Grundlagen der Informatik 1
WS 2008/09

Prof. Mühlhäuser, Dr. Rößling, Melanie Hartmann, Daniel Schreiber
http://proffs.tk.informatik.tu-darmstadt.de/gdi1

Übung 10 Version: 1.0 12.01.2009

---

1 Mini Quiz

Kreuze die korrekten Aussagen an.

Vererbung

- ☐ Eine Klasse kann gleichzeitig von mehreren Klassen erben. zurückgeben.
- ☐ Eine Unterklasse kann immer *alle* geerbten Methoden überschreiben.
- ☐ Vererbung wird in Java durch das Schlüsselwort *inherit* durchgeführt.
- ☐ Aufrufe über **super** werden statisch gebunden.
- ☐ Aufrufe über **this** werden dynamisch gebunden.

Abstrakte Klassen und Interfaces

- ☐ Eine Klasse kann gleichzeitig mehrere Interfaces implementieren.
- ☐ In jeder nicht abstrakten Klasse muss immer mindestens ein Konstruktor implementiert werden.
- ☐ *Interfaces* enthalten nur Methoden und Konstanten, aber keine Attribute.
- ☐ *Abstrakte Klassen* deklarieren nur Attribute und Methoden, aber keine Konstanten.

2 Fragen

1. Ein wichtiges Konzept der Objektorientierung ist die Vererbung. Welche Vorteile ergeben sich dadurch?
2. Erklären Sie mit eigenen Worten den Begriff *Inkrementelles Programmieren*.
3. Worin liegen die Unterschiede zwischen *Interfaces* und *abstrakten Klassen*? Wann macht es Sinn, diese zu nutzen? Welches der beiden Konzepte unterstützt ganz besonders Reusability (Wiederverwendung)?
4. Was versteht man unter dem Begriff des Information Hiding?
5. Vererbungs-Wissenstest: Streichen Sie falsche Antworten, so dass die Aussagen wahr werden.
   - Es gibt eine / keine Basisklasse, von der alle Java-Klassen direkt oder indirekt abgeleitet sind.

- Subklasse ist ein anderer Ausdruck für Unterklasse / Oberklasse.
- Superklasse ist ein anderer Ausdruck für Unterklasse / Oberklasse.
- Eine Klasse kann / kann nicht Superklasse für mehrere Subklassen sein. Eine Klasse kann / kann nicht Subklasse mehrerer Superklassen sein.

3 The Final Countdown

Für die folgende Aufgabe werden Kentnisse über das Schlüsselwort **final** vorausgesetzt. Machen Sie sich bitte damit vertraut und erklären Sie kurz die Auswirkungen von **final** bei der Anwendung auf:

- Attribute und Methodenparameter
- Methoden
- Klassen

Infos zu diesem Schlüsselwort finden Sie hier:

- `http://de.wikipedia.org/wiki/Java-Syntax` (deutsch)
- `http://renaud.waldura.com/doc/java/final-keyword.shtml` (englisch)

Schauen Sie sich nun bitte folgenden Java Code an. Beheben Sie sämtliche eingebauten Fehler, um den Code lauffähig zu machen. Wenden Sie Ihr neu erworbenes Wissen an, um zu erklären, weshalb die Vererbung nicht so funktioniert, wie sie hier ursprünglich gedacht war.
Hinweis: Für die Bearbeitung dieser Aufgabe sollten Sie bitte *keine* Computer-Unterstüzung benutzen. Denken Sie dran: In der Klausur werden Sie ebenfalls keine Hilfsmittel wie Eclipse einsetzen können...

```
1  public final class A {
2    private int value1 = 0, value2 = 0;
3
4    private final int value3 = 0;
5
6    private int getValue1() {
7      return value1;
8    }
9
10   private int getValue2() {
11     return value2;
12   }
13
14   private void setValue1(int newValue1) {
15     value1 = newValue1;
16   }
17
18   private void setValue2(int newValue2) {
19     value2 = newValue2;
20   }
21
22   public final void changeValue3(final int newValue3) {
23     value3 = 0;
24   }
```

```
25 }
26
27 class B extends A {
28   public void changeValue3(final int newValue3) {
29     value3 = newValue3;
30   }
31
32   public static void main(String args[]) {
33     B obj = new B();
34
35     obj.setValue1(30);
36     obj.setValue2(3);
37     obj.value3 = 2008;
38
39     String result = "The final exam for GdI / ICS 1 will be on " +
         getValue1() + "."
40     + getValue2() + "." + obj.value3
41     + ". Please keep this in mind!";
42
43     System.out.println(result);
44   }
45 }
```

