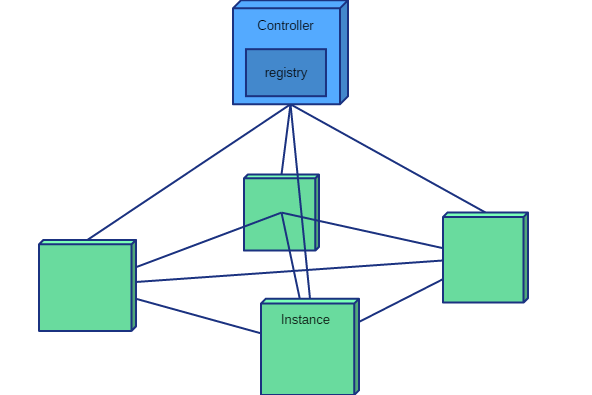
# Dokumentation: "Verteilt Speisende Philosophen"

## Systemarchitektur

In diesem Architekturdiagram sind die wichtigsten Bestandteile unseres Verteilten Systems, sowie ihre Vernetzung skizziert. Instance und Controller können hierbei auf eigenen Laufzeitumgebungen sein.



## Initiierungsphase

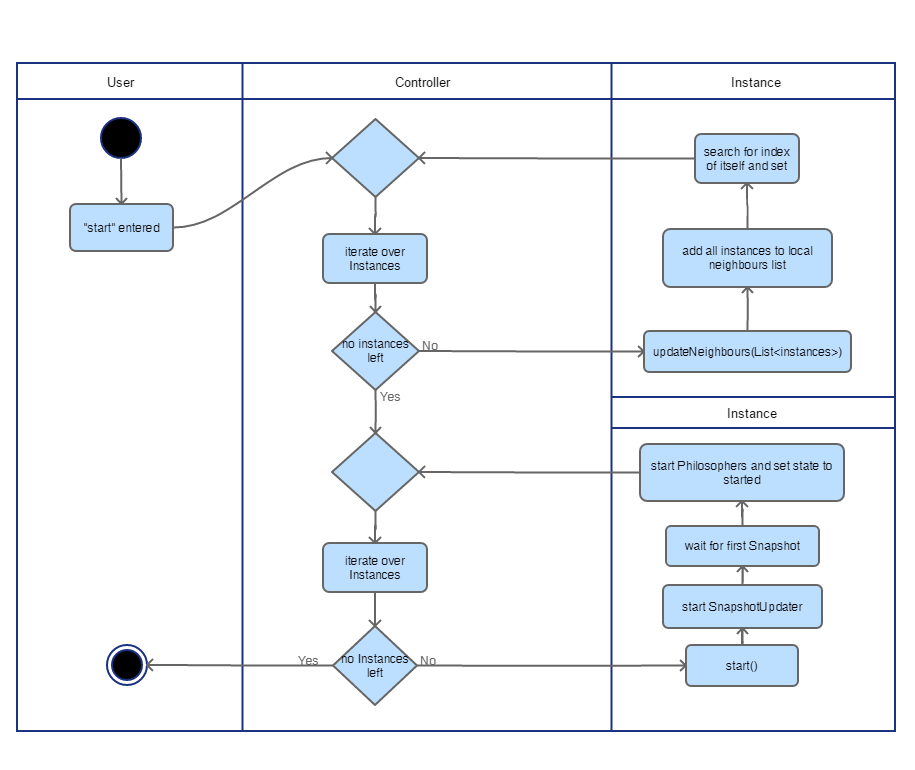
Der Controller kennt jede Instanz und führt eine Liste, die jeder Instanz einen festen Index zuweist. Diese Liste kann nur vom Controller geändert werden, und auch nur, wenn sich das System in der Initiierungsphase befindet.

Ist dies der Fall, können sich Instanzen beim Controller anmelden und die globale Liste wird erweitert.

Wenn die Initiierungsphase abgeschlossen ist, verfügt jede Instanz über eine lokale Kopie der Liste und kennt damit auch alle anderen Instanzen und den eigenen Index.

