Heinrich Heine Universität Düsseldorf

Programmierung – WS 2025/26
Dr. Markus Brenneis
Abgabefrist für Feedback:
12.01.26, 10 Uhr

11. Übungsblatt

Aufgabe 1: Klausuraufgabe

Sie haben inzwischen schon sehr viel in der Programmierung gelernt. Eine gute Methode, um selbst für die Klausur zu lernen, ist, sich selbst Klausuraufgaben auszudenken und die eigene (oder auch die von anderen Personen) zu lösen.

In der Klausur stellen wir Aufgaben, die überprüfen, ob Sie die Lernziele der Veranstaltung erreicht haben. Zu einer Klausuraufgabe gehören somit:

- eine ausformulierte, nachvollziehbare Aufgabenstellung
- ein Lösungsvorschlag
- die/das geprüfte(n) Lernziel(e); unsere Lernziele stehen immer hinten in den Foliensätze

Denken Sie sich selbst eine Klausuraufgabe aus (egal ob Programmier- oder Textaufgabe). Im Klausurarchiv des Fachschaftsrats[1] können Sie sich inspirieren lassen.

Falls Sie die Aufgaben mit uns teilen wollen oder Feedback erhalten wollen, können Sie Ihre Aufgabe (ggf. mit `// Feedback gewünscht`) im Abgabesystem hochladen. Um uns die Arbeit zu erleichtern, nennen Sie die Datei mit der Aufgabenstellung `aufgabenstellung.txt`; weitere Dateien (z. B. java-Code oder Bilder) können auch hochgeladen werden.

Aufgabe 2: Binärer Suchbaum nach VL 19

Wir wollen unseren binären Suchbaum um ein paar weitere, nützliche Methoden ergänzen (und dabei noch ein bisschen Rekursion üben). In der Vorgabe finden Sie die Klasse `BinarySearchTree` mit der privaten inneren Klasse `BinaryNode`, wie Sie sie aus der Vorlesung kennen. Die Methode `insert(int)` ist bereits vorgegeben.

Erweitern Sie die Vorgabe um folgende paket-private Instanzmethoden:

- `int maximumRecursive()`
  Sucht das Maximum im gesamten binären Suchbaum (mithilfe einer **rekursiven** Hilfsmethode) und gibt es zurück. Falls der Baum leer ist, soll eine `java.util.NoSuchElementException` geworfen werden.
- `int maximumIterative()`
  Sucht das Maximum im binären Suchbaum **iterativ**. Falls der Baum leer ist, soll eine `java.util.NoSuchElementException` geworfen werden.[2]

[1] https://fscs.hhu.de/klausurarchiv/
[2] Anmerkung: In der Praxis würde man eher den iterativen Algorithmus nehmen. Warum?

- `int height()`:
  Berechnet die Höhe des gesamten binären Suchbaumes. Ein Baum ohne Elemente hat dabei die Höhe 0, ein Baum mit genau einem Element die Höhe 1.
- `int sum()`: Gibt die Summe aller Zahlen im binären Suchbaum zurück. Für einen leeren Suchbaum soll das Ergebnis 0 lauten.
- `String reverseOrder()`:
  Gibt eine String-Repräsentation des Baums zurück, in der alle Elemente absteigend sortiert sind. Der String soll – analog zur Rückgabe von `toString` aus der Vorlesung – folgende Form haben: `12, 8, 2, 0, -1, ` (mit Komma und Leerzeichen am Ende)
- *Knobelaufgabe:* `void deleteIterative(int)`/`void deleteRecursive(int)`:
  Aus dem Baum soll die übergebene Zahl mithilfe eines iterativen bzw. rekursiven Verfahrens gelöscht werden; falls die Zahl nicht im Baum gespeichert ist, soll nichts passieren. Neben der Codevorgabe finden Sie zwei Ansätze zum Löschen, die leider noch nicht ganz funktionieren; benutzen Sie diese Ansätze und beheben Sie die Fehler im Code. *Tipp: Die Kommentare enthalten keine (absichtlichen) Fehler.*

Hinweise:

- Sie dürfen auch weitere, **private** Hilfsmethoden hinzufügen (bei rekursiven Methoden ist dies beispielsweise erforderlich).
- Die Berechnung der Höhe erfolgt am einfachsten rekursiv. In den Übungsstunden üben Sie, wie Sie rekursive Baum-Algorithmen entwickeln können; zu dem Thema gibt es auch einen Selbsttest auf der Kursseite.
- Testen Sie Ihre Methoden in einer main-Methode.