OperatingSystemDesign

Dining Philosopher Assignment

19102091

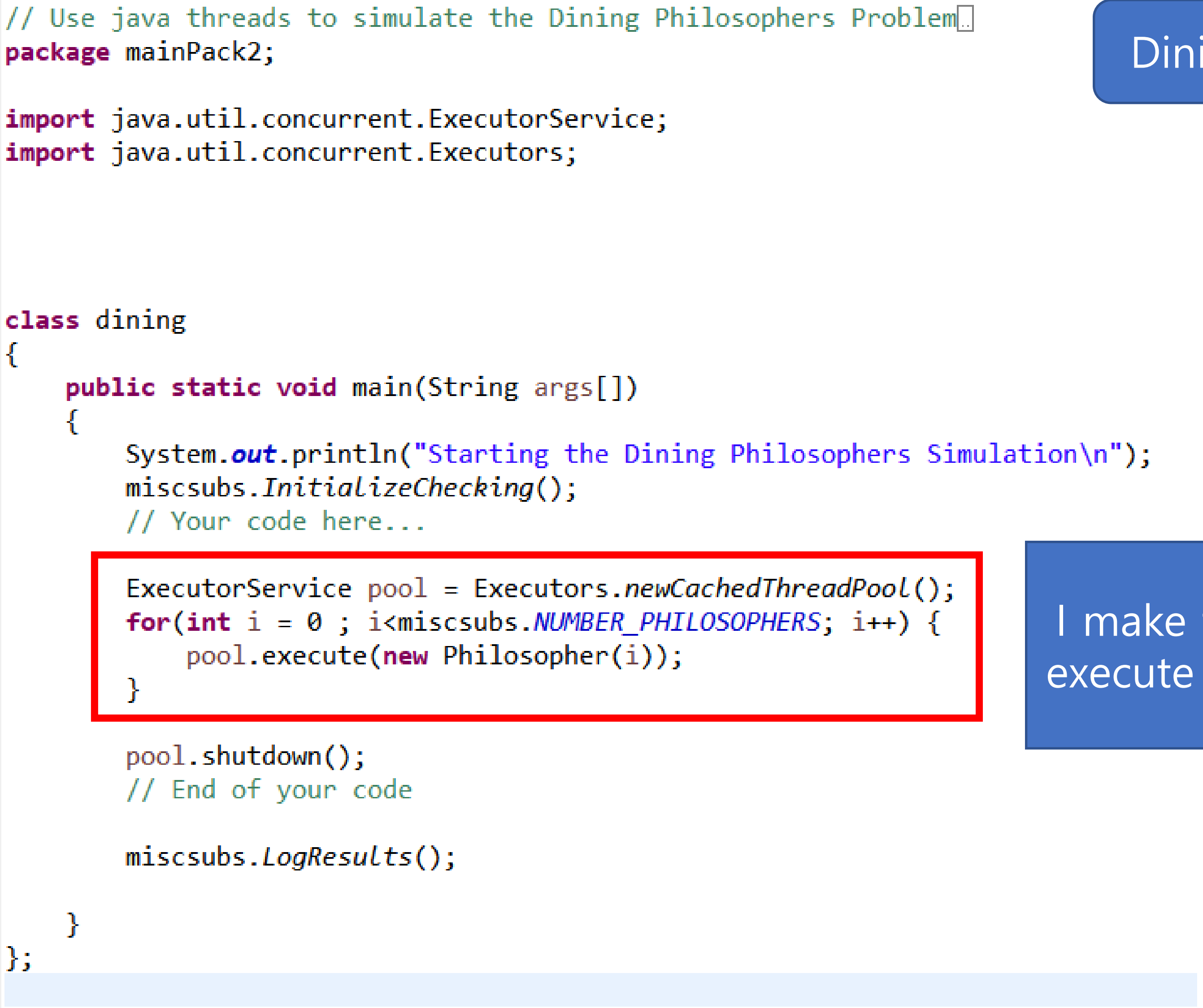
YoungHwanPhan

Before starting

I will explain different part of original code.

