Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Prof. Dr. Peter Becker

Klausur — Wintersemester 2012/13
Datenstrukturen und Algorithmen
18. März 2013

Bevor Sie mit der Bearbeitung dieser Klausur beginnen, lesen Sie bitte folgende Hinweise. Diese Hinweise sind bei der Bearbeitung zu beachten.

1. Prüfen Sie die Vollständigkeit Ihres Exemplars. Es sollte

   - dieses Hinweisblatt und
   - acht Aufgaben auf acht Blättern

   umfassen.

2. Tragen Sie auf jedem Lösungsblatt oben an den vorgesehenen Stellen Ihren Namen und Ihre Matrikelnummer ein. Blätter ohne diese Angaben werden nicht bewertet.

   Hinter den Aufgaben ist jeweils hinreichend Platz für die Lösungen freigelassen. Reicht der Platz nicht aus, benutzen Sie die Rückseiten, wobei die Zuordnungen von Lösungen zu Aufgaben deutlich erkennbar sein müssen.

3. Geben Sie dieses Deckblatt zusammen mit den Aufgabenstellungen und den Lösungen sowie alles weitere beschriebene Papier ab.

4. Zugelassene Hilfsmittel: keine

5. Mit ≥ 45 Punkten haben Sie die Klausur bestanden.

6. Ergebnis (bitte nichts eintragen):

1 (12)	2 (13)	3 (10)	4 (15)	5 (13)	6 (10)	7 (12)	8 (10)	$\sum_{Punkte}$ (95)

Viel Erfolg!

Name: **Matrikel:**

Aufgabe 1 (12 Punkte)

Wir wollen in Java farbige geometrische Figuren in der Ebene (2D) mit den folgenden Eigenschaften modellieren:

- Jede geometrische Figur hat eine Position $(x, y) \in \mathbb{R}^2$ und eine Farbe. Die Farbe ist unveränderlich und wird bei der Erzeugung der Figur festgelegt. Zur Ermittlung der Farbe einer Figur soll es eine Methode `farbe()` geben.
- Jede geometrische Figur ist entweder ein farbiger Kreis oder ein farbiges Quadrat.
- Die Position einer geometrischen Figur kann mit Hilfe einer Methode `verschieben()` verändert werden.
- Zur Berechnung der Fläche einer geometrischen Figur soll eine Methode `berechneFlaeche()` zur Verfügung stehen.
- Eine Methode `drucke()` soll Art, Farbe, Position und Fläche einer geometrischen Figur ausgeben.

Erstellen Sie in Java eine Klassenhierarchie für farbige geometrische Figuren mit den oben genannten Eigenschaften. Verwenden Sie in angemessener Weise Vererbung und abstrakte Klassen.

Name: **Matrikel:**

Aufgabe 2 (5+5+3=13 Punkte)

Wir betrachten integrierbare Funktionen $f : \mathbb{R} \to \mathbb{R}$.

(a) Definieren Sie eine Java-Schnittstelle für integrierbare Funktionen. Die Schnittstelle soll es ermöglichen, Funktionswerte $f(x)$ sowie eine Stammfunktion F mit $F' = f$ zu ermitteln.

(b) Implementieren Sie die Schnittstelle aus (a) am Beispiel der Exponentialfunktionen $f(x) = a\, e^{bx}$ mit $a, \in \mathbb{R}, b \neq 0$.

Hinweis: Die Stammfunktion von $f(x)$ ist $F(x) = \frac{a}{b}\, e^{bx}$.

(c) Erstellen Sie eine Java-Methode, die für integrierbare Funktionen f das Integral

$$\int_a^b f(x)\, dx$$

berechnet.

Name: **Matrikel:**

Aufgabe 3 (2+2+2+2+2=10 Punkte)

Es sei der folgende Quelltext gegeben:

```
import java.io.*;
public class ZigeunerSchnitzel extends Schnitzel {
  public static void main(String[] args) throws IOException {
    new ZigeunerSchnitzel().eat();
  }
  void eat() throws IOException { }
}
class Schnitzel {
  // insert code here
}
```

Für welche der folgenden Methodendefinitionen, jeweils eingefügt für die Zeile mit dem Kommentar "`insert code here`", lässt sich der Quelltext kompilieren?

(a) `void eat() { }`

(b) `void eat() throws Exception { }`

(c) `void eat(int x) { }`

(d) `void eat() throws RuntimeException { }`

(e) `void eat() throws FileNotFoundException { }`

Erläutern Sie jeweils in einem Satz, warum eine Kompilierung möglich bzw. nicht möglich ist.

Name: **Matrikel:**

Aufgabe 4 (15 Punkte)

Implementieren Sie eine generische Warteschlange auf Basis einer doppelt verketteten Liste. Sehen Sie dabei die folgenden Methoden vor.

- `queue()`: Hängt ein Objekt an das Ende der Warteschlange.
- `dequeue()`: Löscht das erste Objekt der Warteschlange.
- `top()`: Liefert das erste Objekt der Warteschlange ohne diese zu verändern.
- `size()`: Liefert die Länge der Warteschlange in Zeit $O(1)$.

