Lösungsvorschlag zur

Nachklausur zu Allgemeine Informatik II (SS 2006)

24. Oktober 2006

Prof. Dr. Franz Schweiggert / Norbert Heidenbluth

Bearbeitungszeit: 120 Minuten

NICHT MIT BLEISTIFT SCHREIBEN!

Name:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Nr	Max	Bewertung	
1	**10**	xxxxx	
(a)	6		xxxxx
(b)	2		xxxxx
(c)	2		xxxxx
2	**10**	xxxxx	
3	**14**	xxxxx	
(a)	4		xxxxx
(b)	6		xxxxx
(c)	4		xxxxx
4	**6**	xxxxx	
5	**8**	xxxxx	
(a)	2		xxxxx
(b)	2		xxxxx
(c)	4		xxxxx
6	**10**	xxxxx	
(a)	3		xxxxx
(b)	3		xxxxx
(c)	4		xxxxx

Nr	Max	Bewertung	
7	**12**	xxxxx	
(a)	2		xxxxx
(b)	2		xxxxx
(c)	2		xxxxx
(d)	2		xxxxx
(e)	2		xxxxx
(f)	2		xxxxx
8	**10**	xxxxx	
9	**10**	xxxxx	
(a)	5		xxxxx
(b)	5		xxxxx
10	**10**	xxxxx	
Summe	**100**		

Generelle Hinweise zur Bearbeitung dieser Klausur:

- Bitte benutzen Sie für die Lösungen den freigelassenen Platz nach der jeweiligen Angabe oder die Rückseite unter Angabe der Aufgabe.
- Sofern Sie bei der Bearbeitung der Aufgaben Annahmen treffen, geben Sie diese bitte mit an!
- Sofern in der Aufgabenstellung nichts anderes vermerkt ist, gilt:
  - Es ist Ihnen freigestellt, ob Sie zur Ein- und Ausgabe die Hilfsklassen aus **IOulm** verwenden oder auf Methoden der Java-Standardbibliothek (java.io) zurückgreifen.
  - Es ist nicht erforderlich, mit Exception-Handling zu arbeiten.

Viel Erfolg!

Aufgabe 1 **(10 Punkte)**

Gegeben sei die folgende Klasse `DummyClass` sowie eine Anwendung `DummyApplication`, welche von dieser Klasse Gebrauch macht:

Klasse `DummyClass:`

```
public class DummyClass {

   private String text;
   private int number;

   public DummyClass() {
      System.out.println("Lege ein neues Dummy-Objekt an...");
   }

   public void setText(String t) { text = t; }

   private String getText() { return text; }

   public void setNumber(Integer i) { number = i; }

   public int getNumber() { return number; }

}
```

Anwendung dieser Klasse:

```
import IOulm.*;

public class DummyApplication {

   public static void main() {
      DummyClass dc = new DummyClass();
      dc.setText("Nachklausur");
      dc.setNumber(100);
      Write.Line("Wir schreiben gerade eine " + dc.getText() +
                 " bei der maximal " + dc.getNumber() +
                 " Punkte zu erreichen sind!");
   }
}
```

(a) 6 Punkte

Würde man die Anwendung nun compilieren (mit dem Compiler aus der Vorlesung – kein Java 5 für „Insider“), bekäme man zwei Fehlermeldungen, weil in der Klasse `DummyClass` zwei Fehler enthalten sind, die ein Compilieren **der gegebenen Anwendung** unmöglich machen. Beschreiben Sie kurz diese Fehler und notieren Sie (im Quellcode auf der vorherigen Seite) die jeweilige Fehlerbehebung.

Lösung:

- Die `getText()`-Methode ist *private* und kann daher in DummyApplication nicht verwendet werden! Man muss die Methode wie folgt ändern:
  `public String getText() { return text; }`
- Der Parameter `Integer` der Methode `setNumber()` ist vom falschen Typ (zumindest in Java < 5!). Man muss den Typ in `int` ändern. Die veränderte Methode sieht also wie folgt aus:
  `public void setNumber(int i) { number = i; }`

(b) 2 Punkte

Angenommen, nach der Bearbeitung von Teilaufgabe (a) liegen nun keine Fehler mehr in der Klasse `DummyClass` vor. Lässt sich die Anwendung (d.h. `DummyApplication.java`) nun compilieren? Lässt sie sich ausführen? Begründen Sie Ihre Antworten!

