Family Name: _____

Given Name: _____

Student Number: _____

Hand In
answers recorded
on question paper

Department of Computing and Information Science
Faculty of Arts and Science
Queen's University

Software Architecture
CISC 322*

Final Examination

Instructor: T.C. Nicholas Graham

- If you need to make assumptions in order to answer a question, record your assumptions in your answer.

Question 1	/10
Question 2	/10
Question 3	/10
Question 4	/10
Question 5	/20
Question 6	/20
Question 7	/20
TOTAL	/100

Question 1 (10 Points)

Specify the formula normally used to measure availability.

Question 2 (10 Points)

Give two examples of techniques used to improve availability.

Technique 1: _____

Technique 2: _____

Question 3 (10 Points)

Stakeholders in a software development project might include a *systems analyst*, a *system designer/architect*, an *auditor/quality analyst* and a *programmer*.

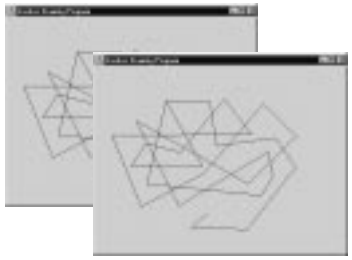
For each of the following project tasks, identify the phase of the waterfall model in which the task would most likely be carried out, and identify the stakeholder most likely to perform the task.

Project Task	Phase of Waterfall Model	Stakeholder carrying out task
Assess risk that the proposed design will not satisfy quality attributes specified in the requirements document		
Talk with customers to determine their needs		
Connect a commercial database, a web server and custom components to complete the system		
Construct a test harness for a component		
Change a program to be XML compliant		

Question 4 (10 Points)

Recall that in the *PAC* (Presentation-Abstraction-Control) architectural style, interactive programs are composed of a tree of PAC components. Each component is split into *Presentation, Abstraction* and *Control* facets.

We wish to implement the following user interface using the PAC architectural style.



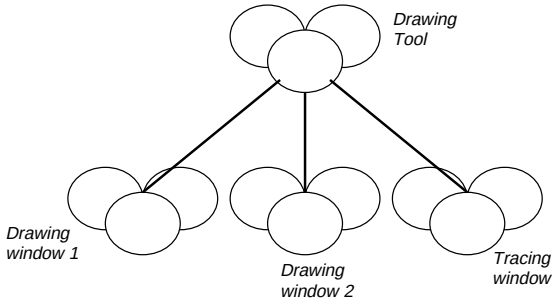
Drawing Windows

```
% java Main
Got line: (130,169)->(365,184)
Got line: (365,184)->(274,56)
Got line: (274,56)->(195,202)
Got line: (195,202)->(238,361)
Got line: (238,361)->(344,200)
Got line: (344,200)->(235,222)
Got line: (235,222)->(126,279)
Got line: (126,279)->(364,360)
Got line: (364,360)->(337,227)
```

Tracing Window

In this program, two *drawing windows* permit users to draw lines. Clicking twice in the window draws a line between the two locations clicked. Drawing a line in either window causes the line to be displayed in *both* windows. In addition, for each line drawn, a message is shown in a tracing window giving the precise coordinates of the line.

The following PAC architecture can be used to construct this program:



On the following page, briefly describe the function of the facets of the specified components in this architecture.

Drawing Tool Component:
Presentation Facet:

Abstraction Facet:

Control Facet:

Drawing Window 1 Component:
Presentation Facet:

Abstraction Facet:

Control Facet:

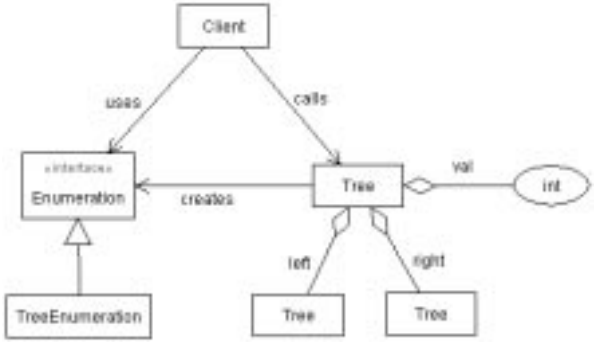
Tracing Window Component:
Presentation Facet:

Abstraction Facet:

Control Facet:

Question 5 (20 Points)

The following class diagram describes a binary tree and an associated iterator. Tree elements contain an integer value, and may have *left* and *right* subtrees. Upon request from a client, a tree element may create an iterator allowing each element of the tree to be visited.



Recall that in Java, iterators implement the *Enumeration* interface. This interface is defined as:

```
public interface Enumeration {

   // Returns true as long as there are more
   // elements to visit
   public boolean hasMoreElements();

   // Returns next element to visit
   public Object nextElement();
}
```

The *Tree* class provides the following method to create an enumeration:

```
Enumeration elements();
```

Part A (10 Points): Implement the *Tree* class in Java. Provide the *elements* method as well as appropriate methods to traverse the tree and extract its value.

Part B (10 Points): Implement the *TreeEnumeration* class in Java.

Question 6 (20 Points)

An inventory contains a number of *items*. Items may be *books* or *compact discs* (CD's). All items have a title (a string). In addition, books have an author (a string), and CD's have a list of track names. Track names are simply strings.

Items may be created through an item factory. This factory should be based on an interface called *ItemFactory*, with implementations responsible for creating Book and CD objects.

Part A (10 Points): Provide a class diagram showing the structure of items in the inventory and the structure of the factories used to create items. Show the inheritance structure amongst the classes, properties implemented by the classes, and any other interesting relationships you wish to highlight amongst the classes.

Part B (10 Points): Provide implementations in Java for the item abstract factory and the concrete factories for CD's and books.

Question 7 (20 Points)

A central service exists for people to call in case of emergency. For example, when a member of the public sees a building on fire, they call the emergency number (e.g., 911) and speak to an emergency operator. An emergency vehicle (i.e., a fire truck) is dispatched to the scene of the emergency.

In building a system to support this service, assume we create the following components:

- A *Telephone Event Generator* is attached to the phone system to provide information about emergency calls from the public. The telephone event generator provides events: when a new call is started, the telephone event generator fires an event with the start time of the call and the phone number of the person making the call. When a call ends, the telephone event generator fires an event specifying that the call is complete.
- A *Call Entry* component is used by the emergency operator to create a *call record* containing information about the call. A call record contains: the start and end time of the call, the name and phone number of the caller, a description of the problem (a string), the address of the emergency, and the kind of vehicle to be dispatched (fire truck, police car or ambulance).
- A *Vehicle Location* component contains a list of all vehicles available, and their current location.
- A *Scheduler* component is responsible for determining which vehicle to dispatch to an emergency. The scheduler is given a call record, and returns a vehicle identifier.
- A *Dispatcher* component sends a message to a vehicle specifying what address the vehicle is to go to next.
- A *Log* component logs all call records and dispatch information.

Part A (10 Points): Provide an architecture diagram for the emergency system described above. Your diagram should contain the six components described above, and appropriate connectors between these components. Provide a legend explaining the symbols you use in the diagram.

Part B (5 Points): Specify the quality attributes that you feel are most important to a system such as this.

Part C (5 Points): Identify any areas in this architecture where there are risks of the quality attributes of part B not being satisfied.