

The Dining philosophers Problem

In this project we are trying to solve the dining philosophers problem, the problem is that there is N philosophers who are sitting at a round table and there is only N chopsticks on the table, each chopstick is shared between two philosophers sitting next to each other, and they can only eat if they have 2 chopsticks and after eating returning the chopsticks each in its proper place, and while they are not eating they think, they just think for some time or eat.

Some ways of solving this problem is to make the philosophers with odd numbers start taking the right chopstick then the left one and the ones with the even numbers to start taking the left chopstick then the right one, another way to solve the problem by checking if both of the chopsticks available and take them or don't take any of them, and another way using busy waiting but in this solution it keeps the CPU active doing nothing just to check if it can take a chopstick or not.

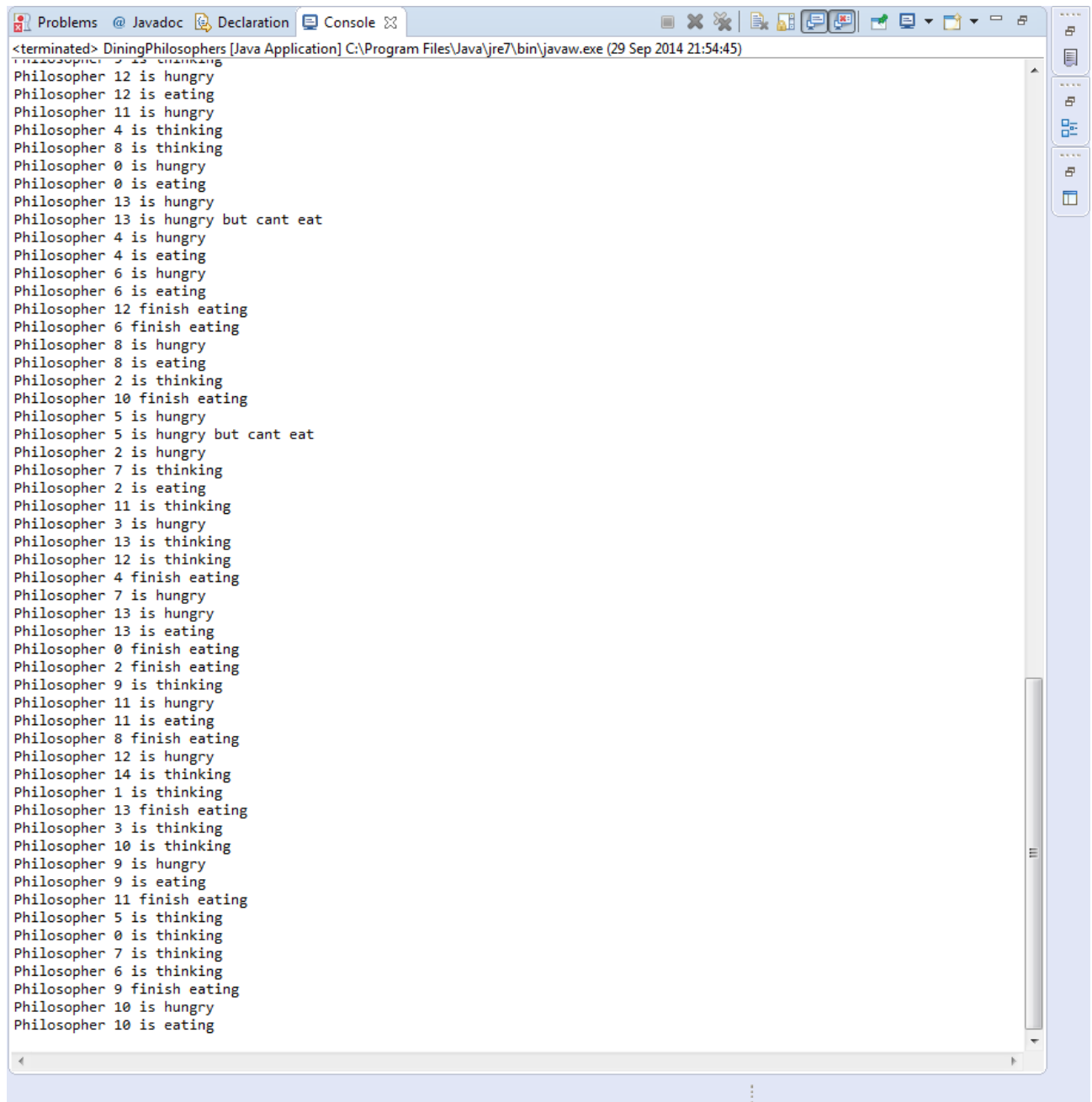
In my solution I tried to combine some of the solutions together to get better solution, which I combined the number of philosophers if it is even or odd what chopstick to start with and limit the busy waiting to try for limited times and if could not take two chopsticks then drop the one you have so others can use it and try again.