Lösung:

Die Anwendung lässt sich nicht compilieren, da die `main()`-Methode in `DummyApplication` keinen Parameter hat. Diese müsste wie folgt aussehen: `public static void main()`

Da sich die Anwendung nicht compilieren lässt, kann man sie auch nicht ausführen.

(c) 2 Punkte

Welche Ausgabe liefert die Anwendung, wenn man die Beseitigung aller versteckten Fehler voraussetzt?

Lösung:

Ausgabe:
Lege ein neues Dummy-Object an...
Wir schreiben gerade eine Nachklausur bei der maximal 100 Punkte zu erreichen sind!

Aufgabe 2 **(10 Punkte)**

Ein Programm soll als Argumente beim Aufruf genau die folgenden Parameter (in der angegebenen Reihenfolge) erhalten:

1. Genau einen Kleinbuchstaben aus der Menge {“d”, “h”, “n”, “m”, “s”},
2. eine positive ganze Zahlen sowie
3. einen String, dessen Länge genau 7 Zeichen betragen muss.

Schreiben Sie ein solches Programm gemäß der nachstehenden Spezifikation:

Sofern die bei Programmstart übergebenen Parameter **nicht** den oben genannten Anforderungen entsprechen, bricht das Programm mit einer entsprechenden Fehler- und Usage-Meldung, die auf der **Standardfehlerausgabe** erfolgen soll, ab. Andernfalls gibt es die Argumente einzeln aus und terminiert dann regulär.

Lösung:

```
public class A2 {
   public static void error(String err) {
      System.err.println(err);
      System.err.println("Usage: java A2 <char> <int> <string>");
      System.exit(1);
   }
   public static void main(String[] args) {
      if (args.length != 3)
         error("Drei Kommandozeilenargumente erwartet");
      if (args[0].length() != 1)
         error("Erstes Argument muss ein (!) Zeichen sein");
      switch (args[0].charAt(0)) {
         case ’d’: case ’h’: case ’n’: case ’m’: case ’s’: break;
         default:
            error("Erstes Argument muss aus {’d’,’h’,’n’,’m’,’s’} sein"
      }
      try {
         if (Integer.parseInt(args[1]) <= 0)
            error("Zweites Argument muss positiv sein");
      }
      catch (Exception e) {
         error("Zweites Argument muss eine Integer sein");
      }
      if (args[2].length() != 7)
```

```
            error("Drittes Argument muss genau 7 Zeichen lang sein");

        System.out.println("1. Argument: "+args[0]);
        System.out.println("2. Argument: "+args[1]);
        System.out.println("3. Argument: "+args[2]);
    }
}
```

Aufgabe 3 **(14 Punkte)**

Gegeben sei die Klasse `List` ähnlich der aus der Vorlesung (Skript Seite 236, Programm 10.1):

```
public class List{
   private class Node {
      Object content;
      Node next;
      Node(Object o){ content = o; next = null; }
   }

   private Node first;

   public List(){
      this.first = null;
   }
   public void add(Object el){
      Node tmp = new Node(el);
      tmp.next = first;
      first = tmp;
   }
   public String toString() {
      StringBuffer sb = new StringBuffer();
      sb.append("(");
      Node x = first;
      while ( x != null ) {
         sb.append(x.content.toString());
         x = x.next;
         if ( x != null ) sb.append(", ");
      }
      sb.append(")");
      return sb.toString();
   }
}
```

(a) 4 Punkte

Schreiben Sie eine Instanz-Methode `sizeOf()`, welche die Anzahl der in der Liste enthaltenen Objekte als `int`-Wert zurückliefert (d.h.: **nicht** als textuelle Ausgabe).

