

THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester 2007

**COMP2110
(Software Design)**

Writing Period: 3 hours duration

Study Period: 15 minutes duration

*Permitted Materials: Textbook Fox: Introduction to Software Engineering
Design, or other reference book,
plus one A4 page of notes*

Instructions

Answer all five questions.

The questions do not have equal value.

Marks for each part are indicated on the exam paper.

Write your answers in the examination book provided.

Start each question on a new page.

Extra exam books are available if you run out of space.

Do not use red ink.

QUESTION 1 [20 marks]

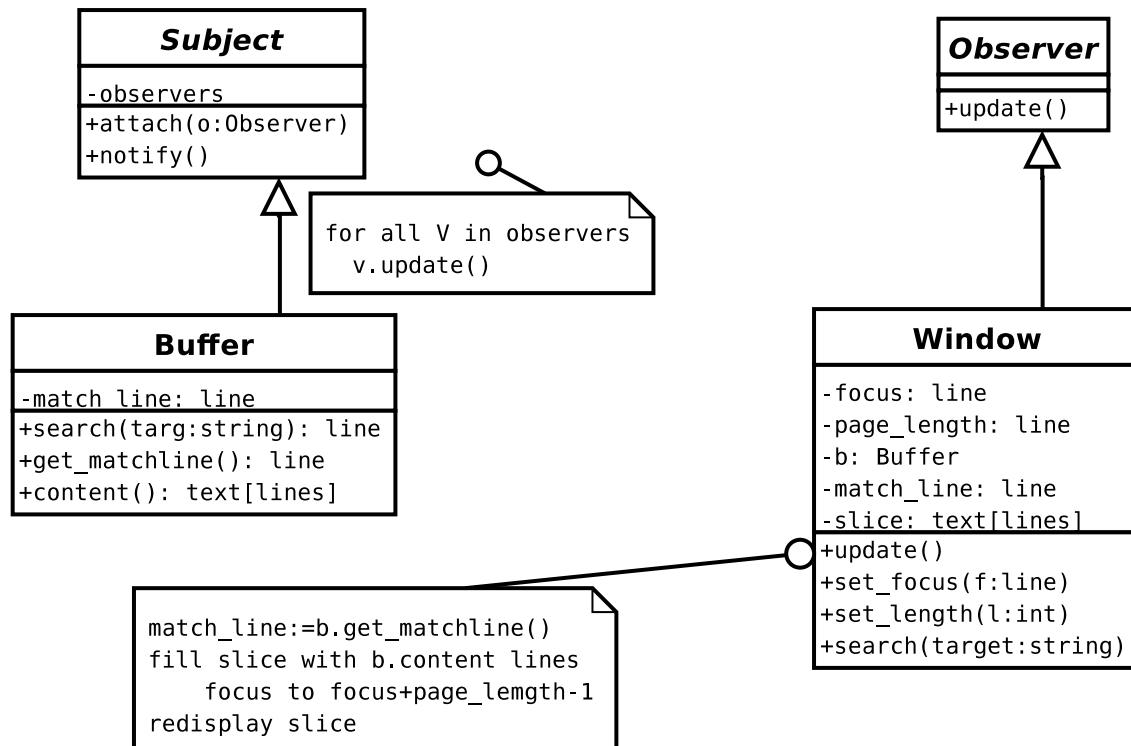
Consider the design of a plain text, multi-window browser program. The program allows users to browse over a plain text file using multiple viewing windows by moving the windows over the text and by searching for target text strings.

The contents of the text file is managed as an array of lines of text (represented by strings) in an object of class Buffer. The viewing windows are objects of class Window. Each Window displays a part of the buffer, called a “slice” of the text. In one window the display always starts at a line called the “focus” and extends for the length of the window, called the “page length” of that window. Each window’s display can be moved over the buffer separately - the different windows can overlap or show independent parts of the buffer, under the user’s control. The user controls the focus of a window by single keystroke commands: **ENTER** or character **d** moves the window focus down one page, **BACKSPACE** or **U** moves up one page.

The program uses the position of the mouse cursor to determine which window (the “current window”) is affected by a keystroke at a particular time, by using normal Graphical User Interface library operations.

One line of the buffer is called the “match-line” and this line is highlighted when it is displayed in any viewing window. The search command **S** requests a target search string from the user and then sets the buffer’s match-line to the next line containing that target. It also changes the display of the current window by moving its focus to equal the match-line.

The Buffer and Window classes can be shown like this



- (a) Describe the use of the Observer pattern here. In particular describe (using words and an informal object diagram) what is the result of creating two new windows for a buffer, and identify what operation(s) play the role of the Observer pattern's `get_state` operation.

[8 marks]

- (b) Consider the sequence of actions that happens in the buffer and the two windows when the Search command key is used in one window.

