Assignment 1

ENGI 5895 Software Design 2016
Electrical and Computer Engineering
Memorial University

Due Mon Jan 25 at 11:30PM.

Q0 [10]. Domain analysis.
Create a class diagram describing the following information.

The library has a number of patrons and items. Each item is classified as one of the following: A book, a periodical issue, a periodical volume, or an electronic resource. All items have a unique acquisition number and a call number. Each item is in one of the following states at each point in time: borrowed, shelved, on hold, returned, missing. Each patron has a unique library card number, and PIN. Patrons also have contact numbers. At each point in time, each patron has a number items that they have borrowed. When an item is borrowed by a patron, there is a due date. The following are system invariants:

- Every item that is in the borrowed state is borrowed by one patron.
- Every item that is not in the borrowed state is borrowed by no patron.
- Borrowed items (and only borrowed items) have a due date and a list of recall notices.

Show all relationships and attributes. Operations need not be shown for this question. Show multiplicities on all associations. Use notes (comments) to express any constraints (i.e., invariants) not expressed with UML notation.

Q1 [15]. Software Design. Mathematically, a finite ordered forest is a structure (V, r, c), where V is a finite set of nodes, r is a sequence of nodes, $c(v)$ is a sequence of V, for each $v \in V$. The nodes on sequence r are called *roots*. If u is on $c(v)$, we say u is a *child* of v and that v is the *parent* of u. We further require that

- no node occurs more than once on r (no duplicate roots),
- no node occurs more than once in any sequence $c(v)$ (no duplicate children),
- each member of r is a child of no node (roots are not children),

- every node not on r is a child of exactly one node (nonroots have one parent),
- the nodes can be numbered such that the nodes on r are numbered 0, and each child's number is one more than its parent's (no cycles).

These structures and structures like them are very useful. For example an HTML or XML document is an ordered forest; so, in an abstract sense, is source file for a computer program; so is a JSON object.

(a) Draw a UML class diagram representing a set of classes and relationships for representing finite ordered forests. Your classes should be mutable, meaning that, at each point in time, the state of a forest object will represent a finite ordered forest, but over time the same object may represent different finite ordered forests.

(b) Elaborate your class diagram by adding attributes and operations as needed. Each node should have a label which is a string. The operations should provide a convenient interface to access the information in a forest object and to change the state of a forest object. The operations should not allow an object to represent something that is not a finite ordered forest (i.e. invariants should be enforced).

(c) Make a sequence diagram that illustrates a sequence of messages a client can use to create a forest object that represents

$$(\{u, v, w\}, r, c)$$

in which $r = [u, v, w]$ and $c(u) = c(v) = c(w) = []$. Your diagram should go on to show show a sequence of operations that mutates the same object until it represents a forest

$$(\{u, v, w\}, r', c')$$

in which $r' = [u]$, $c(u) = [v, w]$, and $c(v) = c(w) = []$. Finally your diagram should go on to show a sequence of operation that mutates the same object until it represents a forest

$$(\{u, v\}, r'', c'')$$

in which $r'' = [v]$, $c(v) = [u]$, and $c[u] = []$.

(More to think about: Although I'm not asking you to hand anything in, you might want to think about how to represent finite ordered forests using a set of immutable objects in such a way that it is convenient to obtain an object representing a forest from an object representing a similar forest. E.g. it should be reasonably easy to compute a forest f' that is similar to forest f except that it is missing one leaf node.)

Q2 [15] Widgets

(a) Create a class diagram for the following set of classes. Include all operations. A widget represents some sort of visual component, such as buttons, text boxes, images, or panels. Panel widgets may contain other widgets (even other panels) as children. There are in fact two kinds of panel widgets. Vertical

panels widgets stack their children vertically, while horizontal panels stack their children horizontally.

Each widget can be queried as to its width, height, x position and y position. Widgets can have their x and y positions set. The x and y positions are relative to the parent of the widget.

A 'lay out' message sent to a widget causes it to first lay out its children (if any) and then to position its children (if any) relative to the widget.

A 'paint' message sent to a widget causes it to draw itself to the screen and then to send 'paint' messages to its children.

(b) Make an object diagram (see Fowler's book for more on object diagrams) showing an panel containing at least two widgets.

(c) Based on your object diagram, make a sequence diagram showing what happens when a 'lay out' message and then a 'paint' message are sent to the top-level panel of your object diagram.

Submitting the assignment.

This is an individual assignment.

Submit as a PDF file using D2L. You are encouraged to use VP to create the diagrams.