Multithreading – recap

- Divide work between multiple threads
- Locate shared data and accesses to it – **critical sections**
- Synchronize with one big lock
- Optimize with fine-grain locks and privatization

A good critical section

- *Mutual exclusion* – 1 thread at a time in critical section

- *No delays* – If no thread holds the lock and a thread arrives at the critical section, it should acquire it immediately

- *Eventual entry* – At the end, every process that wants to execute code in the critical section will succeed

- *No **deadlock*** – at least one process will acquire the lock before the critical section

Deadlocks – example

Problem: Two processes increment A and B acquiring locks before the increment

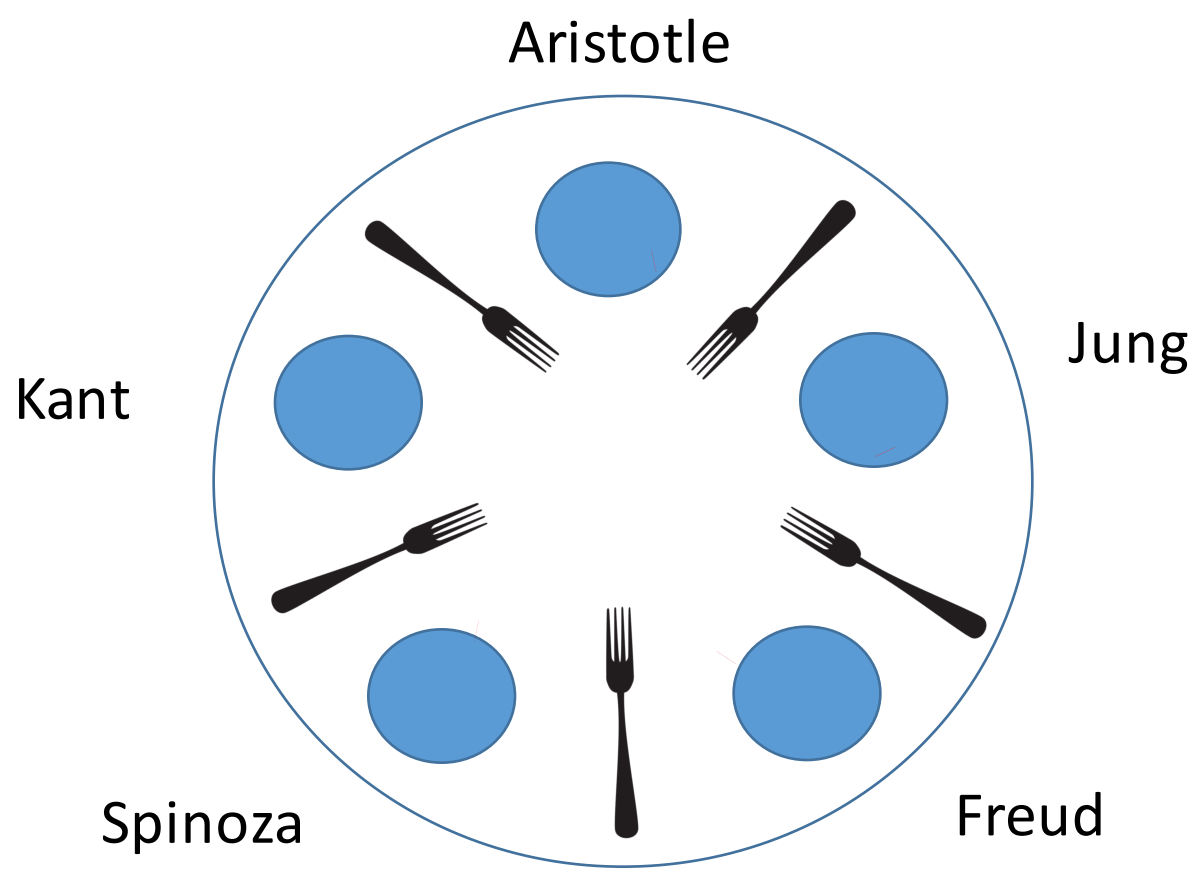
```
pthread_mutex_t mutex1,mutex2;
int a=b=0;
```

```
void P1(){
  pthread_mutex_lock(&mutex1);
  a++;
  pthread_mutex_lock(&mutex2);
  b++;
  pthread_mutex_unlock(&mutex2);
  pthread_mutex_unlock(&mutex1);
}
```

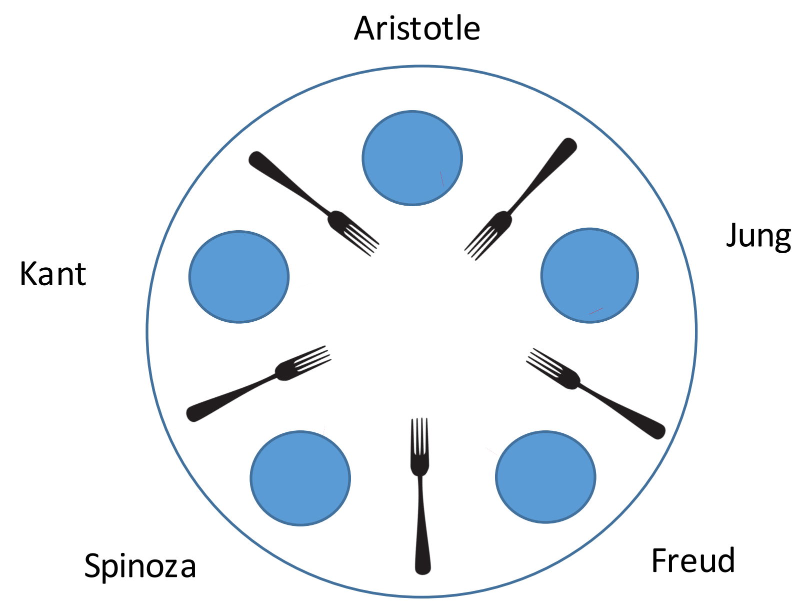
```
void P2(){
  pthread_mutex_lock(&mutex2);
  b++;
  pthread_mutex_lock(&mutex1);
  a++;
  pthread_mutex_unlock(&mutex1);
  pthread_mutex_unlock(&mutex2);
}
```

- P1 acquires lock for A and P2 acquires lock for B
- P1 tries to acquire lock for B before releasing lock on A and P2 tries to acquire lock on A before releasing lock on B
- Both will wait forever

Dining Philosophers



The problem



- 5 philosophers – **processes**
- 5 forks – **shared resources –** only one philosopher can hold a fork at a time
- Each philosopher thinks then eats
- Needs to have both forks in order to eat – **lock on each fork (mutex)**

