JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

A07 Synchronisation

Softwareentwicklung
4CHIT 2016/17

Filip Scopulovic

Version 1.0
Note:
Begonnen am 20. November 2016
Betreuer: W. Rafeiner-Magor
Beendet am 20. November 2016

Inhaltsverzeichnis

1 Aufgabe

1.1 Vorgangsweise

Das, von dem Herr Professor Rafeiner-Magor, zur Verfügung gestellte `pdf`-File[1] über `Events`, `Bedingungsvariablen` und `Queues` war Anhaltspunkt für die Aufgabe. Durch dieses `pdf`-File konnte ich mich sehr leicht an diese Aufgabe gewöhnen. Mit dem `pdf`-File habe ich sofort begonnen meine Ideen zu Implementieren.

1.2 Aufwand

Der Aufwand war gering. Nachdem ich mich über `Queues` informiert habe, war die Aufgabe leicht. Jedoch hatte ich Schwierigkeiten bei dem vollführen der erweiterten Aufgaben. Die Angabe hat mich verwirrt.

1.3 Resultate

Mein Resultat war ein `txt`-File im Ordner `files`. Dieses `txt`-File ist gefühlt mit den ganzen Primzahlen, die gefunden wurden. Bei jedem neuen Aufruf wird das `txt`-File überschrieben. Zudem wird in der Konsole außerdem die Primzahlen ausgegeben. So stehen die Primzahlen im `txt`-File.

```
0
1
2
3
5
7
11
13
17
...
111751
111767
111773
```

Listing 1: Primzahlen im txt-File

So werden die Primzahlen in der Konsole ausgegeben.

```
Found prime number is 0
Found prime number is 1
Found prime number is 2
Found prime number is 3
Found prime number is 5
Found prime number is 7
Found prime number is 11
Found prime number is 13
Found prime number is 17
...
Found prime number is 117259
Found prime number is 117269
Found prime number is 117281
```

Listing 2: Ausgabe in der Konsole

[1] SEW_4_Events_Bedingungsvariablen_Queues_Python.pdf

1.4 Beobachtungen

Ich fand es interessant wie es mit der `Queue` funktionieren konnte und kein `lock` oder Sonstiges benötitgt wurde.

1.5 Schwierigkeiten

Schwierigkeiten haben mir die Erweiterungen der Aufgabe bereitet. Hauptsächlich weil ich sie erst nach mehrfachen lesen verstanden habe. Sie waren recht unklar formuliert.

1.6 Code

```
"""
@author: Filip Scopulovic
@date: 11-11-2016
@use: Consumer prints the prime number out that the producer finds
"""
import threading

class Consumer(threading.Thread):
    """
    class Consumer that inheritance from threading.Thread
    Prints prime numbers, that it takes from the Producer class

    :inheritance threading.Thread:
    """

    def __init__(self, queue, prime_file):
        """
        Constructor of the Consumer class.

        :param Queue queue: takes the queue as a parameter
        :param object prime_file: a file where the Consumer writes the prime numbers in
        """
        threading.Thread.__init__(self)
        self.queue = queue
        self.prime_file = prime_file

    def run(self):
        """
        Endless loop that waits for the producer to get the prime numbers from the queue.
        Prints out the prime number in the console and write it in a .txt-file.

        :return None:
        """
        while True:
            number = self.queue.get()
            print("Found prime number is %s" % (str(number)))
            self.prime_file.write(str(number) + "\n")
            self.queue.task_done()
```

Listing 3: Verbraucher


```python
"""
@author: Filip Scopulovic
@date: 16-11-2016
@use: Producer searches for prime numbers and gives them to the consumer
"""
import threading, queue, math

class Producer(threading.Thread):
    """
    class Producer that inheritance from threading.Thread
    Takes prime numbers and gives it to the Consumer class

    :inheritance threading.Thread:
    """

    def __init__(self, queue):
        """
        Constructor of the Producer class

        :param queue: takes the queue as a parameter
        """
        threading.Thread.__init__(self)
        self.queue = queue

    def run(self):
        """
        Endless loop that calls the method self.is_prime_number().
        This mehtod returns a boolean and if it is True then it puts the number in the queue,
        else it just increments the number var.

        :return None:
        """
        number = 0
        while True:
            if self.is_prime_number(number):
                self.queue.put(number)
                self.queue.join()
            number += 1


    def is_prime_number(self, number):
        """
        Looks if the given parameter number is a prime number and returns a Boolean.
        Idea from the method came from the website http://stackoverflow.com/questions/18833759/python-prime-number-checker

        :param int number: Takes a number
        :return bool is_prime: Returns a Boolean to see if the given number is a prime number
        """
        if number % 2 == 0 and number > 2:
            return False
        for i in range(3, int(math.sqrt(number)) + 1, 2):
            if number % i == 0:
                return False
        return True
```

Listing 4: Erzeuger

```
"""
@author: Filip Scopulovic
@date: 11-11-2016
@use: run_script that initializes producer and consumer and starts them
"""
import queue, consumer, producer

class Run_script:
    """
    Class I made for starting the Consumer and producer
    I just made the class so my docstrings will be recognized by Sphinx
    """

    def __init__(self):
        """
        Initializes the queue, the file, the Consumer and the Producer and calls the start_threads()
            method
        """
        self.queue = queue.Queue()

        self.file = open("files/prime_numbers.txt", 'w')

        self.t_producer = producer.Producer(self.queue)
        self.t_consumer = consumer.Consumer(self.queue, self.file)

        self.start_threads()


    def start_threads(self):
        """
        Calls the methods start() and join() from Producer and Consumer

        :return None:
        """
        self.t_producer.start()
        self.t_consumer.start()

        self.t_producer.join()
        self.t_consumer.join()


rs = Run_script()
```

Listing 5: Start

Listings

