quelldatei) vor die Klassendeklaration gesetzt werden, die zum Paket `paketName` gehören soll. [JLS: § 7]

Kurseinheit 4, Beispiel 24.2-1: Zugriff auf Klassen

In Klasse `L` ist die Klasse `M` sichtbar, es muss somit lauten:

```
public class L {
    // Hier sind die Klassen
    // K, M und N sichtbar
}
```

Kurseinheit 4, Kapitel 25.1, S. 278

Die Beschreibung der Darstellung abstrakter Klassen in der UML ist fehlerhaft und erfolgt nicht über Stereotypen. In der UML wird eine Klassen als abstrakt gekennzeichnet, indem ihr Name kursiv geschrieben wird. Alternativ kann zur Kennzeichnung hinter oder unter dem Klassennamen `{abstract}` ergänzt werden.

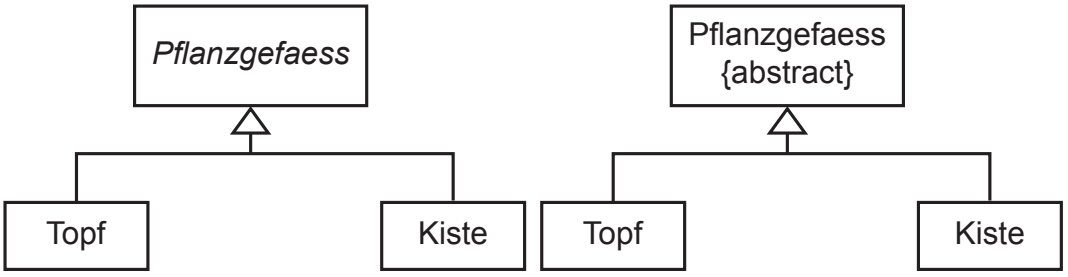


Abbildung 1: Darstellung abstrakter Klassen

Kurseinheit 4, Abbildung 25.1-1

Die Darstellung der abstrakten Klasse ist falsch. Die beiden korrekten Varianten finden Sie in Abbildung 1.

Kurseinheit 4, Kapitel 25.2, S. 285

In UML-Klassendiagrammen werden Schnittstellen durch das Schlüsselwort «`interface`» gekennzeichnet. Es handelt sich hierbei nicht um einen Stereotypen.

Kurseinheit 4, Kapitel 25.2, S. 285

Hier muss ergänzt werden:
Die Subtypbeziehungen zwischen zwei durch Schnittstellen festgelegter Typen ist äquivalent zu der Subtypbeziehung zwischen Klassen definiert. Außerdem ist die Subtypbeziehung transitiv, d.h. wenn `A` ein Subtyp von `B` ist und `B` ein Subtyp von `C` ist, dann ist `A` auch ein Subtyp von `C`.

Kurseinheit 4, Selbsttestaufgabe 26.2-1

Die Lösung löscht leider einen Buchstaben zu wenig, die Lösung muss somit

```
sb.delete(6, 13).insert(20, "un");
```

lauten.

Kurseinheit 4, Selbsttestaufgabe 26.2-3

In der Lösung fehlt bei beiden `return`-Anweisungen das Semikolon am Ende.

Kurseinheit 5, Selbsttestaufgabe 30.5-2

Die Lösung enthält Fehler bei den Vergleichsoperatoren in den ersten beiden Äquivalenzklassen. Diese müssten lauten:

- `0 < tage < 14 && 0 <= km <= 200`
- `0 < tage < 14 && km > 200`

Kurseinheit 5, Selbsttestaufgabe 30.5-3

Die Lösung enthält Fehler beim Aufruf der Methode `berechne()` Dieser kann natürlich nur an einem Objekt erfolgen. Darum muss die assert-Anweisung folgendermaßen lauten:

```
assertEquals("Berechnung bei x = -2 und y = -3 "
        + "fehlerhaft", 6, new Rechner.berechne(-2, -3);
```

Kurseinheit 5, Selbsttestaufgabe 30.5-4

Die Lösung enthält Fehler beim Aufruf der Methode `berechne()` Dieser kann natürlich nur an einem Objekt erfolgen. Darum müssen die assert-Anweisungen folgendermaßen lauten:

```
assertEquals("Berechnung bei x = 4 und y = 5 "
        + "fehlerhaft", 20, new Rechner.berechne(4, 5);
assertEquals("Berechnung bei x = -2 und y = -3 "
        + "fehlerhaft", 6, new Rechner.berechne(-2, -3);
```

