Chapter 11 – Object-Oriented Design: Use Case Realizations

Solutions to End-of-Chapter Problems

Review Questions

1. What is meant by the term use case realization?

The term realization means to create or bring into reality. Hence use case realization means to bring the use case to life or into reality by defining the detail steps and processes required to implement it.

2. What are the benefits of knowing and using design patterns?

Design patterns are a widely held approach to implementing standard programming constructs. It is important to know about and to understand both the idea of design patterns and to have a repertoire of understood patterns. Just as the vocabulary for any discipline is important in order to be educated in that discipline, system developers should be aware and have a working knowledge of design patterns.

3. What is the contribution to systems development by the Gang of Four?

The Gang of Four were the first developers to specifically define the idea of design patterns and to identify several fundamental common design patterns.

4. What are the five components of a standard design pattern definition?

The name, the description of the problem or need, the description of the solution, an example or diagram of the solution, and specific benefits or consequences of this solution.

5. List five elements included in a sequence diagram.

An actor, objects, lifelines, activation lifelines, messages, and loop control boxes.

6. How does a sequence diagram differ from an SSD?

A systems sequence diagram only has one object, the :system. A sequence diagram explodes the system object into all of the internal objects of the system. The idea is similar to a black box test and a white box test – one looks only at the outside while the other observes the details of what is going on inside.

7. What is the difference between designing with CRC cards and designing with sequence diagrams?

CRD cards is a more elementary process. It does not try to identify the messages at the same level of detail as is done with a sequence diagram. CRC cards gives an overview of the process, but leaves many details to the programmer. Sequence diagram goes more in depth.

8. Explain the syntax of a message on a sequence diagram.

'*' – means multiply occurring or looping
[true/false] – means test condition which is tested before the message is sent
return-value – is the return value which is returned to the originator of the message
:= -- is used to denote that a return value exists
message-name – this is the message name or service requested
(parameter-list) – this is the list of data parameters being sent with the message

9. What is the purpose of the first-cut sequence diagram? What kinds of classes are included?

The purpose of the first-cut sequence diagram is to design and describe the logic within the problem domain objects, or the set of messages and connections between the various problem domain classes, e.g. business layer logic. Only problem domain classes are included.

10. What is the purpose of the use case controller?

A use case controller class is a completely artificial class that is used to provide a link between the view layer and the domain layer. A controller helps reduce the coupling between the view and domain layers in that the view layer does not need to know about all the classes in the domain layer. It only needs to know about the controller.

11. What is meant by an activation lifeline? How is it used on a sequence diagram?

An activation lifeline indicates the active execution or 'life' of the destination object. It is used as part of the lifeline of an object, but denoting that period of time when the object is actively executing some logic.

12. Describe the three major steps in developing the set of messages for the first-cut sequence diagram.

1. Start with each input message and identify all information and services that it will need in order to complete the requested service, e.g. identify all internal messages.
2. Identify all of the classes that will be involved to process.
3. Make sure each internal message is complete with parameters, true/false condition, and looping.

13. What assumptions do developers usually make while doing the initial use case realization?

Perfect technology assumption
Perfect memory assumption
Perfect solution assumption

14. When doing multilayer design, what is the order in which layers should be designed? Why?

The business logic layer is done first to yield the first-sequence diagram.
Often the data access layer is done next.
Finally the view layer is added.

15. What is the "separation of responsibilities" principle?

Separation of responsibilities is the concept that each unique task within a use case should be done by a separate method/object. Don't try to jam all of the processing logic into one giant method in one class. Make the methods and the objects cohesive by having them complete the task that applies to their primary purpose or reason for being.

16. Explain the two methods of accessing the database to create new objects in memory.

One method is to create an empty shell of an object, say by the controller or a factory. Then in the constructor of that newly instantiated object, it will call the data access object to read the database and fill in all of the fields.

The other method is to issue a call to the data access object and have it read the database and instantiate a new object, or multiple objects, based on what is returned from the database.

17. What symbols are used in a communication diagram, and what do they mean?

Stick figure is an actor
Boxes mean classes
Lines between the boxes are links to pass messages – much like navigation visibility.
Arrows with labels are the messages

18. Explain the components of message syntax in a communication diagram. How does this syntax differ from that of a sequence diagram message?

[true/false] – is the true/false condition to determine if the message is sent
number – is the sequence number or order of the message on the diagram
return-value – is the value returned to the originating object
message-name – is the name of the requested service
(parameter-list) -- is the data fields that are sent with the message

19. Explain the method syntax on design classes.

'+/-' -- is the visibility. + is visible externally, - is hidden
name – is the name of the method
parameter – is the parameter list required by the method
type – is the type of any returned values by the method

20. What is meant by a dependency relationship? How is it indicated on a drawing?

A dependency relationship is shown by a dashed arrow. It indicates that the item that the arrow points from (tail) is dependent on the item that the arrow points to (head). If something changes in the independent item, then the dependent item probably will also have to change.

21. List the major implementation responsibilities of each layer in a three-layer design.

View layer – display forms and screens, capture external events, capture entered data, edit entered data, forward data to domain layer, start and stop the system.
Domain layer – create and manage problem domain/persistent objects, process business logic.
Data access layer – connect to database, access the database with SQL or equivalent, process data returned from data base and write to database, disconnect from database.

22. What is the purpose of the adapter pattern?

The adapter pattern is used when a new component needs to be "plugged into" an existing system, but the API is not exactly the same. The adapter adapts the API of the new component so that it can fit into the existing system.

23. What common element is found in the singleton pattern and the factory pattern? What is the basic difference between the two patterns?

Both have code to test if the object exists and if not to create the object. If it does exist just return it. The difference is that the singleton has responsibility for itself, where the factory takes responsibility for other objects.

Problems and Exercises

EXPLANATORY NOTE: Chapter 11 refers to two sets problems from Chapter 5 that review the student's skills to develop a problem domain class diagram and a use case diagram. Unfortunately those two sets of problems were not included in the final iteration of Chapter 5.

