Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Leonel Merino

Series 04 — 11.10.2017 – v1.0a
Safety Patterns

Exercise 1 (3 Points)

Please answer the following questions:

a. Why are immutable classes inherently safe?

b. What is "balking"?

c. When is partial synchronization better than full synchronization?

d. How does containment avoid the need for synchronization?

e. What liveness problems can full synchronization introduce?

f. When is it legitimate to declare only some methods as synchronized?

Exercise 2 (2 Points)

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:

```
SAVAGE = (getserving -> SAVAGE).
COOK   = (fillpot -> COOK).
```

Model the behavior of the pot as an FSP process.

Exercise 3 (2 Points)

Please consider the FSP below and answer the questions:

```
property LIFTCAPACITY = LIFT[0],
LIFT[i:0..8]      = (enter -> LIFT[i+1]
                    |when(i>0) exit -> LIFT[i-1]
                    |when(i==0)exit -> LIFT[0]
                    ).
```

a. Which values can the variable `i` take?

b. What kind of property is used and what does it guarantee?

c. Provide an action trace that violates the provided property.

d. Provide an action trace that does not violate the provided property.

Exercise 4 (3 Points)

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

The MessageQueue has the following specifications:

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

Please consider also the order of importance for development:

1. *Logic (Does the implementation work?)*
2. *Concurrency (Do safety- and liveness-property hold?)*
3. *Scalability (Does the approach scale?)*