Prof. Dr.-Ing. R. Roosmann, H. Plitzner

Aufgabenblatt 03

Ziel dieses Aufgabenblatts ist es, Sie weiter mit der Anwendung generischer Typen in Java vertraut zu machen.

Abgabe: 04.11.2019; Max. Punktzahl: 10; Min. Punktzahl: 6 Punkte

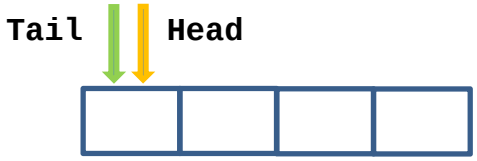
Aufgabe 3.1: Generischer Ringpuffer (6 Punkte)

Ringpuffer werden in unterschiedlichen Bereichen der Informatik eingesetzt. Bspw. für Flugschreiber in Flugzeugen, als Tastaturpuffer oder in Messaging-Systemen. Die Idee ist relativ einfach. Ein Ringpuffer hat eine Kapazität, eine Schreib- sowie eine Leseposition und kennt häufig auch die Anzahl der verwalteten Elemente.

Wird nach mehrfachem Schreiben die Kapazitätsgrenze des Puffers erreicht, wird die Schreibposition – normalerweise, wenn bereits Elemente ausgelesen worden sind – einfach wieder auf 0 gesetzt. Natürlich ist darauf zu achten, dass die Schreibposition die Leseposition nicht überholen darf.

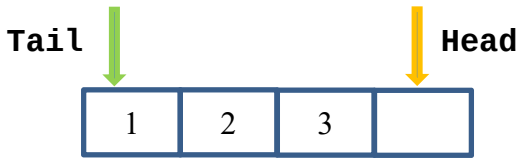
Beispielhafte Darstellung eines Ringpuffers

1. Ringpuffer für Integer-Objekte mit der Kapazität 4 wird initial als FIFO-Container erstellt

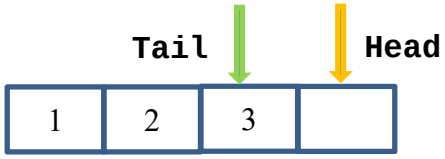


Tail: Index des Elements, das als nächstes ausgelesen werden soll
Head: Index des Elements, in das als nächstes geschrieben werden soll

2. Dem Ringpuffer werden drei Elemente hinzugefügt

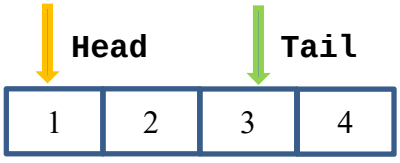


3. Vom Ringpuffer werden zwei Elemente entfernt



„Entfernte“ Elemente sollen physisch in der ArrayList<T> verbleiben. Sie werden nur „logisch“ gelöscht.

4. Dem Ringpuffer wird ein Element hinzugefügt



Vorgaben:

- Umsetzung einer generischen Klasse `Ringpuffer<T>`
- Die Klasse Ringpuffer soll die Interfaces `Deque<T>`, `RandomAccess`, `Serializable` und `Cloneable` implementieren; Informationen zu den Interfaces finden Sie bspw. unter: https://docs.oracle.com/javase/8/docs/api/index.html
- Die Klasse `Ringpuffer<T>` verwendet zur internen Verwaltung der Elemente eine Instanzvariable namens `elements` vom Typ `ArrayList<T>`
- Des Weiteren hält die Klasse folgende private Instanzvariablen:
  - `int head`: definiert die End-Position des Puffers
  - `int tail`: definiert die Anfangs-Position des Puffers
  - `int size`: definiert die Anzahl der tatsächlich verwalteten Elemente
  - `int capacity`: definiert die maximale Anzahl der Elemente des Puffers
  - `boolean fixedCapacity`: legt fest, ob die Kapazität vergrößert werden darf
  - `boolean discarding`: legt fest, ob Elemente überschrieben werden dürfen
- Die Kapazität der `ArrayList<T>` soll einzig über den Ringpuffer verwaltet werden.
- Klienten haben die Möglichkeit die Kapazität initial festzusetzen.
- Klienten haben die Möglichkeit festzulegen, ob der Ringpuffer bei Erreichen der Kapazitätsgrenze a) vorhandene Elemente überschreibt, b) keine neuen Elemente mehr annimmt oder c) die Kapazität automatisch um einen definierten Faktor vergrößert.
- Der Ringpuffer soll als FIFO- und LIFO-Container verwendet werden können.
- Auf sämtliche Elemente soll iterativ lesend zugegriffen werden können.
- Sobald Source-Code in verschiedenen Methoden vorkommt, ist dieser in eine eigene private Hilfsmethode zu verschieben.

Aufgabe 3.2 Generischer Ringpuffer mit Array anstelle einer ArrayList

Führen Sie eine Literaturrecherche durch, ob zur Umsetzung des generischen Ringpuffers anstelle von ArrayList<T> zur Verwaltung der Elemente nicht besser ein generisches Array verwendet werden sollte.

Die Grundlagen zum Verständnis liefert u.a. Angelika Langer in:
http://www.angelikalanger.com/Articles/JavaMagazin/Generics/GenericsPart1.html und
http://www.angelikalanger.com/Articles/JavaMagazin/Generics/GenericsPart2.html

Fassen Sie Ihre Begründung auf maximal einer DIN A4-Seite zusammen (gerne weniger;-).

Aufgabe 3.3 Anwendung und testen des generischen Ringpuffers

a) Simulieren Sie den Flugschreiber in einem Flugzeug unter Verwendung des generischen Ringpuffers aus 3.1.
b) Überlegen Sie sich jeweils ein Beispiel zur Nutzung eines FIFO- und eines LIFO-Containers. Bei beiden Beispielen sollen vorhandene Elemente nicht überschrieben werden. Bei einem Beispiel soll die Kapazität fix und bei einem dynamisch veränderbar sein. Verwenden Sie dazu den generischen Ringpuffer aus Aufgabe 3.1. die Beispiele sollen geeignet sein, die grundsätzliche Verwendbarkeit Ihrer Lösung nachzuweisen.

Viel Erfolg!!