To do the problem sets in Chapter 11, you will need to either supply the students with a class diagram and a use case diagram, or alternatively, you may provide the problem descriptions to allow them to create a class diagram and use case diagram. The second option provides a good review alternative. Both the diagrams and the descriptions are provided below.

Problems 1 through 7 are based on the solutions you developed in Chapter 5 for problems 1 and 2, which involved a university library system. Alternatively, your instructor may provide you with a use case diagram and a class diagram.

University Library System

This case is a simplified version of a new system for the University Library. Of course, the library system must keep track of books. Information is maintained both about book titles and the individual book copies. Book titles maintain information about title, author, publisher, and catalog number. Individual copies maintain copy number, edition, publication year, ISBN, book status (whether it is on the shelf or loaned out), and date due back in.

The library also keeps track of its patrons. Because it is a university library, there are several types of patrons, each with different privileges. There are faculty patrons, graduate student patrons, and undergraduate student patrons. Basic information about all patrons is name, address, and telephone number. For faculty patrons, additional information is office address and telephone number. For graduate students, information such as graduate program and advisor information is maintained. For undergraduate students, program and total credit hours are maintained.

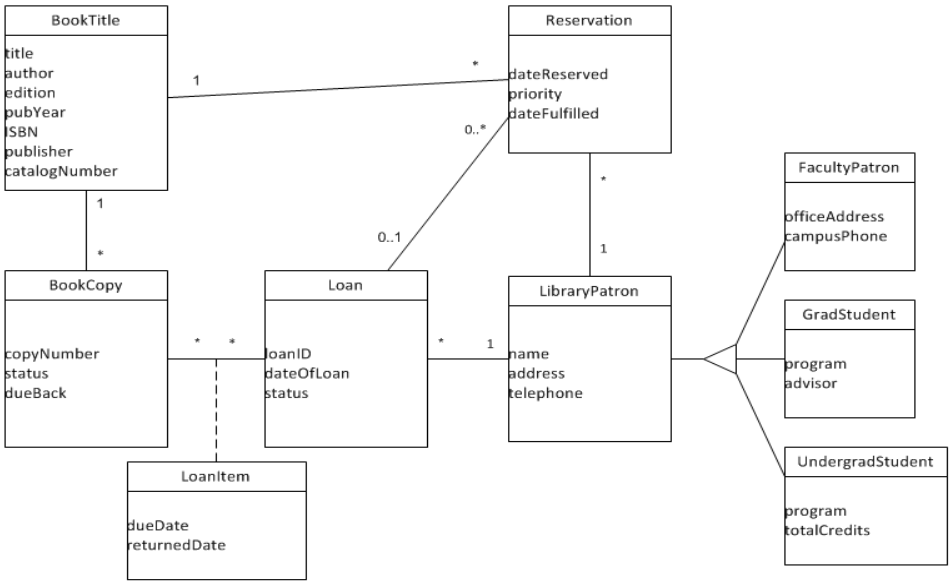
The library also keeps information about library loans. A library loan is a somewhat abstract object. A loan occurs when a patron approaches the circulation desk with a stack of books to check out. Over time a patron can have many loans. A loan can have many physical books associated with it. (And a physical book can be on many loans over a period of time. Information about past loans is kept in the database.) So, in this case, an association class should probably be created for loaned books.

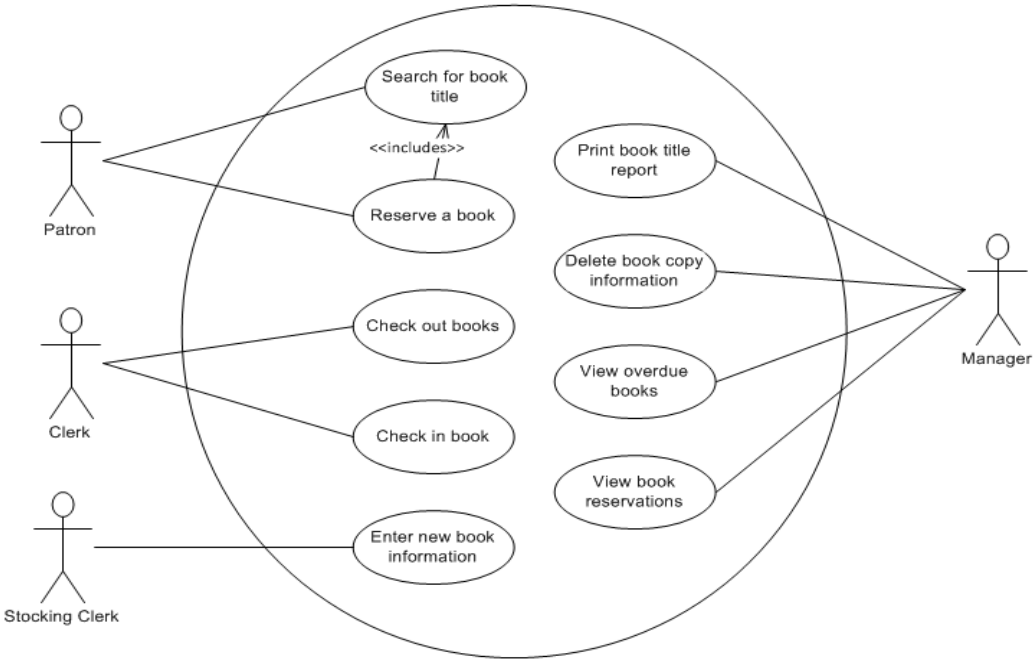
If a patron wants a book that is already checked out, the patron can put that title on reserve. This is another class that does not represent a concrete object. Each reservation is for only one title and one patron. Information such as date reserved, priority, and date fulfilled is maintained. When a book is fulfilled, the system associates it with the loan on which it was checked out.

Patrons have access to the library information to search for book titles and to see whether a book is available. A patron can also reserve a title if all copies are checked out. When patrons bring books to the circulation desk, a clerk checks out the books on a loan. Clerks also check books in. When books are dropped in the return slot, clerks check in the books. Stocking clerks keep track of the arrival of new books.