Naïve solution

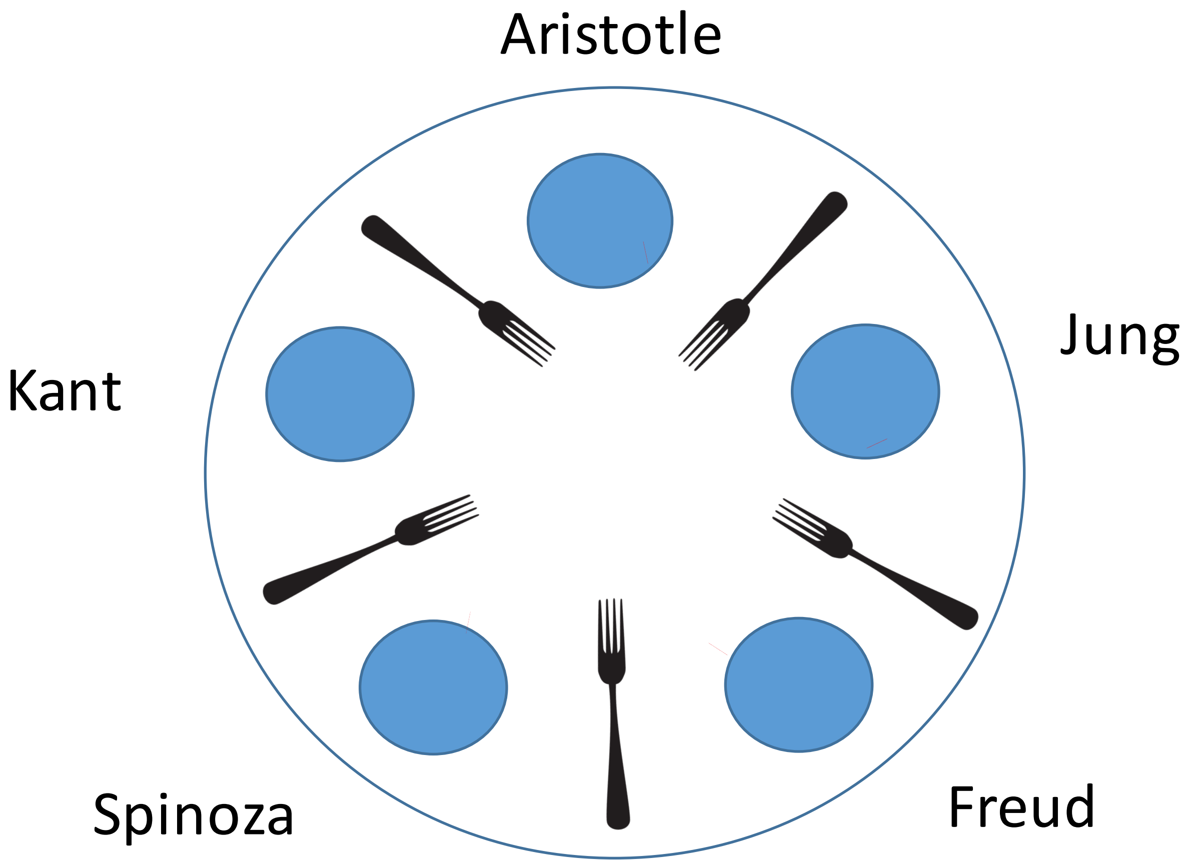
Every philosopher:

- Thinks,
- Tries to acquire lock on right fork
- Tries to acquire lock on left fork,
- Eats
- Releases forks

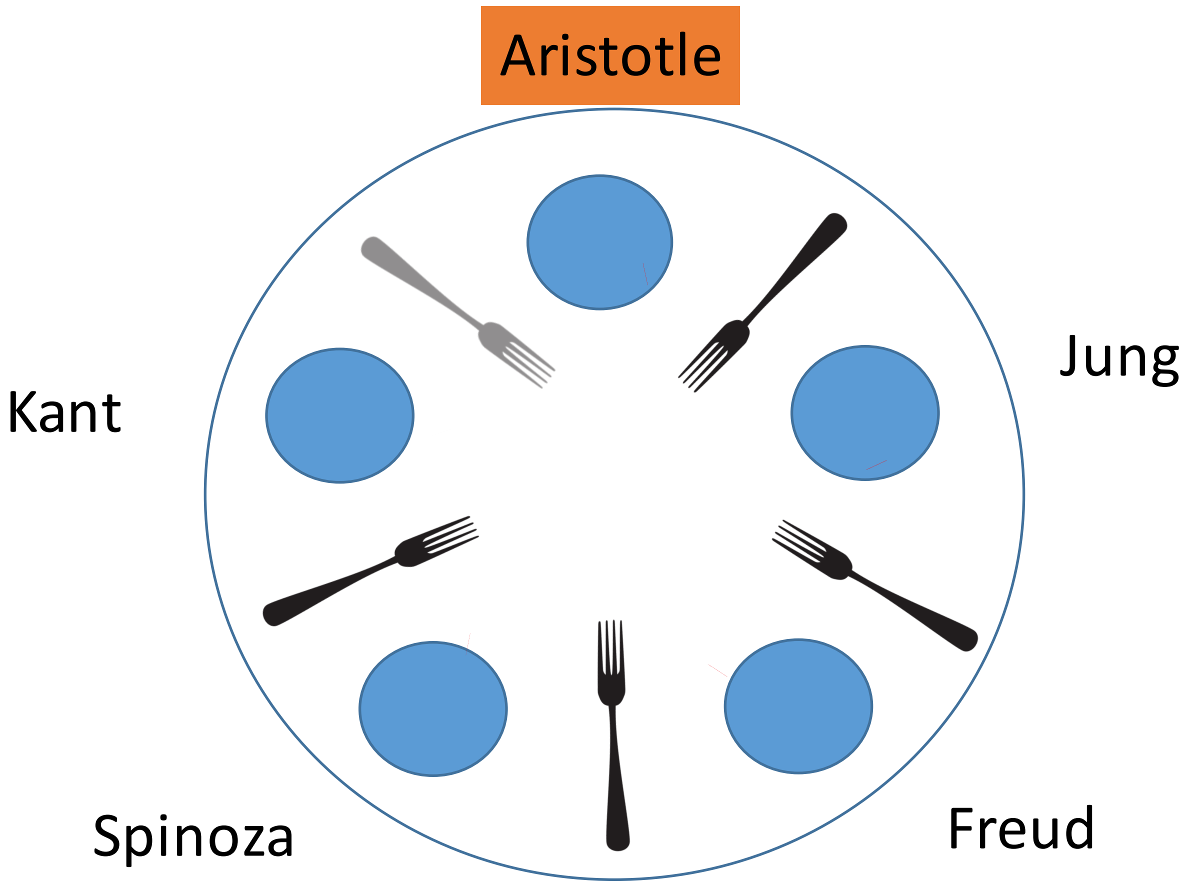
Problems?

```
while(true) {
  for(int i = 0; i<numPhilosophers;  i++){
    philosophers[i].think();

    pthread_mutex_lock(fork[i]);
    pthread_mutex_lock(fork[(i  - 1) % numForks]);

    eat();

    pthread_mutex_unlock(fork[i]);
    pthread_mutex_unlock(fork[(i  - 1) % numForks]);
  }
}
```

