Com S 362
Fall 2002

Object-Oriented Analysis and Design

Exam 3: Analysis, Design, and Implementation

This test has 4 questions and pages numbered 1 through 8.

Exam Process

Question 1 can be done at any time, and should be turned in at the end of the test along with all of the front matter in the test.

There is a use case and system sequence diagram for the remaining questions following the first question.

Starting with question 2 on this exam, each question builds on the answer from the previous question. To aid grading and to prevent you from getting too far off track, when you complete an answer for one question, you will trade your answer to that question for our standard solution for that question. You must then use our standard solution for this question when answering the next question. For example, when you finish question 2, you will trade in your answer for question 2 for our standard solution to question 2, and then you will then use our answer to question 2 to solve question 3. Similarly, you will use our answer for questions 2 and 3 to solve question 4, etc. *For these questions, be sure to put your name on each page you hand in!*

Reminders

This test is open book and notes. However, it is to be done individually and you are not to exchange or share materials with other students during the test. So if you have materials on your team project you wish to refer to during the test, please make copies.

If you need more space, use the back of a page. Note when you do that on the front.

This test is timed. We will not grade your test if you try to take more than the time allowed. Therefore, before you begin, please take a moment to look over the entire test so that you can budget your time.

For diagrams and programs, clarity is important; if your diagrams or programs are sloppy and hard to read, you will lose points. Correct syntax also makes some difference.

1. (10 points) This is a problem about the GRASP design patterns and principles. For each part, write down the name of the GRASP design pattern or principle that would be most useful for addressing the situation described.

   a. You want to assign responsibility for handling a system event, but don't want to have the user interface be tightly coupled to classes in the application logic layer. What pattern would best support this?

   b. You are building a game program which features several types of characters (i.e., kinds of simulated people) with different properties. Yet all types of these types of characters support some common methods; for example, they all can be asked to move in the game board, pick up things they find, and report their status. You want to avoid coupling the code that does such common actions to the different classes that implement these types of characters. What pattern would best support this?

Use Case for the following problems

Use Case: Create Text Document

Primary Actor: Author

Stakeholders and Interests:

- Author: wants non-tedious way to create a text document, with spell checking.

Success Guarantee (Postconditions): A text document is saved.

Main Success Scenario (or Basic Flow):

1. Author asks for a document to be created.
2. System creates a document with a single empty paragraph, and positions the insertion point at the beginning of the paragraph.
3. Author asks to insert a non-blank and non-punctuation character (e.g., a letter) after the insertion point.
4. System places the given character immediately following the insertion point, and moves the insertion point past the given character.

   Author repeats steps 3-4 until Author indicates the end of a word with a blank or punctuation character.
5. System confirms the word's spelling, then places the blank or punctuation character following the insertion point, and moves the insertion point past the given blank or punctuation character.

   Author repeats steps 3-5 until Author indicates end of a paragraph.
6. System places a new paragraph immediately following the insertion point, and then moves the insertion point to the start of the new paragraph.

   Author repeats steps 3-6 until Author indicates done with this document.
7. Author asks to save the completed document, giving a file name.
8. System saves the document in the given file.

Extensions (or Alternative Flows):

*a. If, at any time, Author wants to change (i.e., replace) part of the document:
   1. Author asks to change a character, word, or paragraph.
   2. System shows the replacement in the place where the original character, word, or paragraph was.

*b. If, at any time, Author wants to delete part of the document:
   1. Author asks to delete a character, word, or paragraph.
   2. Author removes the given character, word, or paragraph from the document.

5a. If the word is misspelled
   1. System suggests various corrections.
   2. Author accepts a correction
      2a. If the Author does not like any of the corrections:
         1. Author indicates that the word should be considered correctly spelled.
         2. System records the word in the Author's dictionary.
   3. System shows the correct word in place of the misspelled word.

8a. If the document cannot be saved in the given file.
   1. System tells the Author about the error.
   2. Author tells the system another file name.
      2a. If the Author does not want to try saving the document in a file:
         1. The Author tells the system to stop trying to save the document.
         2. Use case ends.
   3. Use case continues from step 8.