The managers in the library have their own activities. They will print reports of book titles by category. They also like to see (online) all overdue books. When books get damaged or destroyed, managers delete information about book copies. Managers also like to see what books are on reserve.

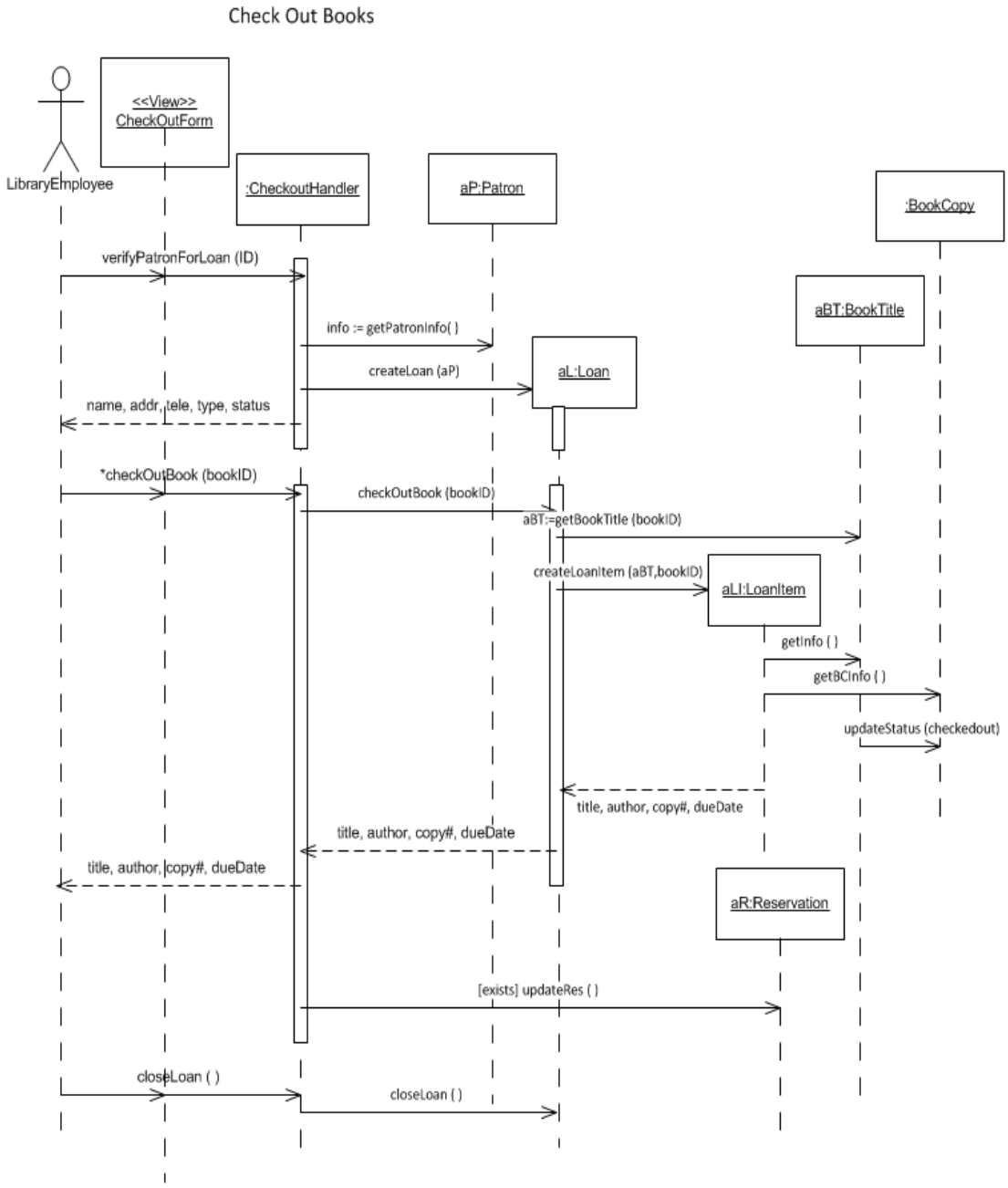
Class Diagram and Use Case Diagram for the University Library System



