CMPT 318 99-1: Assignment #5

Updated: Mar 4
Updates to this assignment will be sent to the mailing list if they are required.

Due Date

April 1 at 12:00 (noon) in the assignment box.

Late Policy:

No penalty will be assessed for late assignments until April 5 at 12:00 (noon), after which there will be a 10% penalty assessed each day for a maximum of two days. No assignments will be accepted after April 7 at 12:00 (noon).

Marks

4% of your final mark

Purpose

Important programming ideas are contained within the object-oriented design patterns (OODP) documented by the so-called "Gang of Four" design patterns book. That is one reason why Eckel devotes a chapter to it in his Java book: it is an important part of Java programming, just like event-based GUI programming, threading, and beans are. However it is difficult to learn OODPs and all their subtleties and implications in a classroom setting. This assignment reinforces the design pattern information you learned in class, but without forcing you to write code.

After this assignment you should have a solid understanding of OODPs in general and know a few design patterns in particular in depth.

Bonus (Optional) Readings

In all prior units I have sought to cover the same sort of material covered by Eckel, but from a slightly *altered* viewpoint in order to give you several views on the same material. In the DP unit, however, we relied less on course textbook material. For one thing not everyone has the GoF "Design Patterns" book, and Eckel's coverage of design patterns is too light for a serious study of OODP. Thus more emphasis is placed on the online notes (in contrast to the notes being more-or-less just a reading guide as it was for prior units). Thus extra reading is generally a good way of reinforcing the classroom material. This is especially true if you skipped classes, dozed through them, or were confused or missed points.

If you own a copy of the GoF "Design Patterns" book, it would be wise to read chapter 1 and 6.

If you do not own a copy of the GoF "Design Patterns" book you should probably read some the following references. They are presented in order of simplicity with the simplest at the beginning.

1. Design Patterns: Elements of Reusable Architectures
   Linda Rising, in the Annual Review of Communications, 1996.
2. Software Design Patterns: Common Questions and Answers
   James O. Coplien. In Proceedings of Object Expo New York, pages 39-42, June 1994. New York, SIGS Publications.
3. Software Patterns
   Douglas C. Schmidt, Ralph E. Johnson, Mohamed Fayad, guest editorial, CACM, vol 39, no 10, 1996.
   (prettier PDF is here)
4. Introduction to Design Patterns
   Class Notes for Doug Schmidt's Design Patterns Course.
5. The Patterns FAQ
   maintained by Doug Lea. A good read because it emphasises the general lack of easy answers which frustrates some people who expect there to be easy answers.
6. Patterns and Software: Essential Concepts and Terminology
   long but valuable web-based document by Brad Appleton.
7. Design Patterns: Abstractions and Reuse of Object Oriented Design
   Proceedings of ECOOP'93, Kaiserlautern, Germany.

   > This is one of the original publications of the GoF, and is the precursor to the GoF book. NOTE: some of the terminology and patterns have evolved since then, but it is a good read so long as you understand the changes.

(even if you do own a copy, these are good sources of reading material.) #1 is most recommended because it is so short, then #5 and #6.

Assigned Work

1. **Patterns Unto Patterns**

Some say that one of the real problems with the "patterns" literature is that if you're not careful *everything* starts looking like a pattern of some sort.

Read: The Pros and Cons of Adopting and Applying Design Patterns In the Real World

by Marshall P. Cline, CACM, v 39, no 10, 1996
(in our ACM Digital Library).

This is a short article on the benefits and drawbacks of using design patterns in software development. There is a strong analogy between "benefits" and "drawbacks" and the

consequences of the object-oriented design patterns. So this is a good place to test your hand at writing patterns.

a. Summarise the contents of the article in the form of a design pattern. We covered GoF form which is a good starting point, but you can customize it or use your own form to your liking. Just be sure to capture the important 4 issues of a design pattern in some reasonable manner.

   When doing this part of the assignment, do **not** do more than necessary. I'm looking for something similar to Thomas Minka's style of writing short patterns in his Decorator Pattern (except obviously without the implementation or known uses).

   Keep in mind these two points when you write:

   i. Use ***short*** descriptions when writing the consequences. I expect you to phrase the consequences in your own words. Strive for one or possibly two brief sentences summarizing the points of the paper. You may need 4-8 sentences to write the problem description (e.g. motivation).

   ii. The article does not directly cover the problem, context, forces, or solution, but these are all implicit in the description of the consequences. Give some thought to these are. You may have to think back on the programming issues covered in the course.

   (15 marks)

b. Is it reasonable to call this a pattern? Explain. (60 words max).

   (5 marks)

(20 marks total)

(Optional question).

Compare the use of a desgin pattern form versus the form of a research article for the presentation of solution knowledge. What does each form highlight? When is one form better than the other?

2. Patterns and Examples

Examples appear to be important for both documenting and learning design patterns.

Read: Non-Software Examples of Software Design Patterns

Michael Duell, Object Magazine, vol 7, no 5, July 1997, pg. 52-57, SIGS Publications.

a. Choose one of the design patterns that Duell covers and provide an *alternate* example of the design pattern being used in some other circumstance. It can be from another object-oriented program or library (like the AWT), or it can be an everyday example (how pizza is delivered, how to break up with a boyfriend or girlfriend, ... whatever).

   The example you choose should not be used in the GoF book (and minor variations of the given examples are not permitted) or one that was used in the class notes. If you do not have access to the GoF book, you should consult the list of online patterns, with preference for the ones I've translated from the GoF book since the same examples are used. Try to borrow the book from a friend for a few days if you can.

   In a manner similar to Duell, briefly describe your example.

   (4 marks)

b. Briefly explain how your example "matches" the pattern's basic idea, that is, how your example is *analogous* to the design pattern's example. Normally you will need to explain how your example matches (1) the intent and (2) the problem. In the GoF form, it should match the *intent motivation* and *applicability*. Duell is sometimes vague on the problem and intent match, but you must be more explicit on why your example is analogous.

   (5 marks)

c. Draw a class diagram or object interaction diagram for the example you chose. Which you should draw will depend upon the pattern you chose for your example. Your diagrams should generally be similar to the ones in the GoF book, or the ones you have seen in class. If you do not use a standard notation (like UML or OMT), give a legend to indicate what all the symbols in your diagrams mean. Capturing the basic structure of the example is more important than getting the diagramming notation correct.

   (4 marks)

d. Write down a reasonable scenario that highlights one of the *consequences* of the design pattern. That is, your scenario should be concrete (rather than abstract) and should seem plausible. You may choose to highlight a consequence that is a drawback in the scenario. In this case, your scenario should point out how the solution failed to meet your needs. Otherwise you must describe a scenario in which failing to use the pattern would result in negative consequences. For instance, a Mediator pattern beneficially decouples objects with complex interrelationships, so you could describe a scenario in where the Mediator pattern is not used and resulted in too high of a coupling.

   (4 marks)

(17 marks total)

3. **Example Clash?**

If you examine the paper in Q2, you can see that the example for the Mediator pattern looks somewhat similar to the example for the Facade pattern. But there are important differences.

a. Briefly explain how the two examples can look structurally similar but be associated with different patterns.

b. Explain why the two examples are different.

c. Is it reasonable for one example to be used for several patterns? Explain.

(3 marks)