Student ID#: ___________________________

CPSC 410 — Advanced Software Engineering
Mid-term Examination (Term I 2003-2004): Solution
Instructor: Gail Murphy

Do NOT start until you are informed you can start!

This examination has 6 questions. The question and answer space is from page 2 to page 9. Check that you have a complete paper when you are told that you can start the examination.

You have **80 minutes** to complete this exam. Before you start, you must write your student id number on the top of **every** sheet of this exam. ***We will not mark your exam without this information***. You must also sign your name in the space provided below.

Write legibly and in **ink** (not pencil). We also have to be able to read it to mark it.

This exam is closed book and closed neighbour. Notes, book, or other materials are not allowed. Candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action:

- Making use of any books, papers or memoranda, calculators or computer, audio or visual cassette players, or other memory aid devices, other than those authorized by the examiners.
- Speaking or communicating with other candidates.
- Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

If you have a question during the exam, raise your hand and one of the invigilators will come and answer your question. We will only answer questions about errors in the exam. If you have to a make an assumption about a question, write down the assumptions you make.

The exam is out of 100 marks. The value of each question is indicated. Use your time wisely. If you do not know the answer to a question, move on to the next question and come back to the question at the end.

The exam is double-sided. There are questions on **both** sides of the page.

Good luck!

Print Name:___________________________________

Signature:_____________________________________

1.	/20
2.	/10
3.	/20
4.	/10
5.	/10
6.	/30
Total	/100

1. ***(Total: 20 points. 2 points each.)*** True or false with justification. For each statement below, **circle** T if the statement is true or F if the statement is false. In the space provided for each statement, write at most **two** sentences explaining your choice.

(a) A system is designed well if its components are loosely coupled.

[✓ T] or F. Justification:

Loose coupling promotes independence between modules, enabling better use, encapsulating change to parts of the system, etc.

(b) The second tier of a 3-tier client-server system attempts to encapsulate the business logic of an application.

[✓ T] or F. Justification:

The first tier provides mostly GUI functionality, and the third tier provides access to shared data, leaving the second tier to encapsulate business logic and application functionality.

(c) The layered and object-oriented architectural styles share many commonalities.

[✓ T] or F. Justification:

They are both substyles of the call-and-return architectural style.

(d) The batch sequential and pipes and filters architectural styles share many similarities.

[✓ T] or F. Justification:

They are both substyles of the data flow architectural style.

(e) In a system designed according to the pipe-and-filter architectural style, the order in which components shall execute is determined at compile-time.

T or [✓ F.] Justification:

The order may also be determined at run-time (e.g., Unix pipe-and-filters).

(f) The implementation of a software system always conforms exactly to the architecture designed for the system.

T or [✓ F.] Justification:

The need for performance in the system, amongst other reasons, may cause the implementation to drift from the architectural design.

(g) The term stakeholder refers only to the customers of a software system.

T or [✓ F.] Justification:

The term stakeholder refers to anyone interested in the construction of a software system.

(h) A major differentiation between the blackboard and repository architectural styles is that control flow can be initiated from the data to the clients in a system conforming to the blackboard architectural style.

[✓ T] or F. Justification:

When data on the blackboard changes it may notify (usually via an implicit invocation mechanism) clients.

(i) Only one level of plug-ins are supported in the Eclipse framework (i.e., a user-created plug-in cannot declare extension points for use by other plug-ins).

T or [✓ F.] Justification:

A user-defined plug-in may offer extension points that are then used by subsequent plug-ins. Much of the Eclipse platform works this way.

(j) A class in an object-oriented system may play the role of either the Subject or the Observer in an Observer design pattern, but may not play the role of both the Subject and the Observer if the pattern is applied two or more times in the system.

T or [✓ F.] Justification:

One class may play Subject in one application of the pattern, and an observer in the other application of the pattern as the interfaces do not overlap.

2) ***(Total: 10 points) Short Answer*** (3 sentences or less for each).

a) ***(2 marks)*** What is the meaning of the term "module interface" to a software developer?

A module's interface refers to the set of services (e.g., procedures, methods, etc.) that the module provides to client modules.

b) ***(2 marks)*** What is the difference between an extension point and an extension in the Eclipse architecture?

