**Dining Philosopher's problem**

**Subject = Operating Systems Design**

**Major = ITM**

**Student ID = 17102062**

**Name = Lee Yong Hoon(이용훈)**

Table

**1. Execution screenshots**

**2. Code analysis**

**1. dining.java - main()**

**2. philosopher.java - static variables and Constructor**

**3. philosopher.java - run()**

**1. check right chopstick whether it is used or not**

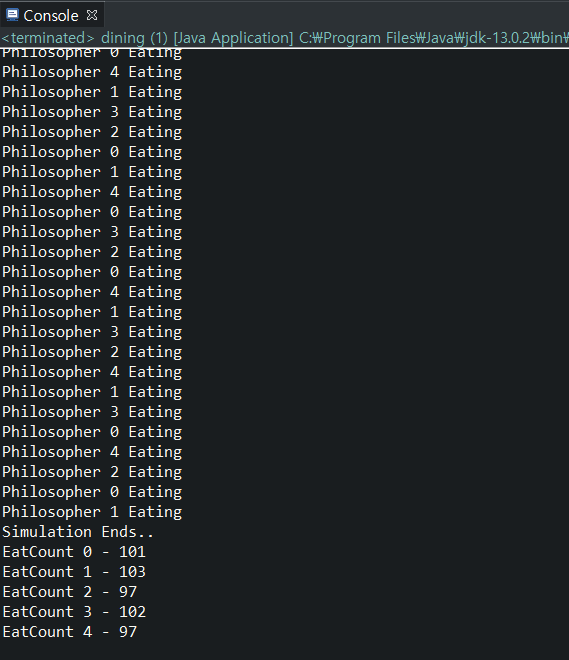
**2. check left chopstick whether it is used or not - avoid deadlock**

**3. After check both chopsticks, start eating**

**1. Execution screenshots**

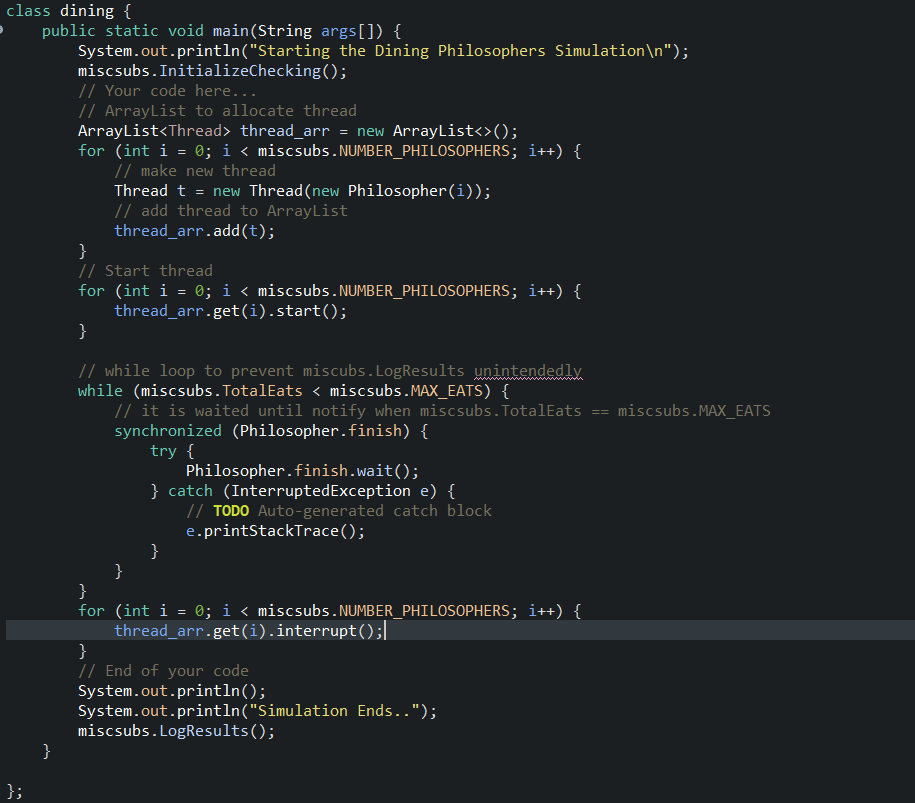
**In dining.java class, you run program**