Wichtig: Für diese Aufgabe darf **nicht** die private Variable `count` verwendet werden!

Lösung:

```
public int sizeOf() {
   Node head = first;
   int size = 0;
   while (head != null) {
      size++;
      head = head.next;
   }
   return size;
}
```

(b) 6 Punkte

Schreiben Sie eine Instanz-Methode `deleteElement(int i)`, welche – sofern vorhanden – das i.te Element (i wird als Parameter übergeben) aus der linearen Liste entfernt. Die Methode soll `true` zurückliefern, sofern der Löschvorgang erfolgreich war, andernfalls `false`.

Tip: Sie dürfen für diese Teil-Aufgabe die Methode `sizeOf()` aus Teilaufgabe a) verwenden.

Lösung:

```
public boolean deleteElement(int i) {
   if (i <= 0 || i >= sizeOf())
      return false;
   if (i == 0)
       first = first.next;
   else {
      Node head = first;
      for (int j = 0; j < i-1; j++)
         head = head.next;
      head.next = head.next.next;
   }
   return true;
}
```

(c) 4 Punkte

Schreiben Sie einen Konstruktor für die Klasse `List`, welcher als Parameter eine Liste (d.h. ein weiteres Objekt der Klasse `List`) erhält und daraus eine neue Liste erstellt, welche die Objekte in umgekehrter Reihenfolge enthält.

Tip: Sie dürfen auch (zusätzlich zum Konstruktor) eine neue private Methode einführen.

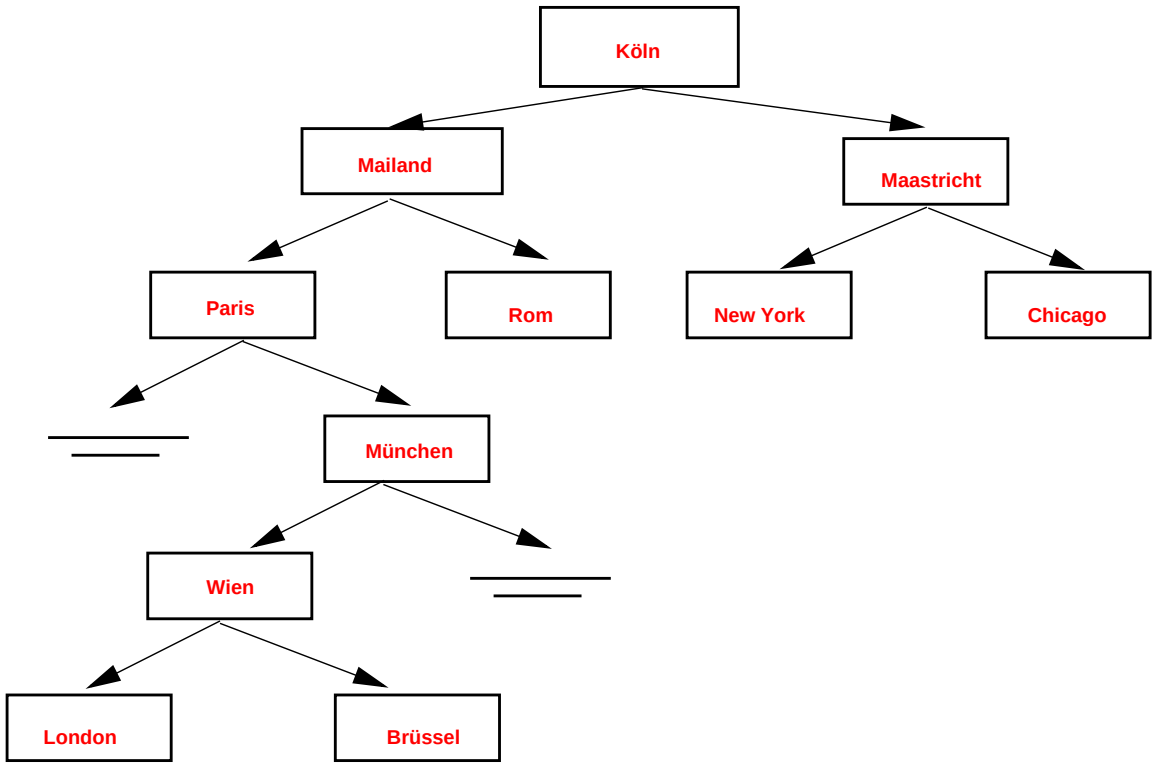
Lösung:

```
public List(List l) {
   this();
   append(l.first);
}
private void append(Node head) {
   if (head != null) {
      append(head.next);
      add(head.content);
   }
}
```

