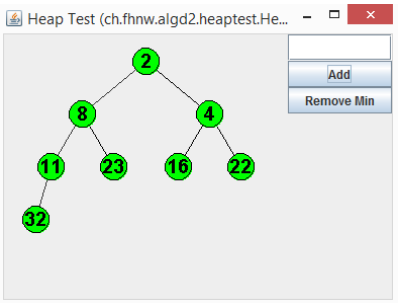
5. Priority Queues - Programmieren

Programmieraufgabe 1 – Heap

Importieren Sie das Programmier-Projekt *ch.fhnw.algd2.heaptest*. Vervollständigen Sie dort die Klasse *Heap*.

Verwenden Sie ein Array zum Speichern der Heap-Elemente. Sie können selbst entscheiden, ob Sie die Wurzel des Heaps bei Index 0 oder 1 platzieren wollen. Ein Konstruktor mit der gewünschten Kapazität der Priority Queue als Parameter ist vorbereitet. Dort soll das Array in passender Länge alloziert werden.

Implementieren Sie die Methoden *add* und *removeMin*. Für die Hilfsmethoden *siftDown* und *siftUp* sind entsprechende Methodenköpfe vorbereitet.



Damit das Testprogramm funktioniert, müssen Sie die Methode *toLongArray* implementieren. Hier sollen die im Heap gespeicherten *Prioritäten* in ein Array kopiert werden. Die Länge dieses Arrays muss der Anzahl der Elemente im Heap entsprechen. An die Stelle mit Index 0 muss die Wurzel des Heaps kopiert werden, anschliessend die restlichen Elemente in der Reihenfolge, wie Sie auch in Ihrem Heap-Array gespeichert sind. Die Werte sollen nicht sortiert werden, sondern in der von Ihnen benutzen Struktur bleiben. Sie werden dann als Baum dargestellt.

Starten Sie zum Testen die Klasse *HeapTest*. *HeapTest* prüft zunächst die grundlegende Funktionsweise der Klasse *Heap* mit ein paar Tests. Schlägt einer davon fehl, wird in der Konsole eine entsprechende Meldung angezeigt. Sind alle Tests bestanden, wird ein Editor geöffnet, mit dem Sie analog zum *TreeEditor* Ihren Heap ausprobieren können.

Geben Sie im Textfeld Zahlen ein um Knoten einzufügen. Sie erhalten eine Baumdarstellung Ihres Heaps. Knoten, deren Vorgänger einen grösseren Wert enthält, werden rot dargestellt, alle anderen grün. Ein korrekter Min-Heap ist vollständig grün. Testen Sie ausgiebig!

Programmieraufgabe 2 – HeapSort

Importieren Sie das Programmier-Projekt *ch.fhnw.algd2.sortdemo*. Vervollständigen Sie dort die Klasse *ch.fhnw.algd.sortalgs.HeapSort*.

Wenn Sie die *Sort Demo* Applikation starten (s. separate Anleitung), können Sie in einem Fenster verschiedene Sortierverfahren animieren. In einer Ebene werden Punkte angezeigt, wobei die X-Koordinate der Position in der Datenfolge und die Y-Koordinate dem Wert entsprechen. Sortierte Daten bilden eine Diagonale von links unten nach rechts oben.

Benützen Sie das Kontextmenu (öffnen mit der rechten Maustaste) um die angezeigten Daten zufällig zu mischen und einen Sortieralgorithmus zu starten.

Insertion-, *Selection-* und *QuickSort* sind bereits implementiert und können ausprobiert werden.

Während des Sortierens können Sie die Animation mit der linken oder der rechten Maustaste anhalten und wieder starten, oder mit dem Kontextmenu einen angehaltenen Sortiervorgang abbrechen.

Hinweise zur Implementierung finden Sie in der vorbereiteten *HeapSort*-Klasse.

Es gibt in diesem Framework keinen direkten Zugriff auf die zu sortierenden Daten. Stattdessen implementiert das übergebene Datenobjekt der Klasse *SortData* folgende Operationen:

> *boolean less(i, j):* Gibt an, ob *x[i] < x[j]*.
>
> *void swap(i, j):* Vertauscht die Werte von *x[i]* und *x[j]*.
>
> *int size():* Gibt die Anzahl der zu sortierenden Elemente an.

Die Operationen *less* und *swap* erwarten Indizes im Interval [0 … size() [.