Stack/Queue

Implementiere die Klassen `Node`, `LinkedStack` und `Queue` und schreibe entsprechende Testprogramme.

1 Node

Eine Instanz der Klasse `Node` steht für einen Eintrag am Stack. Mindestens zu implementierende Mitglieder:

- `Object elem` - Beliebige Objekte sollen am Stack gespeichert werden können (Typsichere Stacks folgen in der 3. Klasse)
- `Node next` - eine Referenz auf den nächsten Knoten, jeder Knoten hat gespeichert, welcher Knoten nach ihm folgt
- Konstruktor(en)

2 LinkedStack

Eine Instanz der Klasse `LinkedStack` steht für einen Stack mit beliebig(?) vielen Einträgen. Ein Stack verwaltet Daten nach dem LIFO-Prinzip (last in-first out). Mindestens zu implementieren:

- `Node tos` - das TopOfStack-Element
- `int size` - die Anzahl der Elemente am Stack
- `public void push(Object o)` - ein neuer Knoten mit dem Eintrag `o` wird angelegt und kommt oben auf den Stapel(also wird der `tos`-Knoten)
- `public Object pop()` - entfernt den obersten Eintrag vom Stack und liefert diesen zurück; das nächste Element rückt nach und wird zum neuen `tos`-Knoten
- `public boolean isEmpty()`
- `public int size()`
- `public Object element()` - liefert den Inhalt des `tos` retour, ohne etwas zu verändern
- die Methode `public String toString()`

3 Testprogramm Stack

Schreib ein Testprogramm, welche einen Stack anlegt und die Datei `test.txt` öffnet. Danach werden die Zeilen der Datei auf den Stack geworfen und danach wieder (in umgekehrter Reihenfolge) ausgegeben.
Beispiel:

```
A
B
C
```

erzeugt als Ausgabe

```
C
B
A
```

4 LinkedQueue

Eine Instanz der Klasse `LinkedQueue` steht für einen Queue mit beliebig(?) vielen Einträgen. Eine Queue verwaltet Daten nach dem FIFO-Prinzip (first in-first out). Beispielsweise eine Schlange an einer Supermarktkassa. Mindestens zu implementieren:

- `Node top` - der erste Eintrag in der Queue
- `Node tail` - der letzte Eintrag in der Queue
- `int size` - die Anzahl an Einträgen in der Queue
- `public boolean isEmpty()`
- `public int size()`
- `public boolean add(Object o)` - ein neuer Knoten mit dem Eintrag `o` wird angelegt und kommt ans Ende der Queue; er wird also der neue `tail`-Knoten
- `public Object get()` - entfernt den ersten Eintrag der Queue und liefert diesen zurück; das nächste Element rückt nach und wird zum neuen `top`-Knoten
- `public Object element()` - liefert den ersten Eintrag der Queue zurück ohne diesen zu entfernen
- die Methode `public String toString()`

5 Testprogramm Queue

Zum Testen der Queue modellieren wir eine Firma, welche ein Produkt erzeugt und verschiedene Kunden hat. Die Firma bekommt Aufträge von Kunden und verarbeitet diese in der Reihenfolge in der Sie hereinkommen.

1. Schreibe eine Klasse `Order`, die einen `String customer` und einen `int quantity` kapselt, sowie eine `toString`-Methode
2. Schreibe ein Programm, das in einer Queue Kundenaufträge verwaltet. Dabei sollen die Quantitäten der Aufträge zufällig zwischen 20 und 50 liegen.
3. Das Programm soll eine zufällige Anzahl an Kundenaufträgen aufnehmen und danach vor jedem Bearbeitungsschritt die Queue ausgeben; Beispielausgabe:

```
********* QUEUE DUMP *********
   Size: 4
   ** Element 1
     Customer: Steakmauss
     Quantity: 20
     ------------
   ** Element 2
     Customer: Hasenjäger
     Quantity: 30
```

```
      ------------
   ** Element 3
      Customer: Dem Reichel reichts!
      Quantity: 40
      ------------
   ** Element 4
      Customer: Schreibers Bar und Bistro
      Quantity: 50
      ------------
*****************************
Bestellung Steakmauss bearbeitet
********* QUEUE DUMP *********
   Size: 3
   ** Element 1
      Customer: Hasenjäger
      Quantity: 30
      ------------
   ** Element 2
      Customer: Dem Reichel reichts!
      Quantity: 40
      ------------
   ** Element 3
      Customer: Schreibers Bar und Bistro
      Quantity: 50
      ------------
*****************************
.
.
.
```

6 Extra

- Ergänze LinkedQueue durch eine Methode `public Object remove(int idx)`, welche das `n`-te Element aus der Queue entfernt. Teste die Methode, indem eine Queue mit 4 Elementen erzeugt und das 2. aus der Queue entfernt wird.
- Schreibe Tests zu den Klassen `LinkedStack` und `LinkedQueue`