Aufgabe 4 **(6 Punkte)**

Gegeben sei der folgende Baum:

Lösung:



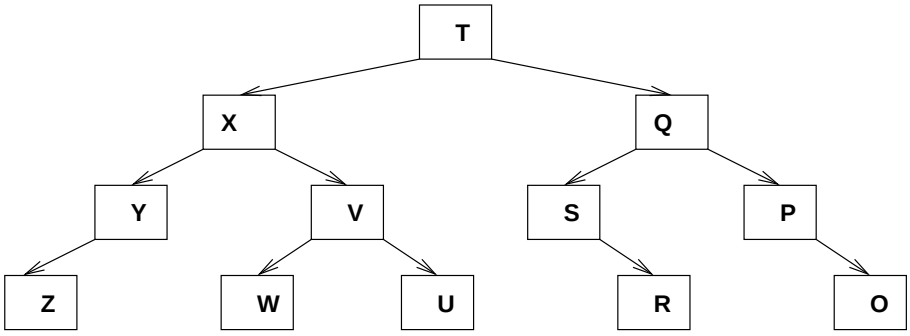
Fügen Sie die nachstehenden Städtenamen so in den gegebenen Baum ein, dass sie, wenn der Baum **inorder** traversiert wird, in der selben Reihenfolge wie in dieser Aufgabenstellung ausgegeben werden:

Paris, London, Wien, Brüssel, München, Mailand, Rom, Köln, New York, Maastricht, Chicago

Hinweis: Der besseren Übersicht wegen sind die `null`-Referenzen der Blätter in dieser Skizze nicht dargestellt.

Aufgabe 5 **(8 Punkte)**

Gegeben ist folgender Binärbaum:



(a) 2 Punkte
Ist dieser Baum nach Höhe ausgeglichen? Begründen Sie auch hier wieder Ihre Antwort.

Lösung:

Der Baum ist nach Höhe ausgeglichen, weil bei jedem Knoten die Höhe der beiden Teilbäume sich höchstens um 1 unterscheidet.

(b) 2 Punkte
Ist dieser Baum vollständig ausgeglichen? Begründen Sie Ihre Antwort kurz.

Lösung:

Der Baum ist vollständig ausgeglichen, weil bei jedem Knoten das Gewicht der beiden Teilbäume sich höchstens um 1 unterscheidet.

(c) 4 Punkte
Traversieren Sie den Baum **preorder** und **postorder** und schreiben Sie jeweils die Ausgabe auf.

Lösung:

preorder: T, X, Y, Z, V, W, U, Q, S, R, P, O

postorder: Z, Y, W, U, V, X, R, S, P, O, Q, T

Aufgabe 6 **(10 Punkte)**

Gegeben sei das folgende Zahlen-Feld:

150	16	12	8	7	180	1	13	9	5	3	170

(a) 3 Punkte

Ein wichtiger Schritt beim Quicksort-Algorithmus ist die Wahl des Elements (Pivot-Element, Bezugselement), mit dem die Partitionierung durchgeführt wird. Ist die Wahl des Elementes **180** in obigem Array als Pivot-Element für den ersten Partitionierungsschritt dafür günstig? Antwort bitte mit Begründung versehen!

Lösung:

In diesem Fall würde bei der ersten Aufteilung eine „Hälfte“ alle Elemente (bis auf 180) enthalten und die andere „Hälfte“ wäre fast leer. Somit müsste beim rekursiven Aufruf wiederum fast das ganze Array sortiert werden.