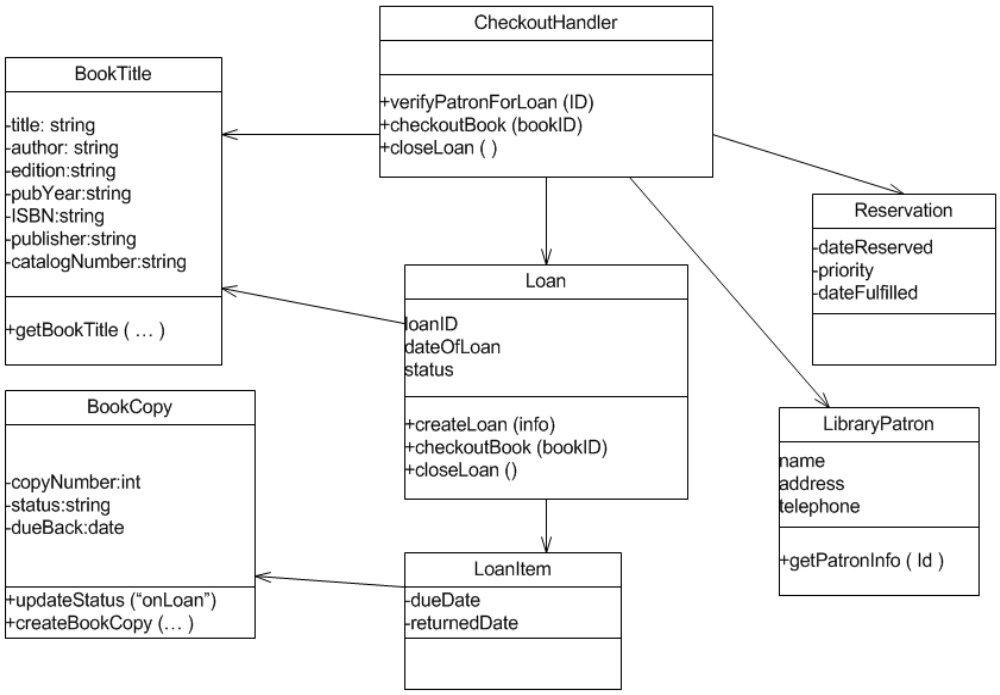


1. Figure 11-25 is an SSD for the use case *Check out books* in the university library system. Do the following:

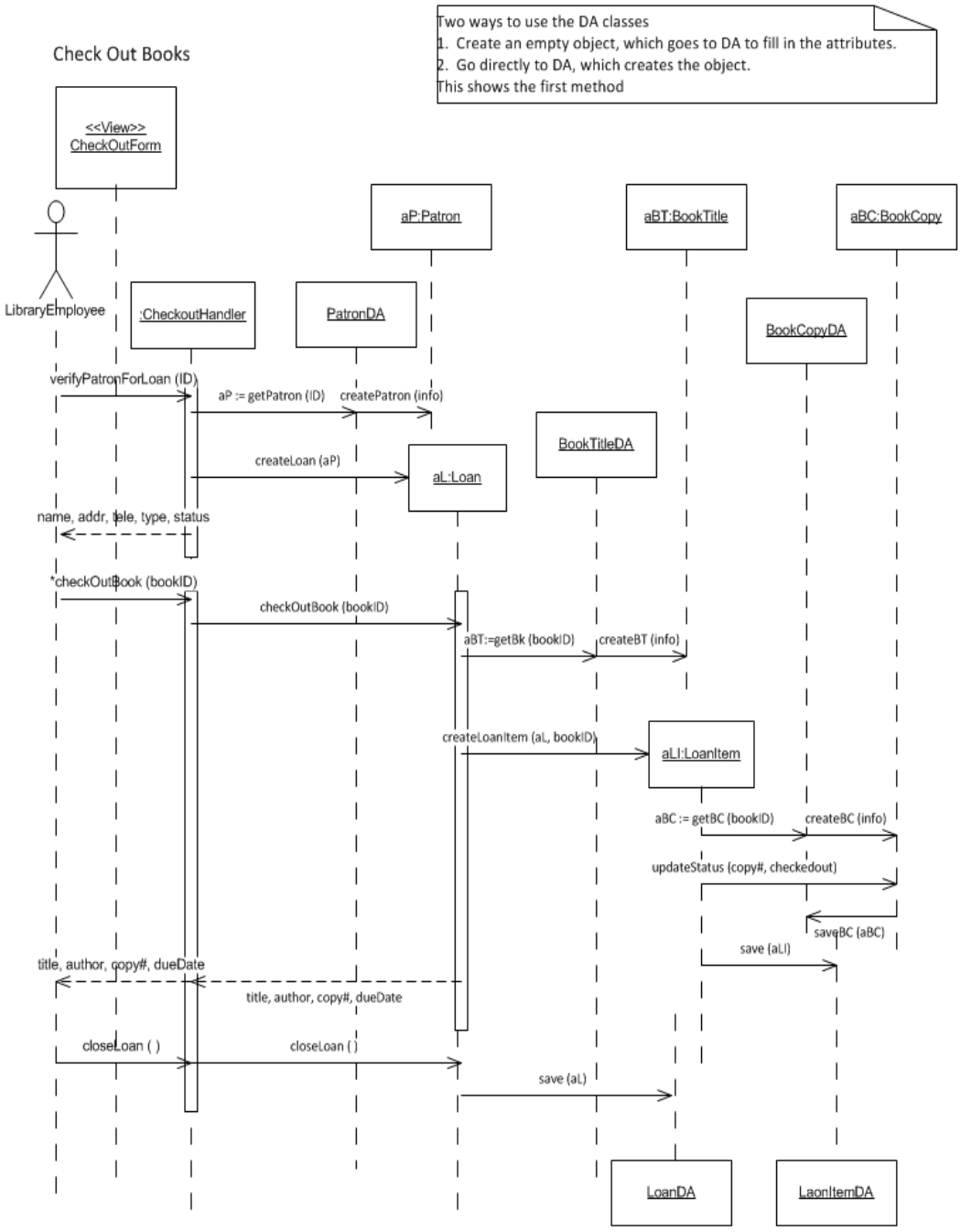
a. Develop a first-cut sequence diagram that only includes the actor and problem domain classes.



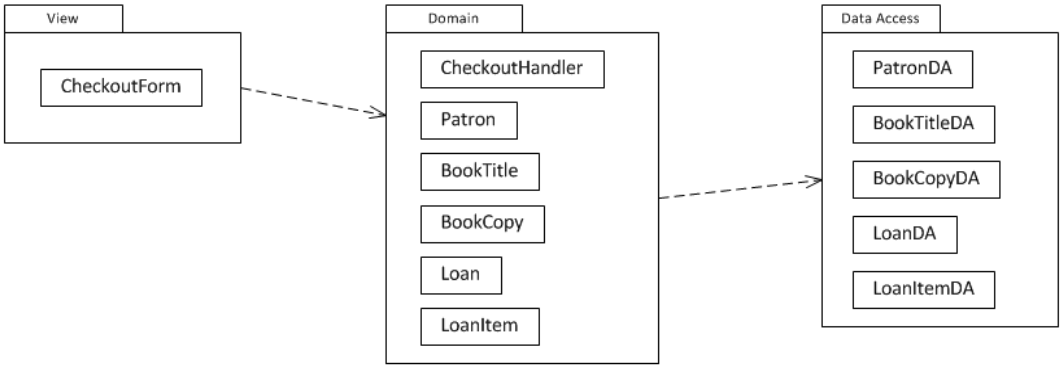
b. Develop a design class diagram based on your solution. Be sure to include your controller class.



2. Using your solution to problem 1, do the following:
a. Add the view layer classes and the data access classes to your diagram. You may do this with two separate diagrams to make them easier to work with and read.

