Com S 362
Spring 2003

Object-Oriented Analysis and Design

Solutions to
Exam 3: Analysis, Design, and Implementation

This test has 5 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. (5 points) This is a problem about the GRASP design patterns. Suppose you want to decide which of two classes should be responsible computing something, and one of the two classes involved has an attribute that is needed in the computation, but the other class does not. What GRASP design pattern would best help you decide what class should be responsible?.

   The Information Expert pattern.

2. (5 points) This is a problem about the Creator pattern. Suppose you want to decide which of two classes in a bus reservation system should be responsible for creating a `TripLeg` object:

   a. A `Controller` object, which knows the information about the leg of the trip the user wants, such as the starting time and the departure and arrival cities.

   b. A `Reservation` object, which has already been given the responsibility of containing all of the trip legs in the user's reservation.

   According to the Creator pattern, which of these would be the best choice for creating the `TripLeg` object?

   The `Reservation` object (b), because it aggregates the TripLeg objects.

Use Case for the following problems

Use Case: Send E-Mail with Attachments

Primary Actor: User
Stakeholders and Interests:
- User: wants fast, accurate sending of e-mail with attachments.
- System Administrators: want few delivery problems.

Preconditions: The User has been authenticated.
Success Guarantee (Postconditions): The e-mail message with attachments has been sent to the recipients.
Main Success Scenario (or Basic Flow):

1. The User tells the System they would like to send an e-mail message.
2. The System asks the user for the name of a recipient.
3. The User tells the system the name of a recipient.
4. The System confirms that it knows the e-mail address of this recipient by showing their address, and remembers the recipient.

   System repeats steps 2-4 until User indicates done.
5. The User tells the system the subject of the e-mail.
6. The System remembers the subject.
7. The User tells the System the body (i.e., the text) of the e-mail message.
8. The System remembers the body of the e-mail.
9. The User tells the System the name of a file they would like to send as an attachment.
10. The System confirms that the named file is readable, and attaches the contents of the named file to the e-mail message.

    User repeats steps 9-10 until indicates done.
11. The User tells the System to send the e-mail message.
12. The System sends the message to the recipients, confirming to the User that the message was sent.

Extensions (or Alternative Flows):

4a. If the recipient's e-mail address is not known:
   1. The System informs the user and asks for the recipient's e-mail address.
   2. The User tells the system the recipient's e-mail address.
   3. The System remembers the address (both permanently and for this e-mail).
   4. The use case continues from step 3 of the main success scenario.

10a.If the named file is not readable:
   1. The System tells the User that the file is not readable, and asks the User for a new file name.
   2. The User tells the System a file name to read the attachment from.
   3. The use case continues from step 10 in the main success scenario.

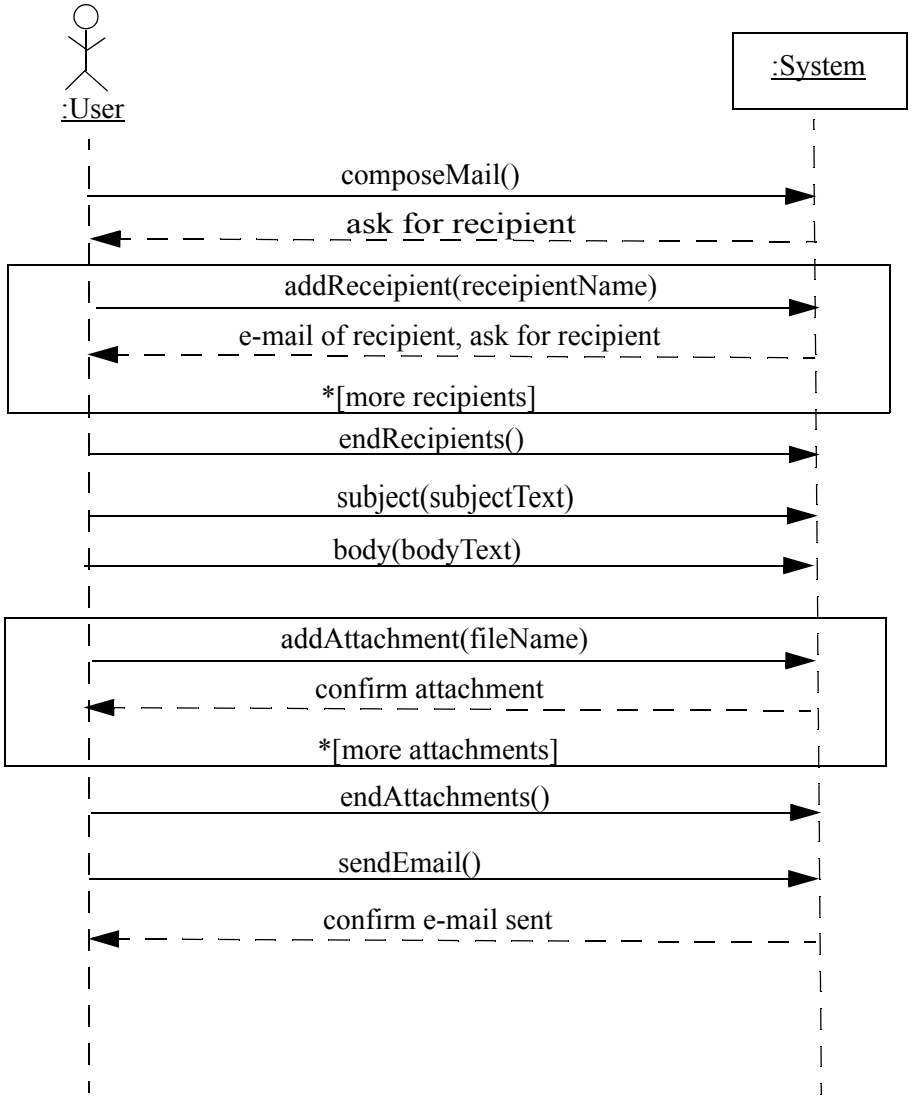
10b. If an input/output error occurs during the reading of the attachment:
   1. The System tells the User about the input/output error and asks the User if they wish to try again.
   2. The User indicates that they wish to try again.
      2a. If the User does not wish to try again:
         1. The User tells the system to cancel processing this attachment.
         2. The System asks if the User is done attaching files.
         3. The User indicates they are done attaching files.
            3a. If the User wants to attach more files:
               1. The use case continues from step 9 in the main success scenario.
         4. The use case continues from step 11 in the main success scenario.
   3. The use case continues from step 10 in the main success scenario.

