Prof. O. Nierstrasz
Jorge Ressia, Jan Kurš

Serie 8 - Asynchrony & Condition Objects

Exercise 1

Answer the following questions:

a) Why are servers usually structured as thread-per-message gateways?

b) Can you think of other well-known programming concept(s) that is (are) similar to Futures? Outline differences (if any) and give some programming languages that support these features.

c) What are condition objects? Name at least two advantages and disadvantages, respectively, of using condition objects.

d) Point out the difference(s) (1) between a simple condition object and a semaphore, and (2) between a simple condition object and a monitor.

e) Why does the SimpleConditionObject from the lecture not need any instance variables?

f) What is the "nested monitor problem"? Give an answer that is as precise and short as possible (in your own words). Also describe briefly a concrete example where a nested monitor occurs.

g) What is the easiest way to avoid the nested monitor problem?

h) What are "permits" and "latches"? When it is natural to use them?

Exercise 2: Futures

Consider the sample code FutureTaskDemo.java and EarlyReplyDemo.java which you both find in CP-Series8.zip in the package asynchrony. In both cases, several client threads request a server to compute fibonacci numbers.

a) Which implementation would you prefer for this kind of problem? Is there any considerable difference at all? Justify your answer!

b) Write a new class FutureTaskExecDemo.java that uses an executor service to compute the future task and to execute the clients, instead of creating explicit new threads. What is the benefit of using executors?

c) In addition, add a time constraint such that the client thread waits for at most a given amount of time for the result.

Exercise 3: Nested Monitor

Farmer Napoleon owns a magic chicken called Clarissa who is supposed to lay infinitely many eggs of different colors. Napolean has hoped to dispose of an endless source of eggs to build up his egg-imperium. But there is a serious deadlock problem hidden behind the story. Your task is to resolve this problem in order to let Napolean to continously retrieve eggs from Clarissa.

You find the code in CP-Series8.zip in the package eggFarm.

Exercise 4: Nested Monitor revisited

Write an LTS model for the modified program from Exercise 3. Make sure that no deadlock occurs; check your model using the LTSA tool. Hint: You should model at least the following processes: `SEMAPHORE`, `THENEST`, `CHICKEN`, `FARMER`, and the composite process `FARM`.