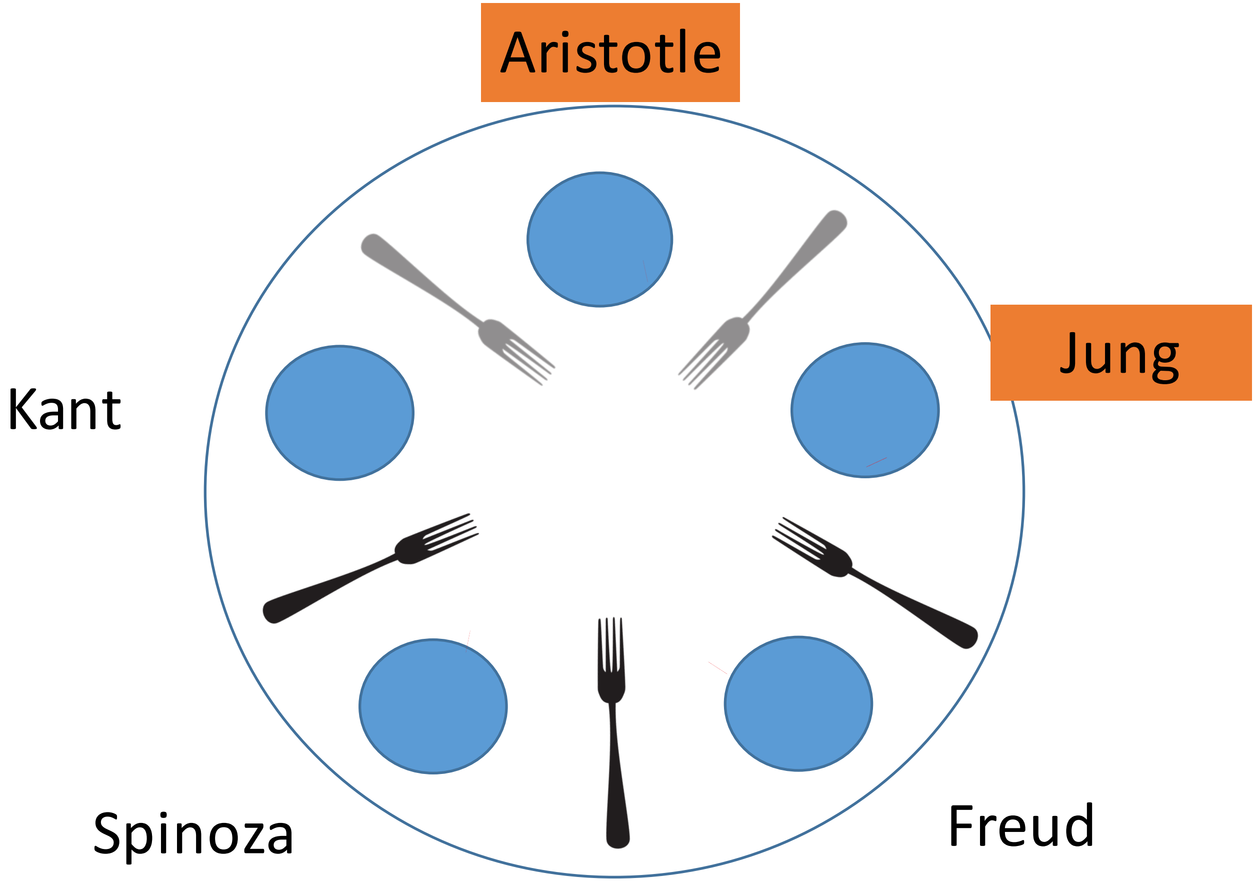
Deadlock - if all acquire their right fork



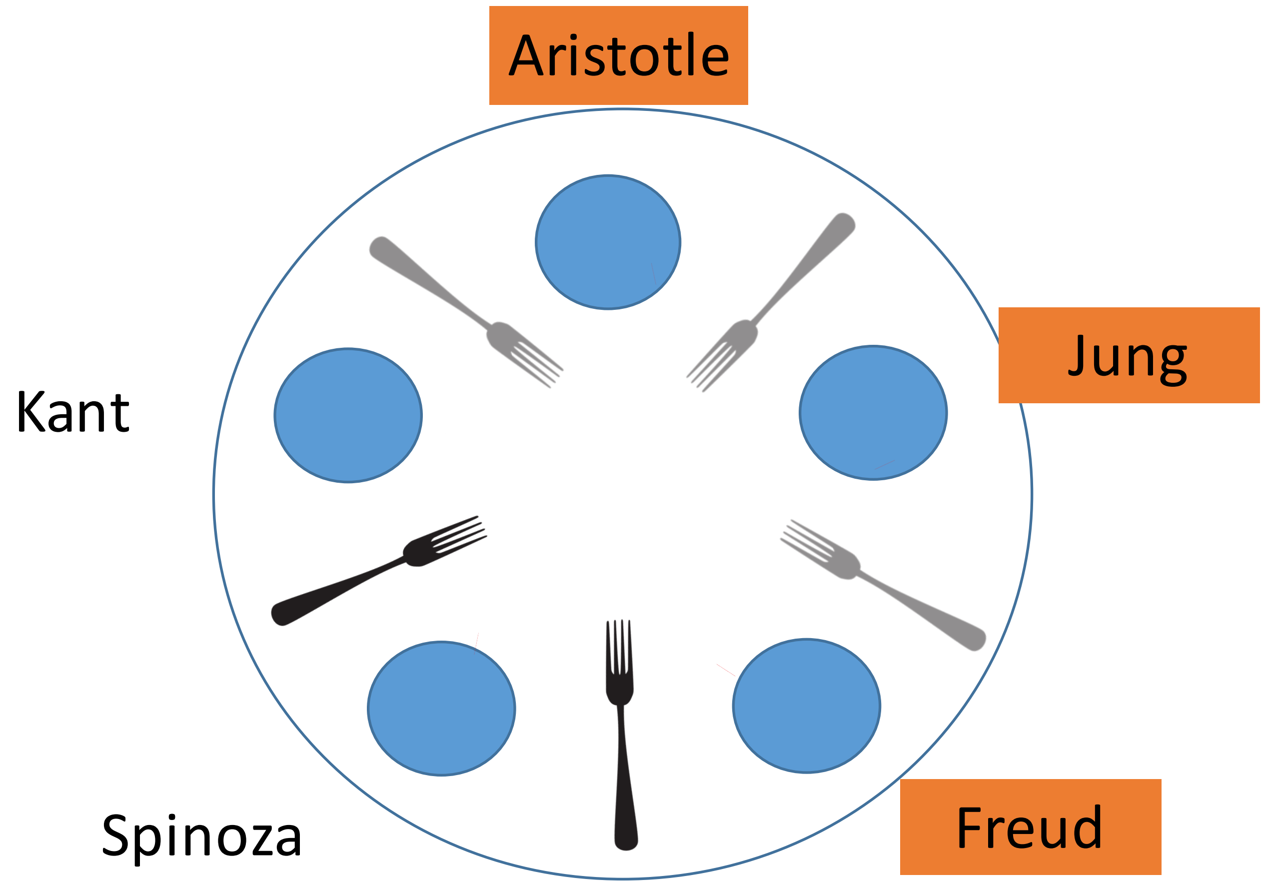
Deadlock - if all acquire their right fork



