Programming Assignment#3

< The Dining Philosophers Problem>

20102122 정효안

1. Dining Class

- Use java threads to simulate the Dining Philosophers Problem

```
package assignment3;

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();

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

        pool.shutdown();

        miscsubs.LogResults();

    }
};
```

I make threadPool to create and execute thread almost same time.

2. miscsubs class

```
package assignment3;

import java.lang.Thread;
import java.util.Random;

class miscsubs
{
    static int NUMBER_PHILOSOPHERS = 5;
    static int NUMBER_CHOPSTICKS   = 5;
    static int MAX_EATS = 500;
    static int TotalEats = 0;
    static int EatCount[] = new int[NUMBER_PHILOSOPHERS];
    static boolean EatingLog[] = new boolean[NUMBER_PHILOSOPHERS];
    static int StarveCount[] = new int[NUMBER_PHILOSOPHERS];

    static Random r;
```

I make global members to using under created method.

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

```
    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.

```
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!

```
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 16 time, 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.

3. Philosopher class

```java
package assignment3;

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];

    }

    @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 global members like that.

And set its Philosopher ID as phId

And default setting as THINKING.

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

Philosopher try pick chopstick

Philosopher eat and return to THINKING

4. Test

```
EatCount 0 - 103
EatCount 1 - 112
EatCount 2 - 92
EatCount 3 - 92
EatCount 4 - 102
Simulation Ends..
```

```
EatCount 0 - 101
EatCount 1 - 96
EatCount 2 - 95
EatCount 3 - 107
EatCount 4 - 102
Simulation Ends..
```

```
EatCount 0 - 87
EatCount 1 - 91
EatCount 2 - 113
EatCount 3 - 109
EatCount 4 - 101
Simulation Ends..
```

```
EatCount 0 - 105
EatCount 1 - 89
EatCount 2 - 114
EatCount 3 - 107
EatCount 4 - 86
Simulation Ends..
```

```
EatCount 0 - 85
EatCount 1 - 99
EatCount 2 - 102
EatCount 3 - 112
EatCount 4 - 103
Simulation Ends..
```

I will explain how much this program satisfies thread fairness.

Error : 20, 12, 26, 28, 27

Average Error: (20+12+26+28+27)/5=22.6

Error Rate: 22.6/185= 0.122=12.2%

Test Fairness Rate= 87.8%