Serie 4 - Safety Patterns — 07.10.2015

Exercise 1

Answer the following questions:

1. Why are immutable classes inherently safe?
2. What is “balking”?
3. When is partial synchronization better than full synchronization?
4. How does containment avoid the need for synchronization?
5. What liveness problems can full synchronization introduce?
6. When is it all right to declare only some methods as synchronized?

Exercise 2

The dining savages: A tribe of savages eats communal dinners from a large pot that can hold M servings of stewed missionary. When a savage wants to eat, he helps himself from the pot unless it is empty in which case he waits for the pot to be filled. If the pot is empty the cook refills the pot with M servings. The behavior of the savages and the cook are described by:

\[
\text{SAVAGE} = (\text{getServing} \rightarrow \text{SAVAGE}). \\
\text{COOK} = (\text{fillPot} \rightarrow \text{COOK}).
\]

Model the behavior of the pot as an FSP process.

Exercise 3

What action trace violates the following safety property?

\[
\text{property } PS = (a \rightarrow (b \rightarrow PS|a \rightarrow PS)|b \rightarrow a \rightarrow PS).
\]

Exercise 4

Implement a thread-safe MessageQueue class using one of the safety patterns. You have to justify your choice.

The MessageQueue has the following specifications:

- Messages are strings.
- The queue has a fixed capacity that is set at construction time.
- There are two methods: \text{add(String s)} and \text{remove}.
- Method \text{add(String s)} inserts a message to the end of the queue. If the queue is full, it waits until a message is removed.
- Method \text{remove} returns the first message in the queue and then removes it from the queue.
- Write a unit test for your MessageQueue to demonstrate the thread-safety of your implementation.