Assignment 4: Monitors

ETH Zurich

1 Unisex bathroom

1.1 Background

This task has been adapted from *Foundations of Multithreaded, Parallel, and Distributed Programming* [1]. In an office there is a unisex bathroom with n toilets. The bathroom is open to both men and women, but it cannot be used by men and women at the same time.

1.2 Task

1. Develop a Java program that simulates the above scenario using Java built-in monitors. Your solution should be deadlock and starvation free.
2. Justify why your solution is starvation free.

2 Signal and continue vs. signal and wait

2.1 Background

Listing 1 shows a monitor class that defines three parts of a job.

Listing 1: three part job class with signal and wait

```
monitor class THREE_PART_JOB

feature
  first_part_done : CONDITION_VARIABLE

  do_first_and_third_part
   do
     first_part
     first_part_done . signal    -- "Signal and Wait" signaling discipline
     third_part
   end

   do_second_part
     do
       first_part_done . wait
       second_part
     end
end
```

The condition variable *first_part_done* is used to ensure that the first and the third part are executed by one thread t_1 and that the second part is executed by another thread t_2 in between the first and the third part. This is the correctness specification.

2.2 Task

1. Assume that the condition variable implements the "Signal and Wait" discipline. Is the code correct? If the code is correct, justify why it works. If the code is not correct, show a sequence of actions that illustrates the problem.

2. Assume now that the condition variable implements the "Signal and Continue" discipline instead. Is the code correct? If the code is correct, justify why it works. If the code is not correct, show a sequence of actions that illustrates the problem.

3. If the program is not correct with the "Signal and Continue" discipline, rewrite the program so that it is correct. To do this, use the "Signal and Continue" condition variables.

3 Smoke Signals

3.1 Background

This task, originally proposed by Patil [2], was a response to Dijkstra's semaphores.

3.2 Task

There is a table with 3 smokers, and a dealer. The smokers continually smoke and make cigarettes. Each smoker also has an infinite amount of one type of supply (papers, tobacco, matches) to make a cigarette. The smokers cannot accumulate supplies that are not their own. They smoke a single cigarette, then try to acquire the required supplies to make a new one, ad infinitum.

The dealer is responsible for non-deterministically selecting two smokers, taking one of each of their supplies, and placing them on the table. He then notifies the third smoker that he/she may take these supplies and make another cigarette when he is finished his current cigarette (if he has one). When the dealer sees the table is again empty, he will repeat the action of placing supplies on the table.

Try to formulate the dealer and each smoker as a separate process. You may use either monitors or semaphores to solve the problem.

References

[1] Gregory R. Andrews: Foundations of Multithreaded, Parallel, and Distributed Programming. Addison Wesley, 1999

[2] Sahus Patil: Limitations and capabilities of Dijkstra's semaphore primitives for coordination among processes. MIT, 1971