Praktikum Algorithmen und Datenstrukturen, Teil 2	Hochschule Bochum Bochum University of Applied Sciences
Praktikum Nr. 2: Lineare Datenstrukturen	

H1: Hauptaufgabe (1 Punkt): Implementieren Sie CountingSort effizient, d.h. in $O(n + range)$,. wobei $range$ den Bereich spezifiziert aus dem die Eingabe stammt.
Sie können annehmen, dass die Eingabe m[] aus ints besteht. Ihr CountingSort sollte allerdings in der Lage sein, das Universum aus dem eine konkrete Eingabe kommt, speichereffizient auf das Counts-Array abzubilden: Letzteres Array sollte nur Einträge für Zahlen aus der Bandbreite ($range$) der Zahlen der Eingabe, d.h. für Zahlen von min(m) bis max(m) besitzen.
Testen Sie Ihre Implementation für die Eingabe m1=[1, 17, 3, 1, 4, 9, 4, 4] und m2=[-1, 13, 3, -1, -4, 9, -4, 4].

Z1: 1. Zusatzaufgabe(1/3 Punkt): Implementieren Sie Radixsort effizient, d.h. in $O(d \cdot n)$, wobei d die Basis des verwendeten Zahlsystems angibt. Sie können annehmen, dass nur positive ints zu sortieren sind. Testen Sie mit der Eingabe Eingabe m3=[1, 19004, 20003, 10009,4, 9, 19007, 19, 9009].

Z2: Zusatzaufgabe (1/3 Punkt): Implementieren Sie einen ADT Queue oder Stack (wie in der Vorlesung beschrieben) in Java.

Ihre Implementierung kann entweder Zeiger- oder Array-basiert sein, sollte sich aber nicht auf in Java bereitgestellte Datenstrukturen wie LinkedList oder Vector abstützen.

Ihre Implementierung sollte zudem den weiter unten spezifizierten Queue bzw. Stack-Anforderungen entsprechen.

1. **Test-Anforderungen für Z2:**

Programmieren Sie ein Programm *DataStructureTest* um Ihre Datenstruktur zu testen.

- In diesem Programm sollen innerhalbeiner Schleife die folgenden Aktionen in jedem Schleifendurchlauf durchgeführt werden: 2 Elemente (z.B. Integer, Double und Strings) werden der Datenstruktur hinzugefügt, und dann eines entfernt.
  - Führen Sie den obigen Test aus für folgende (in 4 Durchläufen der obigen Schleife) Beispieleingabe von insgesamt 8 Eingabeelementen: [1, 2, 3, 4, 5, 6, 7, 8]. Beachten Sie: 1. Alle Eingabeelementen sollten genau einmal eingefügt werden. 2. Der Test soll auch für alternative Eingabearrays zumindest für solche bestehend aus eine geraden Anzahl von Zahlen) funktionieren.
  - Geben Sie vor jedem einzelnen Hinzufügen und Entfernen von Elementen diese auf den Bildschirm aus, und geben sie weiterhin aus welches das derzeit zugreifbare Element der Datenstruktur ist. Beachten Sie: Die Bildschirmausgabe, (d.h. z.B. System.out.println()-Aufrufe) soll nicht von den Methoden der Datenstruktur erfolgen, sondern aus Ihrem Testprogramm heraus.
- Geben Sie nach der obigen Prozedur den Zustand der Datenstruktur aus, indem Sie in einer Schleife alle *verbliebenen* Elemente jeweils erst auf den Bildschirm ausgeben und dann entfernen.

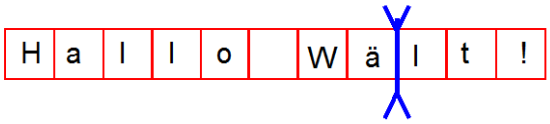
Z3: Zusatzaufgabe (1/3 Punkt):
Benutzen Sie eine selbstgewählte Stack- oder Queue-Implementierung, um einen Cursor in einer Textverarbeitung zu realisieren. Für die Zeile, in der der Cursor sich befindet, soll eine Datenstruktur-Instanz die Zeichen links des Cursors, und eine Datenstruktur-Instanz die Zeichen rechts vom Cursor enthalten. Der Cursor soll folgenden Operationen effizient leisten können, und dabei die beiden Datenstruktur-Instanzen updaten.