Solution

I try to solve the problem and write the code from scratch, I used synchronized methods for taking and dropping the chopsticks, and for each philosopher with even number he try to take the left chop stick then the right one, but for philosophers with odd number they try to take the right chopstick then the left one, and each philosopher who fail to take the chopsticks for 5 times then he drop the chopstick with him and sleep form some time and then try again to get the chopsticks and if he could not eat after 5 tries in which he tried 5 times in each try then he will starve to death.

A try is counted as trying to get the chopsticks for 5 times and not getting any then sleep for some time, it's not the same as busy waiting but I see this is a good approach because many times while checking it might be some other philosopher is dropping his chopsticks so the others will have the opportunity to get them exactly after the other one drop them and before sleeping and not taking the chopstick while he can.

In this solution sometimes it might occur that some philosophers will starve to death but it is rare to happen it actually did not happened with me while testing the code, but it could happen.



```
<terminated> DiningPhilosophers [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (29 Sep 2014 21:54:45)
Philosopher 5 is thinking
Philosopher 12 is hungry
Philosopher 12 is eating
Philosopher 11 is hungry
Philosopher 4 is thinking
Philosopher 8 is thinking
Philosopher 0 is hungry
Philosopher 0 is eating
Philosopher 13 is hungry
Philosopher 13 is hungry but cant eat
Philosopher 4 is hungry
Philosopher 4 is eating
Philosopher 6 is hungry
Philosopher 6 is eating
Philosopher 12 finish eating
Philosopher 6 finish eating
Philosopher 8 is hungry
Philosopher 8 is eating
Philosopher 2 is thinking
Philosopher 10 finish eating
Philosopher 5 is hungry
Philosopher 5 is hungry but cant eat
Philosopher 2 is hungry
Philosopher 7 is thinking
Philosopher 2 is eating
Philosopher 11 is thinking
Philosopher 3 is hungry
Philosopher 13 is thinking
Philosopher 12 is thinking
Philosopher 4 finish eating
Philosopher 7 is hungry
Philosopher 13 is hungry
Philosopher 13 is eating
Philosopher 0 finish eating
Philosopher 2 finish eating
Philosopher 9 is thinking
Philosopher 11 is hungry
Philosopher 11 is eating
Philosopher 8 finish eating
Philosopher 12 is hungry
Philosopher 14 is thinking
Philosopher 1 is thinking
Philosopher 13 finish eating
Philosopher 3 is thinking
Philosopher 10 is thinking
Philosopher 9 is hungry
Philosopher 9 is eating
Philosopher 11 finish eating
Philosopher 5 is thinking
Philosopher 0 is thinking
Philosopher 7 is thinking
Philosopher 6 is thinking
Philosopher 9 finish eating
Philosopher 10 is hungry
Philosopher 10 is eating
```

Figure 1: sample output

Sample output of the code running in Figure 1 above showing that some of the philosophers is eating some cannot eat and some is thinking, and luckily none of the philosophers starve to death all are alive and eating.