ÜBUNGSSERIE 2

Algorithmen & Datenstrukturen AlgDat / HS 2025

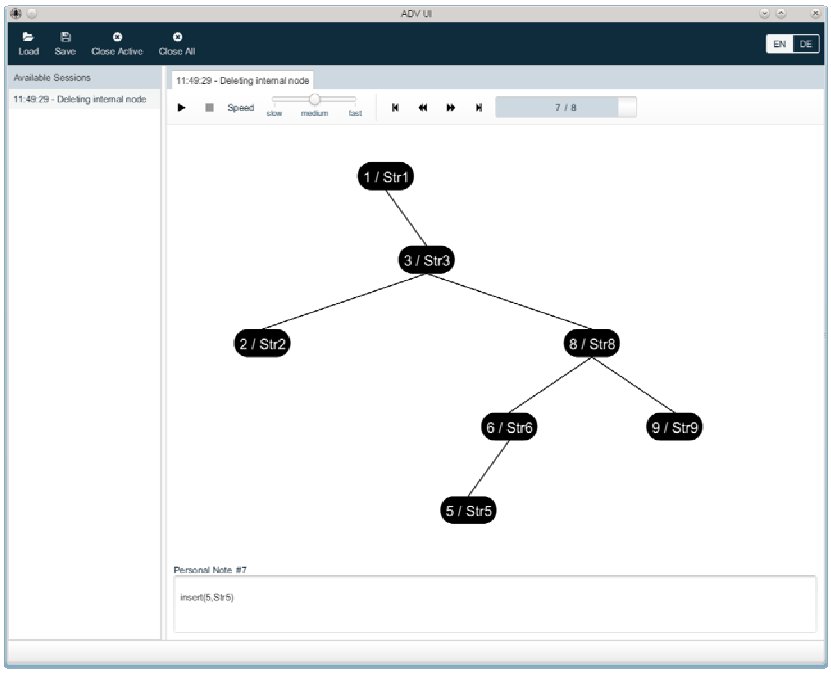
AlgDat Team

Aufgabe 1

Ein zufällig erzeugter binärer Suchbaum mit *n* Knoten hat eine Höhe von ca. ***C*** * *log n*.
Sie sollen die Konstante ***C*** experimentell bestimmen.

Erzeugen Sie dazu grosse binäre Suchbäume mit *n* Elementen und bestimmen Sie deren Höhe. Um die Suchbäume aufzubauen, erzeugen Sie Zufallszahlen zwischen 0 und MAXINT und bestimmen so die Konstante *C* näherungsweise (siehe *BinarySearchTreeTest.java*).
Dazu müssen Sie eine Klasse für binäre Suchbäume mit geeigneten Methoden implementieren.
Implementieren Sie dazu die Klasse *BinarySearchTree .*
Hinweis: Zusätzliche Methoden (interne) können resp. sollen hinzugefügt werden.

Optional kann bei der Entwicklung der Klasse *BinarySearchTree* zur Visualisierung der *"Algorithm & Data Structure Visualizer (ADV)"* eingesetzt werden:



Dazu von *Moodle:Setup Entwicklungsumgebung>Algorithm & Data Structure Visualizer (ADV)* das für die Plattform entsprechende *jar*-File downloaden.
Dann den *ADV-Visualisierungs-Server* in einem Terminal starten, z.B. für Windows:
`java -jar adv-ui_windows_x64_v2.3.0.jar`
Eine allfälliger Hinweis der Firewall kann bestätigt werden (es wird für die Kommunikation mit unserem Test-Programm *PriorityQueueTest* ein Socket geöffnet).

Der *ADV* kann u.a. auch früher gespeicherte *Session's* wieder abspielen.
Die Datei *ex02/BinarySearchTree.adv* ist eine solche Session und kann mit *Load* geladen und dann abgespielt werden.

Konkret ist es genau das Szenario aus *BinarySearchTree.java:main()* (und auch Folie *Löschen (IV/IV)* von der Vorlesung).
Somit ist es schlussendlich das Ziel von dieser Aufgabe, dass unser Programm die selbe Session erzeugt.

Dazu wird in der Klasse *BinarySearchTree* in *main()* dann *ADV* aktiviert, indem anstelle von *BinarySearchTree* neu *BinarySearchTreeADV* instanziert wird:

BinarySearchTree.java:main():

```
...
//BinarySearchTree<Integer, String> bst = new BinarySearchTree<>();
BinarySearchTree<Integer, String> bst =
                        new BinarySearchTreeADV<>("Loeschen (IV/IV)");
...
```

Von nun an werden die Informationen über unseren Baum in *BinarySearchTree.java* jeweils automatisch zum *ADV* gesendet und dort dargestellt.

Dafür sind noch zusätzliche *jar*-Dateien vom *ADV* einzubinden.
Dazu ist in der Entwicklungsumgebung der *Build-Path* um die zwei Dateien aus dem *lib*-Verzeichnis in *Moodle:Setup Entwicklungsumgebung>Algorithm & Data Structure Visualizer (ADV)* zu erweitern: *adv-lib-2.2.jar* und *adv-util-2.0.jar*.
Z.B. in Eclipse: *Project > Properties > Java Build Path > Libraries > Classpath: Add External JARs...*
Hinweis: Sicherstellen, dass der *Classpath* erweitert wird, nicht der *Modulepath*.

Im Weiteren ist sicherzustellen, dass die Java-Version >= 15 ist.