Was müssen Sie darin ändern, um die folgende Ausgabe zu erhalten?
"The final exam for GdI / ICS 1 will be on 30.3.2008. Please keep this in mind!"

4 Wer vererbt wem was...?

Vollziehen Sie bitte nach, wie das folgende Programm arbeitet. Geben Sie die jeweilige Ausgabe des Aufrufs *Z.test()* hinter den Kommentaren der Methode test() in der Klasse Z an.
Es ist *nicht* Ziel dieser Aufgabe, dass Sie Eclipse starten, den Code einfügen und das Ergebnis einfach abzulesen!

```
1  class X {
2    int a = 4;
3
4    int get() {
5      return a;
6    }
7  }
8
9  class Y extends X {
10   static int a = 7;
11
12   int get() {
13     return a;
14   }
15
16   static void set(int x) {
17     a = x;
18   }
19
20   static void set(char c) {
21     a = 2 * c;
22   }
23 }
```

```
24
25 class Z extends Y {
26   static int b = 3;
27
28   int get() {
29     return b + a;
30   }
31
32   static int get(X x) {
33     return x.a;
34   }
35
36   static void set(int i) {
37     a = 3 * i;
38    }
39
40   static void set(X x, int i) {
41     a = i;
42   }
43
44   static void test() {
45     Z z = new Z();
46
47     System.out.println(Y.a); // --------
48     System.out.println(get(z)); // --------
49
50     Z.set('c' - 'a' - 1); // ASCII Werte: 'c' = 99, 'a' = 97
51     System.out.println(get(z)); // --------
52     System.out.println(z.get()); // --------
53
54     Y y = z;
55     Y.set(2);
56     System.out.println(z.get() + 1); // --------
57     Z.set(y, 0);
58     System.out.println(y.get() + 4); // --------
59   }
60
61   public static void main(String args[]) {
62     Z.test();
63   }
64 }
```

Hausübung

Die Vorlagen für die Bearbeitung werden im GdI1-Portal bereitgestellt. Kommentieren Sie Ihren selbst erstellten Code. Die Hausübung muss bis zum Abgabedatum im GdI1-Portal abgegeben werden. Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Mit der Abgabe Ihrer Hausübung bestätigen Sie, dass Sie bzw. Ihre Gruppe alleiniger Autor des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial gestattet war, so müssen Sie dessen Quellen deutlich zitieren.

Abgabe: Bis spätestens Fr, 23.01.2009, 16:00.

5 Eine "bessere" Datenbank

In der letzten Übung haben Sie (hoffentlich) die Datenbankaufgabe programmiert, die die elementaren Funktionalitäten wie das Löschen, Einfügen oder das Finden eines Datensatzes ermöglicht.
Da Datenbanken etwas mehr können als nur diese primitiven Methoden, nutzen wir nun *Vererbung*, um die bestehende Funktionalität der Datenbank um neue Funktionalität zu erweitern. Sie können dazu Ihre eigene Datenbank aus der letzten Hausübung nutzen oder die Bibliothek datenbank.jar, die wir im GDI-Portal zur Verfügung stellen. Laden Sie nun die Vorgabendatei vom Portal und importieren Sie diese über `File -> Import... -> Archive File` in Ihr Projekt.

