CSC 207 summer 2007 – Midterm 2

Tuesday July 17, 2007

Duration: 60 minutes

No Exam Aids Allowed (A tips sheet is provided on page 14)

You must write in **pen** if you wish the option of having your test remarked after it is handed back.

Last Name:_________________________ **First Name:**______________________

Student No. : _______________ **(Please write name & student no. on every page)**

Do **not** turn this page until you have received the signal to start. In the meantime, please fill out the identification section above, and read the instructions below carefully.

This midterm consists of 5 questions (numbered 0 to 4) on 15 pages *(including this one)*, printed on **both sides** of the paper. Pages 5, 7, 9 and 11 are intentionally left blank for your answers to the designated questions. When you receive the signal to start, please make sure that your copy of the examination is complete.

Answer each question directly on the examination paper, in the space provided. Pages 2 and 14 are intentionally left blank for your side work. If you use them, please indicate **clearly** the part of your work that should be marked.

In the programming questions, you do not have to write pre-conditions/post-conditions. Also, you do not have to comment your code but use **meaningful** variable names. You are allowed to use helper procedures.

Be aware that concise, well thought-out answers will be rewarded over long rambling ones. Also, unreadable answers will be given zero (0) so please write legibly.

0.	________________ / 3	(Your Name & ID on every page)
1.	________________ / 28	(Miscellaneous)
2.	________________ / 24	(Software Design)
3.	________________ / 35	(Python)
4.	________________ / 10	(make)

______________ **/ 100 TOTAL**

GOOD LUCK!

This page is intentionally left blank for your side work.

Question 0. [3 mark]
Write your first initial and last name, and your student number on the space provided at the top of every page. Take a deep breath and have fun!

Question 1. Miscellaneous Questions [28 marks total]

(a) [2 marks] What does the X in XML stands for?

Extensible (either 2 or 0).

(b) [2 marks] Which of these tools we have not covered yet in the class: Programming Languages, Scripting Languages, Integrated Development Environment, Profiling Tools, Version Control App, Quality Assurance Framework, Software Build Management Framework, Variance Tracking App, Architecture Tools?

Profiling tools (either 2 or 0)

(c) [2 marks] List the states of a variance (*i.e. the states a variance is assigned from it's discovery to fixing*).

open, assigned, in-progress, on-hold, waiting retest, closed, rejected (either 0,1,2 depending on how many is identified).

(d) [8 marks] Will the following regular expression match the input?

Pattern	Input String	Match?
a*	apple	Yes
^a+	aab	**NO**
a[^bc]?	abcbc	**NO**
(\w)*\d\w	john_2007	**NO**
a*b	baaab	**NO**

