Universität Leipzig
Institut für Informatik

Wintersemester 2025/2026

Einführung in die Objekt-Orientierte Modellierung und Programmierung
–
Praktikum 4

Abgabetermin: 19.12.2025, 23:00 Uhr
Abgabeformat: pdf für das UML, zip-Archiv für die Programme
Maximale Anzahl an Punkten: 62

Rekursion

1. **Beschreibung: Binärer Suchbaum**

   Ein *binärer Suchbaum* (Abbildung 1) besteht aus *Knoten* mit folgenden Eigenschaften

   - genau einer *Wurzel* (Abbildung 1: blauer Knoten):
     keine Elternknoten, höchstens zwei Kindknoten
   - *inneren Knoten* (Abbildung 1: rote Knoten):
     genau ein Elternknoten, mindestens ein und höchstens zwei Kindknoten
   - *Blättern* (Abbildung 1: grüne Knoten):
     genau ein Elternknoten, keine Kindknoten

   Ein *Pfad* ist eine Folge von Knoten, welche durch Kanten verbunden sind. Dabei darf jeder Knoten nur maximal einmal in einem Pfad vorkommen. In Abbildung 1 ist der Pfad von der Wurzel mit dem Wert 23.3 zum Blatt mit dem Wert 21.6 die Folge der Knoten mit den Werten 23.3, 17.7, 19.9 und 21.6, welche durch die cyanen Kanten verbunden sind.

   Die *Länge eines Pfades* ist die Anzahl der Knoten des Pfades. Der Beispielpfad hat also die Länge 4.

   Die *Höhe eines binären Suchbaums* ist sein längster Pfad von der Wurzel zu einem der Blätter: Wurzel, innere Knoten, Blatt.

   - In Abbildung 1 sind die Kanten eines längsten Pfades cyan eingefärbt. Seine Länge und damit die Höhe des binären Suchbaumes beträgt 4.
   - Ist der Baum leer, so ist seine Höhe gleich 0.
   - Besteht der Baum nur aus der Wurzel, so ist seine Höhe gleich 1.

