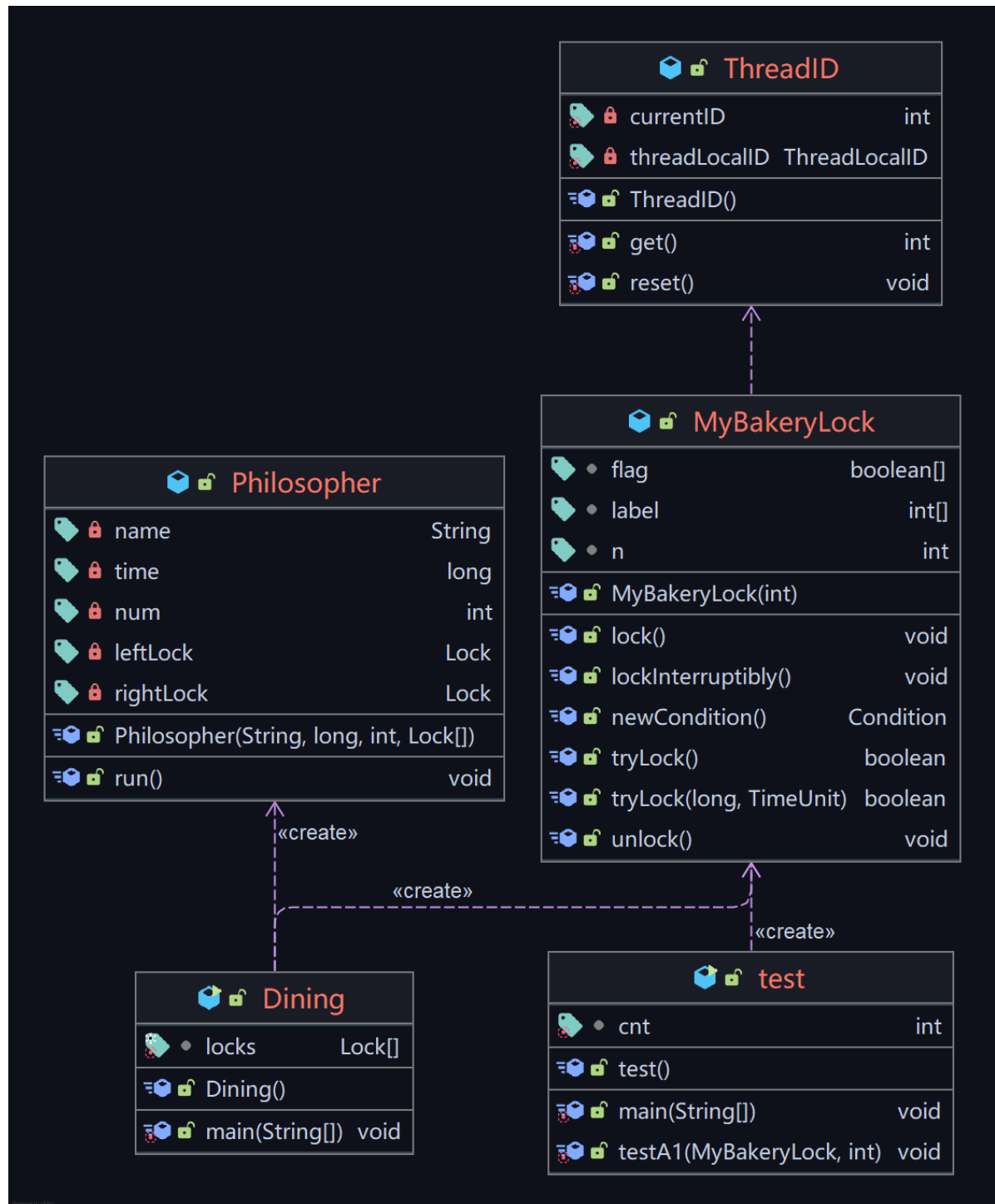


# Lab2开发文档

## 代码结构



- **Philosopher：哲学家类**

leftLock:模拟哲学家左边的筷子

rightLock:模拟哲学家右边的筷子

- **MyBakeryLock：自己实现的锁**

实现lock接口

- **ThreadID：为了方便标记线程，给每个线程存储一个私有的id**

- **test：part A测试类**

- **Dining：part B测试类**

## Part A

---

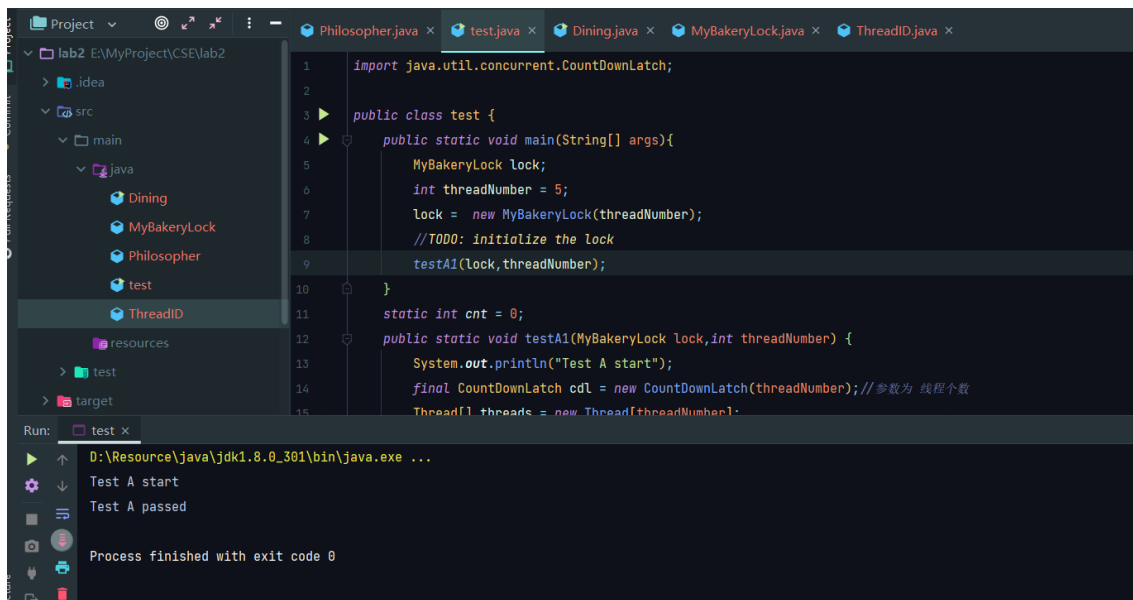
- **实现思路**

参考面包房锁的实现思路，给需要访问公共资源的每个线程登记一个号，号码数逐次加一，按照所给的号码从小到大依次访问公共资源，当线程访问结束后就将其所给的号码归0，如果这个线程需要再次访问这个公共资源就要重新排队。如果两个线程同时拿到了号码，也就是说拿到的号码一样，那就根据线程的id，id小的具有优先权

比如线程t1拿到号码a，线程t2拿到号码b,判断是否可以访问， $(a < b) \vee ((a == b) \wedge (t1 < t2))$ ，则t1访问

```
@Override
public void lock() {
    int i = ThreadID.get(); // 获取线程id
    flag[i] = true;
    int maximum = 0;
    for(int num : label) {
        if(num > maximum)
            maximum = num;
    }
    maximum = maximum + 1 ;
    for (int j = 0; j < n; j++) { // 看是否能加锁
        while((j != i) && flag[j] && ((label[i] < label[j]) || ((label[j] ==
label[i]) && j < i)));
        }
    }
}
```

- **运行结果**



## Part B

### • 实现思路

为了保证哲学家能正常吃饭，需要对五只筷子加锁，对于每位哲学家，每次尝试看自己左右两边的筷子是否可用，如果其他线程占用，那就选择退让，进入思考状态，如果拿到了一边的筷子另一边没拿到，那就放下拿到的筷子，也进入思考状态，拿到了左右两边的筷子就开始吃饭，然后放下左右两边的筷子释放资源。直接用锁模拟筷子，如果尝试加锁失败了，就说明该锁对应的筷子正在被使用；否则就加锁，然后使用筷子，结束后解锁

```
@Override
public void run() {
    while (true){
        System.out.println(num+"号哲学家"+" "+name+" "+"正在思考...");
        //模拟思考的过程
        try {
            Thread.sleep(time);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(num+"号哲学家"+" "+name+" "+"想来吃饭...");
        if (leftLock.tryLock()){//看是否能上锁