(b) 3 Punkte

Als Pivotelement könnte man auch den auf eine ganze Zahl gerundeten Mittelwert (Durchschnittswert) der Elemente im Array verwenden. Warum ist dies für die Effizienz des Quicksort in obigem Beispiel eher ungünstig?

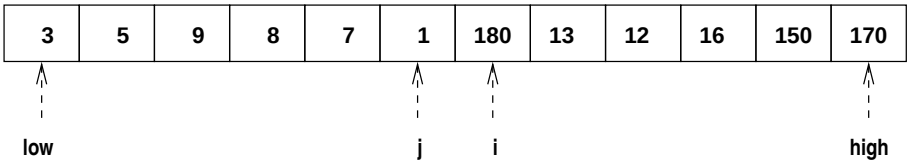
Lösung:

Das Array enthält sowohl sehr kleine als auch sehr große Zahlen. Es gibt aber deutlich mehr kleine Zahlen als große. Der Mittelwert würde genau zwischen den kleinen und den großen Zahlen liegen und somit würde das Array wiederum in zwei deutlich unterschiedlich große „Hälften“ der kleinen und großen Zahlen aufgeteilt.

(c) 4 Punkte

Angenommen, als Pivotelement sei das Element 12 festgelegt. Geben Sie an, wie das Feld nach dem ersten Aufteilungsschritt (Bestimmung der beiden Teilfelder) aussieht! Hierzu sind unten die Zahlen, wie sie im Algorithmus aus der Vorlesung umgestellt werden, wie auch die Teilungsgrenze einzutragen!

Lösung:



Aufgabe 7 **(12 Punkte)**

(a) 2 Punkte

Eine Klasse kann mehrere Konstruktoren besitzen. Worin müssen sich die Konstruktoren voneinander unterscheiden?

Lösung:

... in der Anzahl der Parameter oder den Typen der Parameter (aber Vorsicht bei kompatiblen Parametertypen!).

(b) 2 Punkte

Was unterscheidet in Java den Vergleichsoperator ("==") von der Instanzmethode `equals()`, d. h. worin unterscheiden sich `a == b` und `a.equals(b)`?

Lösung:

Mit `a == b` werden nur die Referenzen verglichen, d.h. es wird überprüft, ob a und b dasselbe Objekt referenzieren. Mit `a.equals(b)` wird hingegen geprüft (kann man selbst implementieren), ob die beiden referenzierten Objekte gleich sind bzgl. ihres Inhaltes. Es muss sich im zweiten Fall nicht um dasselbe Objekt handeln!

(c) 2 Punkte

Welche Java-Schlüsselworte verbinden Sie mit dem Prinzip des *Information Hidings*?

Lösung:

public, *protected* und *private*

(d) 2 Punkte

Was versteht man in Zusammenhang mit Algorithmen unter dem "Divide-And-Conquer"-Prinzip?

Lösung:

Ein Problem wird in kleinere Teilprobleme zerlegt, diese werden rekursiv gelöst und danach werden die Lösungen der Teilprobleme zur Lösung des ursprünglichen Problems zusammen „gebaut".

(e) 2 Punkte

Erläutern Sie kurz die Verwendung der Schlüsselwörter `try`, `catch` und `throws` beim Exception-Handling in Java.

Lösung:

Mit `try` und `catch` kann man Ausnahmen behandeln. Tritt bei einer Anweisung XY evtl. eine Ausnahme vom Typ ABException auf, so kann man diese wie folgt behandeln:

```
try {
   XY
}
catch (ABException e) {
   // Behandle das Auftreten der ABException
   // bei der Ausfuehrung von XY
}
```

Behandelt man eine geprüfte Ausnahme nicht, so muss man sie mit `throws` bei der Methode deklarieren.

(f) 2 Punkte
Welche Bedeutung hat das Java-Schlüsselwort `static`?

