# 

*POSTMORTEM*

# Catherine austria

Dining Philosophers

COSC 460 Operating systems

project 01



# 

*overview*

The dining philosopher’s problem

This concept is associated with the classic concurrency problem dealing with synchronization. In the problem there are five philosophers who are concerned with thinking, becoming hungry and eating. The problem is that although each philosopher has one chopstick, they all have to eat with a shared chopstick without risking starvation. And so the process would be for one philosopher to pick up the chopstick and then eat, then put down or return the chopstick to let other philosophers eat.





*Design choices*

The Monitor solution – server class

The use of the monitor is to observe all the states of the philosopher and make sure that every one of the philosophers (~~zenyattas~~) will be able to eat.

For this the project uses the lock, reentrantlock and condition classes in java 8 to achieve making the monitor class (for the project’s purpose is called the server).

Philosophers class

This class’s job is to constantly make the philosopher do three actions which is:

Think hungry eat

At the beginning of the program the philosopher is set to think.



driver class

This class initializes the philosophers and the monitor-server. It constantly loops to allow infinite executions.

state

This class has the Thinking, hungry and eating state actions for the philosophers. I had planned to make it spout philosophical nonsense while the philosopher was thinking, but I got sick and didn’t feel up to it.

Dining server

To be honest I have no idea why this was in the requirements. It may have been a design that I did not fully understand why the need of the “interface” or what relevance or difference that it makes. I’ll probably learn why in the future and I’ll look back to my code and think it is cancerous just because of how bad it is.

Cheers.