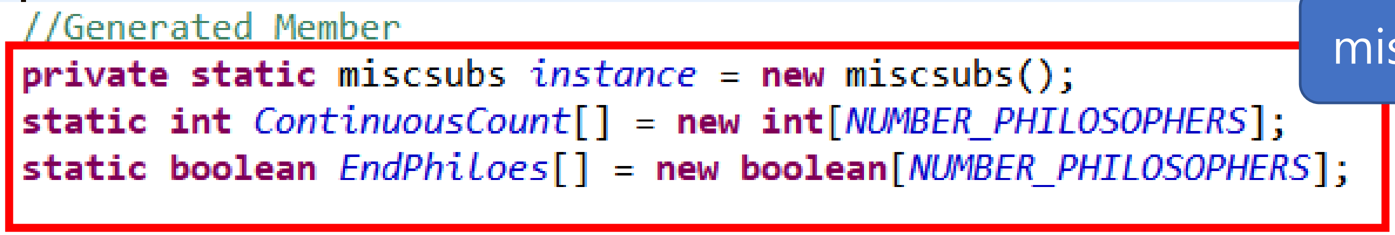
Additional explain will be commented in the code. Not here.

Dining class



```java
// Use java threads to simulate the Dining Philosophers Problem
package mainPack2;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;




class dining
{
    public static void main(String args[])
    {
        System.out.println("Starting the Dining Philosophers Simulation\n");
        miscsubs.InitializeChecking();
        // Your code here...

        ExecutorService pool = Executors.newCachedThreadPool();
        for(int i = 0 ; i<miscsubs.NUMBER_PHILOSOPHERS; i++) {
            pool.execute(new Philosopher(i));
        }

        pool.shutdown();
        // End of your code

        miscsubs.LogResults();

    }
};
```

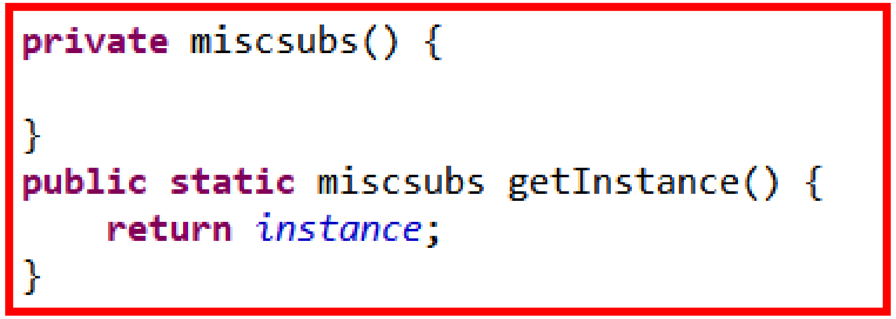
I make threadPool to create and execute thread almost same time



```
//Generated Member
private static miscsubs instance = new miscsubs();
static int ContinuousCount[] = new int[NUMBER_PHILOSOPHERS];
static boolean EndPhiloes[] = new boolean[NUMBER_PHILOSOPHERS];
```

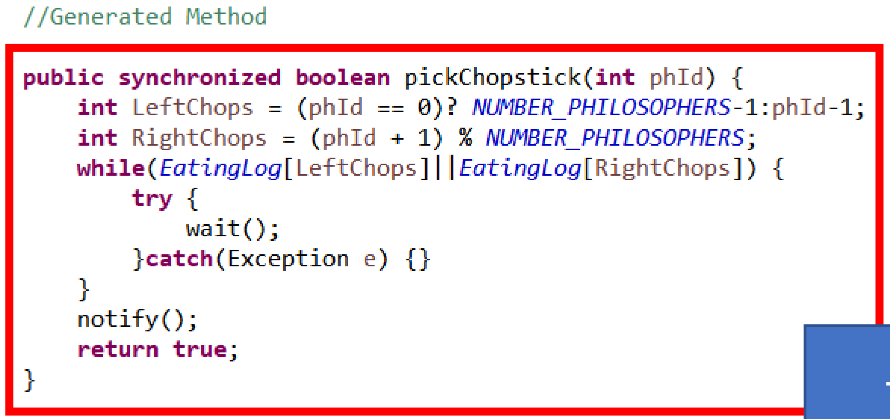
miscsubs class

I make global members to using under created method



```
private miscsubs() {

}
public static miscsubs getInstance() {
    return instance;
}
```

For using pickChopstick method, because this method is not static,
Thread have to use that, I make constructor as private and get its instance by getInstance method.
Miscsubs have to be one. Must not be made two or more.