            try {
                System.out.println(num+"号哲学家"+" "+name+" "+"拿到了左边的筷子! ");
                if (rightLock.tryLock()){//看是否能上锁
                    try {
                        System.out.println(num+"号哲学家"+" "+name+" "+"拿到了右边的
筷子! ");

                        System.out.println(num+"号哲学家"+" "+name+" "+"开始吃
饭! ");

                        //模拟哲学家吃饭的过程
                        Thread.sleep(time);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    } finally {
                        System.out.println(num+"号哲学家"+" "+name+" "+"放下了右边的
筷子! ");
```

```

        rightLock.unlock();
    }
} else {
    System.out.println(num+"号哲学家"+" "+name+" "+"没拿到右边的筷子! 被迫思考...");
}
} finally {
    System.out.println(num+"号哲学家"+" "+name+" "+"放下了左边的筷子!");
    leftLock.unlock();
}
} else {
    System.out.println(num+"号哲学家"+" "+name+" "+"没拿到左边的筷子! 被迫思考...");
}
System.out.println(num+"号哲学家"+" "+name+" "+"思考...");
//模拟思考过程
try {
    Thread.sleep(time);
} catch (InterruptedException e) {
    e.printStackTrace();
}
}
}

```

## • 运行结果

程序一直在运行，这是部分结果的截图

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows the project structure with folders like 'src' and 'main'.
- Code Editor:** Displays the code for the philosopher threads, including the logic for picking up and putting down forks, and the simulation of the thinking process.
- Run Console:** Shows the output of the simulation, including messages like '正在思考...', '饿了, 想来吃饭...', '拿到了左边的筷子!', '拿到了右边的筷子!', '开始吃饭!', and '放下了左边的筷子!'.