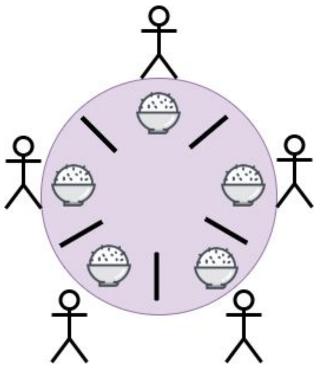
Programming Assignment 3

Read the entire assignment before starting your work.

Introduction

The Dining Philosophers problem is one of the classic problems used to describe synchronization issues in a multi-threaded environment and illustrate techniques for solving them. In this assignment, you will implement a solution to this well-known problem using what we learn in class about multithreading and parallel programming.

The Problem



The diagram above represents the problem. There are five silent philosophers sitting around a circular table, spending their lives eating and thinking.

There are five chopsticks for them to share and to be able to eat, a philosopher needs to have two chopsticks in his hands. After eating, he puts both of them down and then they can be picked by another philosopher who repeats the same cycle.

Implementation

In this assignment, you are provided a zip file named COP3809_PROGRAM3_SYY_XXXX.zip; where YY is 01, 02 or 03, and XXXX is student id. In this zip file is the following:

Chopstick.java
Philosopher.java
DiningPhilosopher.java

Chopstick.java
This class hold information about a Chopstick (whether or not the chopstick is in use, the id, etc)

Must have the following variables:

- id (int) - private
- inUse (bool) - private - indicates whether or not the chopstick is in use

Must have the defined constructor:
- Chopstick(int id)
    - Will set the id and default inUse to false

Will implement the following synchronized methods:
- release() - void
    - sets inUse to false
    - Out prints to the terminal that the chopstick has been released eg “Chopstick 1 has been set down.”
- take() - void
    - sets inUse to true
    - Out prints to the terminal that the chopstick has been released eg “Chopstick 1 has been picked up.”
- isInUse() - boolean
    - returns inUse
- getId() - int
    - returns id

Philosopher.java

Must have the following variables:

- id (int) - private
- leftChopstick(Chopstick) - private
- rightChopstick(Chopstick) - private

Must have the defined constructor:

- Philosopher(int id, Chopstick leftChopstick, Chopstick rightChopstick)
    - Will set the id, leftChopstick, and rightChopstick

Will implement the following methods:
- run() - void
    - Overrides runnable, initial start into the thread
    - Executes eat() method
- eat() - void

    - If the left and right chopstick are not in use, pick up chopsticks and outprint "Philosopher X is eating" X being the id of the philosopher
    - Then have the thread sleep for a random time between 0 and 3 secs (i.e. (int)(Math.random()*3000))
    - Release the chopsticks
    - Call think()
    - This method must handle all exceptions (no throws declaration should be added)
- think() - void
    - Outprint "Philosopher X is thinking" X being the id of the philosopher
    - Then have the thread sleep for a random time between 0 and 3 secs (i.e.(int)(Math.random()*3000))
    - This method must handle all exceptions (no throws declaration should be added)

DiningPhilosophers.java
This is the entry point of the program. In this class, the main method must be implemented.

- 5 Chopsticks must be added to an ArrayList of chopsticks
- 5 Philosophers must be added to an ArrayList of philosophers
- 5 Philosophers must be initialize (setting the id, and the right and left chopsticks)
- Then a thread must be created for each philosopher and started.

Make sure to check your code and insure no deadlocks occur.

Submission
Zip up all the folders and files under dinner package, the name of the zip file

MUST be COP3809_PROGRAM3_SYY_XXXX.zip; where YY is 01, 02, or 03 and

XXXX is student id.