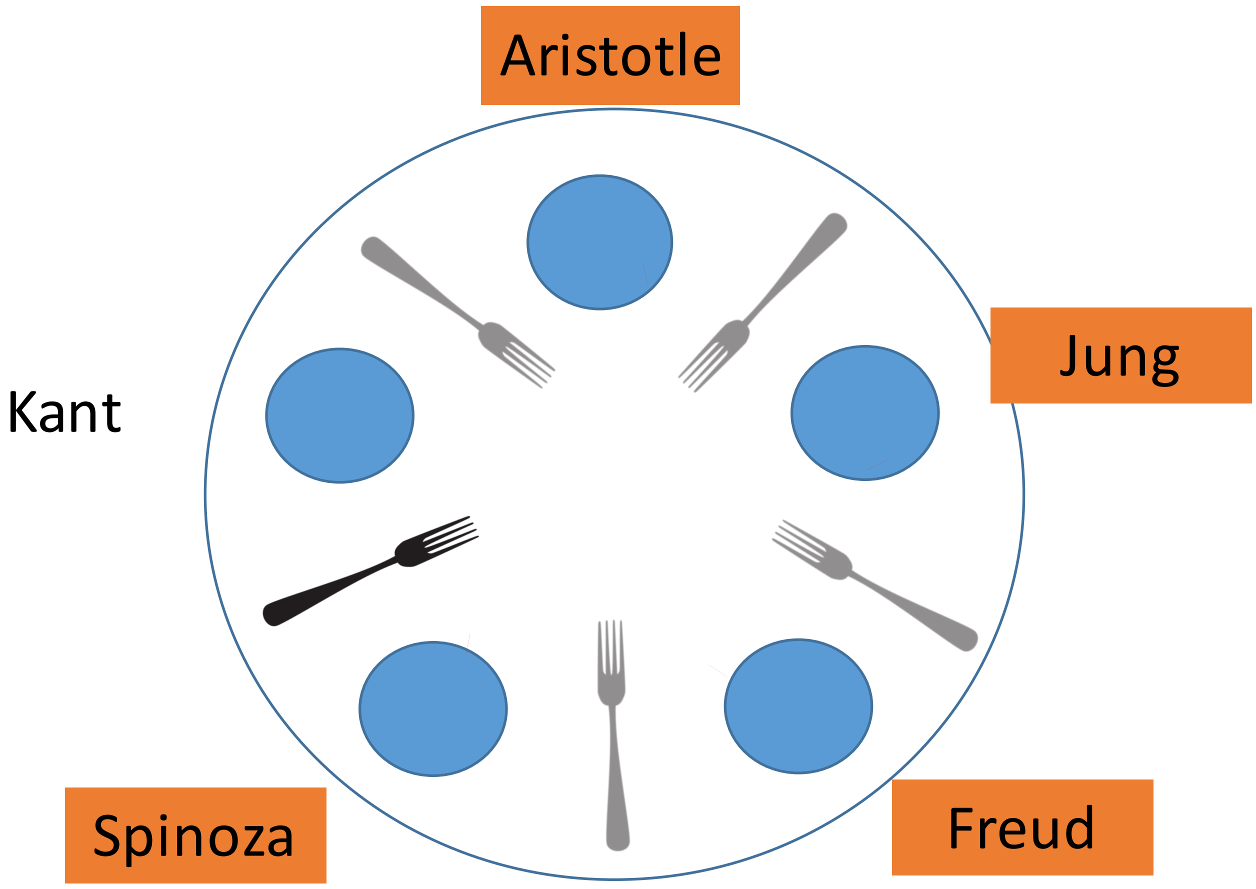
Deadlock - if all acquire their right fork