Lösung:

Man kan mit `static` Klassen-Methoden/-Variablen definieren. Diese unterscheiden sich von Objekt-..., indem es sie genau ein einziges Mal pro Klasse gibt und sie einfach ber die Klasse referenziert werden können (wohingegen man bei Objekt-... immer eine Objekt-Referenz benötigt).

Aufgabe 8 **(10 Punkte)**

Kreuzen Sie an, ob die nachstehenden Aussagen richtig oder falsch sind. Für jede korrekte Antwort gibt es einen Punkt, für jede falsche Antwort ziehen wir einen Punkt ab. Unbeantwortete Aussagen zählen nicht. Es ist also besser, bei Unsicherheit eine Aussage unbeantwortet zu lassen. Die Abzüge wirken sich nur auf diese Aufgabe aus. Schlimmstenfalls erhalten Sie also nur 0 Punkte.

Frage	Richtig	Falsch
Verschiedene Konstruktoren einer Java-Klasse haben stets denselben Namen.	☒	☐
Der Heap-Sort arbeitet nach dem Divide-And-Conquer-Prinzip.	☐	☒
Jeder Knoten in einem Binärbaum muss immer genau zwei nichtleere Nachfolger besitzen.	☐	☒
Beim Exception Handling in Java muss auf einen try/catch-Block stets ein finally-Block folgen.	☐	☒
Geprüfte Ausnahmen (beim Exception Handling in Java) müssen stets deklariert werden.	☐	☒
Ein Stack arbeitet nach dem LIFO-Prinzip.	☒	☐
Primitive Datentypen werden in Java *by value* übergeben.	☒	☐
Die Methode `equals()` muss eine Klassenmethode sein.	☐	☒
Zwei mithilfe der Methode `compareTo()` verglichene Objekte müssen stets Instanzen derselben Klasse sein.	☐	☒
Wenn ein sortierter binärer Baum inorder traversiert wird, dann werden alle Werte in sortierter Form durchlaufen.	☒	☐
Diese Nachklausur ist viel zu schwer.	☐	☒

Aufgabe 9 **(10 Punkte)**

Gegeben sei die folgende Klasse `Room`:

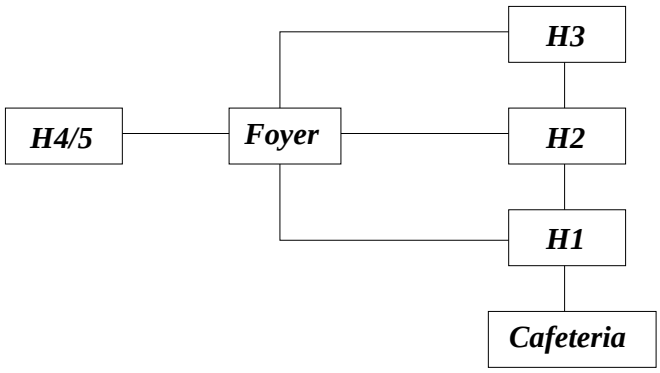
```
public class Room {

   public Room north, east, south, west;
   public String name;

   // Raum-Name wird als Parameter uebergeben, die Referenzen
   // auf Nachbarraeume werden zunaechst alle auf null gesetzt
   public Room(String name) {
      this.name = name;
      this.north = null; this.east = null;
      this.south = null; this.west = null;
   }
}
```

Instanzen dieser Klasse sollen einen Raum mit einer Bezeichnung (Namen) sowie mit maximal vier Ausgängen (jeweils einer im Norden, Osten, Süden und Westen des Raums) repräsentieren. Die Null-Referenz hat im Zusammenhang mit den Feldern `north`, `east`, `south` und `west` die Bedeutung, dass sich in die entsprechende Richtung kein weiterer Raum anschließt.

(a) 5 Punkte

Schreiben Sie den Programmcode auf, der zu dem folgenden Grundriss führt:

(Für eine Anregung, wie dieser Code aussehen könnte: siehe Teilaufgabe b).