An extension point is a named point (entity) in a plug-in to which other plug-ins may contribute functionality. An extension is a contribution by one plug-in made to an extension point of another plug-in.

c) ***(3 marks)*** Why would a developer choose to use the event notification design pattern instead of the Observer design pattern?

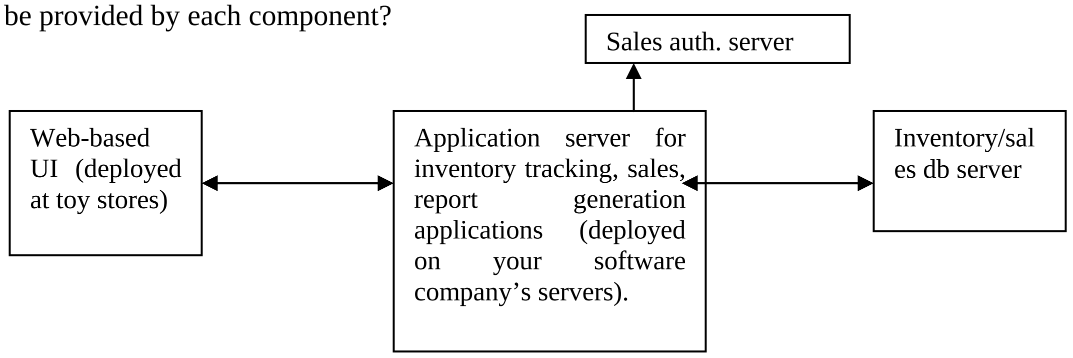
A developer might choose to use the event notification design pattern to simply the dispatch in the Observer to the appropriate handler for an event, or to invoke only those Observers interested in a particular event emitted by the Subject.

d) ***(3 marks)*** On what basis does a software developer select the appropriate architectural styles for a software system under construction?

A software developer will take into consideration the qualities desired in the system to help determine which architectural style to select.

3) ***(Total: 20 points) Client-Server Architecture.*** Imagine your company has decided to introduce a software product that will track the inventory, handle the sales (e.g., authorize credit/debit card payments), and generate reports about the inventory and sales for a collection of small independent toy stores. Your company expects there to be many updates to the system's functionality. The targeted end-users (a toy store and its employees) are not knowledgeable about computers, are spread across Canada, and would like to have a fast turn-around on updates to the functionality of the system.

a) (12 marks) Sketch a client-server architecture for the new software product described above. What will be the major components? How will the components be connected? What functionality will be provided by each component?

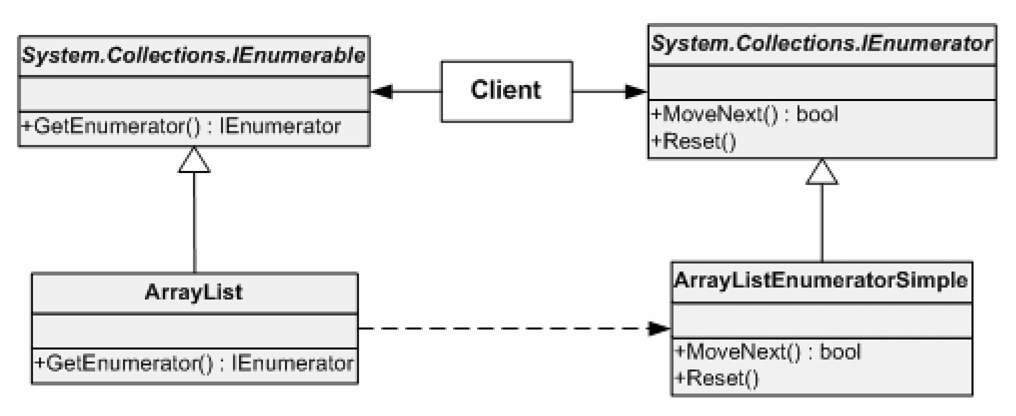

Sales auth. server
Web-based UI (deployed at toy stores)
Application server for inventory tracking, sales, report generation applications (deployed on your software company's servers).
Inventory/sales db server

The major components are indicated in the diagram above. This client-server architecture is 3-tiered and the component. The Web-based UI will connect to the application server via http requests. The application server will connect to the inventory/sales db server via IPC (RMI, etc.); a similar connection will be made to the sales authentication server, which is responsible for authorizing credit and debit card payments.

b) (8 marks) Explain why you chose the client-server architecture you did for part *a* of this question.