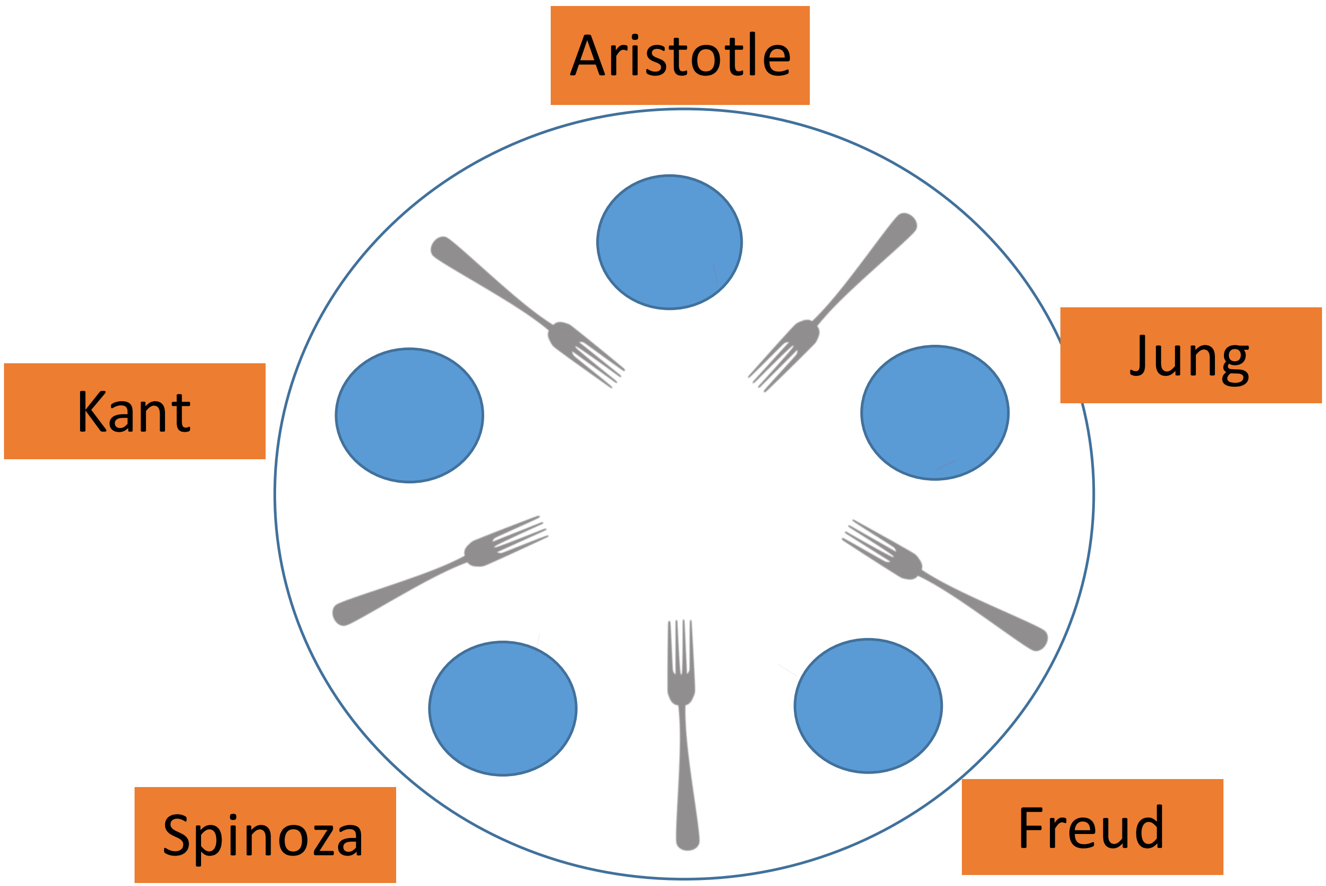
Deadlock - if all acquire their right fork



Deadlock - if all acquire their right fork



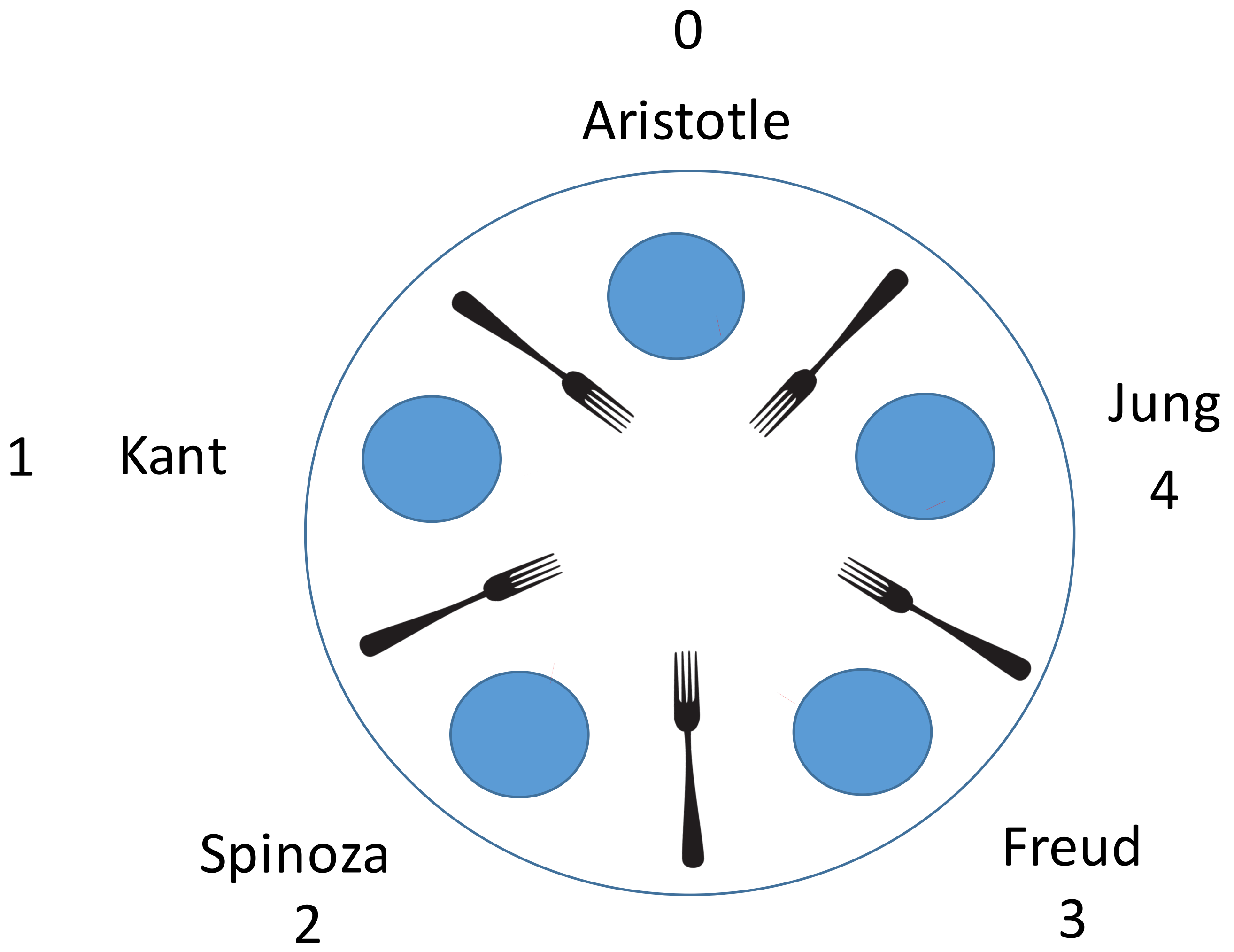
Deadlock - if all acquire their right fork