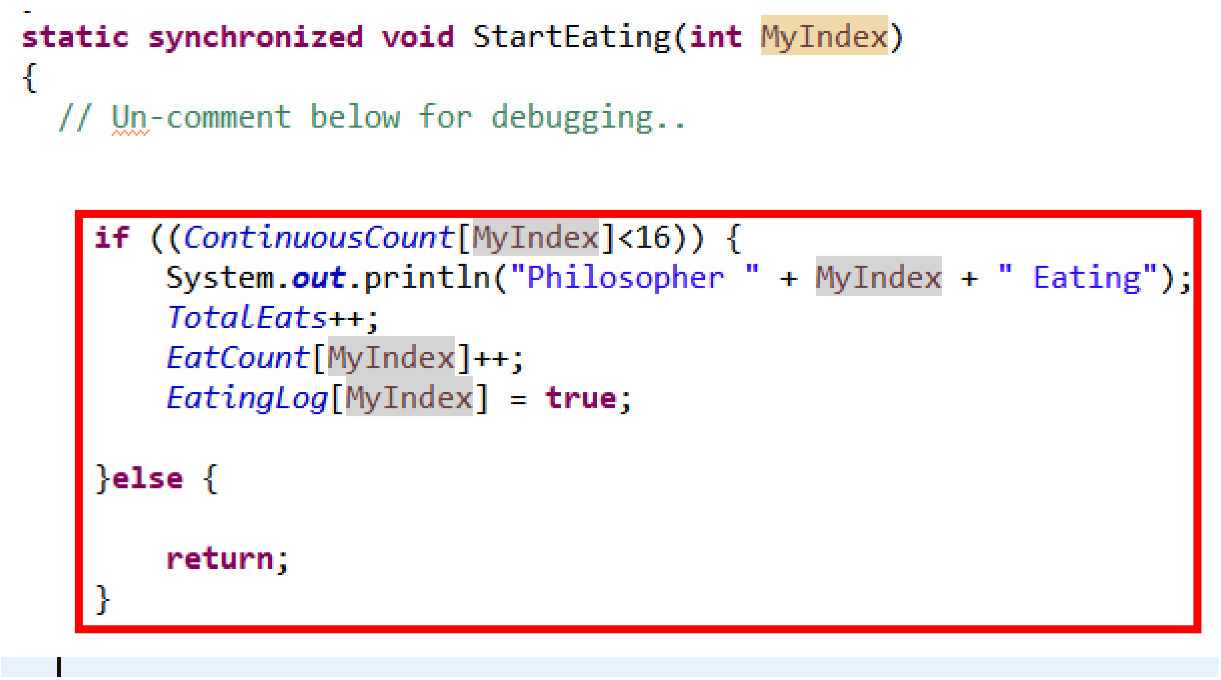


```
//Generated Method

public synchronized boolean pickChopstick(int phId) {
    int LeftChops = (phId == 0)? NUMBER_PHILOSOPHERS-1:phId-1;
    int RightChops = (phId + 1) % NUMBER_PHILOSOPHERS;
    while(EatingLog[LeftChops]||EatingLog[RightChops]) {
        try {
            wait();
        }catch(Exception e) {}
    }
    notify();
    return true;
}
```

This one!

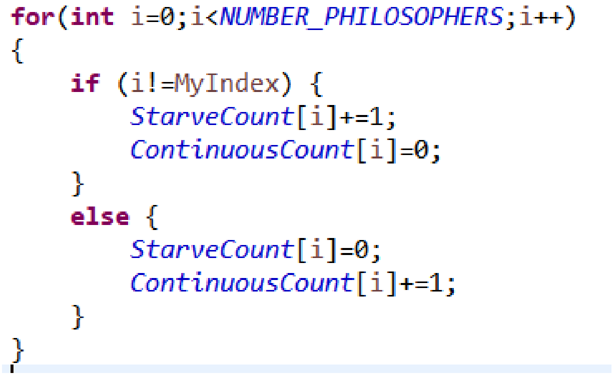
miscsubs class

StartEating Method



```
static synchronized void StartEating(int MyIndex)
{
  // Un-comment below for debugging..

    if ((ContinuousCount[MyIndex]<16)) {
        System.out.println("Philosopher " + MyIndex + " Eating");
        TotalEats++;
        EatCount[MyIndex]++;
        EatingLog[MyIndex] = true;

    }else {

        return;
    }
```

ContinuousCount mean how many this philosopher eat dinner.
For calculating, one philosopher eat continuously 16time,worst case, starvation is occurred.
So I set if condition to prevent it.