12a. If the message could not be sent because of some system error:
1. The System tells the User about the error and asks if the User wants to save the e-mail.
2. The User tells the system they want to save the e-mail message.
3. The System stores the e-mail in a standard place and informs the user that the message was saved.
3a. If an error occurred while saving the e-mail message:
1. The System tells the User that the e-mail message could not be stored.
2. The use case ends.

Special Requirements
- Encoding into MIME format

Frequency of Occurrence: nearly continuous.
Open Issues: Interface with virus checkers; compression; internationalization.

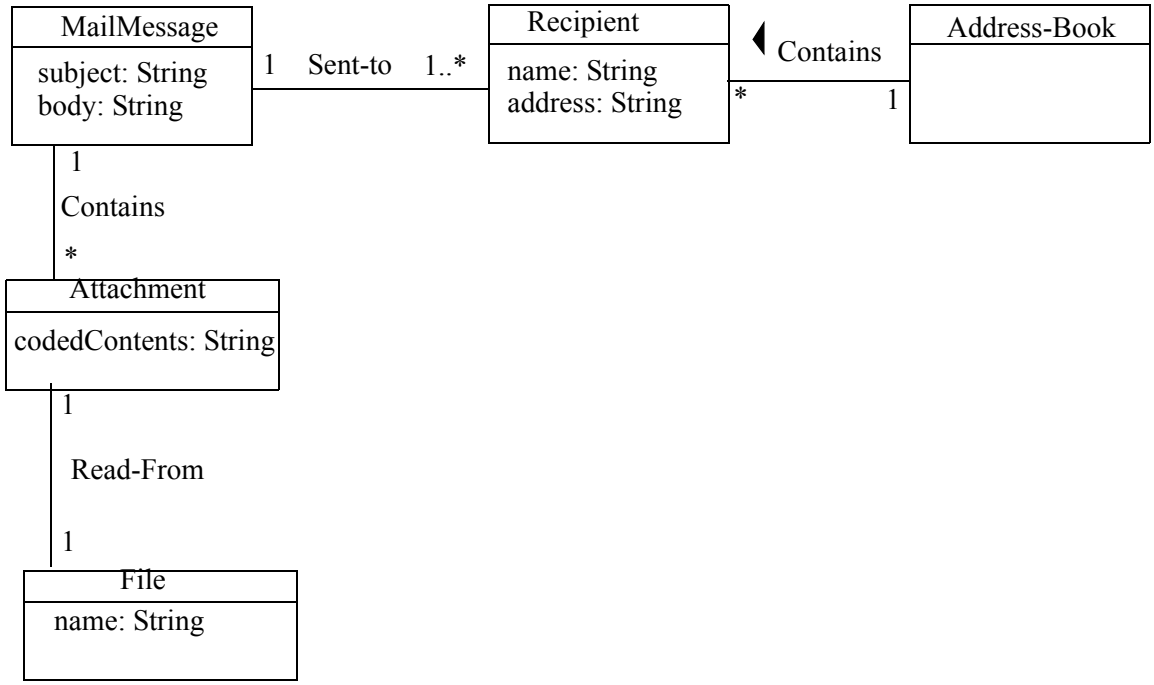
The following is a system sequence diagram (SSD) for the main success scenario of the above use case.