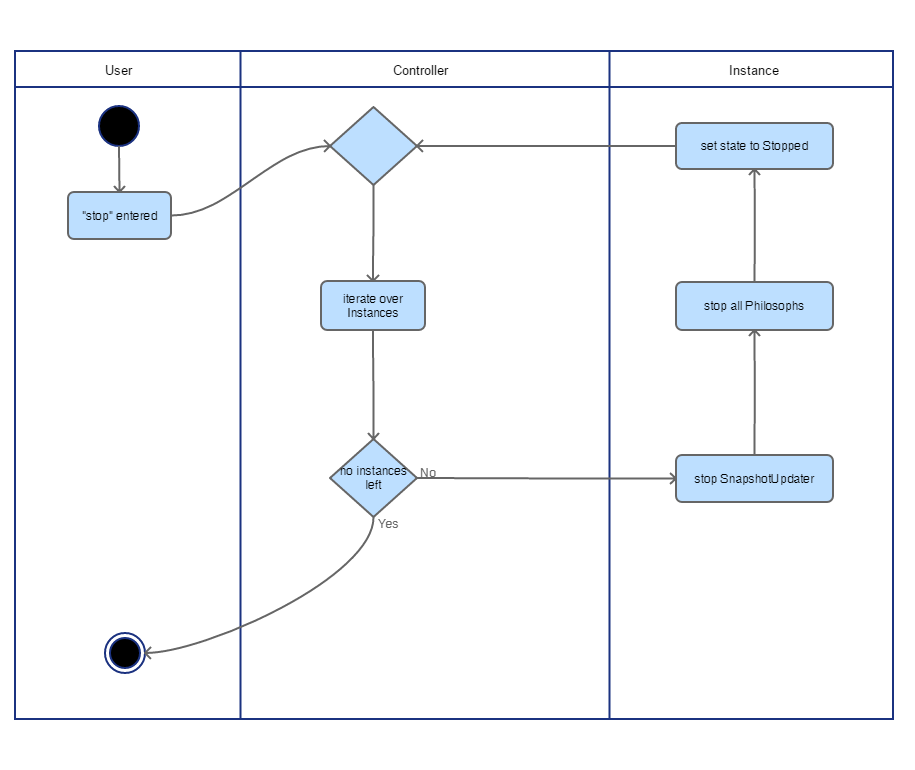
## Systemstart

Um die Initiierungsphase zu verlassen, wird Controllerseitig das System gestartet.



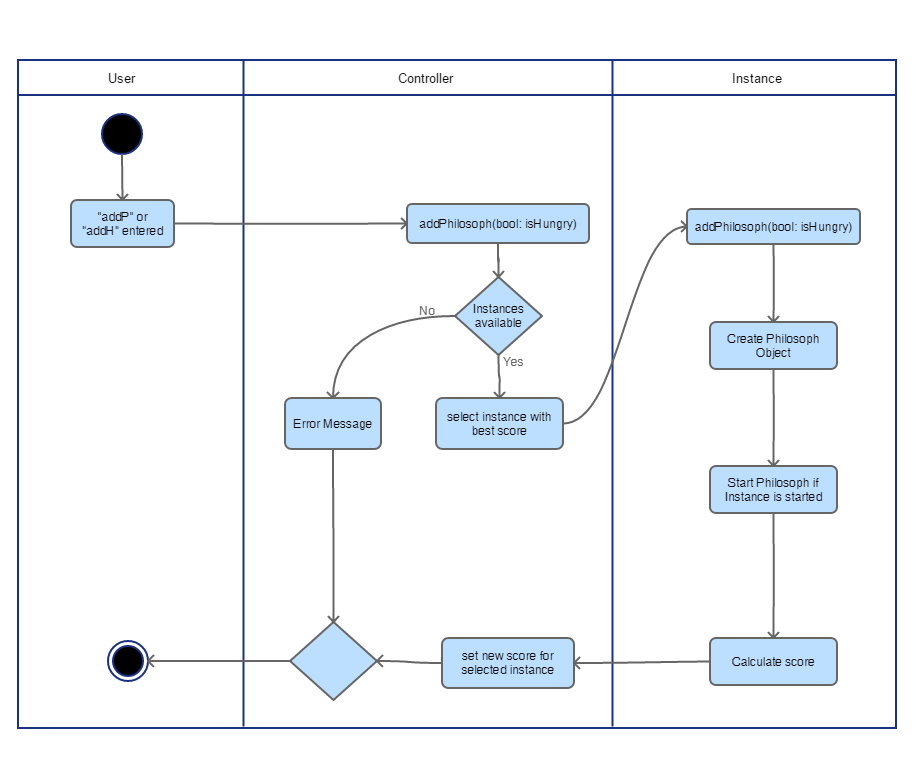
## Systemstopp

Als Rückkehr in die Initiierungsphase wird Controllerseitig der Systemstopp durchgeführt.