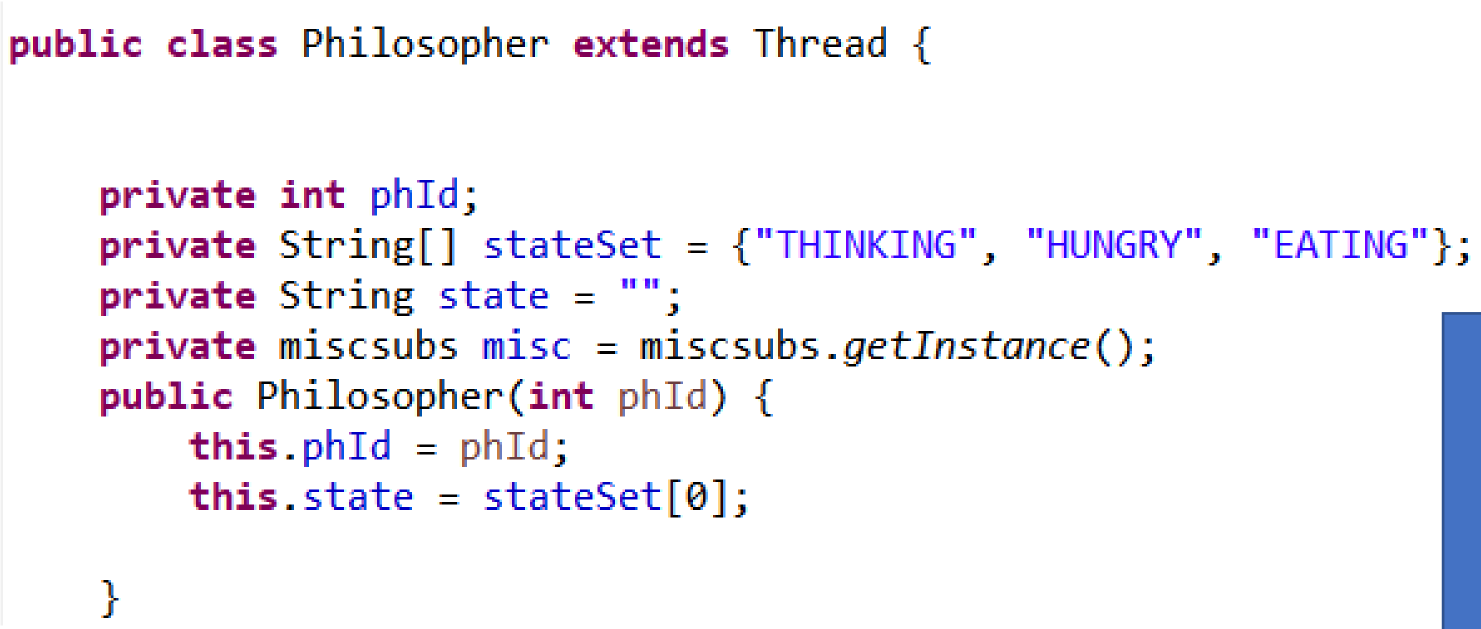


```
for(int i=0;i<NUMBER_PHILOSOPHERS;i++)
{
    if (i!=MyIndex) {
        StarveCount[i]+=1;
        ContinuousCount[i]=0;
    }
    else {
        StarveCount[i]=0;
        ContinuousCount[i]+=1;
    }
}
```

Update here.