3. (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 basic classes that would be expected to be in the programming language, such as strings. You can include classes that are more useful versions of classes you would expect to find in the programming language.

   Answer: There may be other possibilities, of course; but please use this to solve the remaining questions.

Domain Model

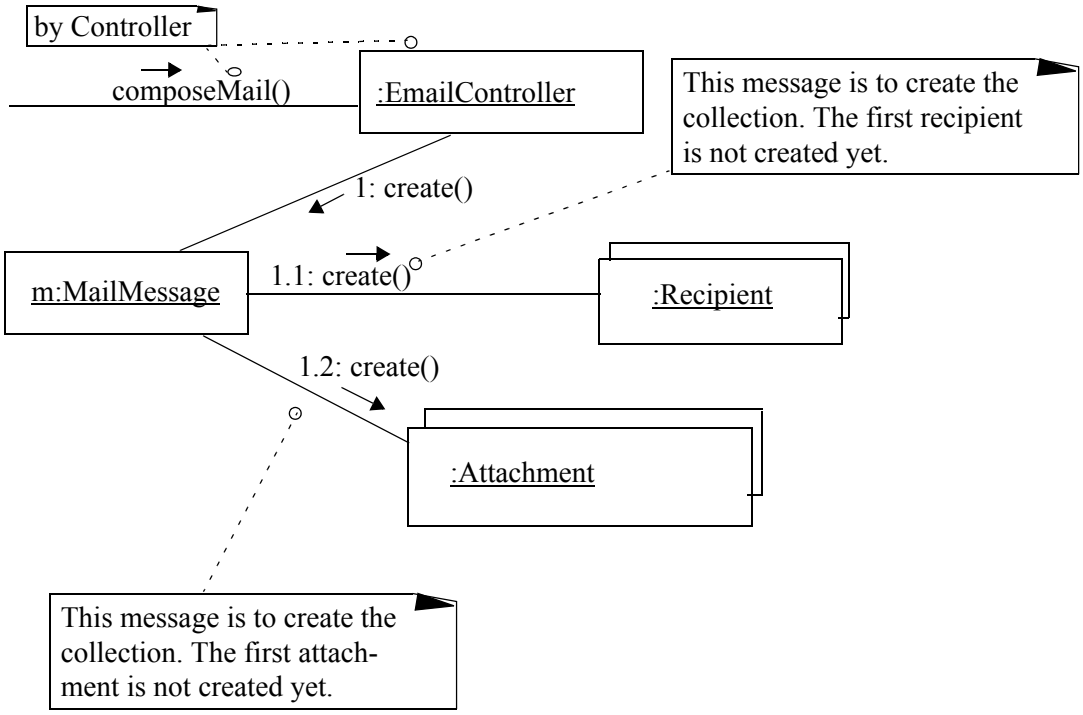


4. (30 points) In this question you will do the design for the `composeMail` system operation from the system sequence diagram found on page 4. Your design should be based on the standard solution for the previous problem.

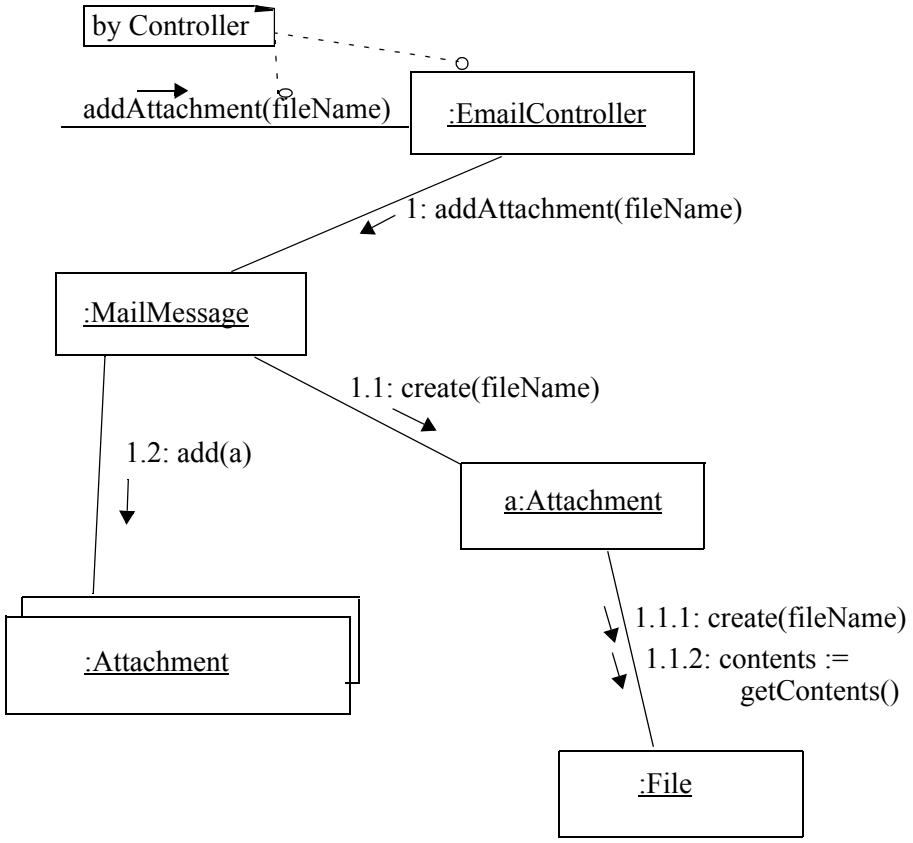
   Your design is to be recorded using an interaction diagram. You may use either the UML sequence diagram or collaboration diagram notation.

