2010 Paper 3 Question 9

Further Java

Fellows at Norisbon College dine at a circular table on which there is a single fork between each Fellow. Fellows either eat or think, and always start dinner thinking. To eat, a Fellow first picks up the fork immediately to his left and, once successful, picks up the fork immediately to his right. When a required fork is not on the table, the Fellow waits, neither eating nor thinking, until the fork is returned to the table. After eating, a Fellow returns both forks to the table. No cutlery is required to think.

Your task is to model the above scenario in Java.

(*a*) Write a class called `Fork` with two public methods, `pickUp` and `putDown`. The methods should take no arguments and return no result. An instance of `Fork` should act as a lock to prevent concurrent access. In other words, once `pickUp` has been called, all further calls to `pickUp` should block until `putDown` is called; when `putDown` is called, one caller (if any) who is blocked should proceed. [7 marks]

(*b*) Write a class called `Fellow` which inherits from the `Thread` class and implements the abstract method `run`. The `Fellow` class should have a single constructor which takes two `Fork` objects, one representing the fork to the Fellow's left, and one to the right. When run, an instance of `Fellow` should think for ten seconds, eat for ten seconds and think for ten seconds before terminating. [7 marks]

(*c*) Describe when and why your implementation may suffer deadlock. [2 marks]

(*d*) By altering the order in which the forks are picked up, describe how you would modify your implementation so that it does not suffer deadlock. [4 marks]