Hinweis: Wenn Sie Ihre eigene Lösung verwenden, implementieren Sie als zusätzliche Funktion **public** Entry getEntry(**int** pos), die den Eintrag an der entsprechenden Position in der Datenbank zurücklie fert. Diese Funktion ist in der zur Verfügung gestellten Bibliothek bereits enthalten.
Hinweis: Wie in der letzten Übung dürfen keine Vectors or Collections verwendet werden.

5.1 Implementieren des Interfaces AdditionalOperations (3 · 1.5 Punkte)

Zunächst soll das Interface `AdditionalOperations` in einer Klasse `ImprovedDatabase` in einem Package *database* implementiert werden. Diese Klasse soll von der Klasse `Database` aus der letzten Hausaufgabe bzw. dem oben erwähnten JAR erben. Achten Sie auf die korrekte Reihenfolge von *extends* und *implements* im Klassenkopf. Es sollen folgende Funktionen, die Sie bereits aus der Mengenlehre kennen, zur Verfügung gestellt werden:

Gegeben zwei Mengen $A = \{a, b, c\}$ und $B = \{b, c, d, e\}$. Daraus ergibt sich

- Vereinigungsmenge: $A \cup B = \{x | x \in A \vee x \in B\} = \{a, b, c, d, e\}$
- Schnittmenge: $A \cap B = \{x | x \in A \wedge x \in B\} = \{b, c\}$
- (Relative) Komplementmenge: $A \diagdown B = \{x | x \in A \wedge x \notin B\} = \{a\}$

Diese Regeln benutzen "echte" Datenbanken ebenfalls, um weitergehende Operationen darauf anzuwenden (Join, Natural Join, Equi Join).
Implementieren Sie nun die Methoden des Interfaces. Beachten Sie dabei auch die Erläuterungen:

- **public int** union(ImprovedDatabase data) verknüpft die aktuelle Datenbank und die als Argument übergebene mittels einer *Vereinigung* der Datensätze. (Faustformel: Kombination der Einträge ohne Duplikate)
- **public int** intersection (ImprovedDatabase data) arbeitet wie *union*, bildet aber die *Schnittmenge*. (Faustformel: nur gemeinsame Einträge verbleiben in der Datenbank)
- **public int** complement(ImprovedDatabase data) bildet die *Komplementmenge* durch Entfernen aller Elemente von *data* aus der aktuellen Datenbank. (Faustformel: gemeinsame Einträge aus der aktuellen Datenbank entfernen)

Alle Operationen sollen auf der *aktuellen Datenbank* arbeiten und diese entsprechend durch das Hinzufügen oder Entfernen von Datensätzen *ändern*. Als Ergebnis ist die Anzahl Datensätze nach dem Ausführen der Operationen zu liefern. Haben wir also eine Datenbank mit den Einträgen $\{a, b, c\}$ und eine mit den Einträgen $\{b, c, d, e\}$ (*data*) , so verbleiben nach Aufruf von *intersection(data)* $\{b, c\}$ in der aktuellen Datenbank und der Aufruf liefert 2 zurück.

5.2 Verbesserter Zugriff: Interface ImprovedAccess (2.5 Punkte)

Die Klasse *ImprovedDatabase* soll nun auch das Interface *ImprovedAccess* implementieren.
Wie in jeder anderen Datenbank auch sollen Datensätze anhand einer Bedingung abfragbar sein. Wir betrachten nur den Fall, dass die Abfrage aus atomaren *Spaltennamen* besteht. Implementieren Sie dazu die folgende Methode aus dem Interface: **public** Entry[] selectXFrom(Column col, String str)
Das Ergebnis ist ein Array von Datensätzen, die in der gewählten *Spalte col* den Text *str* enthalten. Dabei soll es unerheblich sei, an welcher *Position* der Spalte der Text enthalten ist - er muss also nicht am *Anfang* oder *Ende* des Eintrags stehen. Außerdem soll *nicht* zwischen Groß- und Kleinschreibung unterschieden werden.

Hinweis: Sehen Sie in die Dokumentation der Klasse *java.lang.String* und suchen Sie nach geeigneten Methoden. Sie müssen hierbei vorhandene Methoden kombinieren.

