THE DINING PHILOSOPHERS PROBLEM AND METHODS FOR SOLVING IT

Hlavcheva D.M.[1], Yaloveha V.A.[2]
[1]National Technical University «Kharkiv Politechnic Institute», [2]V.N. Karazin Kharkiv National University, Kharkiv

As we know, an object can have synchronized methods or other forms of locking that prevent tasks from accessing that object until the mutex is released. Tasks can become blocked. Thus it’s possible for one task to get stuck waiting for another task, which in turn waits for another task, and so on, until the chain leads back to a task waiting on the first one. Then we get a continuous loop of tasks waiting on each other, and no one can move. This is called deadlock.

The dining philosophers problem, invented by Edsger Dijkstra [1], is the classic demonstration of deadlock. The basic description specifies five philosophers. They spend some time thinking and a part of their time eating. While they are thinking, they don’t need any shared resources, but they eat using a limited number of chopsticks [2]. Clearly, each philosopher will require two chopsticks in order to eat. Philosophers have only five chopsticks (more generally, the same number of chopsticks as philosophers). When a philosopher wants to eat, that philosopher must pick up the chopstick to the left and the one to the right. If the philosopher on either side is using a desired chopstick, our philosopher must wait until the necessary chopsticks become available. If the philosophers spend very little time thinking, they will all be competing for the chopsticks while they try to eat, and we can get deadlock situation, because philosopher is trying to pick up its chopsticks in a particular sequence: first right, then left.

It is considered several ways of solving problem [3]. If the last philosopher is initialized to try to get the left chopstick first and then the right, that philosopher will never prevent the philosopher on the immediate right from picking up their chopstick. Another solution is based on numbering of chopsticks and philosophers took the first chopstick with the smallest number. On the other hand, we can require one person less than the number of chopsticks to sit around the table at the same time [4]. All solutions were programmed on JAVA and C++ languages. We used synchronized methods to prevent tasks from accessing objects applying Brian’s Rule of Synchronization and ExecutorService class to generate Thread.

References:

1. Dijkstra E.W. Some beautiful arguments using mathematical induction / Dijkstra E.W. // Acta Informatica. – 1980. – Vol. 13, No. 1. – P. 1-8.

2. Eckel B. Thinking in Java 4-th / Eckel B. – Massachusetts: Prentice Hall, 2006. – Vol. 8. – 1079 p.

3. Goetz B. Java concurrency in practice / Goetz B., Peierls T. – Pearson Education, 2006. – 234 p.

4. Lea D. Concurrent programming in Java: design principles and patterns / Lea D. – Addison-Wesley Professional, 2000. – 318 p.