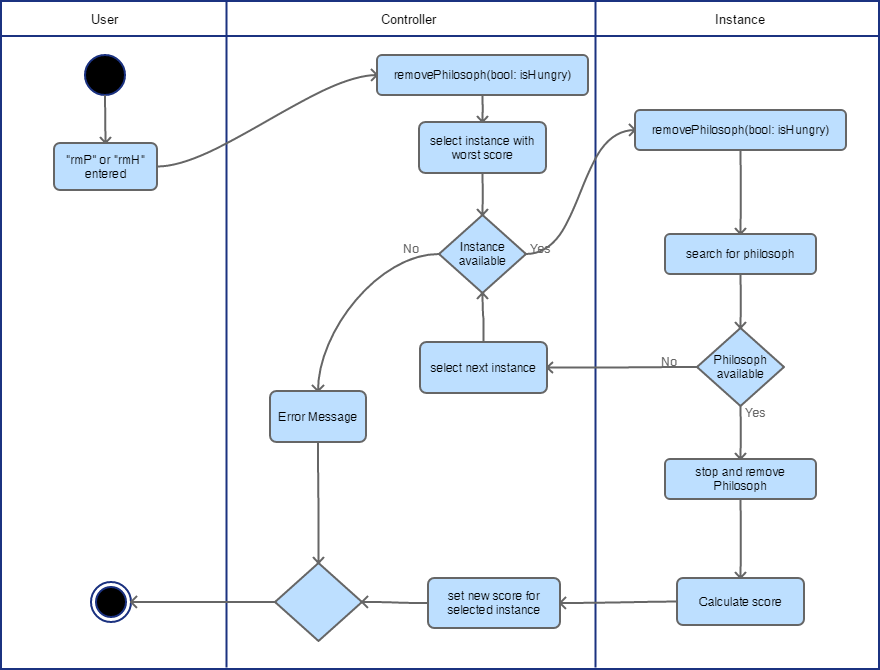
## Philosophen hinzufügen

Philosophen können vom Controller aus hinzugefügt werden. Hierbei kann man auswählen ob ein hungriger – oder normaler Philosoph hinzugefügt werden soll. Der Controller entscheidet daraufhin auf welcher Instanz (geringste Philosophendichte) der Philosoph laufen soll

