Bereich: Threads (2)

Synchronisation

Package: `de.dhbwka.java.exercise.threads.buffer`	**Klasse:** `MyBuffer`

Aufgabenstellung:

Erweitern Sie das `MyBuffer`-Beispiel aus den Vorlesungsfolien (Abschnitt „Synchronisation mit wait und notify“) so, dass der interne Puffer (Speicher) mehrere Elemente enthalten kann.
Für die Anzahl der Elemente im Puffer soll es eine Obergrenze geben (symbolische Konstante `MAXSIZE`, z.B. `MAXSIZE=3`).

Erweitern Sie die Klassen `ProducerThread` und `ConsumerThread` entsprechend, um die Funktion zu demonstrieren.

Hinweise:

- Verwenden Sie für die interne Speicherung der Elemente eine Datenstruktur `LinkedList<Integer>`.
- Um beim Ablauf etwas Zufall (und damit Dynamik) ins Spiel zu bringen, lassen Sie die beiden Threads nach jedem `get()` bzw. `put()` jeweils eine zufällige Anzahl von Millisekunden warten (z.B. zwischen 0 und 1000).
- Geben Sie zum Testen nach jedem `get()` bzw. `put()` den Füllstand des Puffers (Anzahl der Elemente im Puffer) auf die Konsole aus.

Beispielausgabe (Einfügen/Auslesen von 7 Elementen):

```
Put: 0
Fill level after put: 1
Get: 0
Fill level after get: 0
Puffer leer - warten!
Put: 1
Fill level after put: 1
Get: 1
Fill level after get: 0
Put: 2
Fill level after put: 1
Put: 3
Fill level after put: 2
Get: 2
Fill level after get: 1
Put: 4
Fill level after put: 2
Put: 5
Fill level after put: 3
Puffer voll - warten!
Get: 3
Fill level after get: 2
Put: 6
Fill level after put: 3
...
```

Bereich: Threads (2)

Suchmaschine

Package: `de.dhbwka.java.exercise.threads.search`	**Klasse:** `SearchEngine`

Aufgabenstellung:

Internet-Suchmaschinen laden Internet-(HTML-)-Dokumente herunter und extrahieren alle Wörter und Links daraus. Die Wörter gehen in den Index der Suchmaschine ein, die Links werden verfolgt und die dahinter stehenden Seiten werden wiederum indiziert.

Da das Herunterladen von Dokumenten in der Regel einige Zeit dauert, verwenden diese Suchmaschinen Threads, um jeweils eine Seite herunterzuladen und diese dann einem Controller-Programm zur Verarbeitung zu übergeben. Die Verarbeitung des Inhalts (z.B. konvertieren Suchmaschinen PDF- oder Office-Dokumente zunächst in ein internes Format) könnte dann ebenfalls wieder über Threads gesteuert werden.

Schreiben Sie eine Klasse `PageLoader`, welche das Interface `Runnable` implementiert, und der im Konstruktor eine URL übergeben werden kann. `PageLoader` soll den Inhalt dieser URL herunterladen, muss also als Thread gestartet werden!
Eine Methode `pageLoaded()` soll `true` zurückgeben, sobald der Download abgeschlossen ist (ansonsten `false`), eine weitere Methode `getPageContent()` soll dann den Inhalt des Downloads als `String` zurückgeben.

Schreiben Sie weiterhin eine Klasse `SearchEngine`, die eine Liste von URLs, die zeilenweise in einer Datei abgelegt sind oder in einem Array gespeichert sind, einliest und anschließend alle URLs herunterlädt.

Jede URL soll dabei von einem eigenen `PageLoader`-Thread heruntergeladen werden. `SearchEngine` soll die Threads regelmäßig darauf abfragen, ob der Download abgeschlossen ist und sich dann ggf. den Inhalt abholen (und die ersten 40 (oder 60) Zeichen ausgeben).

Erweiterung:
Sorgen Sie dafür, dass `SearchEngine` gleichzeitig nicht mehr als n (z.B. n=3) `PageLoader`-Threads startet.

Hinweise:
Ggf. müssen Sie einen Proxy einstellen um Zugriff zum Web zu haben:

```
System.setProperty( "proxySet", "true" );
System.setProperty( "proxyHost", "myproxy.domain.de" );
System.setProperty( "proxyPort", "8000" );
```

Beispiel zum Lesen des HTML-Inhalts von einer URL. Sie können auch diese Klasse nutzen.

```
package de.dhbwka.java.exercise.threads.search;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;

/**
 * Part of lectures on 'Programming in Java'. Baden-Wuerttemberg
 * Cooperative State University.
 *
 * (C) 2016-2018 by W. Geiger, T. Schlachter, C. Schmitt, W. Suess
 *
 * @author DHBW lecturer
 * @version 1.1
 */
public class ReadURLExample {

   public static String getStringContentFromUrl( String url, String encoding ) {
      StringBuilder buffer = new StringBuilder();
      String line = null;
      try ( BufferedReader br = new BufferedReader(
            new InputStreamReader( new URL( url ).openStream(), encoding ) ) ) {
         while ( (line = br.readLine() ) != null ) {
            buffer.append( line ).append( System.lineSeparator() );
         }
      } catch ( IOException ex ) {    }
      return buffer.toString();
   }

   public static void main( String[] args ) {
      System.out.println( ReadURLExample.getStringContentFromUrl(
            "https://www.tagesschau.de", "UTF-8" ) );
   }

}
```

Beispiel für Testausgabe
(4 URLs; Maximalzahl gleichzeitiger `PageLoader`*-Threads n=3; Ausgabe jeweils der ersten 40 Zeichen der HTML-Seiten, wobei Zeilenumbrüche durch „##“ ersetzt werden):*

```
Gestartet: https://www.tagesschau.de
Gestartet: https://www.sueddeutsche.de
Gestartet: https://www.spiegel.de
Geladen: https://www.tagesschau.de
  Inhalt: <!DOCTYPE html>##<html lang="de">##<head
Gestartet: https://www.kit.edu
Geladen: https://www.sueddeutsche.de
  Inhalt: <!DOCTYPE html>##<html lang="de">##
Geladen: https://www.kit.edu
  Inhalt: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
Geladen: https://www.spiegel.de
  Inhalt: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
```