CMPS 109
Advanced Programming
Winter 2011

Programming Assignment 5

Due Monday March 14

In this project you will construct threaded Java program that solves the classic "Dining Philosophers" problem. In this problem *n* philosophers are seated around a circular table. In front of each philosopher is a plate of spaghetti, and between each plate is a single fork. Each philosopher must alternately think, and then eat. To eat, he or she must pick up both the left and right forks. When both forks are down the philosopher is free to think. When a philosopher holds only one fork, he or she can neither eat nor think. Each fork can be held by only one philosopher at a time, and each has a specified amount of thinking and eating to do. The problem is to arrange it so that all required eating and thinking are actually performed.

You should model this problem on the ConsumerProducer example which will be discussed in class on Wednesday. Each philosopher will be represented by a Thread that is constructed from a class you write called Philosopher, which implements the Runnable interface. You will also write a class called Fork, each instance of which will be shared by two Threads (philosophers). Fork will contain two synchronized methods called pickup() and putdown(). (These are analogous to the Buffer methods get() and put() from the ProducerConsumer example.) When one Thread calls pickup(), the Thread sharing its Fork object is blocked from calling pickup() on that same object.

An obvious problem arises when all philosophers have picked up their right forks. In this case, no philosopher has picked up a left fork, and so none can eat and none can think. This is a form of *deadlock*, which is a common problem in concurrent programming. One possible solution would be to have some philosophers be left handed, and others right handed. The right handed group prefers to pick up the right fork first, and the other group prefers the left first.

Function main for this project will be defined in a file called DininingPhilosophers.java. It will take one command line argument giving the name of a text file. This file will be structured as follows. Line 1 will contain a single integer *n* giving the number of philosophers seated around a table. The remaining *n* lines will each contain two integers giving the total eat time and total think time (in that order) for each philosopher, both in milliseconds. Each philosopher has an identifying integer label in the range 1 to *n*. Thus line *i* of the input file, where $2 \le i \le n+1$, gives the eat and think time for philosopher $i-1$. Output will be written to stdout, and simply reports that a philosopher is either started, eating, thinking, finished eating, or finished thinking. For instance, a portion of the ouput may look like:

```
Philosopher 1: started
Philosopher 2: started
⋮
Philosopher 2: thinking
Philosopher 5: eating
Philosopher 3: thinking
Philosopher 2: eating
Philosopher 1: finished eating
Philosopher 2: finished thinking
Philosopher 1: finished thinking
⋮
```

Program execution halts when all philosophers are both finished eating and finished thinking.

Your Makefile for this project will create an executable jar file called DiningPhilosophers, and will be called on the command line by doing: `DiningPhilosphers inputfile`. If you are unfamiliar with jar files, study the Java examples at: http://www.soe.ucsc.edu/classes/cmps101/Fall10/examples.html

You will submit 5 files for this project:

`README`	A table of contents for the project
`Makefile`	Described above
`Fork.java`	Defines the Fork class with synchronized pickup() and putdown()
`Philosopher.java`	Defines the Philosopher class
`DiningPhilosopohers.java`	Contains function main()