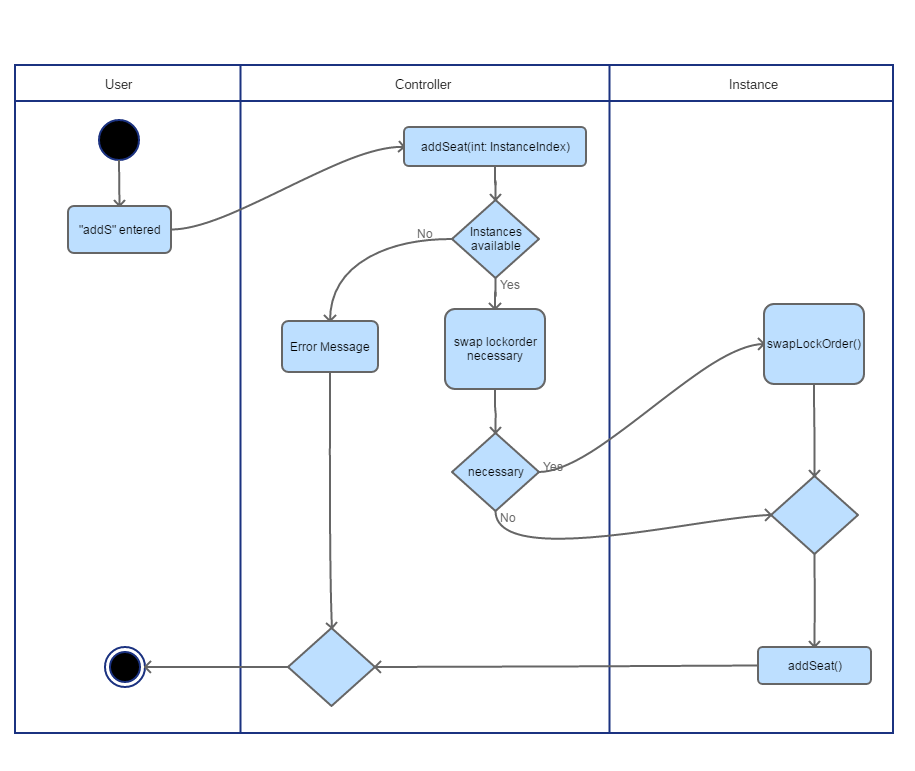


## Philosophen entfernen

Auch das Philosophen entfernen wird vom Controller aus gesteuert. Es wird wieder spezifiziert ob ein Hungriger, oder ein Normaler gelöscht werden soll. Danach wird erst bei der Instanz mit der größten Philosophendichte gesucht. Falls dort nichts gelöscht werden kann, wird in allen anderen Instanzen gesucht.



## Platz hinzufügen

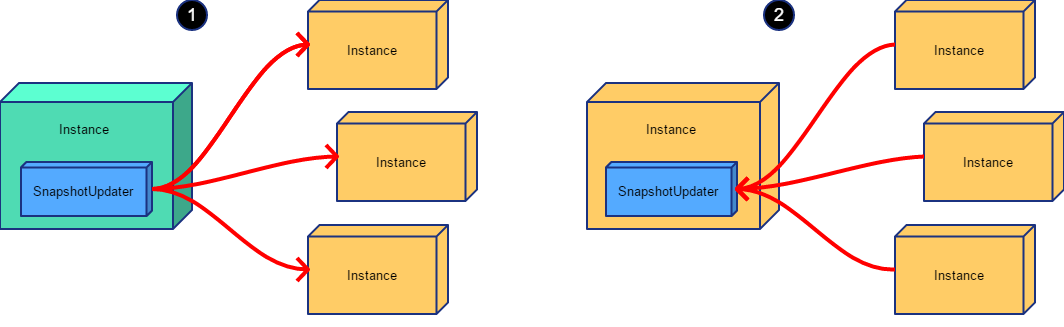


## Instanzen Snapshots

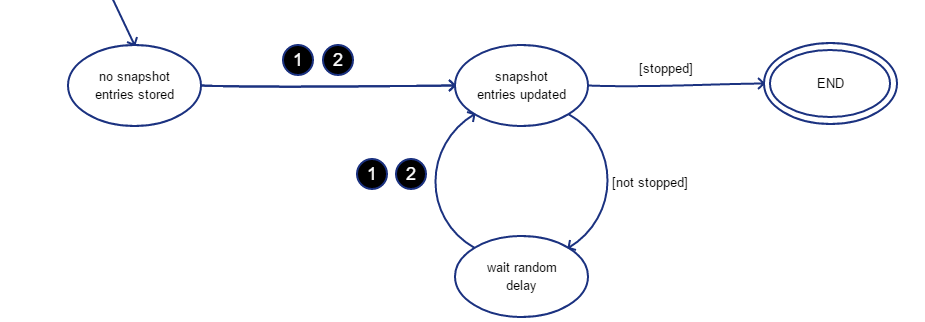
In unserem System sind alle Locks die zur Laufzeit benötigt werden so lokal wie möglich. Um trotzdem die Sicht aufs Gesamtsystem nicht zu verlieren, werden von jeder Instanz regelmäßig Snapshots jeder anderen Instanz erstellt.

Damit dieser Mechanismus nicht zu viel Overhead erzeugt, werden mehrere Maßnahmen ergriffen:

* Lokale Snapshots anderer Instanzen haben Timestamps und werden nur geupdated wenn sie zu alt sind
* Beim updaten der veralteten Snapshots (2) wird auch gleich der eigene neue Snapshot mitgegeben (1)



Jede Instanz verfügt über einen SnapshotUpdater der immer wieder die eigenen Snapshots aktuell hält.