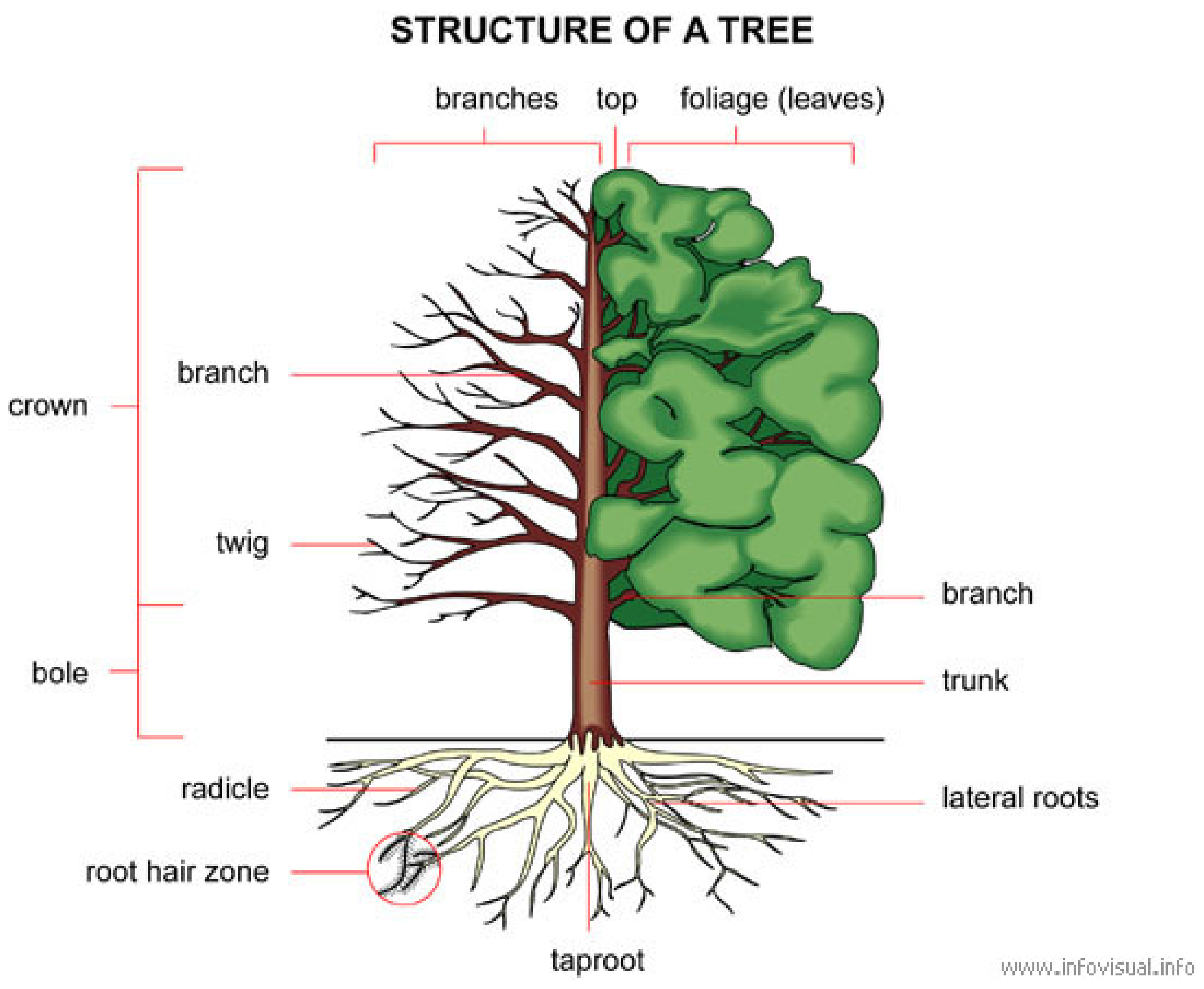
(2 for each)

(e) [4 marks] In a typical software project, there are several types of testing. List those types and specify their order in the project lifecycle.

Unit -> DIT -> SIT -> UAT (2 for correct test names, 2 for correct order)

First Initial & Last Name: ______________________________ **Student No.:** ____________

(f) [10 marks] Write a valid XML-based representation for the components of the following illustrated tree. To save you time: you may abbreviate the tag names you are going to use from the illustration (hint: preserve the composition of the tree as indicated in the illustration).



This page is intentionally left blank for answer to question 1 (f)

- [2 marks] show understanding of the nesting of xml elements
e.g. Separate between below & above ground
nest root elements inside some root element
divide between bole & crown

- [2 marks] Show understanding of the need to identify elements since they repeat either by some id attribute or other means

- [2 marks] Show understanding of the need to identify elements location since they repeat either by some id attribute or other means

- [2 marks] Show understanding of the need to have elements without body because there is no children e.g. <bole/>

