Harder class and sequence diagram exercises

October 9, 2015

Purpose

These exercises are intended to be a little more challenging than the basic ones. Do these if you have time once you are confident with the basic exercises. If you don't do them now, I suggest doing them later for revision. If you would like feedback on your work, write it up and give it to me for marking. You may do this at any time during the course. I should be able to return marked work within a few days under normal circumstances – but of course that won't work if everyone hands in a pile of work a few days before the exam! If I do get overwhelmed, a group feedback session may then be more appropriate.

Exercises

1. Consider a (toy) class `NaturalNumber` which contains an Integer attribute `i`, with the *class invariant* that `i` is always at least 0, and operations `increment()` and `decrement()` which generally have the obvious behaviour of incrementing and decrementing `i`; if `decrement()` is sent to an object of class `NaturalNumber` in which the value of `i` is 0, then instead of decrementing `i`, the object will send to an object `log` (to which it has a reference) the message `triedToDecrementZero()`.

   (a) Draw a behavioural state diagram for `NaturalNumber` that records all the behaviour described above, and has two states.

   (b) Now do the same, but make sure your diagram has three states.

   (c) Generalise... What choices do you have for the number of states in a correct state diagram for `NaturalNumber`? Can you have a correct behavioural state diagram that has one state?

   (d) Explain how this relates to the complete (fully-detailed) state space of `NaturalNumber`.

   (e) Draw a protocol state diagram for `NaturalNumber`, ensuring that you include all the information clients would need to use objects of this class correctly.

2. Deliberately vague question to think about: visibility of the operations and attributes that appear in a state diagram. Is it OK to mention private operations in a state diagram? Private attributes? Does it make a difference whether we're talking about protocol or behavioural state machines? What are the implications for good design?