Faculty of Engineering, Alexandria University
CS 221: PROGRAMMING-2
Dept. of Computer and Systems Engineering

Year: 2nd year
Duration: 180 Minutes
Final Exam: 2013

Read the following scenario CAREFULLY then answer ALL SEVEN questions:

WordNet® is a large lexical database of English language. It consists of *words* and *synsets*. For each word, an internal identifier, a lemma[1] and the most frequently used POS are defined. A word can have more than one part-of-speech (POS). Valid WordNet parts-of-speech are *noun*, *verb*, *adjective*, and *adverb*. For example, the word "table" can be either:

- a noun:
  - as in the table, we sit on, or
  - a database table
- a verb: "to table" is an English verb meaning:
  - putting numbers into a table.
  - to delay or postpone

Words are grouped into sets of synonyms (*synsets*). Each synset expresses a distinct concept. Concepts are independent entities in WordNet. For example, one sysnet that is associated with the word table is *tabulary array* which is a set of data arranged in rows and columns. The word *matrix* (in the POS-form: *noun*) is associated with the same synset. Each synset contains an internal identifier and a brief definition called *gloss*. For example, the gloss for the "tabulary array" synset is "a set of data arranged in rows and columns".

Synsets are interlinked by means of two types of links: *conceptual* and *lexical* relations. WordNet® labels the conceptual relations giving them different types. The most frequently encoded relation among synsets is the IS-A relation. For example, the IS-A of the *tabulary array* is the sysnet *general array*. Other conceptual relations include meronym (a term which denotes part of something but which is used to refer to the whole of it), antonym (a word opposite in meaning to another), and synonym (a word with the same or similar meaning to another word).

Lexical relations are similar to conceptual relations but deal with the lexical nature of the synset. Typical lexical relations are: "verb group" and "lingual derivation".

Question 1 (20%):
Draw the UML class diagram for WordNet. All attributes mentioned in this problem description scenario must be present in the diagram. Make all necessary assumptions.

Question 2 (10%):
Map the UML class Diagram of Question 1 into a code skeleton written in JAVA. All associations between classes must be present in the code. Make all necessary assumptions.

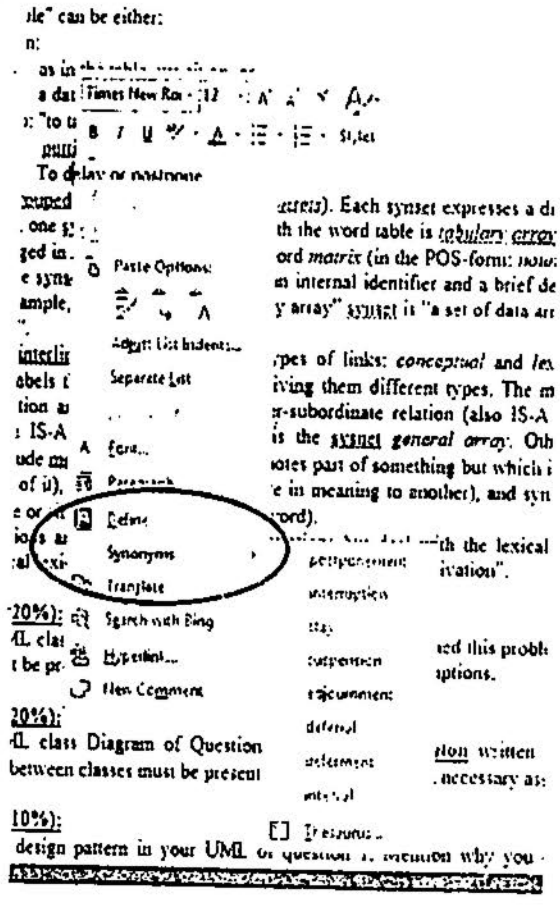
Question 3 (10%):
Identify one design pattern in your UML of question 1. Mention why you employed this pattern.

Having represented WordNet in the previous part, now we want to use it with word processors like MS Word. For example, by right clicking a word, its synonyms should appear. Our new extension of MS Word would show antonyms, and meronyms as two additional options in the popup-window.

[1] A lemma in this context is usually the base form for a word. E.g., the base form of the word *capricious* is *caprice*

Khaled Nagi

N. El-Mahdy




Therefore, we need *one* instance of the WordNet lexicon. The content is loaded into memory from a set of comma-separated text files during the first instantiation. The location of these files should be defined in a Settings-holding class that is read from a properties file. It would be good to hide the implementation details of WordNet from the word processor. Therefore, one class should provide all method calls to WordNet, such as string[] getMeronym(string word, string POS).

Question 4 (20%):
Draw the UML class diagram for this part. Identify two design patterns and mention what they enforce in this system.

Our word processor is designed in an elegant way to allow us to extend it arbitrary. During its startup, it reads the following:

- the names of classes to use to populate the pop-menu.
- the method calls to be executed when the user chooses

Question 5 (15%):
Design this properties file and draw a code skeleton for:

a. populating the menu
b. executing the action of the corresponding right-click button.

Yet, the word processor is not equipped with a POS-detector. When the user right-clicks on the word "table" and chooses "Meronym", the word processor cannot send the POS to the method string[] getMeronym(string word, string POS) as its known interface is just string[] getMeronym(string word). As a workaround, we suggest to take the most frequently used POS for the word and pass it to the string[] getMeronym(string word, string POS).

Question 6 (15%):

a. Which design pattern would you use for this workaround?
b. Draw the UML diagram for this part of the solution.

Question 7 (10%):
Draw the sequence diagram between the RightClickPopMenuObject, the WordProcessorCore and the WordNetObject.

GOOD LUCK

N. El-Makky

2/2

Khaled Nagi