**You can see result like below screenshot**



**2. Code analysis**

**1. dining.java - main()**



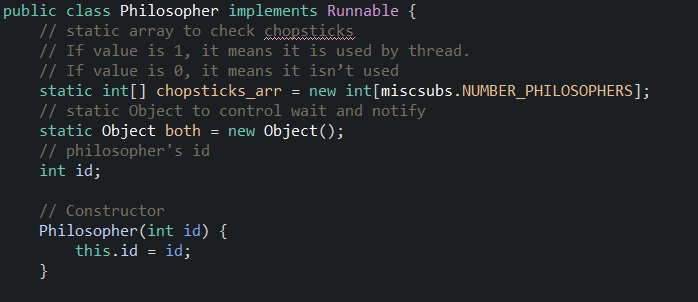
**I make 5 threads by using for loop.**

**And start thread.**

**Wait in while loop and if miscsubs.TotalEats is equal to miscsubs.MAX\_EATS then wakeup.**

**Interrupt all threads and print LogResult**

**2. philosopher.java - static variables and Constructor**



**I use static array named ‘chopsticks\_arr’ to check chopsticks**

**If value is 1, it means it is used by thread.**

**If value is 0, it means it isn’t used**

**I use static object both.**

**It is used to use synchronize function in Java**

**Int id is philosopher’s id and it id is allocated by Constructor**

**3. philosopher.java - run()**

**1. check right chopstick whether it is used or not**



**Firstly, delay to thread by using ‘micsubs.RandomDelay();’**

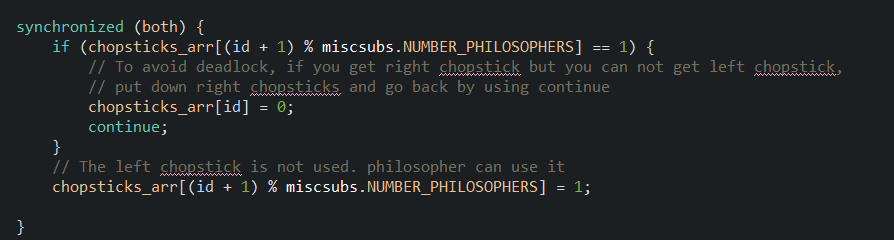
**Use ‘synchronized(both)’ to use wait**

**Check whether right chopstick is used.**

**If used, philosopher should wait. If not used, philosopher can use right chopstick.**

**And after wait, philosopher can use right chopstick**

**2. check left chopstick whether it is used or not - avoid deadlock**



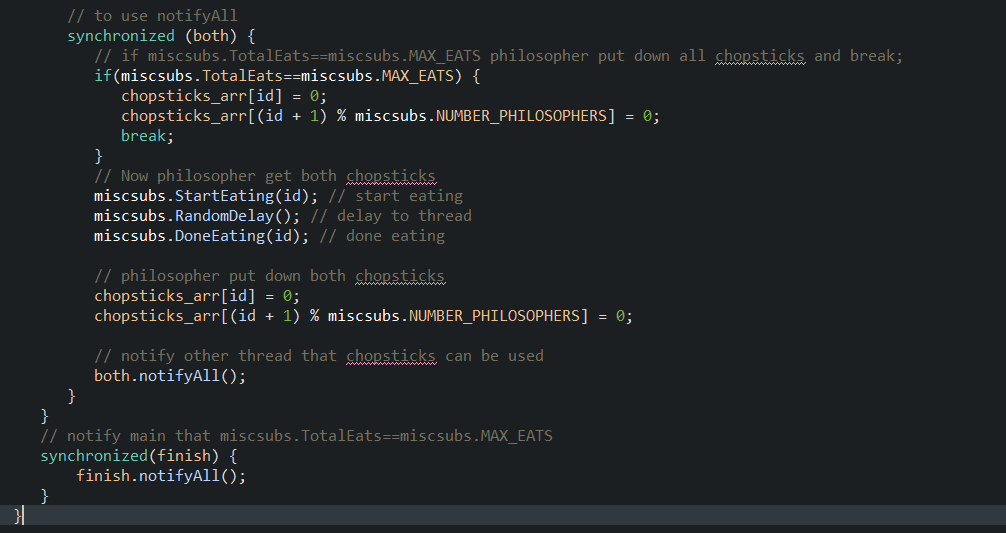
**Check whether left chopstick is used.**

**If left chopstick is used, philosopher put down left chopstick and go back by using continue**

**To avoid deadlock, philosopher should put down both chopsticks. One of the deadlock condition Hold and Wait can be avoided by using this way.**

**And if chopstick is not used, philosopher can use it.**

**3. After check both chopsticks, start eating**



**Use ‘synchronized(both)’ to use notifyAll.**

**Now philosopher get both chopsticks right and left.**

**Philosopher start Eating and give delay to thread.**

**Philosopher finish eating.**

**After eating, philosopher put down both chopsticks right and left.**

**And notify other threads that, chopsticks can be used.**

**if miscsubs.TotalEats==miscsubs.MAX\_EATS philosopher put down all chopsticks and break**

**and then notify to main that miscsubs.TotalEats == miscubs.MAX\_EATS.**