Lösung:

```
Room h45 = new Room("H4/5");
Room f = new Room("Foyer");
Room h3 = new Room("H3");
Room h2 = new Room("H2");
Room h1 = new Room("H1");
Room c = new Room("Cafeteria");

h45.east = f;
f.west = h45; f.east = h2; f.north = h3; f.south = h1;
h3.west = f; h3.south = h2;
h2.west = f; h2.north = h3; h2.south = h1;
h1.west = f; h1.north = h2; h1.south = c;
c.north = h1;
```

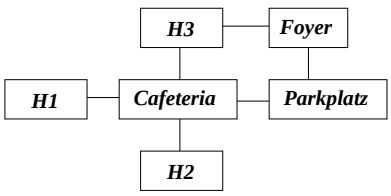
(b) 5 Punkte

Skizzieren Sie, welcher “Grundriss” durch den auf nachfolgenden Programmcode entsteht.

(Für eine Anregung, wie eine solche Skizze aussehen könnte: siehe Teilaufgabe a).

```
// Raeume instanziieren
Room h1 = new Room("H1");
Room h2 = new Room("H2");
Room h3 = new Room("H3");
Room f = new Room("Foyer");
Room c = new Room("Cafeteria");
Room p = new Room("Parkplatz");

// Verbindungen zwischen Raeumen herstellen
c.north = h3; c.south = h2; c.west = h1; c.east = p;
h1.east = c;
h2.north = c;
h3.south = c; h3.east = f;
f.west = h3; f.south = p;
p.west = c; p.north = f;
```

Lösung:



Aufgabe 10 **(10 Punkte)**

Die Umrechnung von Grad Celsius ($^\circ C$) in die amerikanische Skala Grad Fahrenheit ($^\circ F$) erfolgt durch die folgende Formel:

$$T_{\circ F} = T_{\circ C} * \frac{9}{5} + 32$$

Schreiben Sie eine Klasse `DegreeConverter`, so dass das folgende Programm fehlerfrei läuft.

```
import IOulm.*;

public class DegreeConverterEx {

   public static void main (String[] args) {

      DegreeConverter dc = new DegreeConverter(37);
      Write.Line("Ausgabe nach dem Anlegen: " + dc);
      dc.setConvertToFahrenheit(true);
      Write.Line("Ausgabe nach convert (true): " + dc);
      dc.setConvertToFahrenheit(false);
      Write.Line("Ausgabe nach convert (false): " + dc);
   }
}
```

Bei korrekter Implementierung der Klasse `DegreeConverter` produziert das Programm die folgende Ausgabe:

```
theseus$ java DegreeConverterEx
Ausgabe nach dem Anlegen: 37C
Ausgabe nach convert (true): 98F
Ausgabe nach convert (false): 37C
```

Erläuterungen / Anforderungen

- Dem Konstruktor wird eine Integer-Zahl als Parameter übergeben, diese Zahl repräsentiert eine Temperatur in der Einheit Celsius.
- Mit Hilfe der Methode `convertToFahrenheit` lässt sich einstellen, ob die Temperatur in Celesius (`false`) oder Fahrenheit (`true`) ausgegeben wird, wenn die entsprechende Methode (welche?) auf einem Objekt dieser Klasse aufgerufen wird.

- Gestalten Sie Ihre Klasse so, dass der im Konstruktor übergebene Wert auch nach häufigem Umstellen der Ausgabeeinheit nicht aufgrund von Rundungsfehlern beim Umrechnen verändert wurde.

  Tipp: Nehmen Sie dazu nur dann eine Umrechnung vor, wenn Sie auch wirklich benötigt wird.

Lösung

```
public class DegreeConverter {

   private int degree;
   private boolean convertToFahrenheit;

   public DegreeConverter(int deg) {
      this.degree = deg;
      convertToFahrenheit = false;
   }

   public void setConvertToFahrenheit(boolean b) {
      convertToFahrenheit = b;
   }

   public String toString() {
      if (convertToFahrenheit) {
         double f =  (double) degree * 9./5. + 32.0;
         return ( (int) f ) + "F";
      } else {
         return degree + "C";
      }
   }
}
```