Testat 3 der Vorlesung Programmierung

In diesem Testat sollen Sie die Klassen **List**, **Tree** und (optional) **Item** anpassen. Die gegebenen Klassen **TreeNode** und **TreeVisitor** sollen Sie nicht ändern. Die Klasse **Main** enthält einige Tests, mit denen Sie Ihren Code in den Klassen **List** und **Tree** testen können.

Aufgabe 1 - Liste (50 Punkte)
Implementieren Sie die folgenden Methoden in der Klasse **List**:

- void addLast(String): Diese Methode soll den übergebenen String ans Ende der Liste anfügen. Wenn der String null oder leer ist, soll nichts passieren.
- String removeLast(): Diese Methode soll den letzten String in der Liste zurückgeben und aus der Liste entfernen. Wird die Methode auf einer leeren Liste aufgerufen, soll null zurückgegeben werden.
- int size(): Diese Methode soll die aktuelle Anzahl der Strings in der Liste zurückgeben.

Hinweis: Die interne Repräsentation der Liste können Sie frei wählen. Dazu müssen Sie die Klasse **List** um ein oder mehrere Instanzvariablen (oder die optional zu implementierende Klasse **Item**) ergänzen.

Aufgabe 2 - Baum (50 Punkte)
Die Klasse **Tree** implementiert einen geordneten Binärbaum, der intern mit der Klasse **TreeNode** umgesetzt wird. Ein **TreeNode** besitzt hierfür einen double-Wert (value), sowie ein linkes Kind (left) und ein rechtes Kind (right). Die Klasse **TreeVisitor** speichert die Sequenz von TreeNodes, die über die visit-Methode an **TreeVisitor** übergeben werden. Implementieren Sie die folgenden Methoden in der Klasse Tree:

- void traverse(TreeVisitor visitor, TreeNode current): Diese Methode soll den Teilbaum mit der Wurzel "current" vollständig ablaufen (traversieren). Während der Traversierung, soll die Methode visit(TreeNode) am übergebenen Visitor in der Hauptreihenfolge (pre-order) aufgerufen werden (d.h. für jeden Teilbaum erst die Wurzel, dann den Teilbaum des linken Kinds und dann den Teilbaum des rechten Kinds). Tipp: Falls Sie die Hauptreihenfolge (pre-order) nicht verstehen, implementieren Sie irgendeine vollständige Traversierung, um zumindest Teilpunkte zu erhalten.
- int count(double value, TreeNode current): Diese Methode soll ausgehend von "current" die Anzahl der TreeNodes zählen, deren Wert dem übergebenen "value" entspricht. Die Implementierung der Methode soll dabei nur die Teilbäume weiterverfolgen, die den Wert auch enthalten können. Beachten Sie dazu die Ordnung, die die insert-Methode in der Klasse Tree definiert. Tipp: Falls Ihnen die Umsetzung einer effizienten Implementierung nicht klar ist, missachten Sie diese Anforderung, um zumindest Teilpunkte zu erhalten.

Testat 3 der Vorlesung Programmierung

In diesem Testat sollen Sie die Klassen **Stack**, **Tree** und (optional) **Item** anpassen. Die gegebenen Klassen **TreeNode** und **TreeVisitor** sollen Sie nicht ändern. Die Klasse Main enthält einige Tests, mit denen Sie Ihren Code in den Klassen **Stack** und **Tree** testen können.

Aufgabe 1 - Stapel (50 Punkte)
Implementieren Sie die folgenden Methoden in der Klasse **Stack**:

- void addToTop(String): Diese Methode soll den übergebenen String als oberstes Element auf den Stapel legen. Wenn der String null oder leer ist, soll nichts passieren.
- String removeFromTop(): Diese Methode soll das oberste Element des Stapels entfernen und zurückgeben. Wird die Methode auf einem leeren Stapel aufgerufen, soll null zurückgegeben werden.
- int size(): Diese Methode soll die aktuelle Anzahl der Strings auf dem Stapel zurückgeben.

Hinweis: Die interne Repräsentation des Stapels können Sie frei wählen. Dazu müssen Sie die Klasse Stapel um ein oder mehrere Instanzvariablen (oder die optional zu implementierende Klasse **Item**) ergänzen.

Aufgabe 2 - Baum (50 Punkte)
Die Klasse **Tree** implementiert einen geordneten Binärbaum, der intern mit der Klasse **TreeNode** umgesetzt wird. Ein **TreeNode** besitzt hierfür einen double-Wert (value), sowie ein linkes Kind (left) und ein rechtes Kind (right). Die Klasse **TreeVisitor** speichert die Sequenz von TreeNodes, die über die visit-Methode an **TreeVisitor** übergeben werden. Implementieren Sie die die folgenden Methoden in der Klasse **Tree**:

- void traverse(TreeVisitor visitor, TreeNode current): Diese Methode soll den Teilbaum mit der Wurzel "current" vollständig ablaufen (traversieren). Während der Traversierung, soll die Methode visit(TreeNode) am übergebenen Visitor in der Nebenreihenfolge (post-order) aufgerufen werden (d.h. für jeden Teilbaum erst den Teilbaum des linken Kinds und dann den Teilbaum des rechten Kinds und schließlich die Wurzel). Tipp: Falls Sie die Nebenreihenfolge (post-order) nicht verstehen, implementieren Sie irgendeine vollständige Traversierung, um zumindest Teilpunkte zu erhalten.
- int count(double value, TreeNode current): Diese Methode soll ausgehend von "current" die Anzahl der TreeNodes zählen, deren Wert dem übergebenen "value" entspricht. Die Implementierung der Methode soll dabei nur die Teilbäume weiterverfolgen, die den Wert auch enthalten können. Beachten Sie dazu die Ordnung, die die insert-Methode in der Klasse Tree definiert. Tipp: Falls Ihnen die Umsetzung einer effizienten Implementierung nicht klar ist, missachten Sie diese Anforderung, um zumindest Teilpunkte zu erhalten.