Marcel Zauder 16-124-836
Pascal Gerig 16-104-721

2.1 Several Questions

a) *What states can a Java thread be in?*
A thread can be either NEW, when the thread is created but has not yet started, RUNNABLE, thread is executed in the Java Virtual Machine, BLOCKED, a thread that is waiting, i.e. for a monitor lock to be released, WAITING, a thread that waits indefinitely for another thread to finish its task, TIMED-WAITING, a thread is waiting for a specified amount of time, and TERMINATED, a thread that has exited. At one point a thread can only be in one of those states.

b) *How can you turn a Java class into a monitor?*
Each object in Java can be a monitor. For this its public methods must all be synchronized such that the access to this particular object is mutually exclusive.

c) *What is the Runnable interface good for?*
A thread can be instantiated using a Runnable interface, such that the class which specifies the behaviour of the thread can inherit from another superclass which is not possible when already inheriting from the Thread class because Java only allows single inheritance.

d) *Specify an FSP that repeatedly performs hello, but can stop at any time.*
The non-deterministic FSP would look like the following:
HELLO = (hello -> HELLO | hello -> STOP).

2.2 Singleton

a) If no singleton is there yet, then multiple threads can instanciate different singletons simultanously and return those.

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

c) There is a bottleneck because only one thread can get the singleton instance at a certain moment. We can improve this, by only protecting the actual critical section instead of the entire method. In this case, we must guarantee that only one object is created, once that object is created, only accessing it is not critical, thus we can improve the code as follows, by only protecting the creation of the object:

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

Marcel Zauder 16-124-836
Pascal Gerig 16-104-721

2.3 LTSA

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

b) HOLIDAY = (arrive -> relax -> leave -> HOLIDAY).

c) SPEED = (on -> ON),
ON = (off -> SPEED | speed -> ON).

d) LEFTONCE = (ahead -> INTERNALSTATE),
INTERNALSTATE = (right -> LEFTONCE | left -> STOP).

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

f) const COUNTER = 5
FIVETICK = TICK[1],
TICK[n:1..COUNTER] = (when (n<COUNTER) tick -> TICK[n+1]
| when (n==COUNTER) tick -> STOP).

g) PERSON = (workday -> WORKDAY | holiday -> HOLIDAY),
WORKDAY = (sleep -> work -> PERSON),
HOLIDAY = (sleep -> SLEEPHOLIDAY),
SLEEPHOLIDAY = (play -> PERSON | shop -> PERSON).

2.4 Race5K FSP

a) *How many states and how many possible traces occur if the number of steps is 5 (as in the lecture)?*
Because we have 5 steps each competitor would have 6 states. Therefore we would get $6 * 6 = 36$ different states and a total of $\binom{10}{5} = \frac{10!}{5!(10-5)!} = 252$ traces.

b) *What is the number of states and traces for the generalized case (i.e., for n steps)?*
The number of states would be $(n+1)^2$ and the number of traces is $\binom{2n}{n}$.

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