Tutorial: basic class and sequence diagrams

September 19, 2013

Purpose

Check that you know how to read and write basic class and sequence diagrams. In principle, you do from Inf2C-SE! But not everyone did Inf2C-SE, not everyone who did will perfectly remember every detail, and it will help later if you are all rock-solid on these UML basics.

Everyone should become able to do these, and should have a go before the tutorial. If you have trouble:

1. visit the answers page and see whether the video explanations there help. **On your honour**, do not visit this page until you've had a good go at doing the question for yourself!
2. ask your tutor.

If any questions come up in tutorials that aren't easily resolved, please let me know, either by posting to the forum or by mailing me direct.

Do watch the videos on the answers page at some point (there are some points of design discussed, and every feature of UML mentioned in them **is examinable**) but if you're happy with your own answers to these questions, you might prefer to watch them later as revision rather than now.

Exercises

1. Draw a class diagram that shows a class `Party`, a class `Food`, a class `Entertainer` and a class `Venue`. Each of `Food`, `Entertainer` and `Venue` should have an integer attribute `cost` and getter and setter methods for it. `Party` should have an operation with selector `getTotalCost`. (What should the signatures of these operations be?) For now, do not show associations between the classes.
2. It is decided that the `getTotalCost` operation should be implemented by the object of class `Party` asking an object of each class `Food`, `Entertainer`, `Venue` for the value of its `cost` attribute, summing them and returning the result. Draw a sequence diagram that shows this behaviour being invoked by an actor.
3. Since objects of our classes communicate, there should be associations among the classes in the class diagram. Add suitable associations. Show navigability and multiplicities.
4. Next consider a modification of the design in which, instead of containing a fixed `cost`, a `Venue` object will calculate its cost on request, using information about the timing of the event. Suppose that a `Party` can report its date, start time and duration. Consider two possible designs:
   (a) the relevant information is passed to `Venue` along with the request to calculate a cost; or

(b) the `Venue` object will take a no-argument request for a cost as usual, but then will ask for the extra information it needs.

Make versions of, or annotate, your diagrams to show the difference between the two options. Take special care with the second. What are the pros and cons of each design?