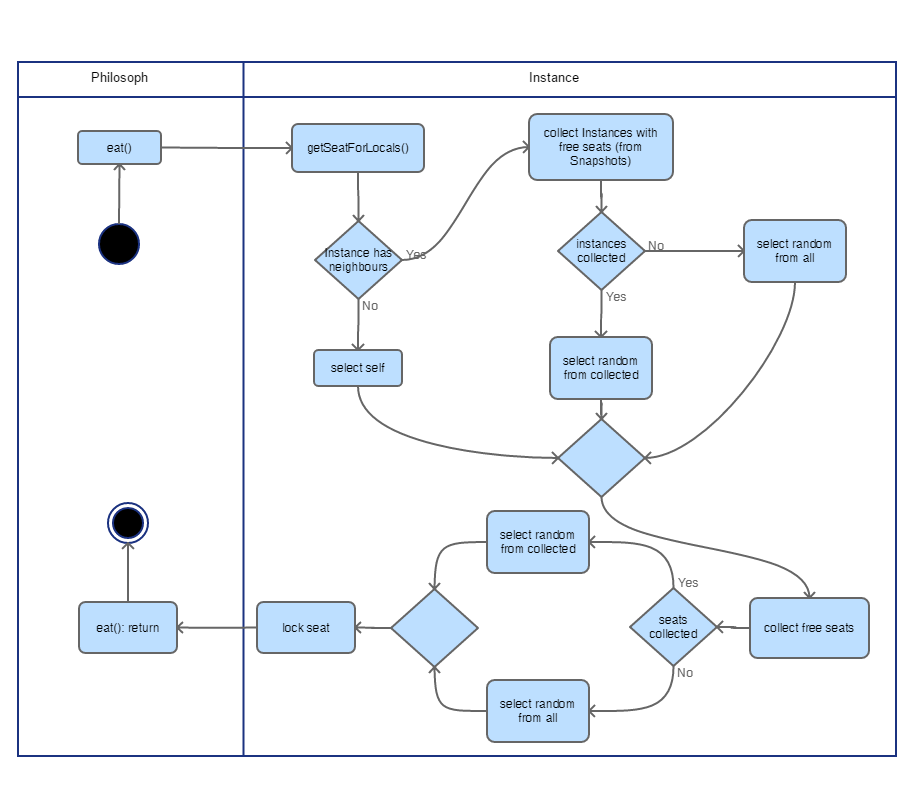


## Global einen freien Platz belegen

Möchte ein Philosoph essen, sucht und belegt dieser Systemweit einen Platz. Hierbei wird als erstes in der eigenen Instanz nach einem freien Platz gesucht. Sollte kein Platz aufzufinden sein, wird in den lokalen Snapshots nach einer freien Instanz gesucht.

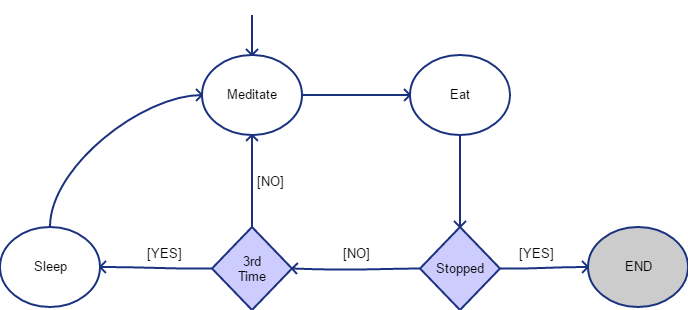
Sowohl für die Auswahl der Instanz in der ein Platz belegt wird als auch für den Platz der in dieser belegt wird, verwenden wir dieselbe Suchstrategie:

1. Sammle alle freien Plätze/Instanzen mit freien Plätzen (nicht synchronisiert also nur Annäherung)
2. Wähle zufällig aus den gesammelten aus
3. Falls nichts gesammelt wurde, wähle zufällig aus



## Philosoph

Der Philosophen-Thread durchläuft drei Zustände (Meditieren, Essen, Schlafen) in einer Schleife, die von einem anderen Thread gestoppt werden kann.



Um essen zu können, muss bei der jeweiligen Instanz nach einem Platz gefragt werden, welcher dann nach dem Essen wieder freigegeben wird.

## Beispielausgabe des Systems

Nachdem das System gestartet wurde und ein paar Sekunden gelaufen ist, wird „pI“ und „printP“ eingegeben um aktuelle Informationen zu Instanzen und Philosophen auszulesen.