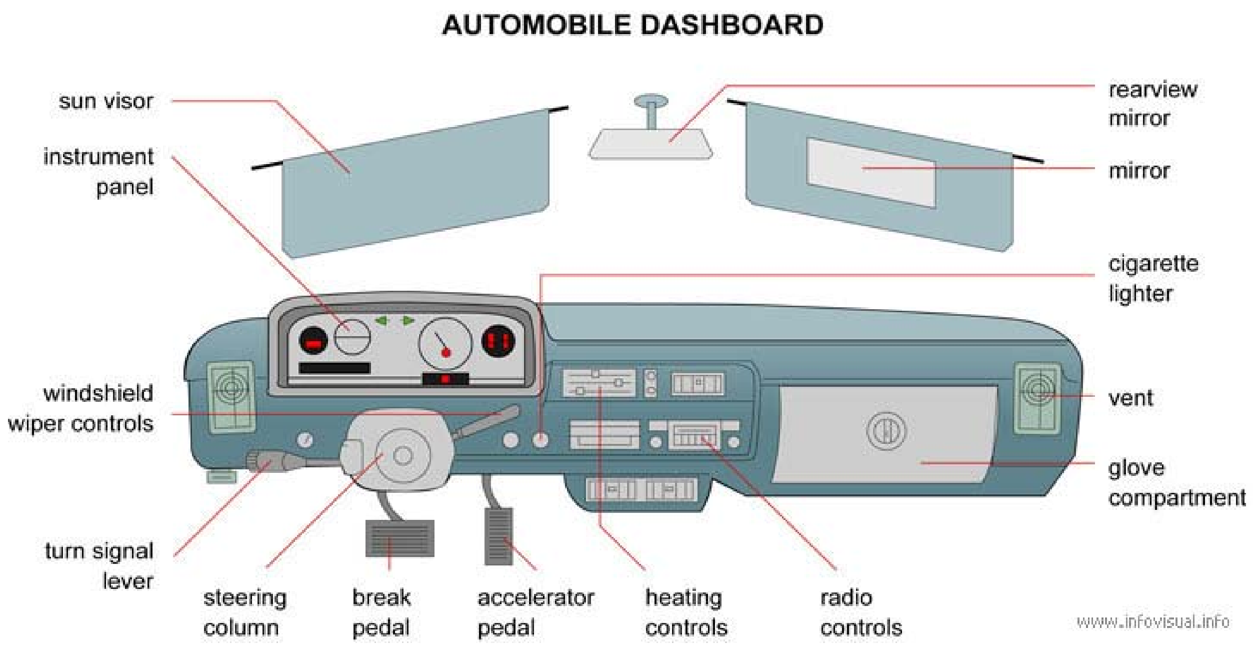
- [2 marks] Attempt to identify elements that are not mentioned in the illustration e.g. trunk height, trunk width

sample answer:

```
<tree>
        <body>
                <trunk>
                        <bole height="" min-width="" max-width="" />
                        <crown height="" min-width="" max-width="">
                                <branch id="1">
                                        <twig id="1">
                                                <leaf id="1"/>
                                                <leaf id="2"/>
                                        </twig>
                                </branch>
                        </crown>
                </trunk>
        </body>
        <roots>
                <root id="1" type="tap">
                        <radicle id="1">
                                <hair id="1"/>
                                <hair id="2"/>
                        </radicle>
                </root>
        </roots>
</tree>
```

Question 2. Software Design [24 marks total]

(a) [14 marks] Assuming you are building a software that models an automobile dashboard. Build a UML class diagram to identify the classes and interfaces in the following illustration of a typical automobile dashboard. State the relations between classes (*whether it is composition, aggregation or inheritance*). You do not need to include the expected methods in each class unless it serves to clarify your design decisions. To save you time: you may abbreviate the class names, e.g. SV for Sun Visor, RC for Radio Controls, etc…and you may simplify the UML class notation to a rectangle without the attributes and methods sections. (*Check tips page for UML notation).*



This page is intentionally left blank for answer to question 2 (a)

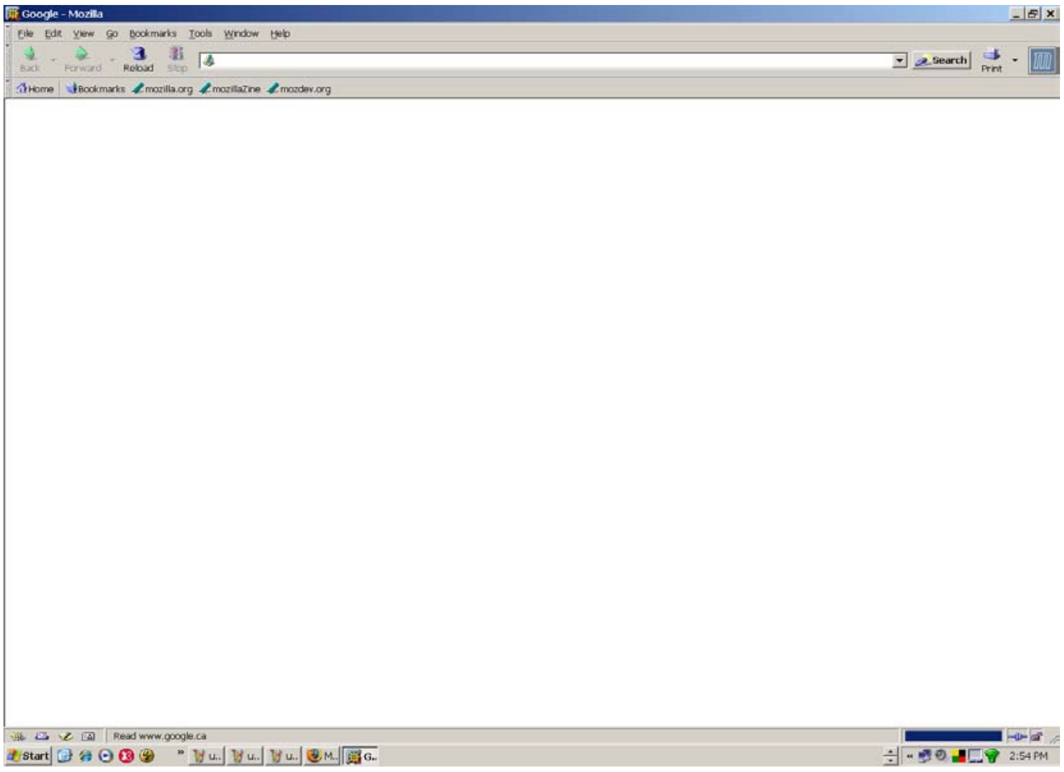
[5 marks] Show understanding of the hirarchey of the classes.
e.g. - Steering_device consists of steering_wheel, steering_column, windshield_wiper_controls, turn_signal_lever
- dashboard consists of steering_device, instrument_panel, heating_controls, radio, cigarette_ligher, glove_compartment, vent, break_pedal, accelerator_pedal

