HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN
WÜRZBURG-SCHWEINFURT

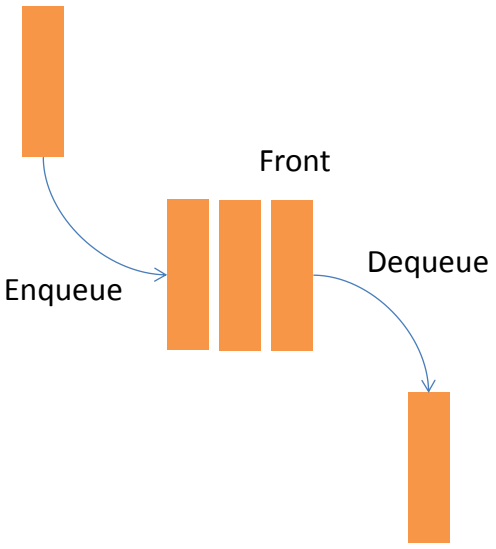
Programmieren II
Sommersemester 2016

Übungsblatt 7 (2 Punkte)

Generische Klassen

Queue

In der ersten Vorlesung haben Sie den abstrakten Datentyp `Queue` kennengelernt. Eine `Queue` ist eine Warteschlange, die das FIFO (First In, First Out) – Prinzip umsetzt, d.h. die Elemente werden in der gleichen Reihenfolge aus der Warteschlange abgerufen, in der sie sich ursprünglich auch eingereiht haben.



Der abstrakte Datentyp `Queue` besitzt vier grundlegende Methoden:

Mit Hilfe der Methode **`enqueue`** werden neue Elemente in die Warteschlange eingereiht.

Die Methode **`front`** erlaubt den Zugriff auf das vorderste Element, ohne die Warteschlange zu verändern.

Mit **`dequeue`** wird das vorderste Element aus der Queue entfernt und zurückgegeben.

Schließlich liefert die Methode **`empty`** einen booleschen Wahrheitswert, ob die Warteschlange gerade leer ist (`true`), oder Elemente warten (`false`).

Mit dem Konzept der generischen Klassen haben Sie nun das Handwerkszeug kennengelernt, eine solche Warteschlange zu implementieren, ohne sich auf einen bestimmten Datentyp für die Elemente festlegen zu müssen.

Eine besondere Herausforderung ist dabei, dass eine solche Warteschlange potenziell unendlich lang werden kann, Sie aber mit einem Array nur eine Speichermöglichkeit mit fester Maximalgröße zur Verfügung haben. Versuchen Sie einen Algorithmus zu finden, der mit der festen Arraygröße auskommt, solange die Länge der Warteschlange nicht diese Grenze überschreitet. Beachten Sie, dass bei alterniertem (d.h. abwechselndem) Aufruf von `enqueue` und `dequeue` auch eine sehr große Anzahl von Elementen die Warteschlange durchläuft, ohne dass die Länge der Warteschlange wächst.

Aufgabe „exercise07“ (Abgabe vom 10.5. - 23.05.2016, bZv-relevant)

Im Verzeichnis Übungsaufgabe_7 finden Sie die Spezifikation des abstrakten Datentyps `Queue` in Form eines gleichnamigen Interfaces vor. Dieses Interface ist bereits generisch angelegt, so dass Ihre Implementierung ebenfalls generisch anzugehen ist. Die in der Klasse `ArrayQueueTest` enthaltene `main`-Methode sollte nach Lösung der Aufgabe ablauffähig sein.

Ihre Aufgabe:

Erstellen Sie eine Klasse `ArrayQueue`, die das Queue-Interface implementiert und intern ein Array für die Datenhaltung verwendet. Die Größe des Arrays soll der Klasse ArrayQueue als Parameter beim Konstruktor-Aufruf mitgegeben werden, z.B.

```
ArrayQueue<String> stringQueue = new ArrayQueue<String>(10);
```

Bitte beachten Sie, dass dieser Konstruktur in jedem Fall öffentlich (`public`) deklariert werden muss, da andernfalls die `main`-Methode in der Klasse `ArrayQueueTest` fehlschlägt.

Überschreitet die Länge der Warteschlange beim Einfügen eines neuen Elements die beim Konstruktoraufruf festgelegte Größe des Arrays, so soll die Methode `enqueue` eine `IllegalAccessException` werfen.

Hinweis:

Leider ist es in Java nicht möglich, in einer generischen Klasse ein Array für Instanzen des Typparameters anzulegen. Allerdings existiert ein Workaround, der mit Hilfe eines Downcasts das Problem löst. Die durch den Downcast verursachte Warnung kann ignoriert werden.

Geht nicht!

```
public class A<T>
{
    private T[] data = new T[10];
}
```

→

Workaround!

```
public class A<T>
{
    private T[] data =
        (T[]) new Object[10];
}
```