Concurrency:
State Models & Design Patterns

Practical Session

Week 03

Assignment 02

Discussion

A02 - Exercise 1

a) What states can a Java thread be in?

There exist six states in the Java runtime environment: `NEW`, `RUNNABLE`, `BLOCKED`, `WAITING`, `TIMED_WAITING` and `TERMINATED`.

b) How can you turn a Java class into a monitor?

It is as simple as declaring all public methods `synchronized`.

A02 - Exercise 1

c) What is the Runnable interface good for?

You can create a running thread based on a `Runnable` object. This is very useful since Java does not support the concept of multiple inheritance: Inheriting from the `Thread` class is not always an available option.

d) Specify an FSP that repeatedly performs hello but may stop at any time.

HELLO = (hello -> HELLO | hello -> STOP).

A02 - Exercise 2

Consider the following Java implementation of a Singleton within a single-threaded application:

```
public class Singleton {
        private static Singleton instance = null;

        private Singleton() {}

        public static Singleton getInstance() {
                if(instance == null) {
                        instance = new Singleton();
                }
                return instance;
        }
}
```

a) What happens if the application is multithreaded?

There exists the possibility that multiple instances of the `Singleton` class are created concurrently, potentially overwriting previously assigned `instance` variable values.

A02 - Exercise 2

b) How to implement a thread-safe singleton in Java?

The `getInstance()` method needs to be synchronized as shown below:

```
public static synchronized Singleton getInstance() {
        if(instance == null) {
                instance = new Singleton();
        }
        return instance;
}
```

A02 - Exercise 2

c) Suppose there arrive 1000 requests/second from different threads at this Singleton. Does your implementation introduce a bottleneck? If yes, how can you improve it?

Yes, a more efficient and fine-grained solution is as follows ():

```
private static volatile Singleton instance;

public static Singleton getInstance() {
	if(instance == null) {
		synchronized (Singleton.class) {
			if(instance == null) {
				instance = new Singleton();
			}
		}
	}
	return instance;
}
```

A02 - Exercise 3

Download LTSA from http://www.doc.ic.ac.uk/~jnm/book/ltsa/download.html. For each of the following processes shown in Figure 1, provide the Finite State Process (FSP) description of the corresponding Labeled Transition System (LTS) graph. You may verify the FSP descriptions by generating the state machines and using the "draw" functionality of the tool.

```
APPOINTMENT = (hello -> converse -> goodbye -> STOP).

TREBLE = (in[i:1..3] -> out[i*3] -> TREBLE).

FIVETICK (N=5) = FIVETICK[1],
FIVETICK[i:1..N] = ( when(i<N) tick -> FIVETICK[i+1]
                   | when(i==N) tick -> STOP).

...
```

A02 - Exercise 4

Consider the full Race5K FSP from the lecture.

a) **How many states and how many possible traces does it have if the number of steps is 5 (as in the lecture)?**

   36 states (cartesian product of individual state spaces) fostering 252 traces.

b) **What is the number of states and traces in the general case (i.e. for n steps)? We assume 2 processes.**

   - Number of states: $(n + 1)^2$
   - Number of traces: (2*n)! / (n!*n!)

c) **Check your solution using the LTSA tool.**

   :-)

Assignment 03

Preview

A03 - Exercise 1

Answer the following questions:

a) What does a safety property do in FSP and how do you model it?

b) Is the busy-wait mutex protocol fair? Justify your answer.

c) Can you ensure safety in concurrent programs without using any kind of locks?

d) The Java language designers decided to implement concurrency based on monitors. What is the main reason behind this decision? What other options except monitors could have been chosen?

A03 - Exercise 2

Consider the following process definitions. Are T1 and T2 equivalent? Why?

```
R = (a->c->R).
S = (b->c->S).
||T1 = (R || S).
T2 = (a->b->c->T2|b->a->c->T2).
```

A03 - Exercise 3

A miniature portable FM radio has three controls. An on/off switch turns the device on and off. Tuning is controlled by two buttons, scan and reset, which operate as follows: (...)

A03 - Exercise 4

Implement a stack in LTS and obey the following requirements:

1) There exists one PUSH process which provides the push functionality.

2) There exists one POP process that provides the pop functionality.

3) There exists one LOCK process that provides the locking functionality.

4) There exists one STACK process that provides the stack functionality.

5) The stack has two slots to store values.

6) The stack operations pop and push must use a lock.

7) The supported data values are only 0 and 1.

8) An error must be thrown for each push to a full stack or pop from an empty stack.

*You have to attend the lecture to reveal such slides.**

:-)

**Disclaimer:*

The content that has been shown on this slide is irrelevant for the exam.