

Artemis 6.4.0

Course Overview

ge64baw

Courses > Praktikum: Grundlagen der Programmierung WS22/23 > Exercises > W12H02 - PUM Server Synchronisierung

W12H02 - PUM Server Synchronisierung

Hausaufgabe

Medium

Points: 6 of 6 Assessment: manual ?

Submission due: 7 months ago

Complaint due: 6 months ago

 100%  (6 months ago)

GRADED

Recent results:



Show all results

Tasks:



PUM Server Synchronisierung

In **Woche 06** hast du den Softwareuinen der PUM bereits geholfen, einen **RingBuffer** und einen **MultiStack** zu implementieren. In der Zwischenzeit haben sie ein wenig weiter an den Klassen gearbeitet, doch jetzt brauchen sie wieder die Hilfe ihres exzellenten Freundes: Deine! Da der Hochleistungsrechner natürlich stark parallelisiert arbeitet, müssen die Datenstrukturen auch für diesen Einsatz geeignet sein.

Update 25.01.: Hinweis zu selbst erstellten Klassen im Zusammenhang mit "NoClassDefFoundError" bei Artemis Tests.

MultiStack



Der **MultiStack** wurde um zwei neue Methoden erweitert. **size** gibt die Anzahl der aktuell im MultiStack enthaltenen Elemente zurück und **search(int)** die 1-indizierte Distanz des übergebenen **ints** zum zuletzt eingefügten Element (vgl. [JavaDoc Stack.search](#)). Deine Aufgabe ist es jetzt, die Methoden des **MultiStack** so zu synchronisieren, dass mehrere lesende Threads gleichzeitig auf den MultiStack zugreifen können, während aber nur ein schreibender Thread diesen zu einem bestimmten Zeitpunkt verändern darf. Während eines schreibenden Zugriffs sind auch keine lesenden erlaubt. Dabei gilt es Folgendes zu beachten:

- Die Signaturen der vorgegebenen Methoden müssen gleich bleiben.
- Die Konstruktorsignatur von **MultiStack** muss gleich bleiben (die von **Stack** darf allerdings verändert werden).
- Vorhandene Attribute müssen gleich bleiben.
- Hilfsmethoden und -Attribute sind erlaubt.
- Die bisherige Funktionalität der Methoden muss natürlich weiterhin bestehen.
- Klassen aus **java.util.concurrent.locks** und **Semaphore** dürfen verwendet werden.
-  **MultiStack Public Tests** [1 of 1 tests passing](#)
Hier siehst du, ob deine Änderungen die ursprüngliche Funktionalität der Methoden beeinflusst.
-  **MultiStack Private Tests** [3 of 4 tests passing](#)
Hier siehst du nach der Deadline das Ergebnis der HiddenTest. Artemis und Threads vertragen sich leider nicht 100%ig gut, daher wird die Aufgabe auch noch manuell überprüft und die Tests dienen nur als grober Anhaltspunkt.

RingBuffer

Der **RingBuffer** soll in einen "BlockingRingBuffer" umgewandelt werden (die Klasse soll aber nicht umbenannt werden!). Das heißt, dass **put(val)** das übergebene **val** immer in den Buffer einfügt. Falls der Buffer voll ist, soll der einfügende Thread warten, bis wieder Kapazität vorhanden ist. Deshalb wurde der Rückgabebetyp auch auf **void** geändert (es wird ja immer eingefügt). Ebenso soll **get** immer ein Element zurückgeben. Falls der Buffer leer ist, wartet der Thread, bis wieder ein Element vorhanden ist. Sollte ein wartender Thread interrupted werden, soll in beiden Methoden eine **InterruptedException** geworfen werden. Während eines **toString**-Aufrufs darf nichts am Buffer geändert werden. Für diese Teilaufgabe gilt Folgendes:


- Die Signaturen der vorgegebenen Methoden und des Konstruktors müssen gleich bleiben. **put** und **get** wurden bereits um ein **throws InterruptedException** erweitert.
- Die Attribute außer **mem** dürfen beliebig geändert werden und neue dürfen hinzugefügt werden. Einzig **mem** darf nicht geändert werden. Hilfsmethoden sind erlaubt.
- Die bisherige Funktionalität der Methoden muss natürlich weiterhin bestehen.

- Klassen aus `java.util.concurrent.locks` und `Semaphore` dürfen verwendet werden.
-  **RingBuffer Public Tests** [1 of 1 tests passing](#)
Hier siehst du, ob deine Änderungen die ursprüngliche Funktionalität der Methoden beeinflusst. (Dies kannst du auch mit den [Test von W06H02](#) selber überprüfen, du musst die Tests teilweise allerdings anpassen.)
-  **RingBuffer Private Tests** [3 of 4 tests passing](#)
Hier siehst du nach der Deadline das Ergebnis der HiddenTest. Artemis und Threads vertragen sich leider nicht 100%ig gut, daher wird die Aufgabe auch noch manuell überprüft und die Tests dienen nur als grober Anhaltspunkt.

Hinweis: Bei der Verwendung selbst erstellter Klassen kann es bei den Tests zu "NoClassDefFoundError" kommen. Dies ist wohl ein Problem von Artemis, für das wir bisher keine verlässliche Lösung haben. Ein Fix, der zumindest manchmal funktioniert ist, entsprechende Attribute nicht bei der Deklaration, sondern erst im Konstruktor zu initialisieren. Falls diese nicht funktioniert, bzw. generelle Informationen dazu gibt es in [dieser Zulip Ankündigung](#).

[Lösungsvorschlag](#)
[Tests](#)

Exercise details	
Release date:	Dec 22, 2022 18:30
Start date:	Dec 22, 2022 18:30
Submission due:	Jan 29, 2023 18:00
Assessment due:	Feb 6, 2023 18:00
Complaint due:	Feb 13, 2023 18:00

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left. 

How useful is this feedback to you?