Philosopher class

Constructor and Member

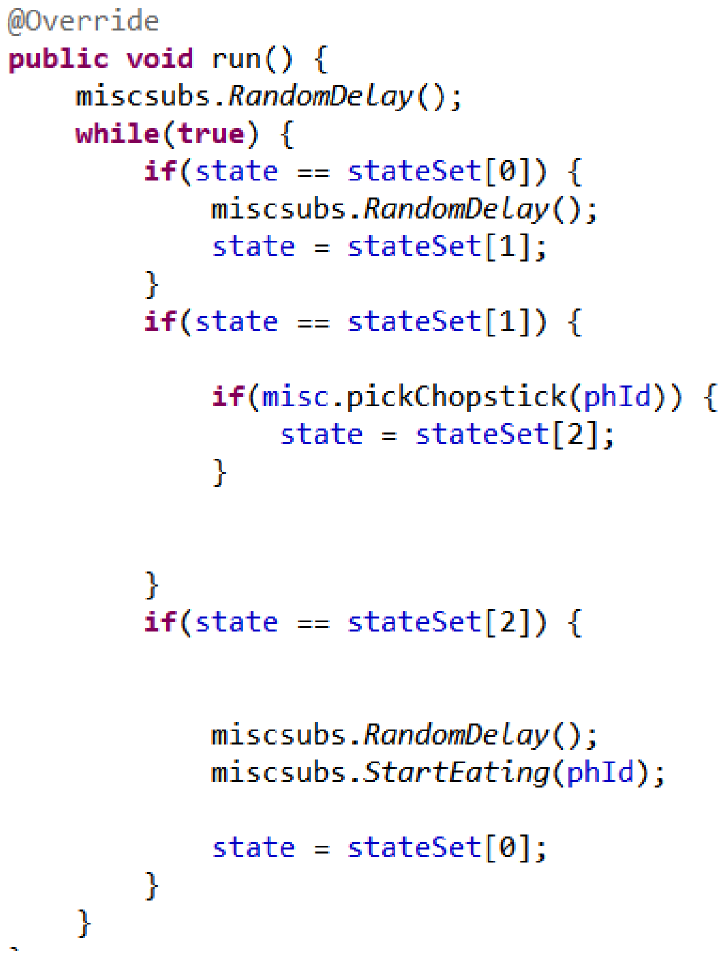


```java
public class Philosopher extends Thread {


    private int phId;
    private String[] stateSet = {"THINKING", "HUNGRY", "EATING"};
    private String state = "";
    private miscsubs misc = miscsubs.getInstance();
    public Philosopher(int phId) {
        this.phId = phId;
        this.state = stateSet[0];

    }
```

I make global members like that.
And set its Philosopher ID as phId
And default setting as THINKING.

Philosopher class

Run method



```java
@Override
public void run() {
    miscsubs.RandomDelay();
    while(true) {
        if(state == stateSet[0]) {
            miscsubs.RandomDelay();
            state = stateSet[1];
        }
        if(state == stateSet[1]) {

            if(misc.pickChopstick(phId)) {
                state = stateSet[2];
            }


        }
        if(state == stateSet[2]) {


            miscsubs.RandomDelay();
            miscsubs.StartEating(phId);

            state = stateSet[0];
        }
    }
```