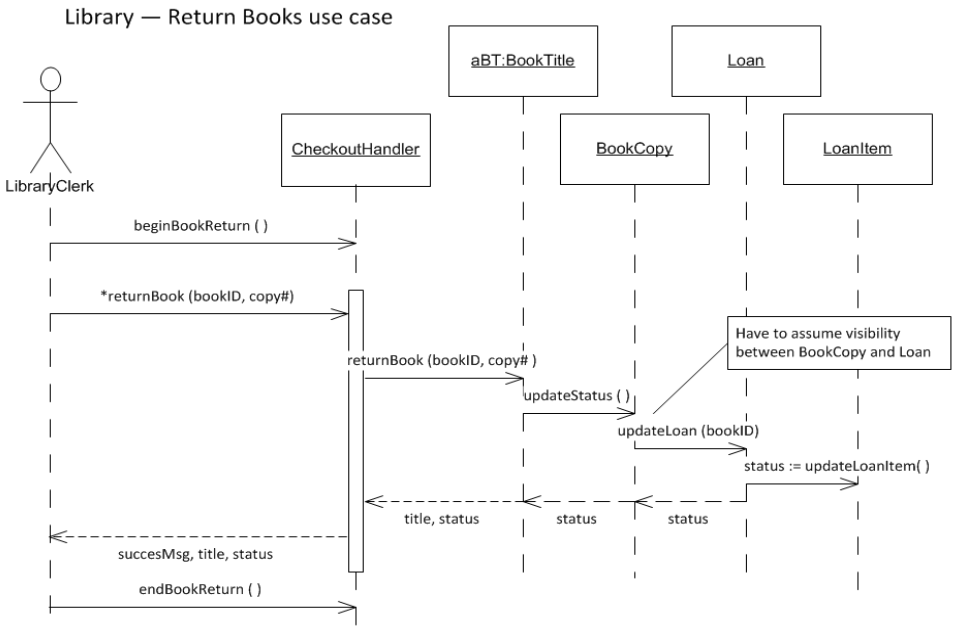


b. Develop a package diagram showing a three layer solution with view layer, domain layer, and data access layer packages.

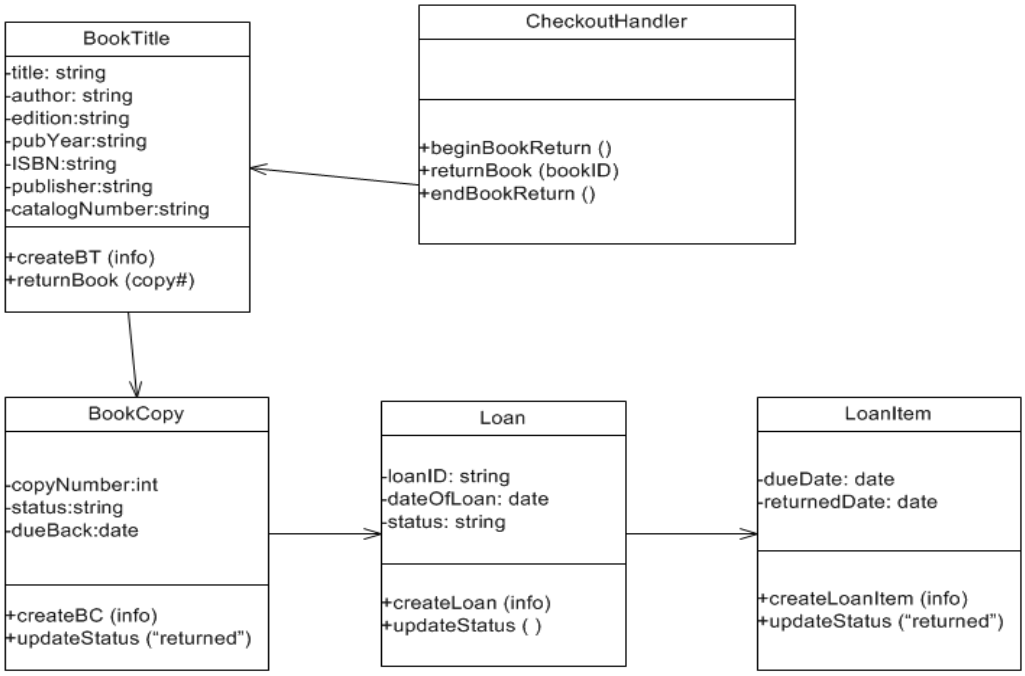


3. Figure 11-26 is an activity diagram for the use case *Return books* in the university library system. Do the following:

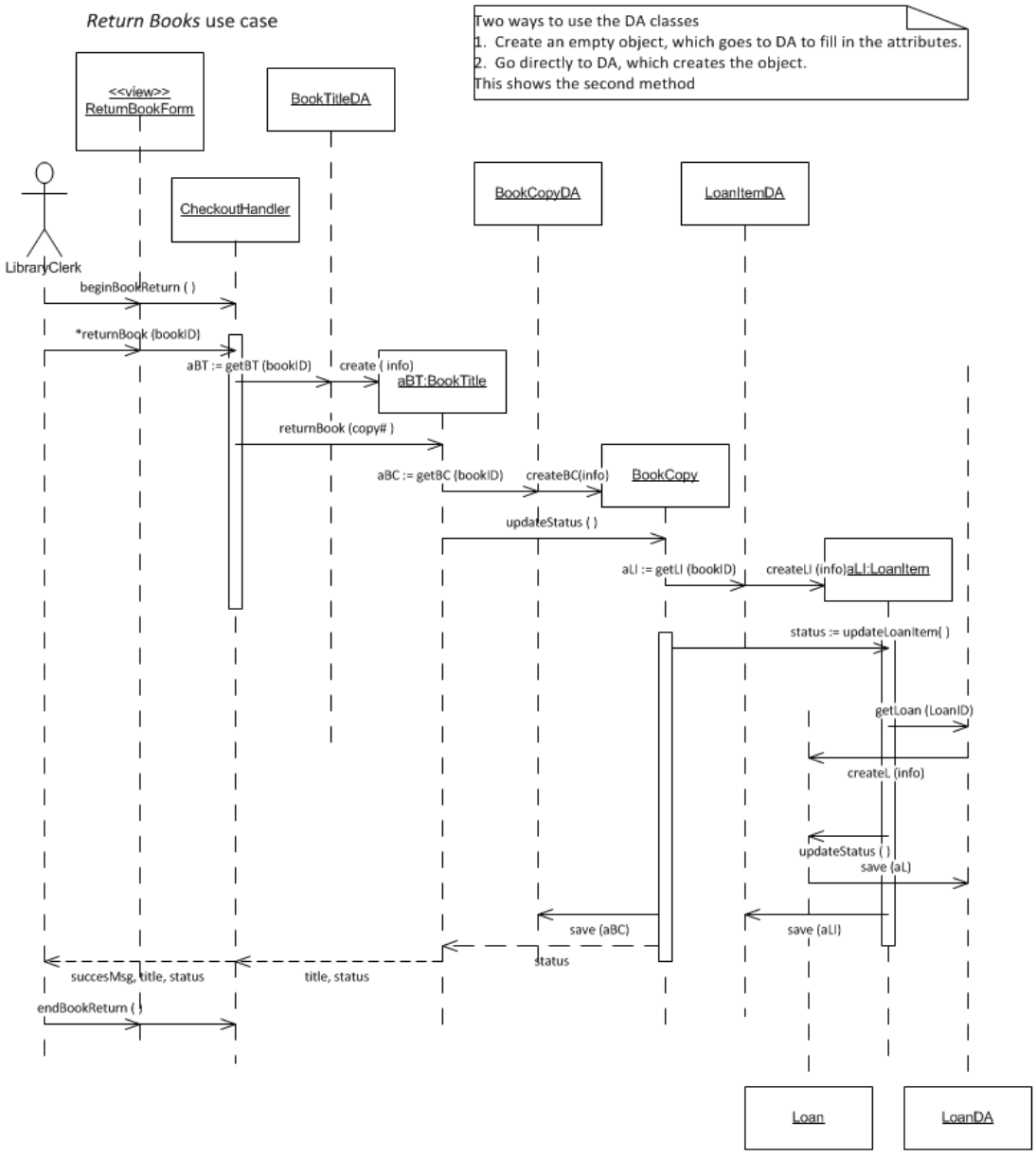
a. Develop a first-cut sequence diagram that only includes the actor and problem domain classes. Note: As with all design solutions, there are several ways to implement the solution. This one assumes that information that can be scanned from the book includes a bookID and a copy#.



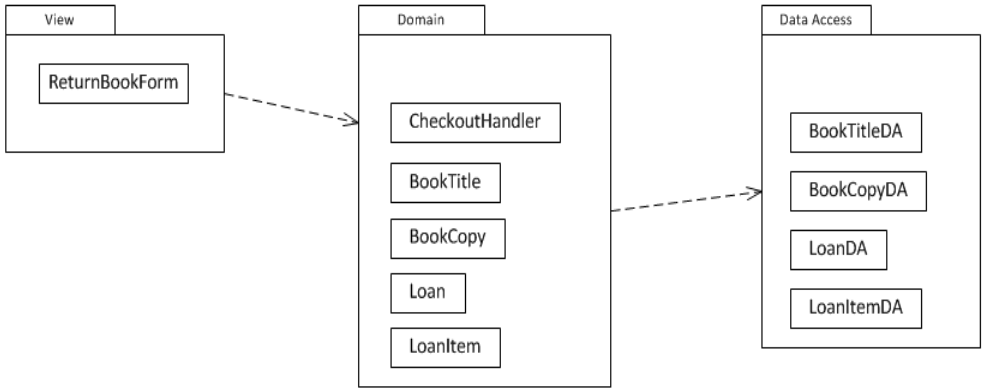
b. Develop a design class diagram based on the domain class diagram.



4. Using your solution to problem 3, do the following:
a. Add the view layer classes and the data access classes to your diagram.

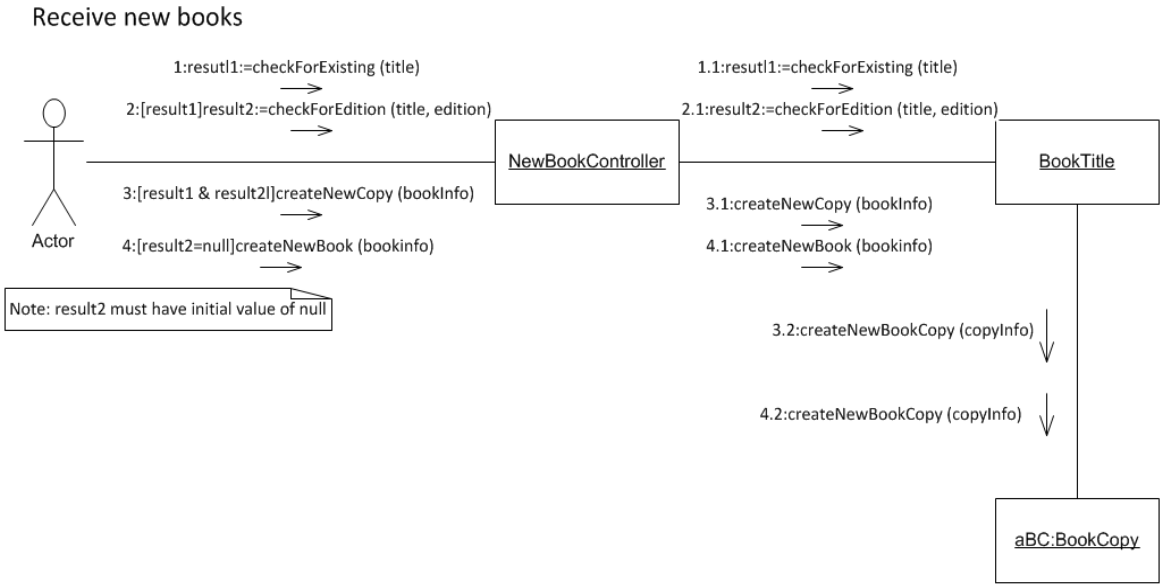


b. Develop a package diagram showing a three layer solution with view layer, domain layer, and data access layer packages.

