Übung 2: Stack & Queue

Abgabetermin: 20.03.2018

Name: **Matrikelnummer:**

Gruppe: ☐ **G1** Di 10:15-11:00 ~~☐ **G2** Di 11:00-11:45~~ ☐ **G3** Di 10:15-11:00

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Punkte
Aufgabe 1	12	☐	Java-Programm Zeichensequenz	Projekt Archiv	☐	
Aufgabe 2	12	☐	Java-Programm Zeichensequenz	Projekt Archiv	☐	

Aufgabe 1: Stack für Objekte (12 Punkte)

Implementieren Sie einen Stack für Objekte, einmal mit einem Array in der Klasse *ArrayStack* und einmal als verkettete Liste in der Klasse *LinkedListStack*. Die Schnittstelle beider Klassen ist durch die abstrakte Klasse *Stack* gegeben (für Methodenbeschreibungen siehe Java-Dokumentation in dem Vorgabeprojekt). Die Methode *peek()* liefert das nächste Element, das *pop()* liefern würde, ohne dieses zu entfernen. Die Methode *reverse()* dreht den gesamten Stack um.

Implementieren Sie die mit **TODO** markierten Funktionen in den Skeleton-Klassen *ArrayStack* und *LinkedListStack* im Paket *at.jku.students.stack*. Des Weiteren implementieren Sie einen *Iterator* für den *ArrayStack* in der Klasse *ArrayStackIterator*. Iteratoren besuchen all Elemente des Stacks, beginnend mit dem das von *peek()* zurückgeliefert werden würde.

Zeigen Sie die Funktion der beiden Klassen mit folgender Zeichenfolge:

R A S ^ T < ^ W S ^ < R A ^ ^ * ^ $ < ^ $

Ein Buchstabe bedeutet *push,* ein Sternchen (*) *pop*, eine Raute (#) *peek,* ein Kleiner Zeichen (<) *reverse* und ein Zirkumflex (^) *pushDown*. Bei einem Dollar Zeichen ($) wird der Inhalt des Stack in Iterationsreihenfolge ausgegeben. Geben Sie die Sequenz der Zeichen an, die die Pop-, Peek-, und $-Operationen zurückgeben.

Abzugeben ist: Projekt Archiv, gesuchte Zeichensequenz

Aufgabe 2: Queue für Objekte (12 Punkte)

Implementieren Sie eine FIFO-Warteschlange (First-in-first-out) für Objekte, einmal mit einem Array in der Klasse *ArrayQueue* (zyklischer Puffer, Array mit fixer Größe) und einmal als verkettete Liste in der Klasse *LinkedListQueue*. Die Schnittstelle beider Klassen ist durch die abstrakte Klasse *Queue* gegeben (für Methodenbeschreibungen siehe Java-Dokumentation in dem Vorgabeprojekt). Die Methode *peek()* liefert das nächste Element, das *dequeue()* liefern würde, ohne es zu entfernen. Die Methode *peekLast()* liefert das zuletzt mit der Methode *enqueue()* hinzugefügte Element.

Implementieren Sie die **TODO** markierten Funktionen in den Klassen *ArrayQueue* und *LinkedListQueue* im Paket *at.jku.students.queue*. Für die *LinkedListQueue* verwenden Sie wie in Aufgabe 1 die Klasse *LinkedList*. Des Weiteren implementieren Sie einen Iterator für die *ArrayQueue* in der Klasse *ArrayQueueIterator*.

Zeigen Sie die Funktion der beiden Klassen mit folgender Zeichenfolge:

R A @ Y # * * Q * U E * # @ * #

Ein Buchstabe bedeutet *enqueue,* ein Sternchen (*) *dequeue,* eine Raute (#) *peek* und ein Klammeraffe (@) *peekLast*. Geben Sie die Sequenz der Zeichen an, die die *dequeue, peek*- und *peekLast*-Operationen zurückgeben.

Abzugeben ist: Projekt Archiv, gesuchte Zeichensequenz

Implementierungshinweise:

- Verwenden Sie das Vorgabeprojekt **PI2_UE02.zip**.
- Fügen Sie Ihre Implementierung in den mit **TODO** markierten Teilen der Klassen *LinkedListStack*, *ArrayStack*, *ArrayStackIterator*, *ArrayQueue*, *ArrayQueueIterator* und *LinkedListQueue* ein.
- Ändern sie **keine public Interfaces** vorgegebener Skeleton Klassen (mit Ausnahme der Sichtbarkeiten).
- Halten Sie sich an die **Codierungsrichtlinien** auf der Kurs Website.
- In der Klasse *ArrayStack* initialisieren Sie das Array mit Länge 2 und verdoppeln Sie die Länge, wenn das Array voll ist.
- In den Klassen *LinkedListStack/LinkedListQueue* verwenden Sie die Klasse *at.jku.ssw.list.impl.LinkedList* mit der Schnittstelle *at.jku.ssw.list.List* aus der Vorgabedatei. Implementieren Sie den Kellerspeicher, indem Sie die Stack-Operationen auf die passenden Listen-Operationen abbilden.