Intro to OO: after-course work

12 March 2010

1 Consolidation of basic idea

Read the following very basic OO tutorial and make sure this is all familiar (except for paragraphs on Smalltalk environments!). Note that it was written when Java was new!

`http://www.softwaredesign.com/objects.html`

Consider a system that allows balancing a chequebook against a bank statement, i.e. works out which cheques have been cleared, which payments in have arrived, and identifies any anomalies on the statement.

Identify the domain classes and as much as you can about states and behaviour.

2 Drafting an object model using CRC cards

Read Beck and Cunningham's short paper "A Laboratory For Teaching Object-Oriented Thinking":

`http://c2.com/doc/oopsla89/paper.html`

Ideally with a few colleagues:

Consider the following rough system description. This is a classic, or even chestnut, addressed in many OO books: this version is quoted from Wilkinson's *Using CRC Cards*.

> This application will support the operations of a technical library for an R&D organization. This includes the searching for and lending of technical library materials, including books, videos and technical journals. Users will enter their company ids in order to use the system; and they will enter material ID numbers when checking out and returning items.
>
> Each borrower can be lent up to five items. Each type of library item can be lent for a different period of time (books 4 weeks, journals 2 weeks, videos 1 week). If returned after their due date, the library user's organization will be charged a fine, based on the type of item (books $1 per day, journals $3/day, videos $5/day.

> Materials will be lent to employees with no overdue lendables, fewer than five articles out, and total fines less than $100.
>
> ... (Design Constraints) ...

Identify the main domain objects that may be needed. (Any dubious/borderline cases? Make a note. Anything that needs to be added to/clarified in the system description? Feel free to imagine, e.g., that you asked the customer and they agreed the changes were correct.)

Following the procedure discussed in the course, make CRC cards and play through some scenarios, refining the cards as you go. Record your final collection of classes with their responsibilities and collaborations.

Next, consider various changes in the requirements that might arise in future. Play through any new or modified scenarios and consider what changes need to be made to the classes. What assumptions are you making? (Of course, precise answers will depend on the final implementation of the system.)

- Changes to the dollar amounts in the description
- Libraries to stock DVDs and CDs in addition to the other lendables.
- Adaptation of the system to a country with a different currency
- Adaptation of the system to manage a network of libraries for an international organization.
- Borrowers to be allowed to reserve items and, where the library has several copies of the same lendable, to be notified as soon as any copy becomes available.
- Anything else that seems interesting...

3 Thought provocation

Consider the following quotation from Dijkstra's "On the role of scientific thought".

> Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects. We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day, so to speak. In another mood we may ask ourselves whether, and if so: why, the program is desirable. But nothing is gained –on the contrary!– by tackling these various aspects simultaneously. It is what I sometimes have called "the separation of concerns", which, even if not perfectly possible, is yet the only available technique for

> effective ordering of one's thoughts, that I know of. This is what I mean by "focusing one's attention upon some aspect": it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously.

What, if anything, does object orientation have to contribute to this broader notion of separation of concerns? Critically investigate *aspect oriented software development* and consider alternative approaches. In practice, what techniques do you use to separate concerns in your normal work?

4 Challenge

This question asks you to think like a language designer, identifying the pros and cons of different choices that might be made and their pros and cons. Assume you are designing a new statically typed, class-based object oriented language. At first, try to work issues out for yourself without using any knowledge you may have of particular OOPLS (and certainly without looking it up!)

1. We said that a subclass may override a method from the superclass, to implement behaviour that is specific to the subclass. Should the subclass ever be allowed to change an argument type, or the return type, of a method it is overriding? If you think so, in what way, and why? You might like to reconsider the Animals example in the inheritance slides.

2. We talked about objects hiding their data, and we talked about classes defining a collection of similar objects. What access, if any, do you think an object should have to hidden parts of another object *of the same class*? Why?

   Now consider how *binary methods* should work. For example, suppose you are writing a `Vector` class, with all its data hidden, and you want to give it a method `add` that will take another `Vector` as argument and add it to the receiving `Vector`. How will this be implemented?

   Next think about how these issues interact with inheritance and subtyping. What may happen if someone later writes a `SpecialVector` class inheriting from `Vector`? Consider the various ways in which `SpecialVector` may differ from `Vector`. For example, suppose that `SpecialVector`s have colour, as well as magnitude and direction, or consider that `SpecialVector` is in a different number of dimensions from `Vector`, or that it is represented using a different coordinate system. In what situations will it make sense to add two vectors? Will any code need to be written or changed to make `add` still work appropriately, and where will any changed or new code be?