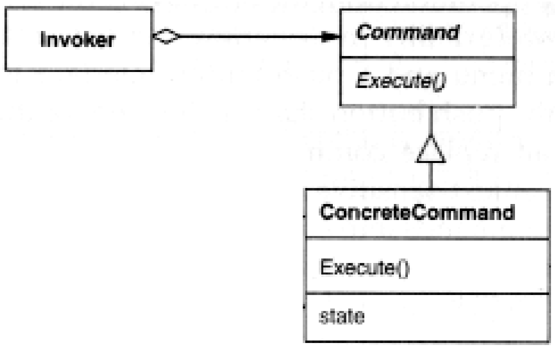
[3 marks] Show understanding that rearview_mirror and sun_visor are not part of the dashboard. It doesn't mean it is mentioned in the illustration that it is part of a dashboard. However, including it as attribute in a higher class as car is ok. So, Car class consists of dashboard, sun_visor and rearview_mirror

[3 marks] Show an understanding that there are entities not described in the illustration
e.g. - instrument_panel has instruments (not mentioned in illustration)
- radio (not mentioned in illustration) has radio_controls

[3 marks] Show an understanding of the difference between uses and has relations.
e.g. - dashboard has instrument_panel, steering_device
- dashboard uses radio and heating_controls and vent since these are not mandatory for the functionality of a dashboard which is to drive a car.

(b) [10 marks] Assuming you are designing your own browser and you are using the command design pattern, write the names of 10 classes which would inherit from a command interface in such an application (*hint: each of those classes, will encapsulate the implementation of a typical action in a web browser*).


Google - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
Search
Print
Home Bookmarks mozilla.org mozillaZine mozdev.org
Read www.google.ca
Start
2:54 PM



First Initial & Last Name: ____________________________ Student No.: ____________

This page is intentionally left blank for answer to question 2 (b)

(1 mark each)

sample answer:

BackwardCmd	**(to go to previous webpage)**
ForwardCmd	**(to go to next page)**
PrintWebPageCmd	
CloseBrowserCmd	
SaveWebPageCmd	
SendPageByEmailCmd	
NewWindowCmd	**(to open new browser window)**
LoadPageFromInternetCmd	
LoadPageFromHarddiskCmd	
FindOnPageCmd	

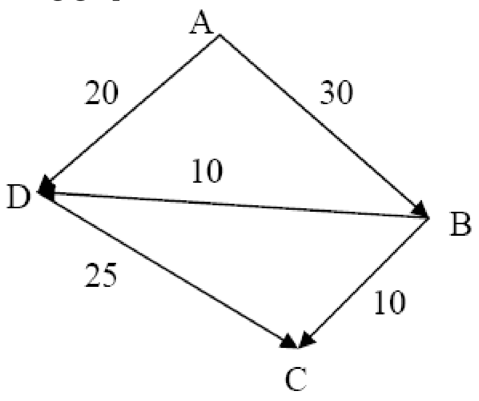
Question 3. Python [35 marks total]

(a) [25 marks] In this question, you will be working with Python dictionary:

Consider a dictionary structure that contains information about a directed graph that does not have any circuit/cycle:

```
{
‘A’: [’B’, 30, ‘D’, 20],
‘B’: [‘C’, 10, ‘D’, 10],
‘D’: [‘C’, 25],
‘C’: []
}
```

This represents the following graph:



As you can see, each key in the dictionary is a single letter that represents the starting vertex. Its corresponding value is always a list that contains all destination vertices together with the arc length. Note that a vertex is in this list if and only if there is an arc from starting vertex to this vertex. For example, the first key-value pair should read “The length of arc A--B is 30, the length of arc A--D is 20.

