

---

# Protokoll

## A07 Synchronisation

---

Softwareentwicklung  
4CHIT 2016/17

Filip Scopulovic

Note:  
Betreuer: W. Rafeiner-Magor

Version 1.0  
Begonnen am 20. November 2016  
Beendet am 20. November 2016

# Inhaltsverzeichnis

<b>1</b>	<b>Aufgabe</b>	<b>1</b>
1.1	Vorgangsweise . . . . .	1
1.2	Aufwand . . . . .	1
1.3	Resultate . . . . .	1
1.4	Beobachtungen . . . . .	2
1.5	Schwierigkeiten . . . . .	2
1.6	Code . . . . .	2

# 1 Aufgabe

## 1.1 Vorgangsweise

Das, von dem Herr Professor Rafeiner-Magor, zur Verfügung gestellte pdf-File<sup>1</sup> 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 gefüllt 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.

```
1 0
  1
3 2
  3
5 5
  7
7 11
  13
9 17
  ...
11 111751
   111767
13 111773
```

Listing 1: Primzahlen im txt-File

So werden die Primzahlen in der Konsole ausgegeben.

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

Listing 2: Ausgabe in der Konsole

---

<sup>1</sup>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 benötigt 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

```

1  """
   @author: Filip Scopulovic
3  @date: 11-11-2016
   @use: Consumer prints the prime number out that the producer finds
   """
5  import threading
7
9  class Consumer(threading.Thread):
10     """
11     class Consumer that inheritance from threading.Thread
12     Prints prime numbers, that it takes from the Producer class
13
14     :inheritance threading.Thread:
15     """
16
17     def __init__(self, queue, prime_file):
18         """
19         Constructor of the Consumer class.
20
21         :param Queue queue: takes the queue as a parameter
22         :param object prime_file: a file where the Consumer writes the prime numbers in
23         """
24         threading.Thread.__init__(self)
25         self.queue = queue
26         self.prime_file = prime_file
27
28     def run(self):
29         """
30         Endless loop that waits for the producer to get the prime numbers from the queue.
31         Prints out the prime number in the console and write it in a .txt-file.
32
33         :return None:
34         """
35         while True:
36             number = self.queue.get()
37             print("Found prime number is %s" % (str(number)))
38             self.prime_file.write(str(number) + "\n")
39             self.queue.task_done()

```

Listing 3: Verbraucher

```

1  """
2  @author: Filip Scopulovic
   @date: 16-11-2016
4  @use: Producer searches for prime numbers and gives them to the consumer
   """
6  import threading, queue, math
8
9  class Producer(threading.Thread):
10     """
11     class Producer that inheritance from threading.Thread

```

```

12     Takes prime numbers and gives it to the Consumer class
13
14     :inheritance threading.Thread:
15     """
16
17     def __init__(self, queue):
18         """
19         Constructor of the Producer class
20
21         :param queue: takes the queue as a parameter
22         """
23         threading.Thread.__init__(self)
24         self.queue = queue
25
26     def run(self):
27         """
28         Endless loop that calls the method self.is_prime_number().
29         This method returns a boolean and if it is True then it puts the number in the queue,
30         else it just increments the number var.
31
32         :return None:
33         """
34         number = 0
35         while True:
36             if self.is_prime_number(number):
37                 self.queue.put(number)
38                 self.queue.join()
39                 number += 1
40
41     def is_prime_number(self, number):
42         """
43         Looks if the given parameter number is a prime number and returns a Boolean.
44         Idea from the method came from the website http://stackoverflow.com/questions/18833759/python-prime-number-checker
45
46         :param int number: Takes a number
47         :return bool is_prime: Returns a Boolean to see if the given number is a prime number
48         """
49         if number % 2 == 0 and number > 2:
50             return False
51         for i in range(3, int(math.sqrt(number)) + 1, 2):
52             if number % i == 0:
53                 return False
54         return True

```

Listing 4: Erzeuger

```

1     """
2     @author: Filip Scopulovic
3     @date: 11-11-2016
4     @use: run_script that initializes producer and consumer and starts them
5     """
6     import queue, consumer, producer
7
8     class Run_script:
9         """
10        Class I made for starting the Consumer and producer
11        I just made the class so my docstrings will be recognized by Sphinx
12        """
13
14        def __init__(self):
15            """
16            Initializes the queue, the file, the Consumer and the Producer and calls the start_threads()
17            method
18            """
19            self.queue = queue.Queue()
20
21            self.file = open("files/prime_numbers.txt", 'w')
22
23            self.t_producer = producer.Producer(self.queue)
24            self.t_consumer = consumer.Consumer(self.queue, self.file)

```

```
24         self.start_threads()
26
28     def start_threads(self):
29         """
30         Calls the methods start() and join() from Producer and Consumer
31
32         :return None:
33         """
34         self.t_producer.start()
35         self.t_consumer.start()
36
37         self.t_producer.join()
38         self.t_consumer.join()
39
40 rs = Run_script()
```

Listing 5: Start

## Listings

1	Primzahlen im txt-File . . . . .	1
2	Ausgabe in der Konsole . . . . .	1
3	Verbraucher . . . . .	2
4	Erzeuger . . . . .	2
5	Start . . . . .	3