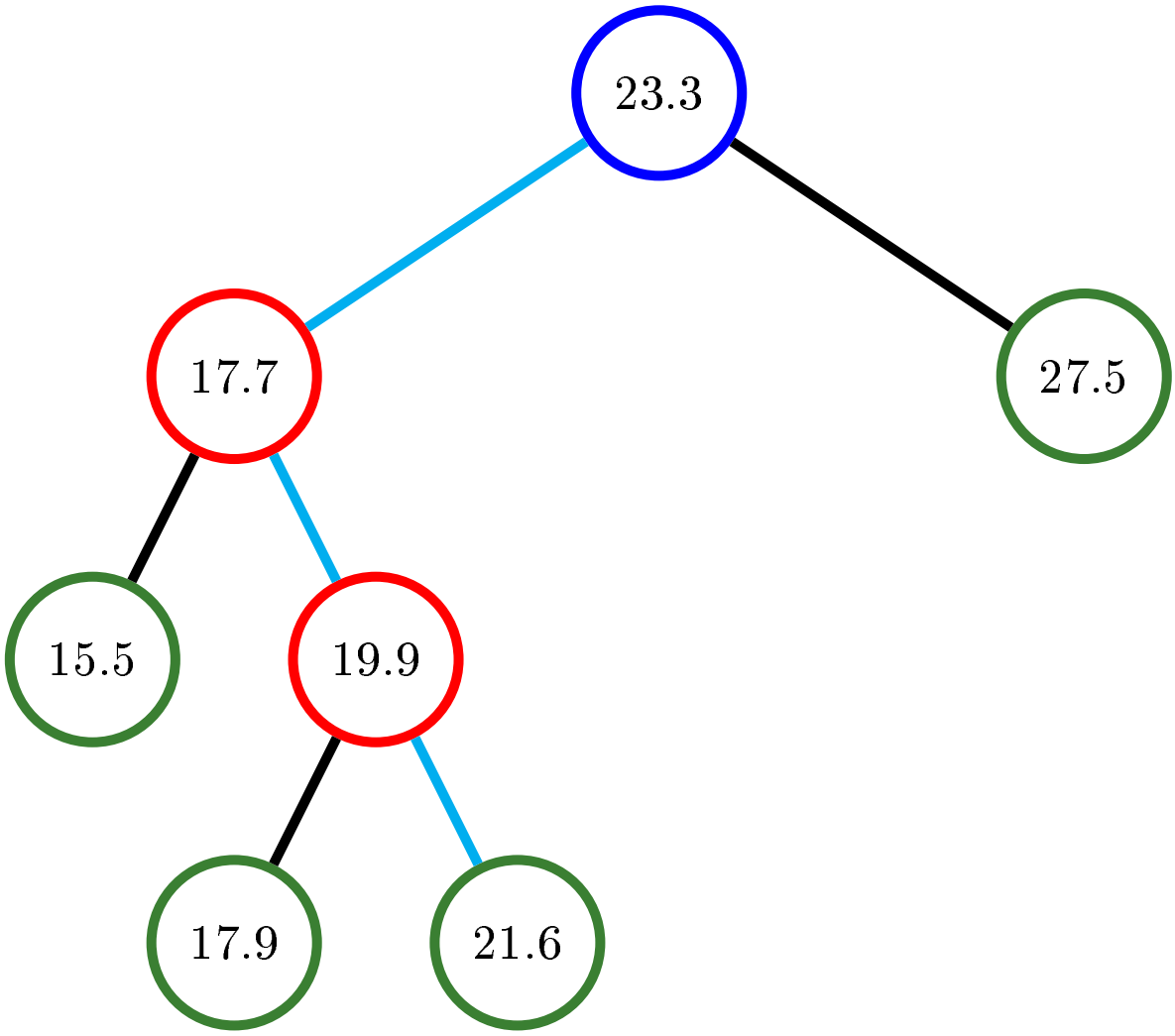


Abbildung 1: Binärer Suchbaum der Höhe 4
blau: Wurzel
rot: Innere Knoten
grün: Blätter
cyan: Kanten des Pfades (23.3, 17.7, 19.9, 21.6)
schwarz: Andere Kanten

Ein binärer Suchbaum wird wie folgt aufgebaut. Zu Beginn ist der Baum leer.

a) Falls die Wurzel noch nicht existiert, erzeuge die Wurzel.

b) Setze den *aktuellen Knoten* auf die Wurzel.

c) Ist die Zahl im aktuellen Knoten nicht gesetzt, so setze sie auf die einzufügende Zahl.

d) Ansonsten: Ist die einzufügende Zahl gleich der Zahl im aktuellen Knoten, so tue nichts (die Zahl ist bereits im Baum enthalten).

e) Ansonsten: Ist die einzufügende Zahl größer als die Zahl im aktuellen Knoten:

   i. Hat der aktuelle Knoten kein rechtes Kind, so lege es an.

   ii. Gehe zum rechten Kind des aktuellen Knotens, indem der aktuelle Knoten auf das rechte Kind gesetzt wird.

   iii. Gehe zu Schritt c)

f) Ansonsten: Ist die einzufügende Zahl kleiner als die Zahl im aktuellen Knoten:

   i. Hat der aktuelle Knoten kein linkes Kind, so lege es an.

   ii. Gehe zum linken Kind des aktuellen Knotens, indem der aktuelle Knoten auf das linke Kind gesetzt wird.

   iii. Gehe zu Schritt c)

Bemerkung: Das Setzen des aktuellen Knotens entspricht dem (rekursiven) Aufruf der Methode der Instanz des jeweiligen Knotens, auf den er gesetzt wird (Wurzel, rechtes Kind oder linkes Kind).

Beispiel: Die Zahlen 23.3, 17.7 und 19.9 werden nacheinander wie folgt eingefügt. Der aktuelle Knoten ist jeweils orange markiert.

a) 23.3

i. Der Baum ist leer.

ii. Die Wurzel existiert noch nicht und wird angelegt. (Schritt a))

iii. Der aktuelle Knoten wird auf die Wurzel gesetzt. (Schritt b))

iv. Die Zahl im aktuellen Knoten ist nicht gesetzt und wird auf 23.3 gesetzt. (Schritt c))

b) 17.7

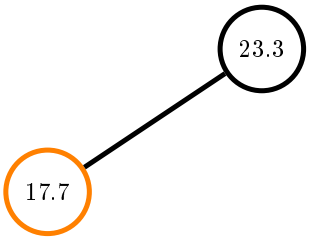
i. Der aktuelle Knoten wird auf die Wurzel gesetzt. (Schritt b))

ii. Die einzufügende Zahl ist kleiner als die Zahl im aktuellen Knoten. (Schritt f))

iii. Der aktuelle Knoten hat noch kein linkes Kind: das linke Kind wird angelegt. (Schritt f) i.)

iv. Der aktuelle Knoten wird auf das linke Kind gesetzt. (Schritt f) ii.)

v. Die Zahl im aktuellen Knoten ist nicht gesetzt und wird auf 17.7 gesetzt. (Schritt c))

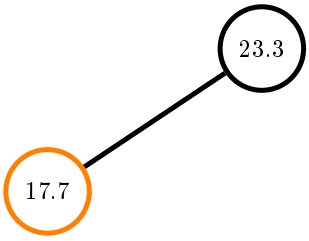


c) 19.9

i. Der aktuelle Knoten wird auf die Wurzel gesetzt. (Schritt b))

ii. Die einzufügende Zahl ist kleiner als die Zahl im aktuellen Knoten. (Schritt f))

iii. Der aktuelle Knoten wird auf das linke Kind gesetzt. (Schritt f) ii.)