Name: **Matrikel:**

Aufgabe 5 (4+6+3=13 Punkte)

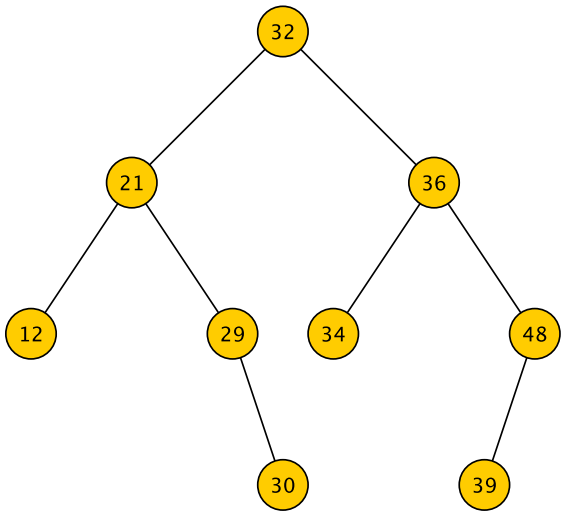
Gegeben sei der folgende Klassenrahmen zur Implementierung generischer Suchbäume:

```
class SearchTree<T extends Comparable<T>> {

    class Node {
        T    value;
        Node left;
        Node right;
    }

    private Node root;
    ...
}
```

(a) Implementieren Sie eine Methode zur Bestimmung der Höhe eines Suchbaums.

(b) Implementieren Sie eine Methode, die ermittelt, ob der Suchbaum einen Schlüssel k enthält oder nicht. Verwenden Sie bei der Implementierung keine Rekursion!

(c) Gegeben sei der folgende Suchbaum:

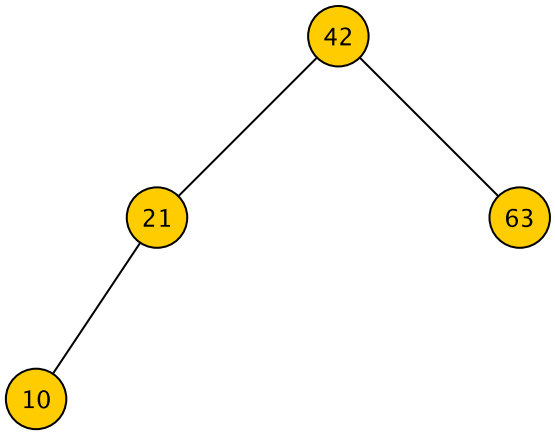


Wir löschen die 48 aus dem Baum von (c). Wie sieht der Baum jetzt aus? Was passiert, wenn wir als nächstes die 32 löschen?

Name: **Matrikel:**

Aufgabe 6 (7+3=10 Punkte)

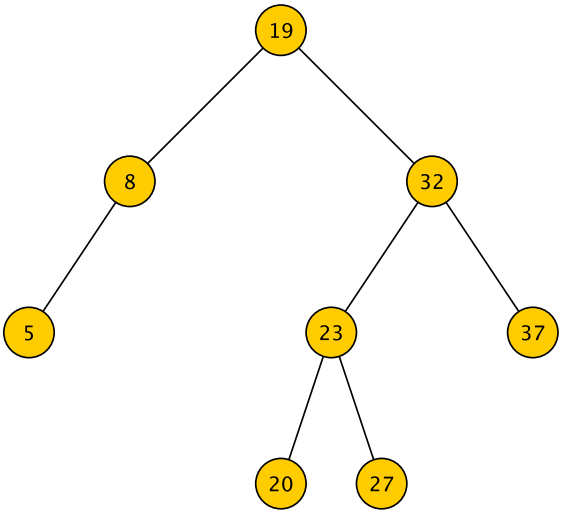
(a) Gegeben sei der folgende AVL-Baum:



Nun werden nacheinander die folgenden Schlüssel eingefügt:
5, 30, 70, 85, 50

Geben Sie für jeden Schlüssel an, wie der AVL-Baum nach dem Einfügen des Schlüssels (inklusive eventuell notwendiger Ausgleichsoperationen) aussieht.

(b) Gegeben sei der folgende AVL-Baum:



Wie sieht der Baum nach dem Löschen der 37 aus?

Name: **Matrikel:**

Aufgabe 7 (4+6+2=12 Punkte)

Gegeben ist die folgende Zahlenfolge:

$$45, 17, 29, 36, 18, 34, 52, 67, 39, 8, 12, 88, 55, 32, 61$$

(a) Geben Sie für die Zahlenfolge einen Heap an.

(b) Führen Sie ausgehend von dem Heap aus (a) drei Iterationen der Sortierphase von Heapsort durch. Geben Sie dabei jeweils an, welche Zahl aus dem Heap entfernt wird und wie der Heap für die nächste Iteration aussieht.

(c) Welchen Zeitaufwand hat Heapsort? Ist Heapsort stabil? (Jeweils ohne Begründung)

Name: **Matrikel:**

Aufgabe 8 (5+5=10 Punkte)

Gegeben Sei eine (zunächst leere) Hash-Tabelle der Größe 7, sowie die Hashfunktion

$$h(x) \quad = \quad x \bmod 7$$

(a) Fügen Sie die folgenden Zahlen in die Hash-Tabelle ein, wobei Kollisionen durch Verkettung behandelt werden:

$$45, 19, 24, 21, 71$$

(b) Implementieren Sie für die folgende Klasse `Foo` auf sinnvolle Weise eine Hash-Funktion.

```
class Foo {
    int a;
    int b;
    String c;
    String d;
}
```