The Dining Philosophers problem

Diaconescu Bogdan-Florin
CR 3.2 A, Anul 3

December 22, 2019

1 Problem statement

There is a group of philosophers (usually 5) who eat together at a round table. There are forks placed between the philosophers. Philosophers spend their time either thinking or eating. In order to eat, a philosopher must pick up exactly two forks, one on his immediate left, and the other on his immediate right. When he is done eating, he will put his forks down so that his neighbors may use them, and he thinks again.

2 Implementation/Solution

The programming problem is to construct a simulation that will allow philosophers to move between their eating and thinking states while properly controlling the forks.

A typical solution has each philosopher doing something like this:

Algorithm 1

```
1: while true do
2:    busy thinking
3:    wait for left fork
4:    wait for right fork
5:    busy eating
6:    drop left fork
7:    drop right fork
8: end while
   =0
```

To solve this problem, I created a generic class "Channel ¡T¿" that has 2 functions: "synchronized void send(T mes)" and "synchronized T receive()". I defined a int variable ready that it's used to indicated the message's status. When ready = 0, the message can be sent, then the ready variable is incremented. When ready = 1, the message can be receive, then the ready variable is decremented. Each processes announces the other processes when the message has been sent or received using the function notifyAll().

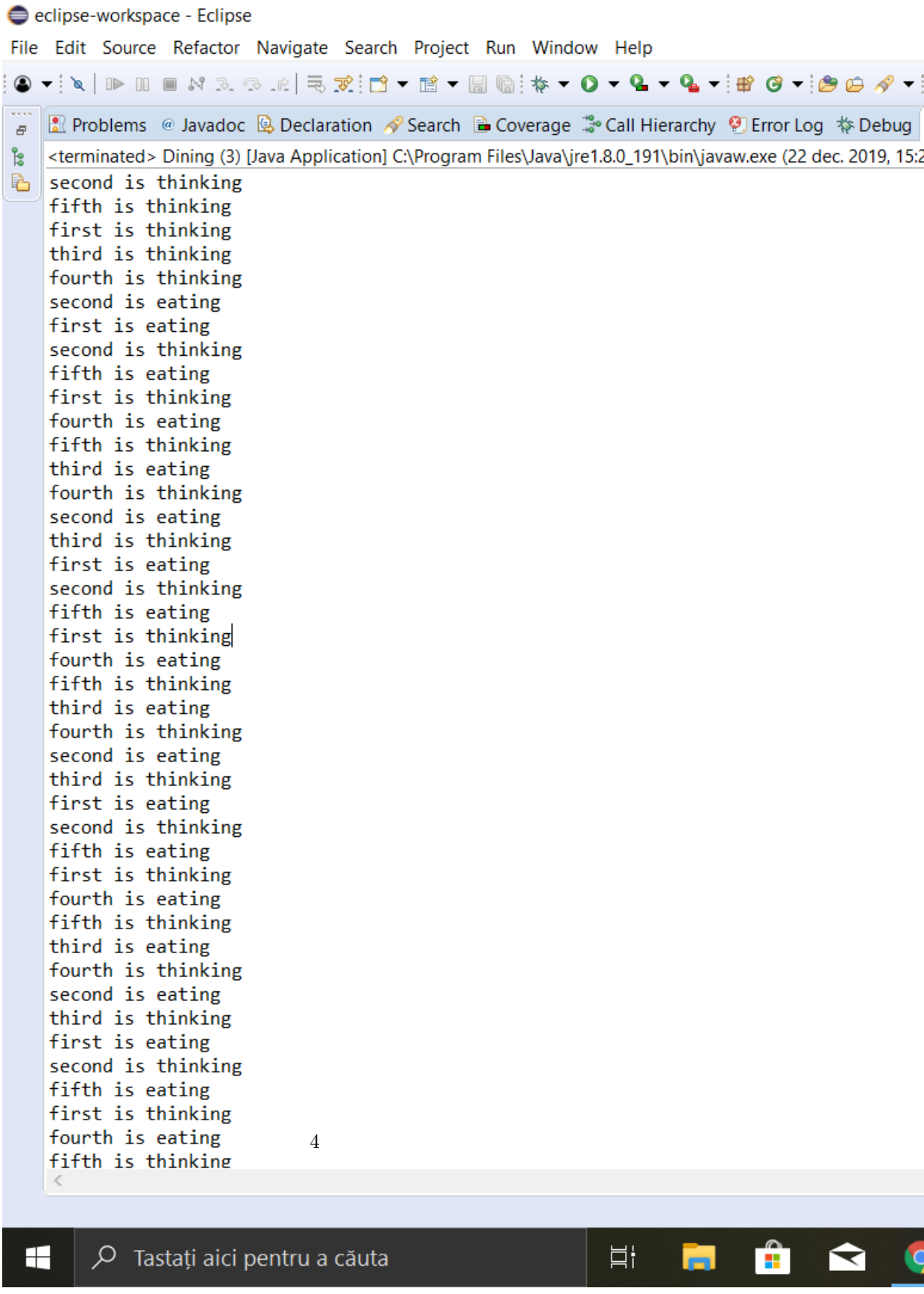
Fork class extends Thread and has a function "void run()". In this function, for each element of forks array, call send(true) and receive() in an infinite while loop. - Forks are implemented trough independent processes which communicate with the philosophers using channels. A philosopher communicates with a fork i using the channel Forks i (that represents the left fork) and with a fork i+1 (right fork) using the channel Forks i+1.

In Philosopher class we create the Philosopher. When both forks are available, message transmitted through the channel using the method get, the philosopher eats.

Dining class corresponds to the Main class, where the philosopher are instantiated. Each philosopher gets a name, the index of the right and left forks

and a forks instance which represents the channel. The threads are launched in execution.

3 Experimental data and outputs


eclipse-workspace - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Search Coverage Call Hierarchy Error Log Debug
<terminated> Dining (3) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (22 dec. 2019, 15:2
second is thinking
fifth is thinking
first is thinking
third is thinking
fourth is thinking
second is eating
first is eating
second is thinking
fifth is eating
first is thinking
fourth is eating
fifth is thinking
third is eating
fourth is thinking
second is eating
third is thinking
first is eating
second is thinking
fifth is eating
first is thinking
fourth is eating
fifth is thinking
third is eating
fourth is thinking
second is eating
third is thinking
first is eating
second is thinking
fifth is eating
first is thinking
fourth is eating
fifth is thinking
third is eating
fourth is thinking
second is eating
third is thinking
first is eating
second is thinking
fifth is eating
first is thinking
fourth is eating
fifth is thinking
Tastați aici pentru a căuta

4 Conclusion

I solve this problem using channels. In other words, the forks communicate with the philosophers trough the channels. When a fork sends a message it means that it is available. When the philosopher sends a message it means that the fork was released.