Betriebssysteme ITS3 – Entwurf zum Aufgabenblatt 2
Johannes Oehlers und Kristoffer Schaaf

1.) Der Aufbau und Ablauf der einzelnen Threads:

Main Thread:

1. initialisiere Queue
2. erstelle alle Producer Threads
3. erstelle alle Consumer Threads
4. erstelle Observer Thread
5. warte auf Erreichen der 1000 produzierten Pakete
6. terminiere alle Producer Threads
7. warte auf Erreichen der 1000 konsumierten Pakete
8. terminiere alle Consumer Threads
9. warte auf Terminierung des Observer Threads
10. beende Programm

Queue:

- wird als doppelt verkettete Liste implementiert
- diese besteht aus structs, welche einen Vorgaenger, einen Nachfolger und die PaketID enthalten
- das Startelement hat die PaketID -1 und das letzte 100
- dazwischen werden 25 leere structs miteinander verknüpft
- die Queue hat eine produce- und eine consume Funktion, in der die Pakete eingefügt, bzw. entfernt werden

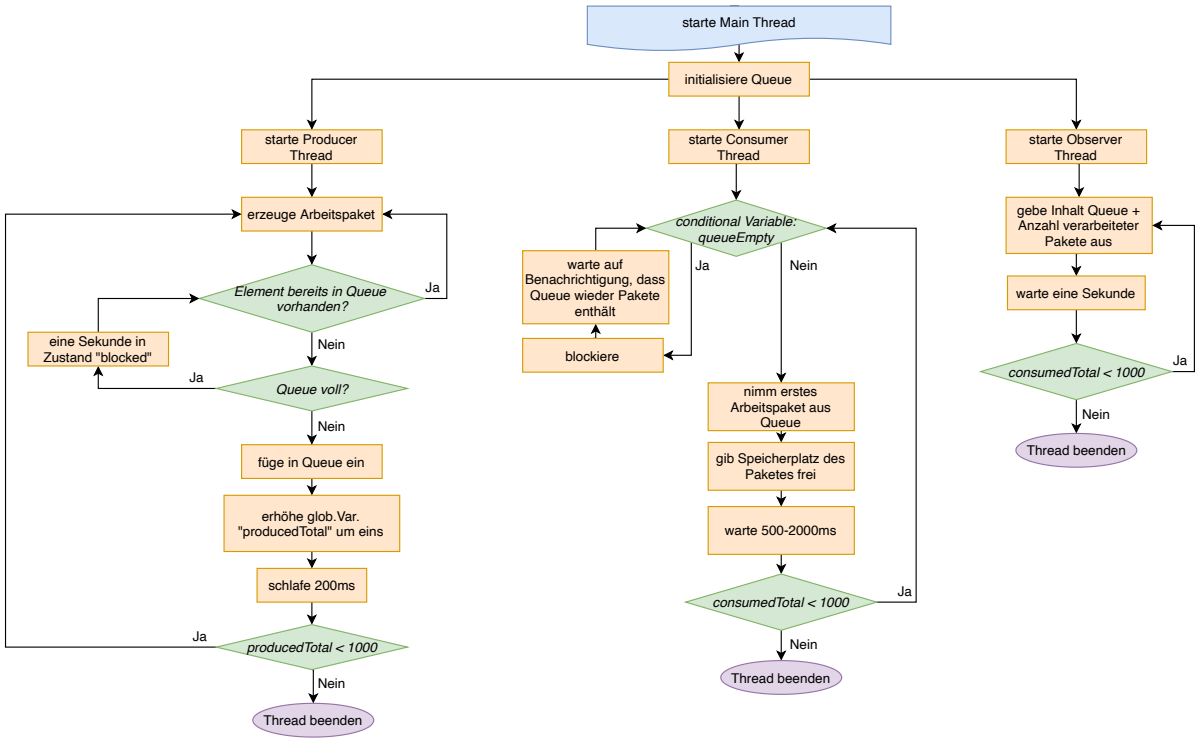

Abbildung 1: Ablauf der einzelnen Threads

2.) Synchronisation:

In diesem Projekt haben wir mehrere kritische Bereiche.
Diese sind die Pakete in der Queue:

- Mehrere Producer/Consumer Threads wollen gleichzeitig auf das gleiche Paket in der Queue zugreifen:

  ➔ ein Paket hat seinen **eigenen Mutex,** wird dieses Paket jetzt von einem Consumer/Producer benutzt, wird dieser Mutex gesperrt.
  Wenn ein anderer Producer/Consumer nun gleichzeitig auf dieses Paket zugreifen will, dann wird dies durch den Mutex verhindert.
  ➔ Somit bleibt die Queue weiterhin für die anderen Threads zugreifbar
- Um bei der Verkettung beim Einfügen oder Entfernen aus der Queue Fehler zu vermeiden, werden auch die Vorgänger und Nachfolger der gerade benutzten Elemente mit Mutexen gesperrt

Bei einer leeren Queue kommt es zu folgender Problematik:

- Wenn die Queue leer ist und Pakete entnommen werden sollen, dann wird dies durch eine **Conditional Variable** verhindert.
  Diese blockiert den Thread so lange, bis wieder Pakete in der Queue sind. Das Signal zum Entsperren kommt aus dem Producer Thread.

Gleichzeitiges Beenden aller Consumer Threads:

- in der main wartet eine Conditional Variable darauf, dass alle Pakete konsumiert wurden
- sobald dies geschieht, werden alle Consumer abgebrochen

3.) Liste der Testfälle:

Anzahl der Threads pro Modul:

Producer	**Consumer**	**Observer**
50	50	1
1	1	1
10	50	1
50	10	1
200	200	1