A 3-tiered version of client-server architecture was chosen to ensure the product is modifiable and scalable. Modification is supported by hosting the applications on an application server at your company. If an application needs updating, or a variant needs to be introduced, this can be accomplished centrally without visiting each toy store and upgrading software. Similarly, a new kind of payment type can be supported centrally. Scalability is achieved because as new customers are added, the application server, sales authentication server and inventory/sales db servers can be replicated as needed (perhaps even in different sites across the country).

4) ***(Total: 10 points***) We studied three design patterns. A sketch of a class diagram representing part of the .NET framework is given below. The `System.Collections.IEnumerator` interface provides a common way of traversing a collection of objects. The `ArrayList` class is one example of a collection class.


System.Collections.IEnumerable
+GetEnumerator() : IEnumerator
Client
System.Collections.IEnumerator
+MoveNext() : bool
+Reset()
ArrayList
+GetEnumerator() : IEnumerator
ArrayListEnumeratorSimple
+MoveNext() : bool
+Reset()

a. ***(5 points.)*** State which of the design patterns we studied is evident in the class design given above and what parts of the design contribute to those patterns.

The factory method design pattern is evident. The IEnumerable interface is the Creator and the ArrayList is the ConcreteCreator. The IEnumerator is the Product and the ArrayListEnumeratorSimple is the ConcreteProduct.

b. ***(5 points.)*** Explain the role of the pattern you identified in the design (i.e., why was the pattern used?)

The pattern was used to ensure that clients of enumerable data structures can have a common interface for iterating through items in the data structure.

5. (Total: 10 points) Object-oriented Design Fundamentals

a. (3 marks) Many object-oriented designs include aggregations. What is an aggregation and why are aggregations used in an object-oriented design?

An aggregation represents a whole-part relationship between two classes (objects of those classes).

b. (2 marks) Do common object-oriented languages, such as C++ or Java, support the direct implementation of an aggregation? If so, how? If not, how are aggregations implemented?

No. An aggregation must be implemented using a field on at least the class representing the whole part of the relation. This class will typically contain a collection of the associated part objects. A part object may also contain a field that references the object playing the role of the whole.

c. (2 marks) What is a type?

A type defines the set of values that a variable can "hold".

d. (3 marks) Is the ability to define a type hierarchy in an object-oriented language sufficient to support the abstraction of invocation in an object-oriented program? Why or why not?

No. Being able to abstract the invocation (not know exactly who will be called) requires support for polymorphism (and dynamic dispatching) and assignment rules that allow a object of the subtype to be assigned to a variable whose type is that of a supertype.

6. ***(Total: 30 points)*** "The keyword-in-context index system accepts an ordered set of lines, each line is an ordered set of words, and each word is an ordered set of characters. Any line may be circularly shifted by repeatedly removing the first word and appending it at the end of the line. The keyword-in-context index system outputs a listing of all circular shifts of all lines in alphabetical order." (Parnas, 1972)

a) ***(12 points)*** Sketch a software architecture for the keyword-in-context index system that uses the pipe-and-filter architectural style. Briefly describe the function of each component of the system. State any assumptions you make.

A sample solution is at http://www2.cs.cmu.edu/~Compose/html/ModProb/KWICsol4.html

b) ***(12 points)*** Sketch a software architecture for the keyword-in-context index system that uses the implicit invocation style. Briefly describe the function of each component of the system. State any assumptions you make.

A sample solution is at

http://www-2.cs.cmu.edu/~Compose/html/ModProb/KWICsol3.html

(6 points) Compare and contrast the two architectures from parts a) and b) of this question. What are the benefits of one compared to the other? The limitations of one compared to the other?

The components in both solutions are potentially reusable in different applications. The first solution also can be extended easily in some ways by the insertion of new filters (e.g., one could filter out common words like "the" from the index by inserting a new filter). The second solution is likely to be even more extensible than the first as new events can be introduced and new components that respond to those events inserted into more places within the architecture. It is more difficult to reason about, and change the ordering of the compution, in the second solution than the first.