Practical Task 2

Aim: *Practical experience with synchronization concepts*

Begin Date: *November 22, 2016*

End Date: *December 6, 2016*

Submission: *an archive file (zip) that includes all files relevant to execute and test your program (please note that you should not include the Java Runtime Environment (JRE)) should be submitted through MyMoodle; if the submitted program does not run (execute) using the instructions provided in the ReadMe.txt file, your solution is not accepted.*

Instructions: *you are allowed and encouraged to use the literature recommended in the course; use of other literature is also encouraged; assignment must be completed independently and without the help of other colleagues or the teacher.*

Problem definition

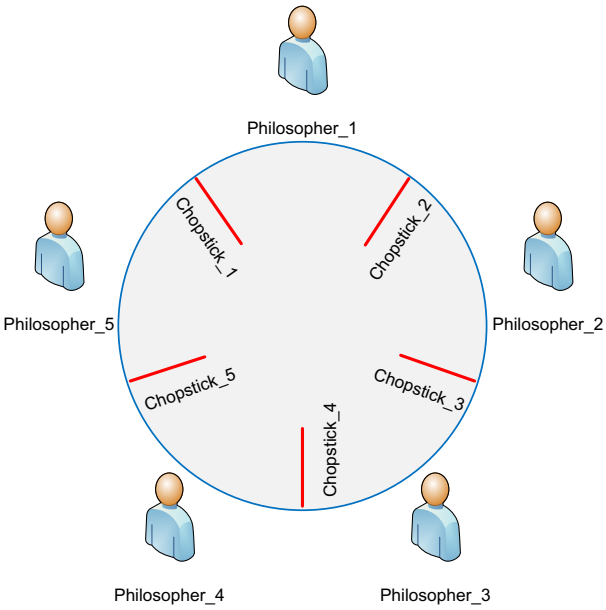

Figure 1. Dining Philosophers

Develop a Java application that simulates the Dining Philosophers problem (see Chapter 6 of the book[1]). Five philosophers sit around a table and in front of each philosopher there is a bowl of food (see Figure 1). A philosopher can be in three states: EATING, THINKING, or HUNGRY. There are five chopsticks available. A hungry philosopher needs two chopsticks (his left and right chopstick) for eating. If a chopstick is used by a neighbor, then the philosopher must wait until the chopstick is released. The philosopher first picks up the left chopstick, and thereafter the right chopstick. Initially all five philosophers are in THINKING state. Time for THINKING and EATING is generated randomly from a discrete uniform distribution [1, 10]. Basically, a philosopher is in THINKING or EATING state for 1, 2, 3, …, or 10 time units (milliseconds); one of these 10 values is selected in random[2] to simulate thinking or eating time.

You should:

1. Create a class *DiningPhilosopher* that among others contains:
    a. A list *(ArrayList)* of Philosophers
    b. A list *(ArrayList)* of Chopsticks
    c. A method *getPhilosophers*() that returns the list of philosophers
    d. A method *setSimulationTime(int millis)* to set the simulation time (that indicates for how many milliseconds the simulation will run). The default value should be set to 10000 milliseconds (or 10 seconds).

[1] Operating System Concepts, 9th Edition, by Abraham Silberschatz, Peter B. Galvin, Greg Gagne
[2] http://docs.oracle.com/javase/7/docs/api/java/util/Random.html

   e. A method *initialize()* to setup the environment (for instance, creating the philosophers, assigning IDs, creating chopsticks etc.)
   f. A method *start*() that starts the simulation process
2. Create a class *Philosopher* that beside others will contain the following methods:
   a. getAverageThinkingTime();
   b. getAverageEatingTime();
   c. getAverageHungryTime();
   d. getNumberOfEatingTurns();
3. Calculate the average[3] EATING, THINKING, and HUNGRY times. When the *start()* method has finished the average eating, thinking, and hungry times for each philosopher should be known.
4. Generate a trace file (named *Log.txt*) with major events that occur during the simulation (for instance, *Philosopher_1 is THINKING*, *Philosopher_4 is EATING*, *Philosopher_3 is HUNGRY, Chopstick_3 released, Chopstick_1 picked-up*, *Deadlock detected* …)
5. Detect and report the deadlock (that is a state where all philosophers hold the left chopstick and wait for the right chopstick). When the deadlock is detected, the program execution should be stopped.

Folder structure

- Create a new project and name it 1dv512.userid (e.g. 1dv512.sm222bt)
- Put all your files in one package (default)
- Pack whole project directory (src) into zip archive. Your zip should contain some root directory (e.g. sm222bt), not just list of files.

Example of the folder structure:

- sm222bt
  - src
    - Philosopher.java,
    - DiningPhilosopher.java,
    - Chopstick.java, …
  - ReadMe.txt,
  - Log.txt

Instructions for running and testing the application

Provide any information that is relevant for running your application in the ReadMe.txt file.
Expect that we test your solution using a method like this:

```
public static void main(String args[]) {
        DiningPhilosopher dp = new DiningPhilosopher();
        dp.setSimulationTime(10000);
        dp.initialize();
        dp.start();
        ArrayList<Philosopher> philosophers = dp.getPhilosophers();
        for (Philosopher p : philosophers) {
                System.out.println(p.getId() + "- " + p.getAverageThinkingTime());
                …
        }
}
```

[3] http://mathworld.wolfram.com/ArithmeticMean.html