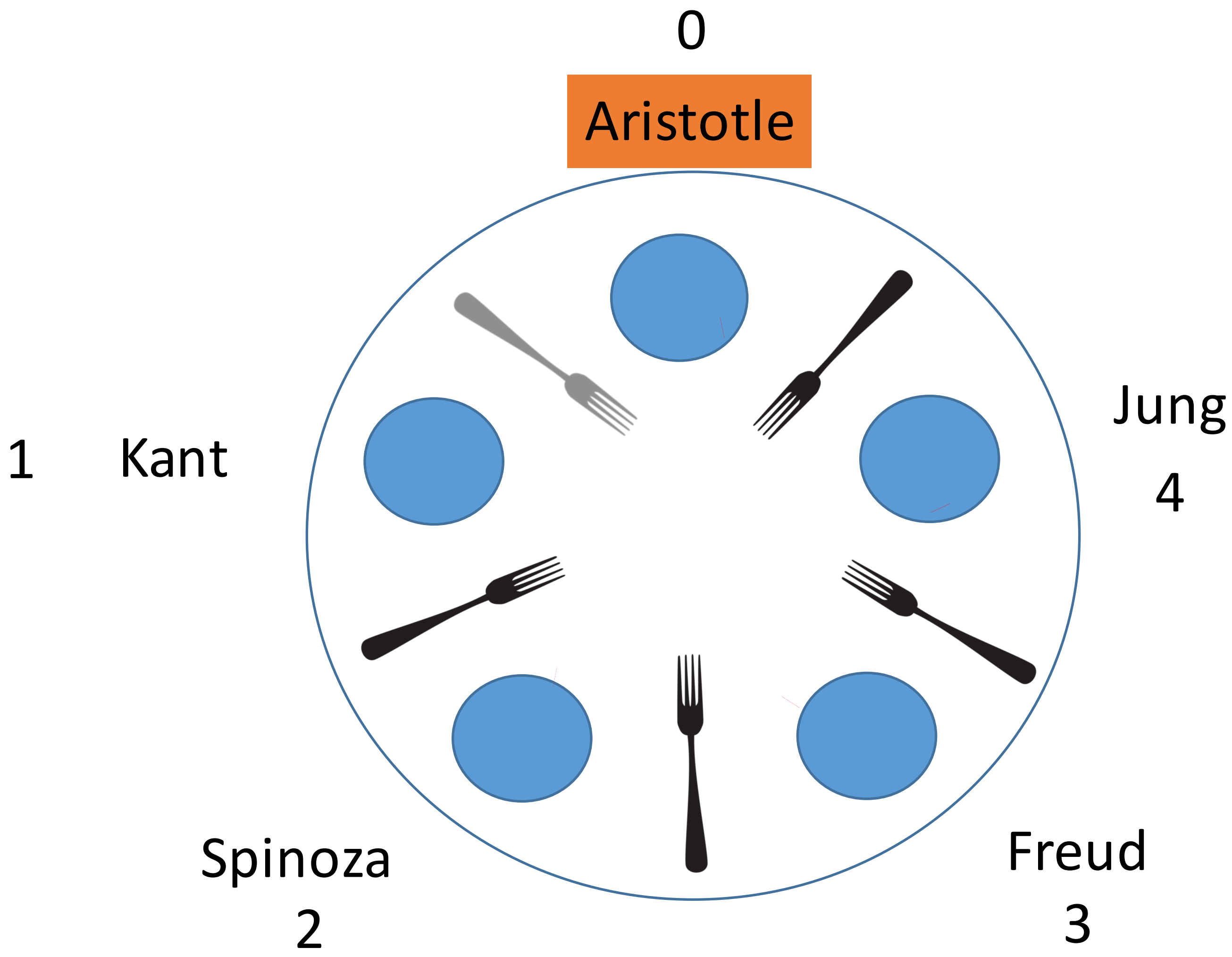
To avoid deadlock – change order of acquiring locks

```
while(true) {
  for(int i = 0; i<numPhilosophers;  i++){
    philosophers[i].think();

    if(i%2 == 0){
      pthread_mutex_lock(fork[(i  - 1) % numForks]]);
      pthread_mutex_lock(fork[i]);
    }
    else{
      pthread_mutex_lock(fork[i]);
      pthread_mutex_lock(fork[(i  - 1) % numForks]);
    }

    eat();

    pthread_mutex_unlock(fork[i]);
    pthread_mutex_unlock(fork[(i  - 1) % numForks]);
  }
}
```

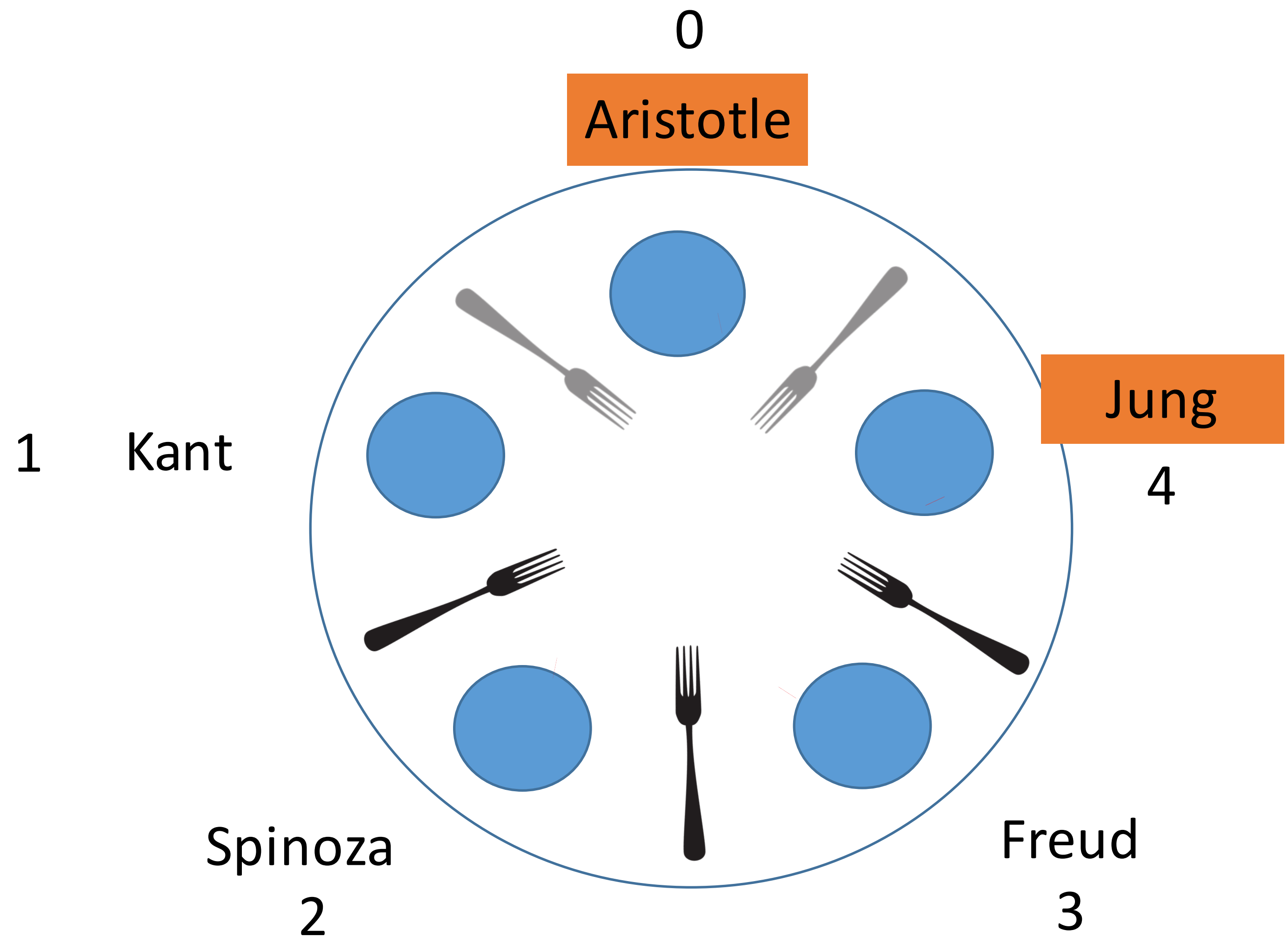
Deadlock - if all acquire their right fork