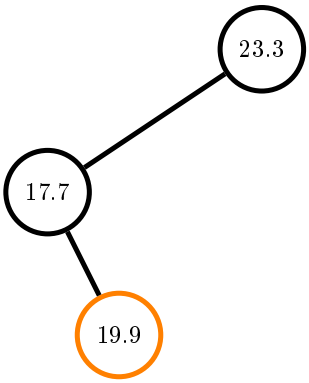


iv. Die einzufügende Zahl ist größer als die Zahl im aktuellen Knoten. (Schritt e))

v. Der aktuelle Knoten hat noch kein rechtes Kind: das rechte Kind wird angelegt. (Schritt e) i.)

vi. Der aktuelle Knoten wird auf das rechte Kind gesetzt. (Schritt e) ii.)

vii. Die Zahl im aktuellen Knoten ist nicht gesetzt und wird auf 19.9 gesetzt. (Schritt c))



2. **Klasse** `BinarySearchTreeNode` ***(37 Punkte)***

   Anforderungen Klasse `BinarySearchTreeNode`:

   - Attribute:
     - Linkes Kind
     - Rechtes Kind
     - In diesem Knoten gespeicherte rationale Zahl
     - Alle Attribute werden direkt mit `null` initialisiert.
   - Methoden:
     - Konstruktor
       - Parameter: keine
       - Funktionalität: keine
     - `add`
       - Parameter: eine rationale Zahl
       - Rückgabe: keine
       - Funktionalität: füge die rationale Zahl in den binären Suchbaum ein (siehe "Beschreibung: Binärer Suchbaum")
     - `getHeight`
       - Parameter: keine
       - Rückgabe: die Höhe des binären Suchbaumes
       - Funktionalität: Berechnung und Rückgabe der Höhe des binären Suchbaumes (siehe "Beschreibung: Binärer Suchbaum")

   a) Modellieren Sie die Klasse `BinarySearchTreeNode` entsprechend den Anforderungen.

   b) Implementieren Sie die Klasse `BinarySearchTreeNode` entsprechend den Anforderungen und dem Modell.

   Allgemeine Anforderungen:

   - Verwenden Sie hierzu ausschließlich eigene sowie die folgenden Klassen:
     - `java.lang.Double`
     - `java.lang.Math`
   - Verwenden Sie ausschließlich die in den Aufgabenstellungen angegebenen Klassen und deren Methoden aus Bibliotheken.
   - Achten Sie auf angemessene Sichtbarkeiten.
   - Achten Sie auf angemessene Benennungen der Parameter und der Attribute.

3. **Klasse** `BinarySearchTree` ***(19 Punkte)***

   Anforderungen Klasse `BinarySearchTree`:

   - Attribute:
     - Wurzelknoten des binären Suchbaumes
     - Wird zu Beginn auf `null` initialisiert.
   - Methoden:
     - Konstruktor
       - Parameter: keine
       - Funktionalität: keine

- `add`
  - Parameter: eine rationale Zahl
  - Rückgabe: keine
  - Funktionalität: füge die rationale Zahl in den binären Suchbaum ein (siehe "Beschreibung: Binärer Suchbaum")
- `getHeight`
  - Parameter: keine
  - Rückgabewert: die Höhe des binären Suchbaumes
  - Funktionalität: Berechnung und Rückgabe der Höhe des binären Suchbaumes (siehe "Beschreibung: Binärer Suchbaum")

a) Modellieren Sie die Klasse `BinarySearchTree` entsprechend den Anforderungen.

b) Kombinieren Sie die Modelle der Klassen `BinarySearchTree` und `BinarySearchTreeNode` und modellieren Sie deren Zusammenhang.

c) Geben Sie das entstandene Modell (UML) ab.

d) Implementieren Sie die Klasse `BinarySearchTree` entsprechend den Anforderungen und dem Modell.

Allgemeine Anforderungen:

- Verwenden Sie hierzu ausschließlich eigene sowie die folgenden Klassen:
  - `java.lang.Double`
- Verwenden Sie ausschließlich die in den Aufgabenstellungen angegebenen Klassen und deren Methoden aus Bibliotheken.
- Achten Sie auf angemessene Sichtbarkeiten.
- Achten Sie auf angemessene Benennungen der Parameter und der Attribute.

4. **Klasse** `TestBinarySearchTree` ***(6 Punkte)***

   Implementieren Sie entsprechend der folgenden Beschreibung die Klasse `TestBinarySearchTree`.

   - Die Klasse enthält ausschließlich die `main`-Methode. Hilfsmethoden der `main`-Methode sind erlaubt.
   - Erzeugen Sie einen binären Suchbaum mit folgender Eingabe:

     $$50.0, 25.0, 75.0, 12.5, 87.5, 37.5, 62.5$$

     Geben Sie die Höhe des binären Suchbaums aus.
   - Erzeugen Sie einen binären Suchbaum mit folgender Eingabe:

     $$12.5, 25.0, 37.5, 50.0, 62.5, 75.0, 87.5$$

     Geben Sie die Höhe des binären Suchbaums aus.

   Allgemeine Anforderungen:

   - Verwenden Sie ausschließlich eigene sowie die folgenden Klassen
     - `java.lang.Double`
     - `java.lang.System`