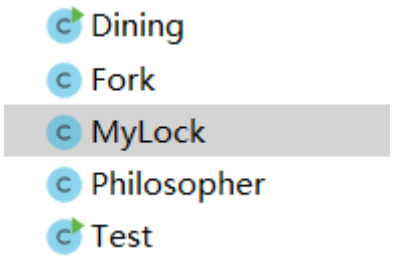
Lab2实验文档

程茜 19302010084

代码结构：



Part A：

运行截图：

```
Test A start
5
Test A passed

Process finished with exit code 0
```

设计思路：

在MyLock类中实现了lock和unlock方法，MyLock类中有一个lockID用于记录当前占用锁的线程id，id为-1表示锁未被占用，每个线程调用lock()方法时会先判断当前的lockID，如果不是-1则会等待，直到lockID 变为-1再尝试获得锁，并将id改为其自身id，修改完后为了防止多个线程同时对锁进行了修改，之后需要再等待一段时间确保所有改了id的线程操作均已完成；而unlock()则是简单将lockID置为-1；

PartB

运行截图

```
Philosopher 3 1188133695580900: Thinking
Philosopher 4 1188133695622300: Thinking
Philosopher 1 1188133720799600: Picked up fork 1
Philosopher 1 1188133721191500: Picked up fork 2
Philosopher 1 1188133721235500: Eating
Philosopher 4 1188133758995900: Picked up fork 4
Philosopher 4 1188133759464000: Picked up fork 5
Philosopher 4 1188133759537600: Eating
Philosopher 3 1188133768697300: Picked up fork 3
Philosopher 1 1188133799276100: Put down  fork 1
Philosopher 1 1188133799425300: Put down  fork 2
Philosopher 1 1188133799498000: Thinking
Philosopher 5 1188133799564200: Picked up fork 1
Philosopher 2 1188133799724100: Picked up fork 2
Philosopher 4 1188133816186800: Put down  fork 4
Philosopher 4 1188133816383300: Put down  fork 5
Philosopher 4 1188133816458200: Thinking
Philosopher 3 1188133816478000: Picked up fork 4
Philosopher 3 1188133816560400: Eating
Philosopher 5 1188133816687100: Picked up fork 5
```

设计思路

Fork类中含有一个partA中实现的MyLock对象，每次有哲学家拿起勺子时会调用lock()方法尝试加锁，放下勺子会调用unlock()方法

```
public void pickUp(int id) throws InterruptedException {
    lock.lock();
    System.out.println("Philosopher " + id + " " + System.nanoTime() + ": Picked up fork " + index);
}
public void putDown(int id){
    lock.unLock();
    System.out.println("Philosopher " + id + " " + System.nanoTime() + ": Put down  fork " + index);
}
```

对勺子进行编号，为了防止所有哲学家同时拿起左边或者右边的勺子的死锁，对勺子进行1-5编号，哲学家每次会先尝试拿起编号小的勺子，再尝试拿起编号大的勺子

```
doAction(System.nanoTime() + ": Thinking");
pickUpMinFork();
pickUpMaxFork();
doAction(System.nanoTime() + ": Eating");
putDownMinFork();
putDownMaxFork();
```