Die zu durchsuchende Spalte wird durch den Parameter *col* angegeben. Hierbei handelt es sich um eine *Aufzählung (Enumeration)*. Sie können alle in *Column* angegebenen Werte wie Konstanten benutzen, etwa als *Column.GIVEN_NAME*.
Als Beispiel betrachten wir die folgende Datenbank:

```
1 Hans    Meiser  0170-1567952    {Roßmarkt 26, 60311, Frankfurt} Tänzer
2 Tom     Müller  0163-4574567    {Hainer Weg 12, 60486, Frankfurt}
    Clown
3 Tanja   Knechtel        0190-6666666    {Igelweg 3, 64293, Darmstadt}
    Pilotin
4 Timo    Ulbrich 0179-8888888    {Poststrasse 9, 80210, Mainz}   Klempner
5 Josh    Smithers        0180-2222222    {Wallstreet 7, 99551, Stuttgart}
          Broker
6 Karl    Reichert        0162-1111111    {Hainer Weg 42, 04504, Berlin}
    Arzt
```

Die Abfrage selectXFrom(Column.PHONE_AREA_CODE, "017"); ergibt folgende Elemente als Entry[]:

```
1 Hans    Meiser  0170-1567952    {Roßmarkt 26, 60311, Frankfurt} Tänzer
2 Timo    Ulbrich 0179-8888888    {Poststrasse 9, 80210, Mainz}   Klempner
```

Die Abfrage selectXFrom(Column.ADDRESS_STREET_NAME, "weg"); liefert folgendes Entry[]:

```
1 Tom     Müller  0163-4574567    {Hainer Weg 12, 60486, Frankfurt}
    Clown
2 Tanja   Knechtel        0190-6666666    {Igelweg 3, 64293, Darmstadt}
    Pilotin
3 Karl    Reichert        0162-1111111    {Marbachweg 42, 04504, Berlin}
    Arzt
```

Beachten Sie, dass die Suche hier sowohl Einträge mit dem Abfragewert *weg* als auch mit *Weg* zurückliefert, wie oben beschrieben.

Hinweis: wir empfehlen, eine Hilfsmethode String getContentFor(Column col, **int** index) zu schreiben. Diese Methode soll den zu der gegebenen Spalte *col* gehörenden Wert aus Zeile *index* der Datenbank liefern. In der Beispieldatenbank mit sechs Einträgen oben gäbe getContentFor(Column.GIVEN_NAME, 1) das Ergebnis *Tom*.

5.3 Sortieren der Einträge (3 Punkte)

Implementieren Sie nun die Funktion **public void** sort(Column col). Diese Methode soll die Einträge der aktuellen Datenbank aufsteigend nach der Spalte *col* sortieren.
Lesen Sie dazu in die Java-Dokumentation zur Methode myString.compareTo(a String) des Interfaces *java.lang.Comparable*, das von der Klasse String implementiert wird.
Für die Sortierung dürfen Sie einen beliebigen Algorithmus nutzen. Wir empfehlen (trotz schlechter Sortierleistung) die Nutzung einer angepassten Version des *BubbleSort* aus Übung 8, Aufgabe 4.3, da Ihnen dieser Code schon vorliegen sollte. *Benutzen Sie keine sonstigen Bibliotheken oder Klassen mit vorgefertigten Sortieralgorithmen, wie etwa java.util.Arrays.sort().*
Um Änderungen im *private Entry[] entries* durchführen zu können, verwenden Sie die bereits implementierte Methode **protected void** swap(**int** pos1, pos2).
Für Testzwecke ist die Implementierung von JUnit-Tests ebenfalls sehr sinnvoll, diese wird von Ihnen angesichts des Umfangs der Aufgabe hier aber nicht erwartet.

Hinweis: die String-Sortierung bei den Hausnummern entspricht *nicht* der Ordnung der Zahlen: der String 513 ist "kleiner" als der String 6, da das Zeichen "5" vor dem Zeichen "6" im Zeichensatz steht. Dies muss aber in Ihrer Lösung nicht berücksichtigt werden.