Datenstrukturen: die Queue (bzw. Schlange)

Die Datenstruktur „Queue“ verhält sich wie eine Schlange vor der Kasse bei einem Supermarkt: Am hinteren Ende der Schlange können sich die Kunden anstellen (Elemente werden hinzugefügt) und am vorderen Ende, bei der Kasse, werden die Kunden bedient und entlassen (Elemente werden entnommen).

Eine solche Queue kennt nur vier Operationen:

`boolean isEmpty()` liefert die Information, ob die Queue leer ist
`void enqueue(Object o)` fügt ein Objekt hinten an der Schlange an
`Object front()` liefert eine Referenz auf das erste Objekt
`void dequeue()` entfernt das erste Objekt von der Queue

Mit diesen Befehlen könnte man folgendes Miniprogramm schreiben:

```
Queue q = new Queue();
System.out.println(q.isEmpty()); // Ausgabe: ____
q.enqueue("Angela Ameland");
q.enqueue("Berthold Brecht");
q.enqueue("Claus Cramer");
System.out.println(q.front()); // Ausgabe: ____
q.dequeue();
System.out.println(q.front()); // Ausgabe: ____
q.dequeue();
q.dequeue();
System.out.println(q.front()); // Fehler, denn: ____
```

Für das Informatik-Abitur gibt es auch eine (vom Ministerium) vorgefertigte **`Queue`**-Klasse, die einer **`Stack`**-Klasse sehr ähnlich ist und ebenfalls mit einer inneren **`Node`**-Klasse arbeitet.

Aufgabe 1: Benutze/Teste die Abitur-Stack-Klasse mit obigem Miniprogramm.

Aufgabe 2: Erweitere die Abitur-Queue-Klasse um eine **`toString()`**-Methode, welche den Inhalt des Stacks ausgibt (ohne den Stack zu verändern)

Aufgabe 3: Schreibe eine einfache Supermarkt-Kassen-Simulation, welche mit zwei Threads arbeitet: Ein Thread sorgt dafür, dass Kunden (String-Objekte) sich an die Schlange hinten anstellen, ein weiterer dafür, dass die Kunden vorne aus der Schlange entfernt werden. Sinnvollerweise warten beide Threads jeweils eine zufällige Zeit, um ihre nächste Operation durchzuführen.