I make philosopher follow routine
{THINKING, HUNGRY, EATING}

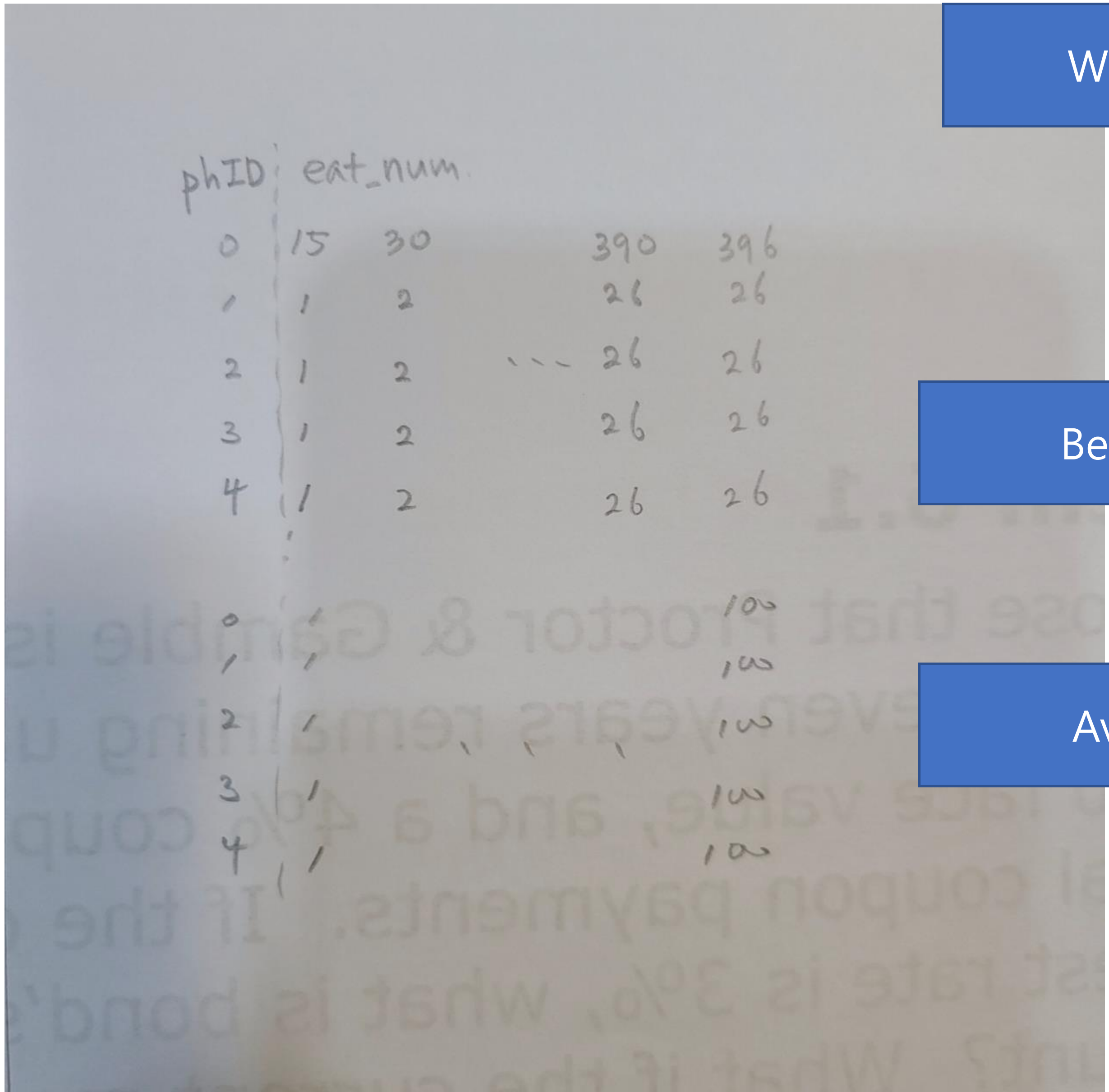
Philosopher try pick chopstick

Philosopher eat and return to THINKING

Performance

Now, I will explain how much this program satisfies thread fairness.

500 Time eat



phID	eat_num				
0	15	30		390	396
1	1	2		26	26
2	1	2	...	26	26
3	1	2		26	26
4	1	2		26	26
0	1				100
1	1				100
2	1				100
3	1				100
4	1				100

Worst case

The most eater eat 396 times
and The worst eater eat 26 times.
In this case , Error is 370.

