# Softwareverfikation Projekt 1

# Aufgabenstellung und Beschreibung des Round-Robin-Scheduler´s:

Unsere Aufgabe ist es einen Round-Robin-Scheduler (RRS) mit Dafny zu implementieren und verifizieren. Der RRS wird mit einer Warteschlange nach dem FiFo-Prinzip (First in - First out) verwaltet. In der Warteschlange befinden sich noch zu bearbeitete Prozesse. Der Ablauf des RRS ist wie folgt. Der erste Prozess wird aus der Warteschlange genommen. Er bekommt sein Quantum an Rechenzeit von der CPU. Wenn das Quantum abgelaufen ist, wird der Prozess am Ende der Warteschlange hinzugefügt. Danach wird ein neuer Prozess aus der Warteschlange genommen. Sollte ein Prozess keine Rechenzeit mehr benötigen, wird er nicht mehr in die Warteschlange eingefügt.

# Zweite Implementierung des RRS:

Unsere zweite Implementierung war einen Prozess-Block und ein System (OS) mit einer Queue zu implementieren. Der Aufbau der einzelnen Komponenten sieht wie folgt aus.

Der Prozess (PCB\_t):

* + speichert seine Prozess-ID (pid) zur eindeutigen Wiedererkennung.
  + speichert seine benötigter Rechenzeit (duration).
  + speichert die ID vom Prozessersteller (ownerID).
  + speichert die bereits erhaltene Zeit von der CPU (usedCPU).
  + Hat einen Konstruktor „Init(pid: int, duration: int, ownerID: int)“ zur Erstellung eines Prozesses mit Initialwerten.

Die Queue (Queue):

* + wird mittels einer Sequenz aufgebaut, die Prozesse speichert (que).
  + hat das Prädikat „Valid()“. Es überprüft, ob es in der Sequenz keine Null-Werte sowie keine doppelten Prozess-ID´s vorkommen.
  + hat das Prädikat „inQue(pid: int)“. Es überprüft, ob sich die pid schon in der Sequenz befindet.
  + hat einen Konstruktor „Init()“ zur Erstellung einer leeren Sequenz.
  + hat eine Methode „enQueue(prozess: PCB\_t)“ zum hinzufügen eines Prozesses.
  + Hat eine Methode „deQueue()“ zum entnehmen des ersten Elements aus der Sequenz.

Das System (OS):

* + speichert eine Sequenz von Prozessen (que).
  + speichert den aktuellen Prozess (pcb).
  + speichert das Quantum (quantum).
  + Hat das Prädikat „Valid()“ wurde von der Queue oben hier eingefügt.
  + Hat das Prädikat „inQueue(pid: int)“ wurde von der Queue oben hier eingefügt.
  + hat einen Konstruktor „Init(quantum: int)“ zum festlegen des Quantums. Es wird pcb auf null gesetzt. Die Sequenz que wird leer erstellt.
  + hat die Methoden „enQueue(prozess: PCB\_t)“ und „deQueue()“ von der Queue oben hier hinzufeügt.
  + benötigt die Methode „addPCB(prozess: PCB\_t)“ nicht mehr, da die Methode enQueue(prozess: PCB\_t) das gleiche macht.
  + Hat die Methode „getPCB()“ gibt den ersten Prozess in der Queue (que) zurück.
  + Hat die Methode „operate()“, dass dem aktuellen Prozess (pcb) sein Quantum (quantum) hinzufügt. Sollte der Prozess noch nicht fertig sein, wird er der Queue (que) hinzugefügt. Anschließend wird pcb auf null gesetzt.

Diese Implementierung scheint leichter verifizieren zu sein.

Quellen:

<https://de.wikipedia.org/wiki/Round_Robin_%28Informatik%29>

<https://searchcode.com/codesearch/view/28108541/>