Write a python function “pathLength.py” that takes two arguments: a list that contains a path on the graph, and a dictionary such as the one above. The program should return the length of this path, if such path exists. Otherwise, it should print -1. For example, let the dictionary above be called g1, then

```
If __name__==”__main__”:
g1={ ‘A’: [’B’, 30, ‘D’, 20], \
‘B’: [‘C’, 10, ‘D’, 10], \
‘D’: [‘C’, 25], \
‘C’: [] }
Length1 = pathLength([“A”, “B”, “D”, “C”], g1)
Length2 = pathLength([“A”, “C”], g1)
Length3= pathLength([“A”,”A”], g1)
print length1, length2
```

should output:

```
65 -1 0
```

Note that you may use as many helper functions as you want.

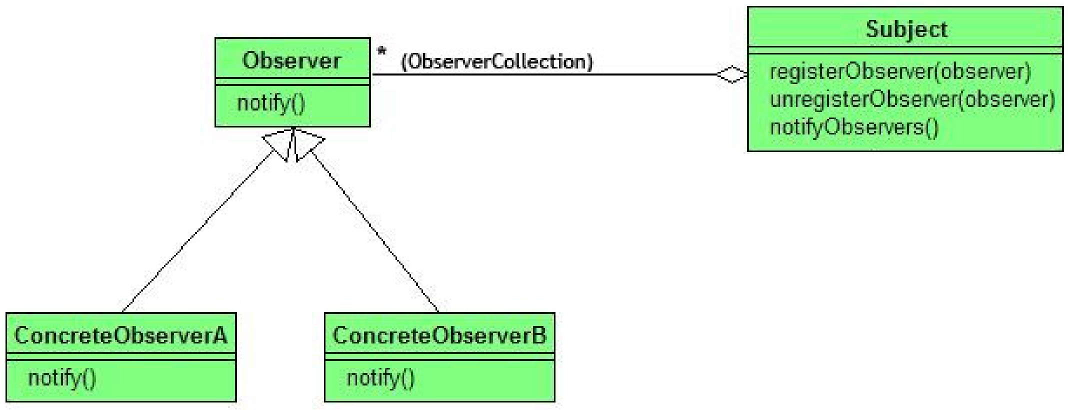
This page is intentionally left blank for answer to question 3a

```
0 marks for no solution or showing no understanding of the
subject matter
-5 mark per serious error
-5 mark for serious style errors (not per error, but overall)
-5 marks for serious syntax errors (not per error, but overall)
```

sample answer:

```
def pathLength(path, g1):
    length=0
    for i in range(len(path)-1):
        currNode=path[i]
        if currNode in g1:
            myValue=g1[currNode]
            nextNode=path[i+1]
            if currNode==nextNode: ## same nodes in a row
                length+=0
                continue
            for j in range(len(myValue)) :
                if (nextNode==myValue[j]):
                    length+=myValue[j+1]
                    break
            # nextNode cannot be at the last position
            if (j==len(myValue)-1) :
                return -1
            else :
                return -1
    return length
```