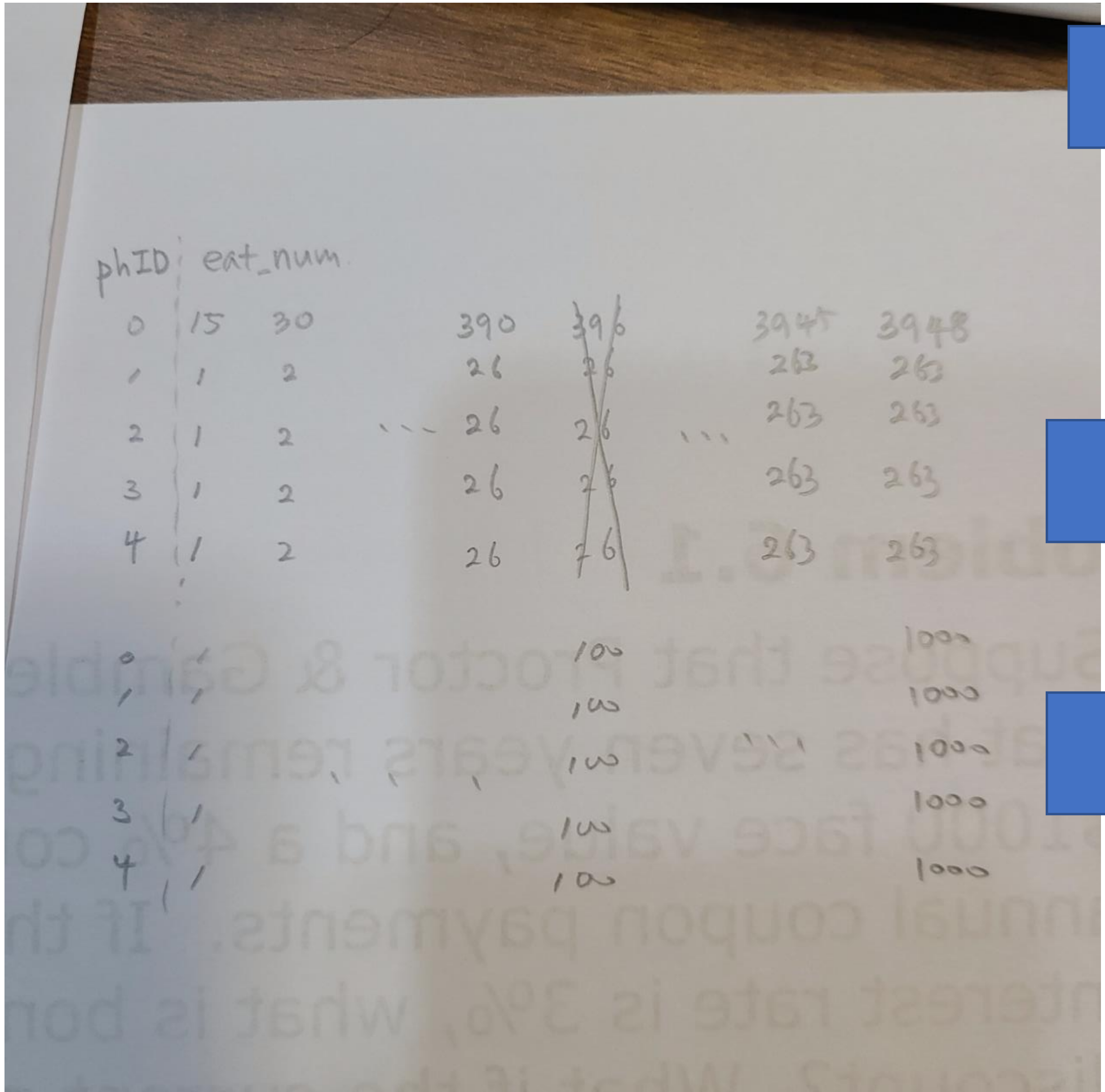
Best case

Error is 0.

Average

So average Error is 185.

5000 Time eat



phID	eat_num							
0	15	30		390	~~396~~		3945	3948
1	1	2		26	~~26~~		263	263
2	1	2	...	26	~~26~~	...	263	263
3	1	2		26	~~26~~		263	263
4	1	2		26	~~26~~		263	263

0	1		100		1000
1	1		100		1000
2	1	...	100	...	1000
3	1		100		1000
4	1		100		1000

Worst case

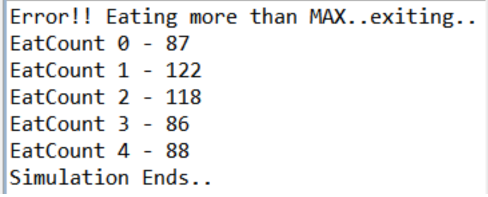
The most eater eat 3948 times and The worst eater eat 263 times. In this case , Error is 3685.

Best case

Error is 0.

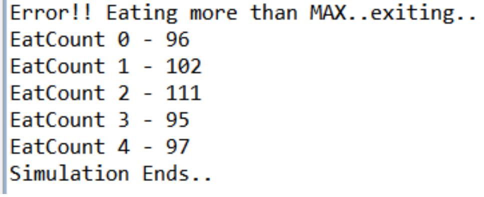
Average

So average Error is 1842.5.



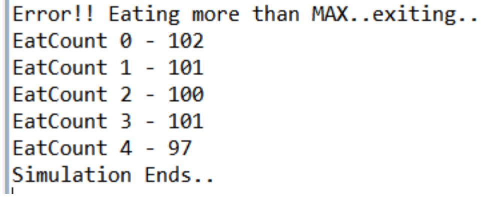
```
Error!! Eating more than MAX..exiting..
EatCount 0 - 87
EatCount 1 - 122
EatCount 2 - 118
EatCount 3 - 86
EatCount 4 - 88
Simulation Ends..
```

Error = 36



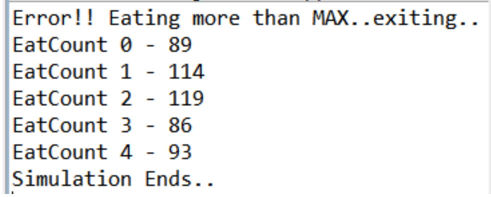
```
Error!! Eating more than MAX..exiting..
EatCount 0 - 96
EatCount 1 - 102
EatCount 2 - 111
EatCount 3 - 95
EatCount 4 - 97
Simulation Ends..
```