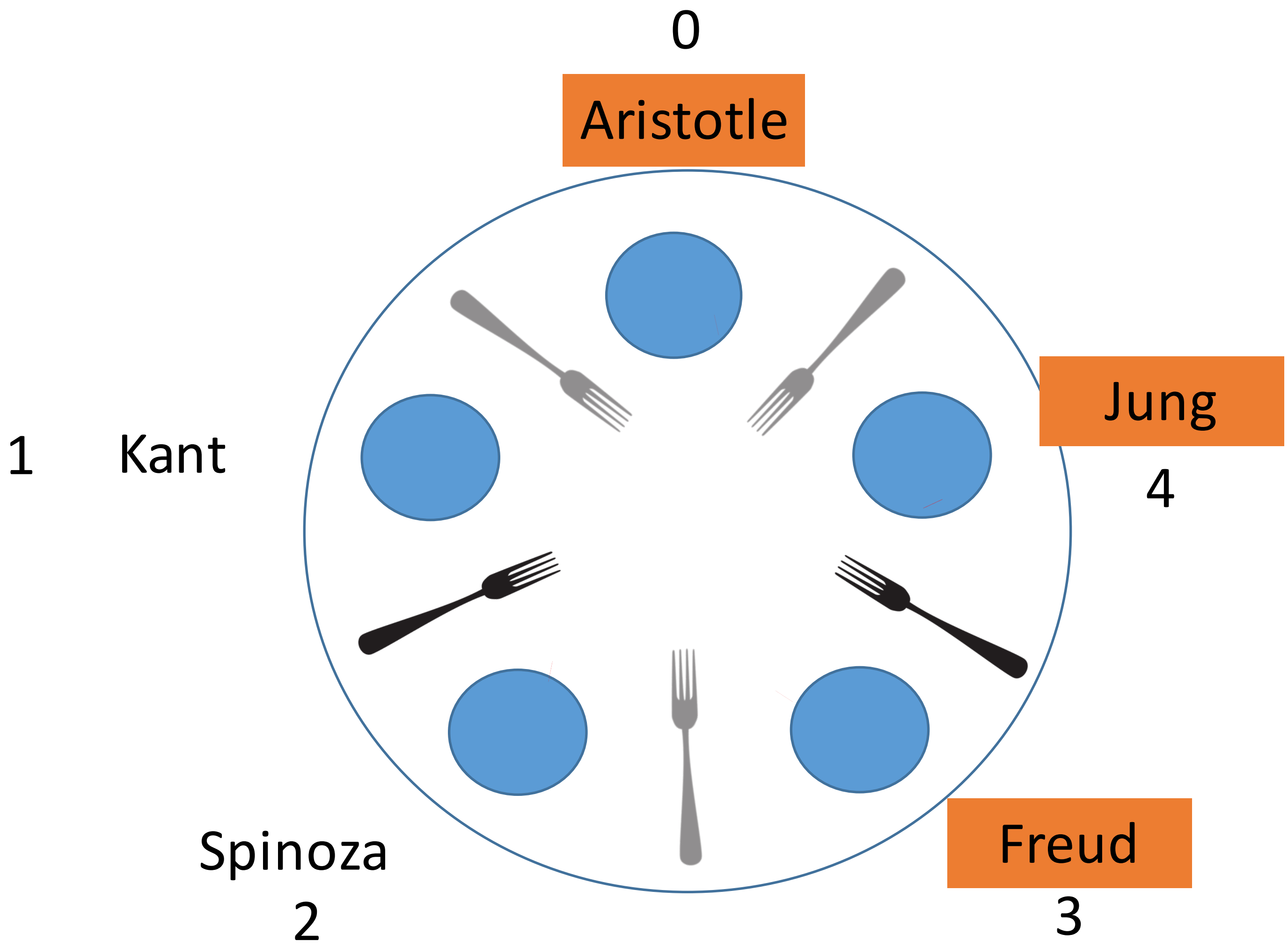
Deadlock - if all acquire their right fork