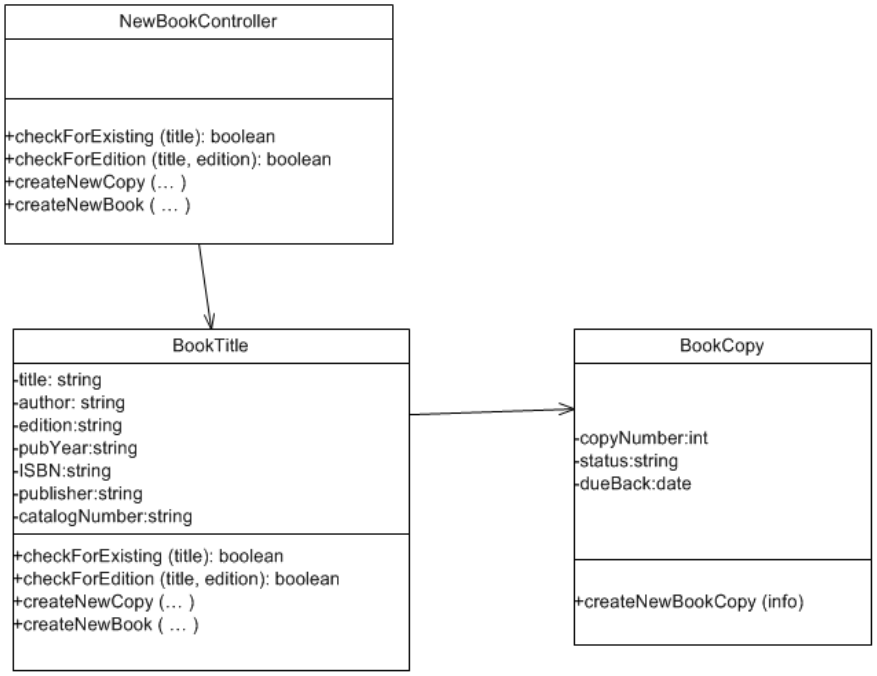


5. Figure 11-27 is a fully developed use case description for the use case Receive new book in the university library system. Do the following:

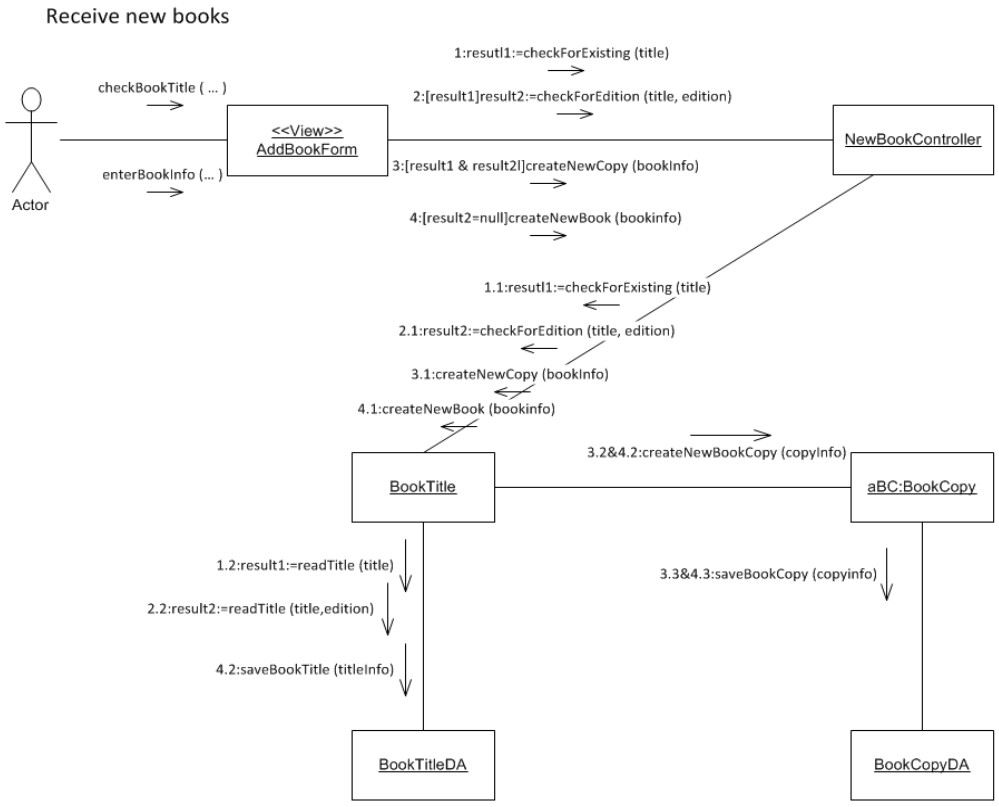
a. Develop a first-cut communication diagram that only includes the actor and problem domain classes.



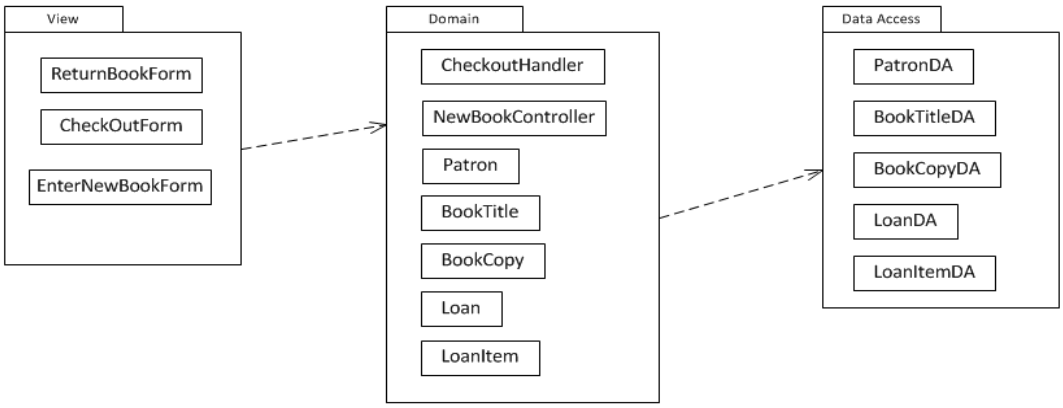
b. Develop a design class diagram based on the domain class diagram.