Kurseinheit 5, Selbsttestaufgabe 30.5-5

Die Lösung enthält Fehler beim Aufruf der Methode `berechne()` Dieser kann natürlich nur an einem Objekt erfolgen. Darum müssen die assert-Anweisungen folgendermaßen lauten:

```
assertEquals("Berechnung bei x = 4 und y = 5 "
        + "fehlerhaft", 20, new Rechner.berechne(4, 5);
assertEquals("Berechnung bei x = -2 und y = -3 "
        + "fehlerhaft", 6, new Rechner.berechne(-2, -3);
assertEquals("Berechnung bei x = -3 und y = 7 "
        + "fehlerhaft", 21, new Rechner.berechne(-3, 7);
assertEquals("Berechnung bei x = 5 und y = -2 "
        + "fehlerhaft", 10, new Rechner.berechne(5, -2);
```

Kurseinheit 6, Lösung zur Selbsttestaufgabe 34-12

Hier hat sich in der vorletzten und letzten Zeile ein Fehler eingeschlichen. Statt der 3 müsste dort eine 6 stehen (siehe Abbildung 2 auf Seite 8).

Kurseinheit 6, Lösung zur Selbsttestaufgabe 34-13

Hier hat sich in der letzten Zeile ein Fehler eingeschlichen. Statt der 3 müsste dort eine 6 stehen(siehe Abbildung 3 auf Seite 8).

Kurseinheit 6, Kapitel 36.4, S. 423

Hier hat sich im ersten Absatz des Kapitels ein Tippfehler eingeschlichen. Es muss Ausgangsgrad statt Ausgangsgraf lauten.

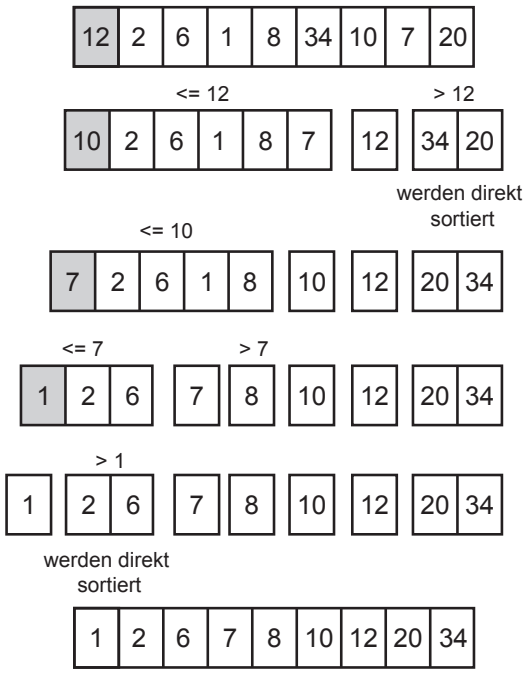


Abbildung 2: Sortieren eines Beispielfeldes mit Quicksort

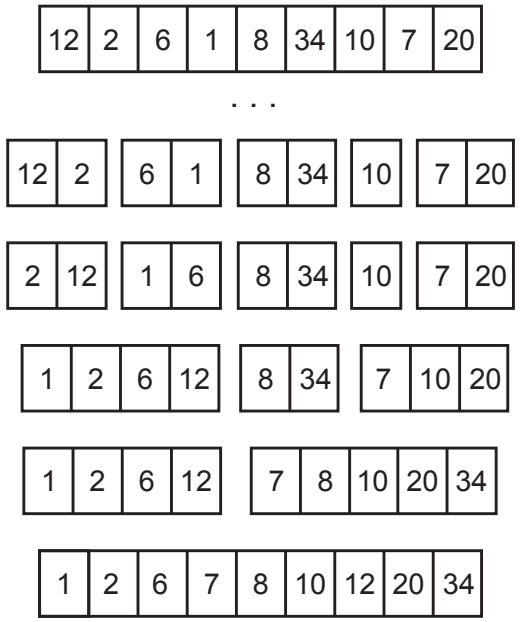


Abbildung 3: Sortieren eines Beispielfeldes mit Sortieren durch Verschmelzen

Kurseinheit 6, Abbildung 36.5-4

Hier sind ab dem vierten Schritt in der Warteschlange F und I vertauscht. Die korrekte Abbildung finden Sie in Abbildung 4 auf Seite 10. Und statt zweimal einem sechten Schritt sollten es natürlich ein fünfter und ein sechter Schritt sein.