- `moveLeft()` : Bewege den Cursor ein Zeiche nach links
- `moveRight()` : Bewege den Cursor ein Zeiche nach links
- `delete()`: Lösche das Zeichen links vom Cursor
- `type(String c)` oder `type(char c)`: schreibt das übergebenene Zeichen (d.h. einen String der Länge 1, oder alternative einen char) an die Stelle links vom Cursor.

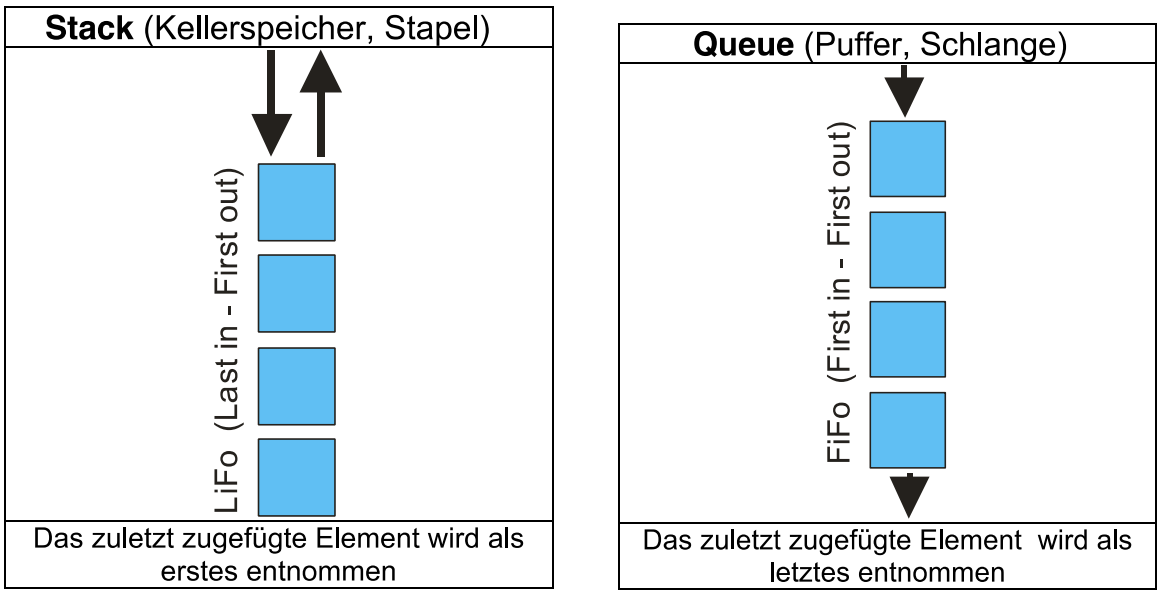
Testen Sie Ihre Implementierung, anhand der unten dargestellten Situation (Cursor in blau), und benutzen Sie ausschließlich die obigen Cursor-Methoden.

0. Erzeugen Sie einen Cursor, ohne Text.
1. Schreiben Sie mit dem Cursor (mit insgesamt 11 Aufrufen) den Text „Hallo Wält!“.
2. Bewegen Sie den Cursor an die im Beispiel dargestellte Stelle.
3. Löschen Sie das „ä“
4. Fügen Sie ein „e“ ein.
5. Platzieren Sie den Cursor dann vor dem „W“ von „Welt“,
6. Geben Sie nun den Inhalt der beiden Datenstrukturen aus.

Hinweis: Sie können in Ihrer Implementierung auf Exception-Handling, etwa für das Herauslaufen aus der Zeile verzichten.



Zu Implementierungen von Stack und Queue



Lineare Datenstrukturen sind essentiell in der Informatik. Die mit elementarsten linearen Datenstrukturen sind Stack und Queue. Diese Datenstrukturen sollten von dynamisches Zufügen und Entfernen von beliebig vielen Elementen zulassen und daher auch von dynamischer Größe sein, das heißt, eine einfache Realisierung durch ein statisches Array ist nicht möglich. Während der Stack nach dem *Last-In-Last-out*-Prinzip operiert, operiert die Queue nach dem *First-In-Last-out*-Prinzip.