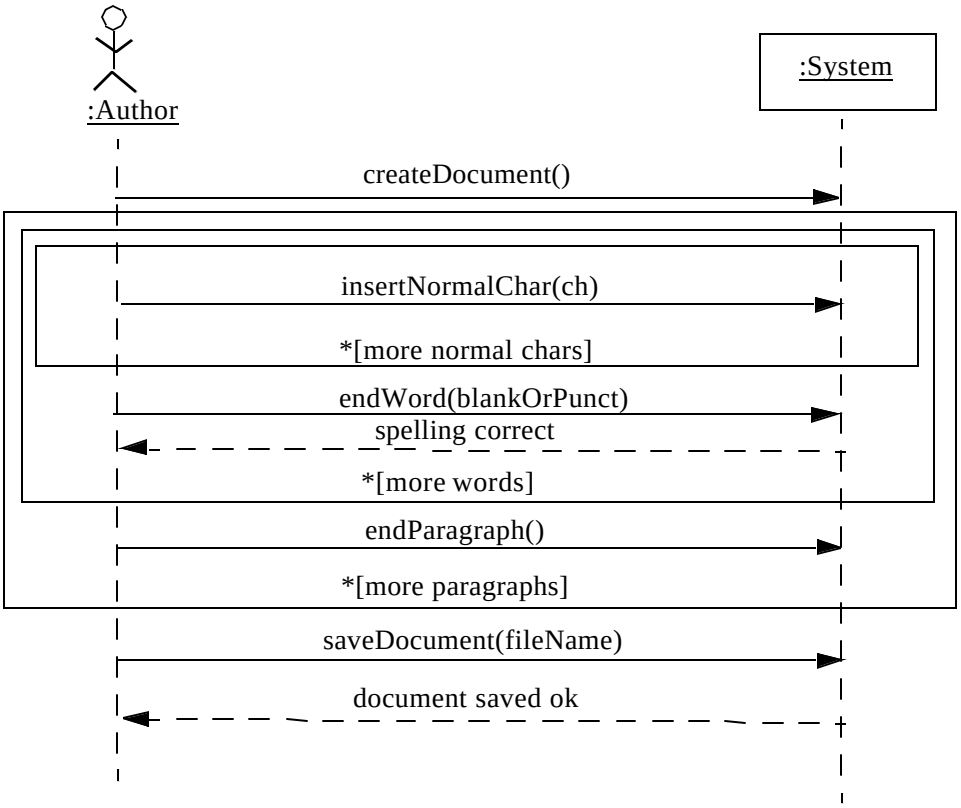
Special Requirements

- Interface with external spell checkers

Frequency of Occurrence: one time only per run of the program.

Open Issues: More flexible spelling correction; Internationalization.

The following is a system sequence diagram (SSD) for the main success scenario of the above use case.



2. (30 points) In this problem you will write a domain model, i.e., a conceptual class diagram, which should be relatively complete for the above use case. It should include associations and any attributes that are useful for the application logic layer. Consider the entire use case. However, do not include conceptual classes for external systems or for classes that would be expected to be in the programming language.

   Hint: think about how the system can know where to insert characters. A document should not simply be an array of characters; model the concepts in the domain!

3. (30 points) In this question you will do the design for the `createDocument` system operation from the system sequence diagram found on page 4.Your design should be based on the standard solution for the previous problems.

   Your designs are to be recorded using interaction diagrams. You may use either the UML sequence diagram or collaboration diagram notation.

   You need only show the design patterns or principles are used to assign responsibilities that are *different than* Creator or Information Expert.

 Name_____

4. (30 points) Based on the standard solution for the previous problems, draw a UML design class diagram that summarizes the design of the two system operations `createDocument` and `insertNormalChar`. (That is, you *don't* have to include classes, attributes, or methods needed to implement other system operations. In particular, you don't have to deal with classes for spell checking words and writing out files.) Include types for all method arguments and results (for methods that have results). Don't include classes that are part of Java's built-in libraries.