Kurseinheit 7, Kapitel 41.2, S. 482

Hier muss in der `for`-Schleife die Ausgabe mit Hilfe des `PrintWriter pw` erfolgen und nicht auf `System.out`:

```
    for (Rechnungsposten rp : posten) {
        pw.println(rp.liefereAnzahl() + " x Nr. "
            + rp.liefereArtikel().liefereArtikelnummer() + " "
            + rp.liefereArtikel().liefereBeschreibung());
}
```

Kurseinheit 7, Kapitel 41.2, S. 483

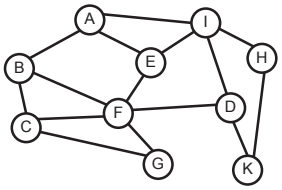
Hier wird im ersten `catch`-Block die Variable `artikeldatei` verwendet, die aber nicht deklariert wurde. Der Fehler kann behoben werden, indem der Anfang der Methode `leseArtikelEin()` folgendermaßen angepasst wird:

```
private void leseArtikelEin() {
    this.artikeldaten = new HashMap<Long, Artikel>();
    String artikeldatei = "artikelliste.txt";
    File f = new File(artikeldatei);
    // ...
```

oder indem im `catch`-Block die Fehlermeldung folgendermaßen angepasst wird:

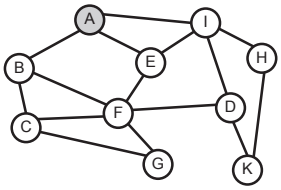
```
System.err.println("Die Datei " + f.getName() +
        " mit den Artikeldaten konnte nicht gefunden werden.");
```

1.



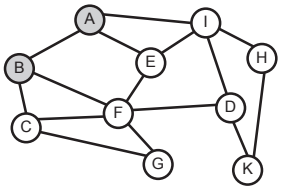
Warteschlange
A

2.



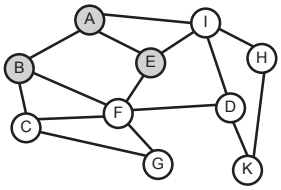
Warteschlange
B, E, I

3.



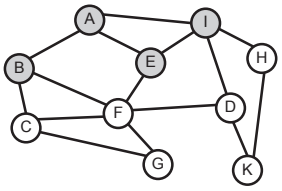
Warteschlange
E, I, C, F

4.



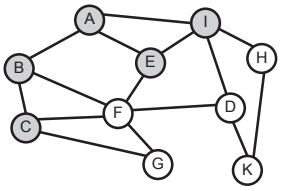
Warteschlange
I, C, F, F, I

5.



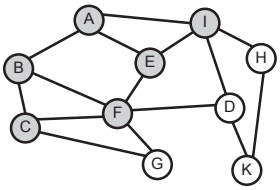
Warteschlange
C, F, F, I, D, H

6.



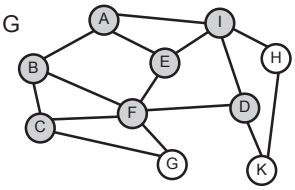
Warteschlange
F, F, I, D, H, F, G

7.



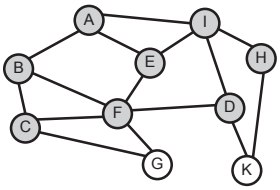
Warteschlange
F, I, D, H, F, G, D, G

8.



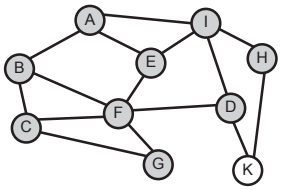
Warteschlange
H, F, G, D, G, K

9.



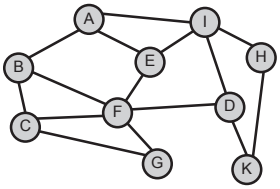
Warteschlange
F, G, D, G, K, K

10.



Warteschlange
D, G, K, K

11.



Warteschlange
K

Knoten K gefunden

Abbildung 4: Breitensuche