CSE505 – Spring 2019

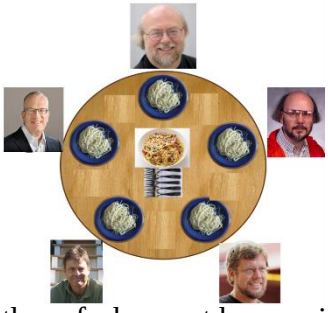
Assignment 3

Due Date: **April 15 (11:59 pm)**

You may work in pairs for this assignment.

There are two problems in this assignment. Problem 2 will be posted after a couple of days.

Problem 1: This is an elaboration of the ***Dining Programmers*** problem discussed in Lecture 15. As discussed in class, this problem is similar to the classic *Dining Philosophers* problem but there is an important difference which necessitates a different approach to concurrency control.



Shown below are five famous programmers around a table. These programmers being more practical than philosophers decided that all five forks shall be placed at the center of the table, as shown in the picture. Each programmer cycles through three states:

$$\{\textit{coding}\ ;\ \textit{hungry}\ ;\ \textit{eating}\ \}^{+}$$

Prior to eating, a programmer must acquire two forks in order to help himself to the spaghetti located at the center of the table. Furthermore, these forks must be acquired one at a time. After eating, the programmer must return both forks back to the center of the table.

By locating the forks at the center of the table it is possible for any two programmers to be eating simultaneously, *including adjacent* programmers. In this way the programmers enjoy more flexibility and concurrency than the philosophers.

Being savvy programmers, they know that an uncoordinated approach to acquiring forks will result in a deadlock – each programmer could acquire one fork each and would end up waiting indefinitely for the second. Hence, the programmers agree to adopt the following protocol:

1. No programmer may acquire a fork if this is the only remaining fork on the table ***and*** it is the ***first*** fork that the programmer is trying to acquire.
2. However, a programmer may acquire the only remaining fork on the table if it is the ***second*** fork that the programmer is trying to acquire.

This problem requires you to develop a solution to the *Dining Programmers* problem in Java. The file `DiningProgrammers.java` gives the outline of the solution. Your task is to complete the definition of methods `pickup` and `drop` in class `Forks` so that no deadlock can arise and maximum concurrency is permitted. Develop your solution using `synchronized` methods and the `wait-notify` constructs of Java.

Run the program under JIVE and save the execution trace in a file `DP.csv`. Load this file in the *Finite State Machine plugin* and obtain two state diagrams as follows:

1. Obtain a diagram for the field `Forks:1->n`. Save the diagram in the file `forks.svg`.

2. Obtain a diagram for the fields `Programmer:1->state` … `Programmer:5->state`.

Draw the diagram ***after*** performing abstraction with respect to the condition:

`=E,=E,=E,=E,=E`

Note: Do not put spaces after the commas. Save the resulting state diagram in file `states.svg`.

A correct implementation of the *Dining Programmers* problem should have a maximum of **16**[1] states (excluding the initial 'null' states) in the abstracted FSM diagram and no state of the FSM will have more than two components with value `E`.

You are free to experiment with different sleep times and number of iterations so that you obtain a state diagram with 16 states. Do not modify other parts of the code.

A screen-cast showing how to install and run the FSM plugin will be uploaded shortly.

WHAT TO SUBMIT:

Make a directory called `A3_Problem1_UBITId` if working solo or make a directory called `A3_Problem1_UBITId1_UBITId2` if working as a pair (give UBITId's in alphabetic order).

Put in this directory the files `DiningProgrammers.java, DP.csv, forks.svg`, and `states.svg`.

Compress the directory, and submit it using the `submit_cse505` command.

[1] An abstracted state is one in which every component of the state vector has the value `E` or `~E`. The notation `~E` stands for "not eating," i.e., coding or hungry. There are five abstracted states with exactly one `E`; there are 10 abstracted states with 2 `E`'s; and there is one state with 0 `E`'s. Hence the total number of abstracted states for the given abstraction is 16.