Error = 16



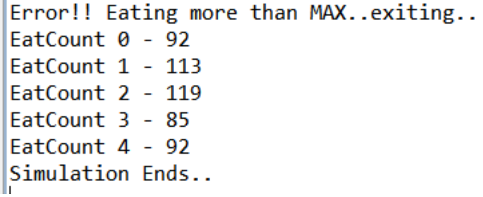
```
Error!! Eating more than MAX..exiting..
EatCount 0 - 102
EatCount 1 - 101
EatCount 2 - 100
EatCount 3 - 101
EatCount 4 - 97
Simulation Ends..
```

Error = 5



```
Error!! Eating more than MAX..exiting..
EatCount 0 - 89
EatCount 1 - 114
EatCount 2 - 119
EatCount 3 - 86
EatCount 4 - 93
Simulation Ends..
```

Error = 33



```
Error!! Eating more than MAX..exiting..
EatCount 0 - 92
EatCount 1 - 113
EatCount 2 - 119
EatCount 3 - 85
EatCount 4 - 92
Simulation Ends..
```

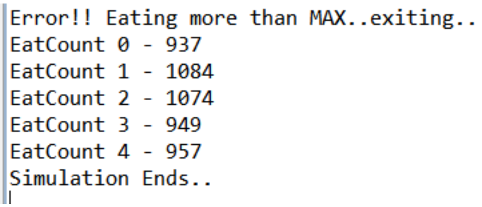
Error = 27

Average test Error = 23.4

Test unfairness rate = 23.4 / 185 = 12.6%

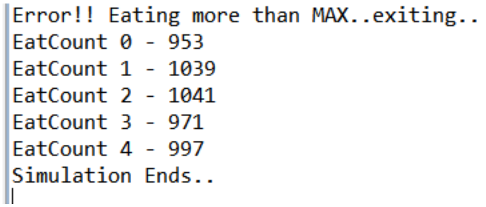
Test fairness rate = 87.4%

If do set maxTotal eat to 5000...



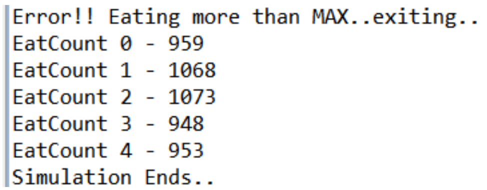
```
Error!! Eating more than MAX..exiting..
EatCount 0 - 937
EatCount 1 - 1084
EatCount 2 - 1074
EatCount 3 - 949
EatCount 4 - 957
Simulation Ends..
```

Error = 147



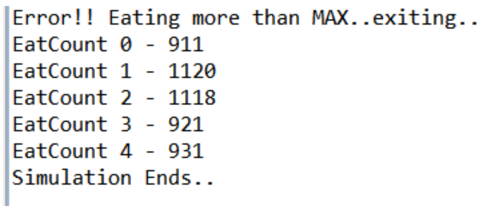
```
Error!! Eating more than MAX..exiting..
EatCount 0 - 953
EatCount 1 - 1039
EatCount 2 - 1041
EatCount 3 - 971
EatCount 4 - 997
Simulation Ends..
```

Error = 88



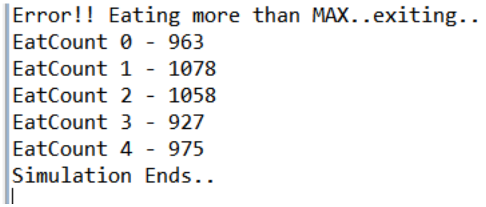
```
Error!! Eating more than MAX..exiting..
EatCount 0 - 959
EatCount 1 - 1068
EatCount 2 - 1073
EatCount 3 - 948
EatCount 4 - 953
Simulation Ends..
```

Error = 125



```
Error!! Eating more than MAX..exiting..
EatCount 0 - 911
EatCount 1 - 1120
EatCount 2 - 1118
EatCount 3 - 921
EatCount 4 - 931
Simulation Ends..
```

Error = 209



```
Error!! Eating more than MAX..exiting..
EatCount 0 - 963
EatCount 1 - 1078
EatCount 2 - 1058
EatCount 3 - 927
EatCount 4 - 975
Simulation Ends..
```

Error = 151

Average test Error = 144

Test unfairness rate = 144 / 1842.5 = 7.8%

Test fairness rate = 92.2%

I think this program designed well ☺

THANK YOU