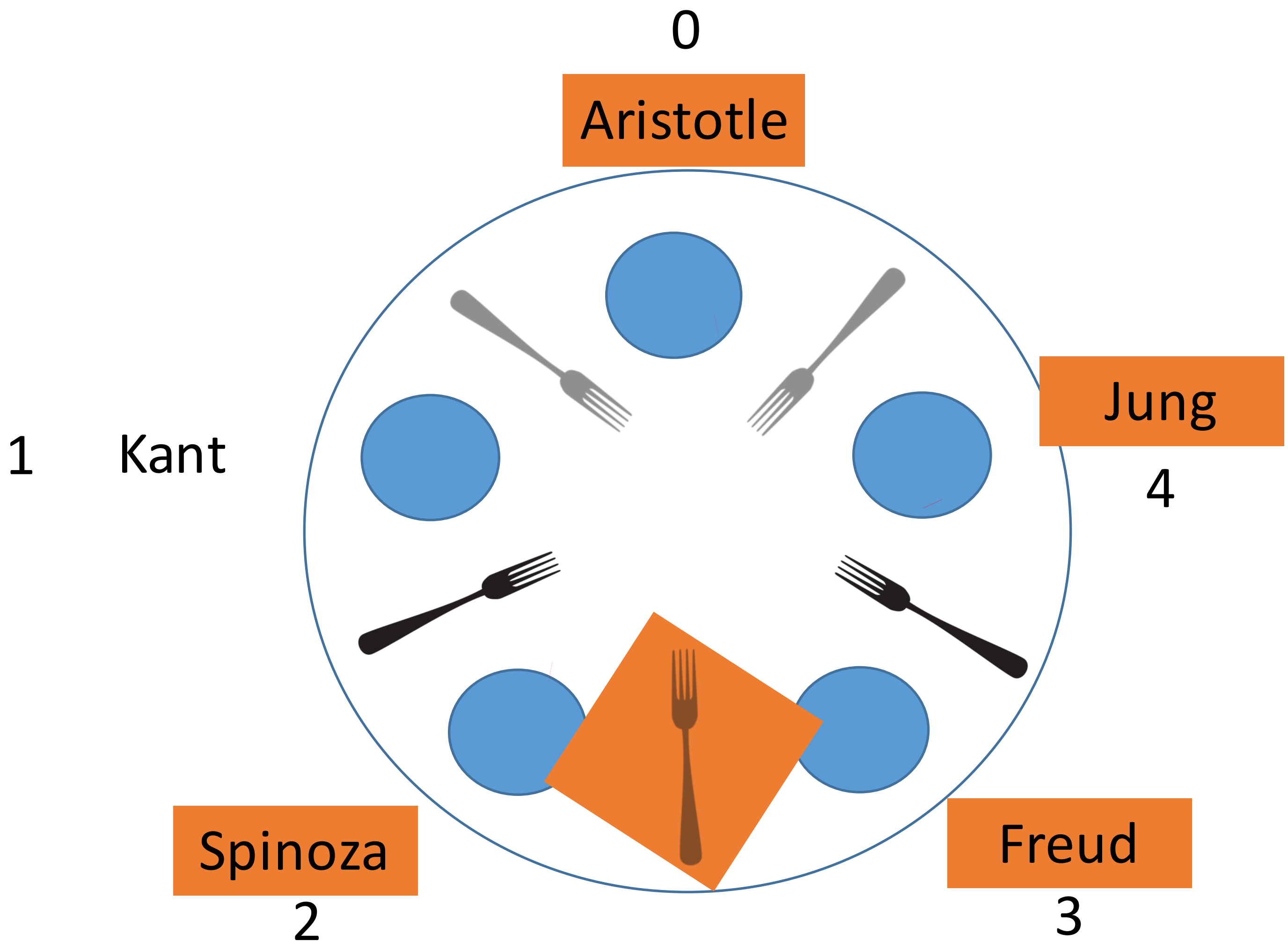
Deadlock - if all acquire their right fork



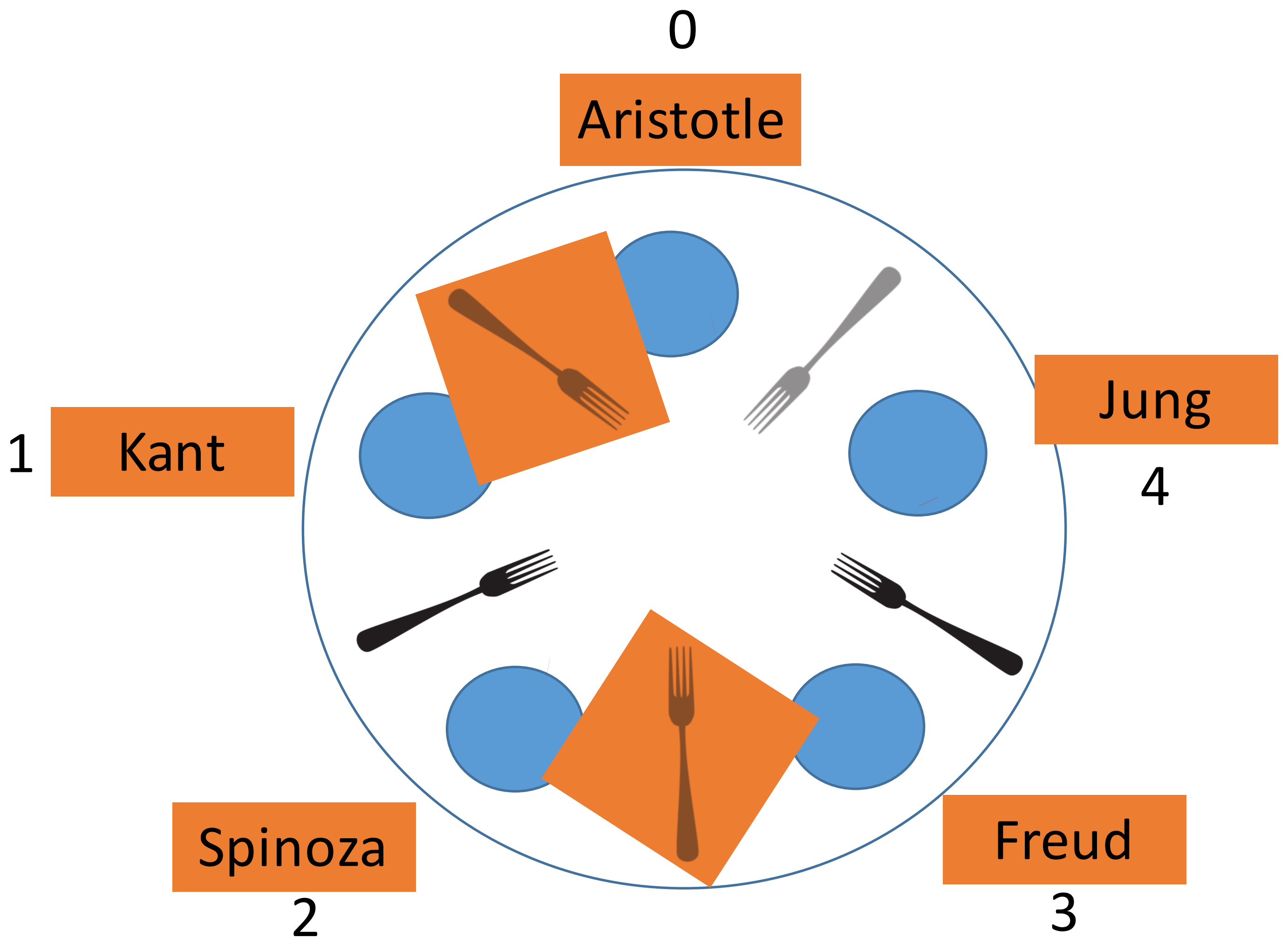
Deadlock - if all acquire their right fork



Deadlock - if all acquire their right fork



Deadlock - if all acquire their right fork



Deadlock - if all acquire their right fork