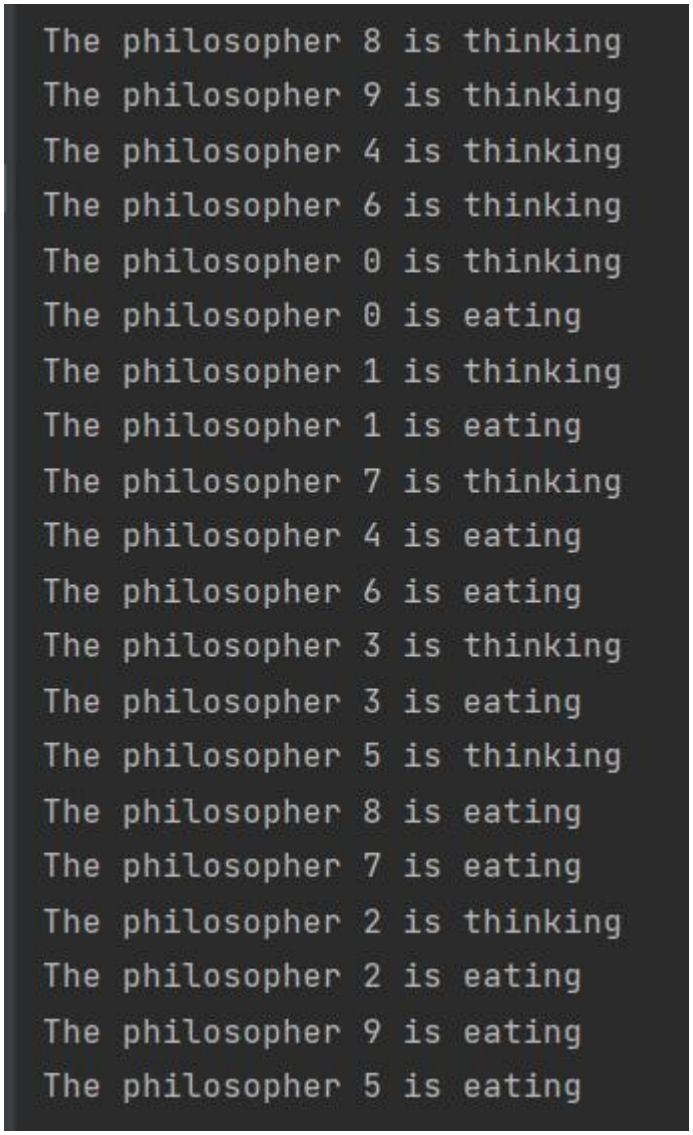
Core idea:
When a philosopher puts a fork down, they call V() on the condition semaphore of their neighbor, who eventually wakes up “with the forks in their hands”

Result:



Code:
Note: There are 10 philosophers. The id varies from 0 to 9.

```
public class philosopher_solution {

    static int THINKING = 0;

    static int HUNGRY = 1;

    static int EATING = 2;
```

```
static int N=10;

static int []state=new int[N];

static Semaphore []c=new Semaphore[N];

static Semaphore mutex = new Semaphore(1);


public static void get_forks(int i) throws InterruptedException {

    mutex.P();

    state[i] = HUNGRY;

    test(i);

    mutex.V();

    c[i].P();

}

public static void put_forks(int i) throws InterruptedException {

    mutex.P();

    state[i] = THINKING;

    offer_fork(i==0?N-1:i-1);

    offer_fork(i==N-1?0:i+1);

    mutex.V();

}


public static void test(int i) { /* called under mutex conditions */

    if (state[i==0?N-1:i-1] != EATING && state[i==N-1?0:i+1] != EATING) {
```

```
            state[i] = EATING;
            c[i].V();
        }
    }
    public static void offer_fork(int i) {
        if (state[i] == HUNGRY && state[i==0?N-1:i-1] != EATING &&
state[i==N-1?0:i+1] != EATING) {
            state[i] = EATING;
            c[i].V();
        }
    }

    public static void main(String[] args) throws InterruptedException {
        for (int i = 0; i < 10; i++) {
            c[i]=new Semaphore(0);
        }
        for (int i = 0; i < 10; i++) {
            Philosopher ph = new Philosopher(i);
            ph.start();
        }
    }
}
```

```
class Semaphore{
    private int  count;
    public Semaphore(int n){
        count=n;
    }

    public synchronized void P() throws InterruptedException {
        count--;
        if(count<0)
            wait();
    }

    public synchronized void V(){
        count++;
        if(count<=0)
            notify();
    }
}
class Philosopher extends Thread{
    int id;

    public Philosopher(int i) {
```

```
        id=i;
    }

    @Override
    public void run() {
        for(int i=0;i<1;i++) {
            think();
            try {
                philosopher_solution.get_forks(id);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            eat();
            try {
                philosopher_solution.put_forks(id);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }

    public void think(){
```

```
        System.out.println("The philosopher "+id+" is thinking");
    }


    public void eat(){
        System.out.println("The philosopher "+id+" is eating");
    }
}
```