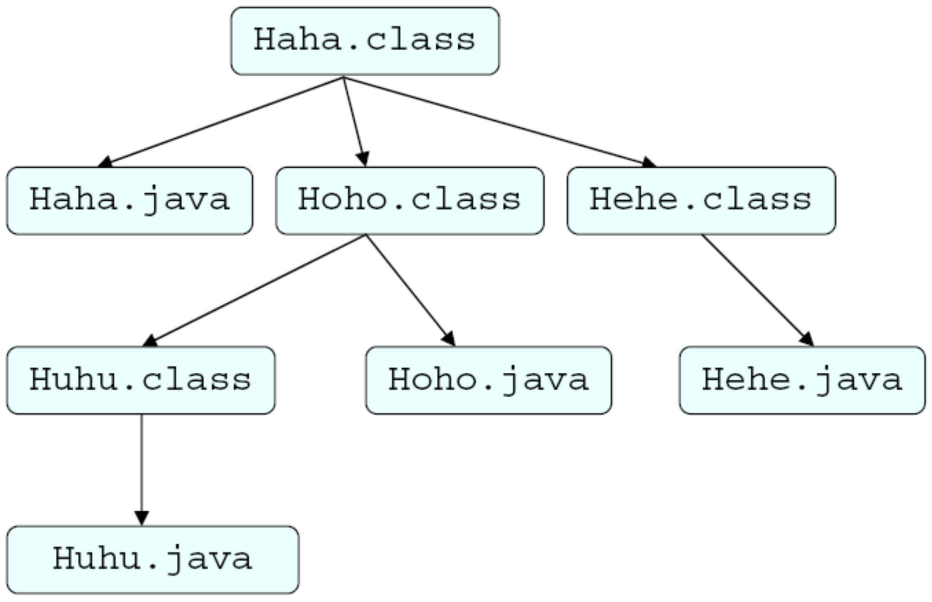
(b) [10 marks] The following UML illustrates the observer design pattern. Provide a python implementation. The code snippets at the bottom are provided to guide you to the classes and methods that you need to implement.



```
class Listener:
    def __init__(self, name, subject):
        self.name = name                                      [3 marks]
        subject.register(self)

    def notify(self, event):
        print self.name, "received event", event             [1 mark]

class Subject:
    def __init__(self):
        self.listeners = []                                   [1 mark]

    def register(self, listener):
        self.listeners.append(listener)                       [1 mark]

    def unregister(self, listener):
        self.listeners.remove(listener)                       [1 mark]

    def notify_listeners(self, event):
        for listener in self.listeners:                       [3 marks]
            listener.notify(event)


subject = Subject()
listenerA = Listener("<listener A>", subject)
listenerB = Listener("<listener B>", subject)
# the subject now has two listeners registered to it.
subject.notify_listeners ("<event 1>")
# outputs:
#    <listener A> received event <event 1>
#    <listener B> received event <event 1>
```

Question 4. make [10 marks total]

Suppose you are building a java project. The relationships of dependencies are drawn below:



When running haha.class, the output will be redirect to a file called output.txt. Complete the following make file. **You must enable –ea option when running each java program.**

```
JC = javac –source 1.5
JR= java –ea                                                [1 mark]

run: Haha.class
        @echo Running Haha.class
        ${JR} Haha > output.txt                             [1 mark]

Haha.class : Haha.java Hoho.class Hehe.class                [2 marks]
        ${JC} Haha.java (this line can be omitted)

Hoho.class : _Huhu.class Hoho.java                          [2 marks]
        ${JC) Hoho.java (this line can be omitted)

%.class: %.java                                             [2 marks]
          ${JC} $<

### clean rule will erase all class files and append output.txt
### to the end of a file called oldOutput.txt
clean :
        rm *.class                                          [2 marks]
        cat output.txt >> oldOutput.txt
```

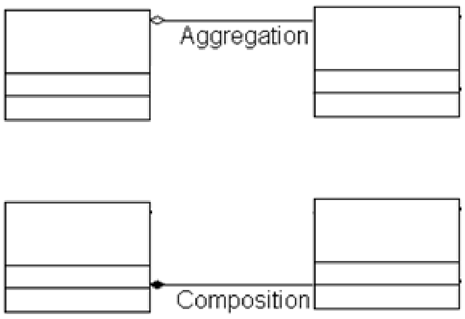
This page is intentionally left blank for your side work.

End of Midterm 2

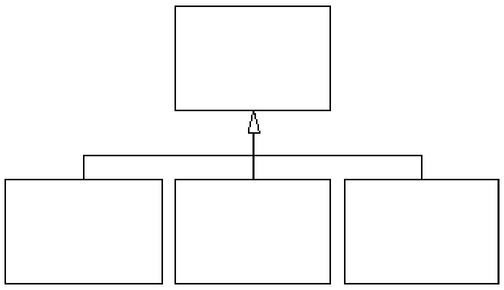
<u>Tips Sheet</u>

UML notation example:

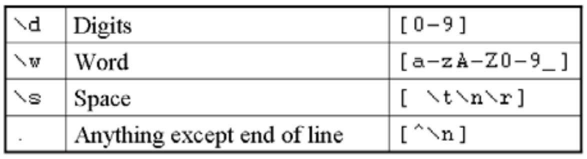
Composition/aggregation



inheritance



Regular expression:



\d	Digits	[0-9]
\w	Word	[a-zA-Z0-9_]
\s	Space	[\t\n\r]
.	Anything except end of line	[^\n]