Exercise 2: Synchronisation mechanisms in Java.
Rendez-vous.

Hand-out: 26 April
Due: 3 May

The goal of the exercise is to get acquainted with the concepts of rendez-vous and active objects. Several ways of programming a rendez-vous will be discussed and illustrated with simple examples.

1. Warm-up: Santa Claus
We were interested in answering the following questions concerning the Santa Claus scenario:

1.1. What synchronisation patterns can you see in the scenario?
   Which of these did you identify:
   - mutual exclusion (with a mutex or a binary semaphore),
   - synchronisation with a general semaphore,
   - producer-consumer,
   - barrier synchronisation,
   - reader-writer?

1.2. Is it possible to implement all of them using mutexes? Semaphores? Monitors? A combination of these?
   What was your conclusion?

1.3. Can you think of a more advanced mechanism that would help?
   Would rendez-vous help?

1.4. How do you implement the priority scheduling (reindeer vs. elves)?
   Would rendez-vous help?

2. Rendez-vous in Java: the beast and the beauty
We illustrate the use of rendez-vous with three simple examples: client-server synchronisation (unconditional rendez-vous), bank account (conditional rendez-vous), and dining philosophers (conditional rendez-vous). We will compare a solution based on a language with no native support for rendez-vous (Java) and a solution written in sJava (or Synchronous Java) that has built-in mechanisms for the support of rendez-vous synchronisation. Discussed examples are available online (see course page).

Questions:
2.1. Which solution (Java or sJava) is more elegant?
2.2. Which solution of the dining philosopher problem is more elegant: monitor-based (as seen last week) or rendez-vous based? Which should be more efficient and why?

2.3. When would you use (active objects with) rendez-vous synchronisation?

2.4. Do you think rendez-vous can emulate other synchronisation mechanisms? How efficient would such emulation be? How elegant would it be?

3. Homework: Synchronisation mechanisms and object-oriented techniques
Think about the problem of fitting the different synchronisation mechanisms into the object-oriented framework.

Questions:

3.1. Can you see any problems with the mechanisms such as inheritance and polymorphism? Try to figure out a few examples.

3.2. How about information hiding?

The topic will be discussed in the next lecture and we will see a few examples in the exercise session.

Have fun!