JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

Queues mit WatchDog

Softwareentwicklung
4CHIT 2016/17

Martin Wölfer

Version 0.2
Begonnen am 21.11.2016
Beendet am 27.11.2016

Note:
Betreuer: RAFM

Inhaltsverzeichnis

1 Einführung

Schreibe ein Programm, welches ein simples Erzeuger-Verbraucher-Muster implementiert!

1.1 Grundanforderungen

- Klasse Watchdog
- Zwei eigene Klassen (Consumer und Producer) erben von Thread
- Zumindest 2 Erzeuger und zwei Verbraucher
- Die zwei Klassen sind über einen Queue verbunden
- Die Erzeuger ermitteln eine Zufallszahl (0 bis 254). Jede gefundene Zahl wird über die Queue an die Verbraucher geschickt
- Die Verbraucher geben Ihre erhaltene Zahl auf der Konsole aus
- Erzeuger und Verbraucher werden mittels Watchdog ordnungsgemäß beendet
- Erzeuger: Ausgabe der erstellten Zahl
- Verbraucher: Ausgabe der erhaltenen Zahl
- Die Queue hat eine Maximalgröße von 20 Elementen
- Kommentare und Sphinx-Dokumentation
- Kurzes Protokoll über deine Vorgangsweise, Aufwand, Resultate, Beobachtungen, Schwierigkeiten, ... Bitte sauberes Dokument erstellen! (Kopf- und Fußzeile etc.)

1.2 Erweiterungen

- Erzeuger: Anzahl der vorhandenen Queue-Elemente nach dem Hinzufügen
- Verbraucher: Anzahl der vorhandenen Queue-Elemente nach dem Entfernen
- Erzeuger: Ausgabe auf der Konsole, falls die Queue voll ist
- Verbraucher: Ausgabe auf der Konsole, falls die Queue leer ist
- GUI: Darstellung des aktuellen Status der Queue (grafische Darstellung des Füllungsgrades)

2 Vorgangsweise Grundanforderung

Um die Grundanforderungen zu lösen habe ich zuerst den `WatchDog` und `Stoppable` so implementiert, wie es in dem bereitgestellten `.pdf` stand.

Nachdem diese Klassen erstellt wurden habe ich den `Producer` und den `Consumer` erstellt. Dieser erben von `threading.Thread` und `Stoppable` und es wird die abstrakte Methode `stopping()` implementiert.

In der Methode `run()` vom Producer werden lediglich Zufallszahlen von 0 bis 254 erstellt und danach in die queue geschickt, welche sich alle `Producer` und `Consumer` teilen.

In der Methode `run()` vom Consumer werden Zahlen aus der `Queue` entnommen und simpel ausgegeben.

Damit die `Queue-Size` nur maximal 20 groß ist, wird bei der Initialiserung der queue das Attribut `maxsize` auf 20 gesetzt:

```
1 queue.maxsize = 20
```

Die Sphinx Dokumentation zu erstellen hat sich als sehr leicht dargestellt, ich habe einfach 2 `.rst` Files erstellt, eines erhält alle Informationen zu dem `WatchDog` und `Stoppable`, während das andere den tatsächlich selbst geschrieben Code enthält, also `Producer` und `Consumer`.

3 Vorgangsweise Erweiterungen

Die Anzahl der Inhalte einer `Queue` erhält man durch `queue.qsize()`.

Um auszugeben wann sie voll ist muss man sie nur mit ihrer ihrer maxsize vergleichen

```
1 if self.queue.qsize() == self.queue.maxsize:
```

Um auszugeben wann sie leer ist muss man nur vergleichen ob qsize 0 ist

```
1 if self.queue.qsize() == 0:
```

Das schwierigste an den Erweiterungen war die GUI, was mich zu dem Thema Schwierigkeiten bringt

4 Schwierigkeiten

Die GUI umzusetzen hat leider nicht funktioniert aus mehreren Gründen, aber zuerst beschreibe ich meinen Ansatz.

Meine Idee war noch eine Klasse zu schreiben welche auch von `threading.Thread` und `Stoppable` erbt und dieser Thread kümmert sich mit `tkinter` nur um die GUI. Es musste irgendwie möglich sein außerhalb dieser Methode, mit einer statischen Methode, von diesem GUI irgendwelche Quadrate oder was auch immer zu setzen damit man eine Art Status anzeigen kann.

Es hat alles recht gut funktioniert ich bin bald auf mein erstes großes Problem gestoßen:

Man kann tkinter nicht in der init-funktion initialisieren.

Wenn ich tkinter.Tk() in der `__init__()` funktion aufgerufen habe hat es nie funktioniert, und dewegen musste ich mein `canvas` und die `root` Variable in meiner `run()` Funktion haben. Dies

hat aber dazu geführt dass `Canvas` nicht zugreifbar war für die methode welche den Status setzt.

Dies hat zu noch viel mehreren Problemen geführt was mich zum Entschuss gebracht hat dass es viel zu Schwer für mich ist ein tkinter-GUI in einem seperaten Thread zu erstellen.

4.1 Github-Link

Hier Klicken für Git Reposiory