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

   Answer: The following is a collaboration diagram for this operation. (A sequence diagram would also be fine.) .



The following is a collaboration diagram for the `addAttachment` system operation, it will be used in the next problem.

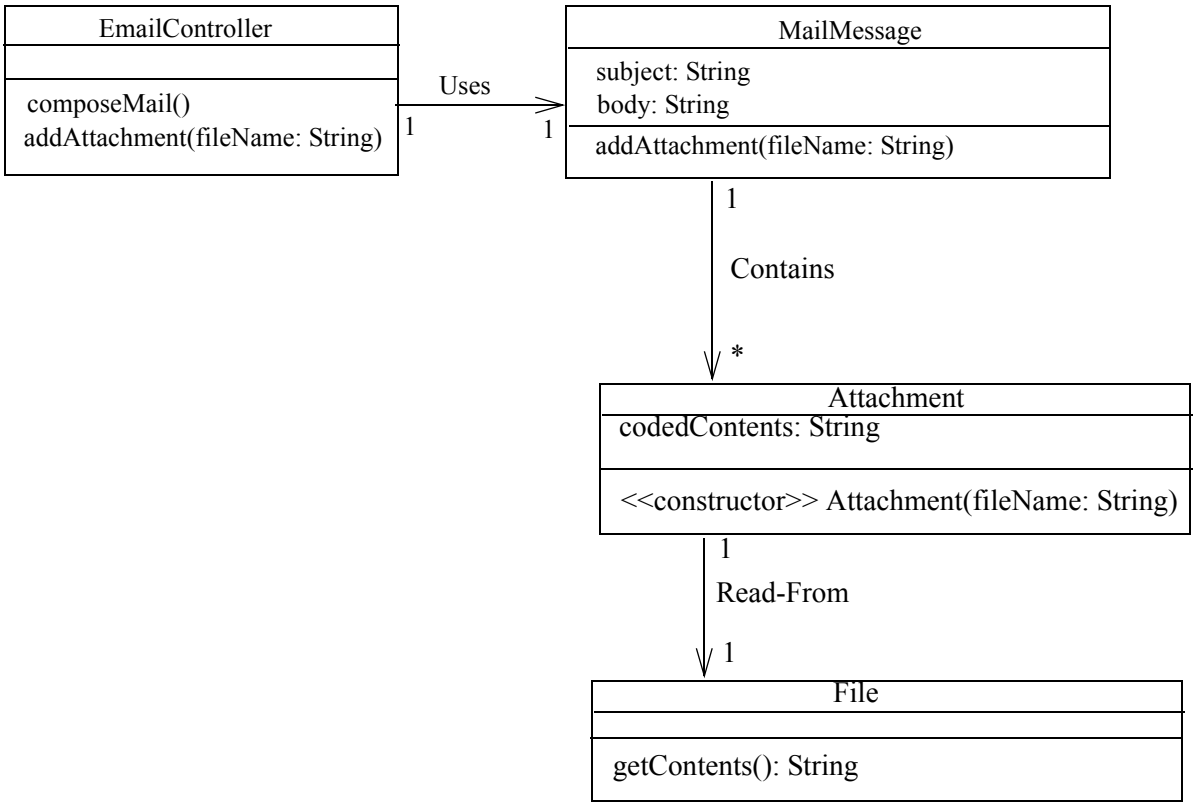


5. (30 points) Based on the standard solution for the previous problems, draw a UML design class diagram that summarizes the design of the two system operations `composeMail` and `addAttachment`. (That is, you *don't* have to include classes, attributes, or methods needed to implement other system operations. In particular, you don't have to deal with classes involved in recording the e-mail's recipients, sending the e-mail, etc.)

   On your diagram, 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.

   Answer: The following is our answer to this problem.

Design Class Diagram



Please don't show this to others still taking the test.
And please hold any arguments about the test until after everyone is done.