Describe what occurs, using sequence diagrams, written explanation, and pseudocode for any Window, Buffer or GUI operations that are involved. Your description should show enough detail to show how this design achieves the change to `match_line` in the buffer and the change of the `slice` and the display in both windows.

You can assume that `Window.search` can use the operation `GUI.get_input` to input the search target string. (Do not describe the `get_input` operation). [7 marks]

- (c) The Subject and Observer parent classes are actually unnecessary in this design. Show how they can be removed by redrawing the class diagram, and annotate it with pattern roles. Briefly discuss what advantages and disadvantages this change would have for the *quality* of the design. [5 marks]

QUESTION 2 [20 marks]

- (a) List 4 important quality factors in Software Design. Explain with examples how the designer may need to make trade-offs between any two (or three) of these when designing a software system such as the diary-calendar application.

[8 marks]

- (b) Explain what are the important defining characteristics of pseudocode.

What is the purpose of using pseudocode in a software design document? [4 marks]

- (c) Explain what is meant by a *module interface* in a detailed design document. Give an example. [4 marks]

- (d) Explain what is meant by the *description of module internals* in a design document. Give an example. [4 marks]

QUESTION 3 [25 marks]

This question is about the design of a software component that allows the user to manipulate files and directories. The files and directories are organised in a tree. The requirements are:

1. A *Client* needs information about a file system.
2. The model of the file system is composed of *items*.
3. Each *item* is either a *file* or a *directory*.
4. Every item has a *name*.
5. Every directory has *contents*, which consists of some number of *items* (files or directories).
6. Every file has a size which is an integer (indicating the number of bytes in the file).
7. Every item has a method called *totalSize* that returns an **integer** result. *totalSize* shall return the total of the sizes of all items that lie anywhere at or below this item in the file system tree. Each directory contributes a size of 1024 bytes to the total for itself, as well as the total size of its contents.

Tasks for this question

- (a) Draw a UML class diagram showing how to use the Composite pattern to model this situation. Assume that a simple recursive traversal of the tree is used to implement the *totalSize* operation (that is, this operation plays the role of the Operation or DoIt of the Composite design pattern)

Include:

- the classes CLIENT, ITEM, FILE and DIRECTORY;
- inheritance relationships;
- association and aggregation relationships, with multiplicities and labels;
- pattern roles for all the classes; and
- all attributes and routines (otherwise known as fields and methods) including *totalSize*.

Label abstract classes and features with an asterisk (*).

You do not need to mark access control/visibility of features.

[6 marks]

- (b) Write a pseudocode algorithms for each version of the *totalSize* method. (That is, write a separate implementation of *totalSize* for each concrete class that has it.)

[6 marks]

- (c) Show a sequence diagram to describe how *totalSize* operates in a tree with one directory that contains two files.

[5 marks]

(d) There are several new operations waiting to be added to the classes. These will all involve traversing the tree in different ways extracting and/or modifying information. They include listing all files and directories recursively, counting the total number of files and/or directories, etc. The initial design suggests adding each new operation by adding a new feature to each of the three classes ITEM, FILE and DIRECTORY.

A passing patterns-guru suggests modifying the design by using the Visitor pattern to simplify the process of adding all these new operations.

Explain how to use the Visitor pattern to do this.

Include a UML class diagram indicating the relationships between the new classes introduced. (You do not have to repeat your diagram of Composite from part (a), but you must indicate in words any changes that would be necessary.)

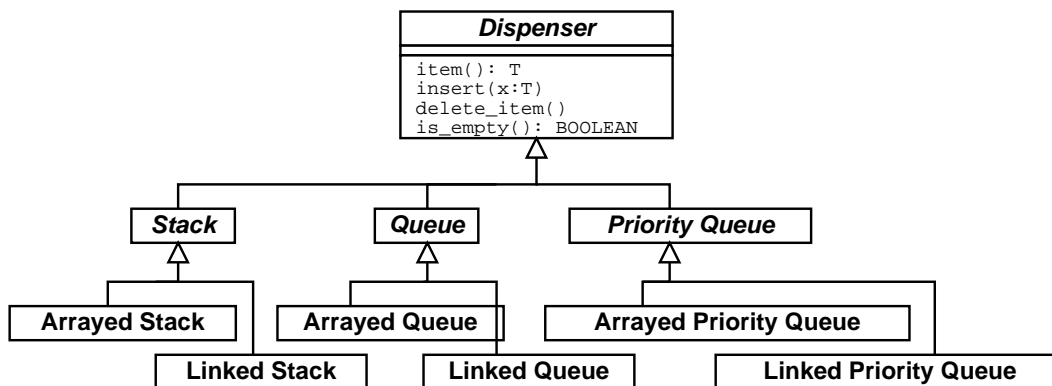
Although you are required to draw the class diagram, it will **not** on its own be a satisfactory answer to this question. You must also answer the question by writing an explanation.

