Pseudocode

Overview

To solve the drinking philosophers problem, we assume that the philosophers are all connected at the table to each other.

We can represent this as an adjacency matrix:

```
class PhilMatrix
    PhilRow, PhilRow, ...
```

```
class PhilRow
    mutex, mutex, mutex, ...
```

We can initialize this with n PhilRows with n mutexes within them.

Philosopher

A philosopher will have an enum representing each state:

```
enum PhilState
    Tranquil
    Thirsty
    Drinking
```

A philosopher's process goes as follows:

```
drinks = check_desired_drinks()
if drinks >= 0
    check_drink_mutex(drink)
    if no_lock(mutex)
        drink(mutex)
    drinks --
loop
```

We then create threads to handle each philosopher.