You do not need to give the implementation of the visitors, although you should give interfaces and pseudo-code indicating the implementation of any action added to the existing classes. **[8 marks]**

QUESTION 4 [20 marks]

As part of the design of a set of general purpose library packages a designer decides to create a reusable Dispenser library. A “dispenser” is a container data structure into which items can be placed, by applying its *insert* procedure. The dispenser has a “current item”, which can be accessed with its *item* method. The current item can be removed from the dispenser with the *delete_item* method. Examples of dispensers are stacks, queues and priority queues (like the event queue in a discrete-event simulation). In each case, the difference is in how the “current item” is determined. For a stack the *current item* is the item that was most recently inserted; for a queue it is the item that has been in there the longest; and for a priority queue it is the item with the highest priority value. Other types of dispenser abstractions are possible.

Dispensers can be implemented using either an array or a linked list. The first design for this library looks like this:



Although the diagram doesn't show it, the classes are all generic classes (like class ARRAY) with the items all belonging to some class T. For the priority queue you may assume that the priority of the items can be compared by using the “<” operator (the “less than” operator) between its items.

- (a) Explain how to modify this design using the Bridge pattern to separate the abstractions (Dispenser, Stack, Queue and PriorityQueue) from their different implementations (Array and LinkedList). As part (but not all) of your answer draw a new UML class diagram showing how this works. Be sure to include both aggregation and inheritance relationships, and to provide annotations to indicate that this is an instance of the Bridge pattern, and the different pattern roles played by the classes and associations. You may assume that there is a new abstract class DISPENSER IMPLEMENTATION, which has the property that the items are stored in a structure that can be indexed by integers, and provides the interface

```
add_at (item: T; index: INTEGER)
```

```
remove_at (index: INTEGER)}  
item (index: INTEGER): T  
count: INTEGER
```

[8 marks]

- (b) Discuss whether or not this redesign is an improvement over the original design, giving reasons for and against your position. You should refer your argument to the quality factors of *extendability*, *efficiency*, *maintainability* and *simplicity* and any others you think are relevant. **[6 marks]**
- (c) The designer is allowed to use an existing class library which already contains classes ARRAY and LINKEDLIST. Discuss whether it would be good sense to use the Adapter pattern for this. Remember that there are two variants of the Adapter pattern. Which one would be more appropriate? **[6 marks]**

QUESTION 5 [15 marks]

Consider the problem of providing a computer application to manage a personal photograph album. The application will allow the user to store a large number of pictures. The pictures can be labelled with captions and with metadata. The album is organised in chunks called chapters, sections, and subsections. The album is stored in a persistent storage device (hard disk) in the form of a group of folders and files. The user application can access the album by navigating and display on a computer screen, through a graphical interface.

The preliminary analysis of the problem has found a small number of requirements for this software application.

Glossary

metadata: Information about a picture such as the photographer's name, date taken, location, description of the subjects in the picture (for example:
photographer="Chris Johnson"; date="15.45 27/9/2003";
location="Nelson, New Zealand"; subject="landscape, river, pinetree")

caption: The title of a picture for example: "A sunny afternoon in Trafalgar St."

Requirements.

1. an album shall contain any number of pictures and album sections, referred to as chapters, sections and subsections.
2. each album section has a title, and can contain any number of pictures and album sections (an album contains chapters, chapters contain sections, sections contain subsections)
3. allow a user to add and edit text for the caption and the metadata of a picture
4. display a table of contents to a chapter, section or subsection, showing titles of album sections and thumbnail images and captions of pictures
5. copy and move pictures, chapters, sections and subsections between album sections
6. enable user to view any picture and to zoom in and out
7. load up to 200 pictures from an external digital camera into a new album item of any kind.
8. allow the user to search for pictures and album sections using a text match to titles, captions and metadata,
9. allow the user to select a picture from a table of contents
10. allow a user to navigate around the album by moving the view to any chapter, section or subsection in any table of contents
11. delete pictures, or whole chapters, sections, subsections
12. save and load an album efficiently between memory and hard disk

13. export selected pictures into external storage formats such as image types that can be used in a Web page or email
14. print an individual picture
15. print all of the pictures in a selected chapter, section or subsection including the pictures in all of the album sections that it contains or encloses
16. change the preferred display format for tables of contents (small or large icon, or lists with details, in order of name or date)

(a) This set of requirements is badly written. Identify the main problems or faults (do not correct the faults—just briefly explain what the faults are). **[4 marks]**

(b) Working by any method such as the methods of use cases, CRC cards, brainstorming or other method of your choice:
List the names *and responsibilities* of a well-chosen group of high level domain classes for this problem. You do not need to show your working. **[7 marks]**

(c) Using some or all of these domain classes, show how these requirements can be reorganised into groups that are better organised.
(Do not write out the requirements in full: you only need to list a name for each group with the original numbers of the specifications that it contains). **[4 marks]**

5 questions, total marks 100