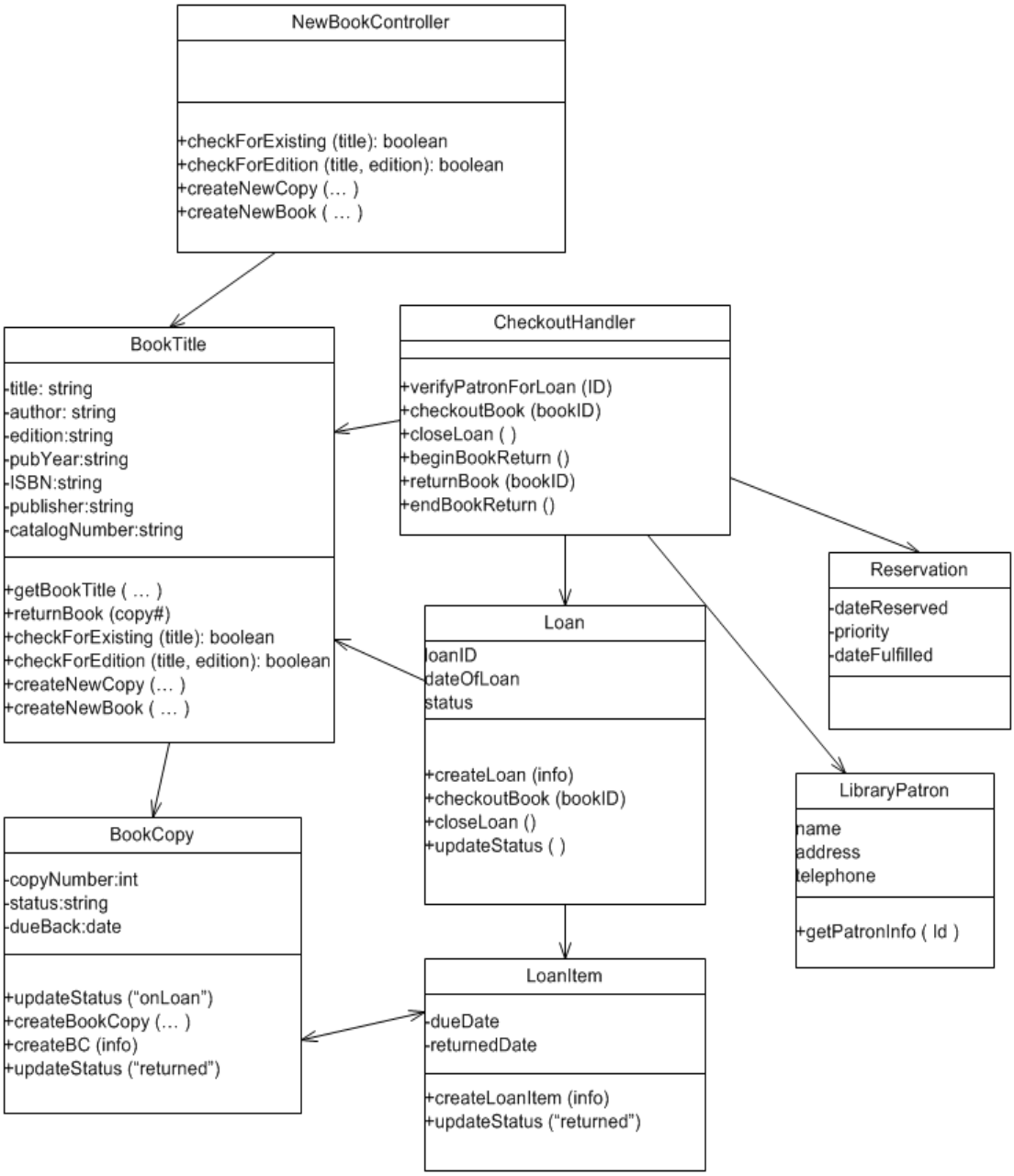
6. Using your solution to problem 5, do the following:
a. Add the view layer classes and the data access classes to your diagram.



b. Develop a package diagram showing a three layer solution with view layer, domain layer, and data access layer packages.



7. Integrate the design class diagram solutions you developed for problems 1, 3, and 5 into a single design class diagram.



Problems 8 through 14 are based on the solutions you developed for problems 3 and 4 in Chapter 5, which involved a dental clinic system. Alternatively, your instructor may provide you with a use case diagram and a class diagram.

Dental Clinic System

A clinic with three dentists and several dental hygienists needs a system to help administer patient records. This system does not keep any medical records. It only processes patient administration. Each patient has a record with his or her name, date of birth, gender, date of first visit, and date of last visit. Patient records are grouped together under a household. A household has attributes such as name of head of household, address, and telephone number. Each household is also associated with an insurance carrier record. The insurance carrier record contains name of insurance company, address, billing contact person, and telephone number.

In the clinic, each dental staff person also has a record that tracks who works with a patient (dentist, dental hygienist, x-ray technician). Because the system focuses on patient administration records, only minimal information is kept about each dental staff person, such as name, address, and telephone number. Information is maintained about each office visit, such as date, insurance copay amount (amount paid by the patient), paid code, and amount actually paid. Each visit is for a single patient, but, of course, a patient will have many office visits in the system. During each visit, more than one dental staff person may be involved by doing a procedure. For example, the x-ray technician, dentist, and dental hygienist may all be involved on a single visit. In fact, some dentists are specialists in such things as crown work, and even multiple dentists may be involved with a patient. For each staff person does procedure in a visit combination (many-to-many), detailed information is kept about the procedure. This information includes the type of procedure, a description, the tooth involved, the copay amount, the total charge, the amount paid, and the amount the insurance company denied.

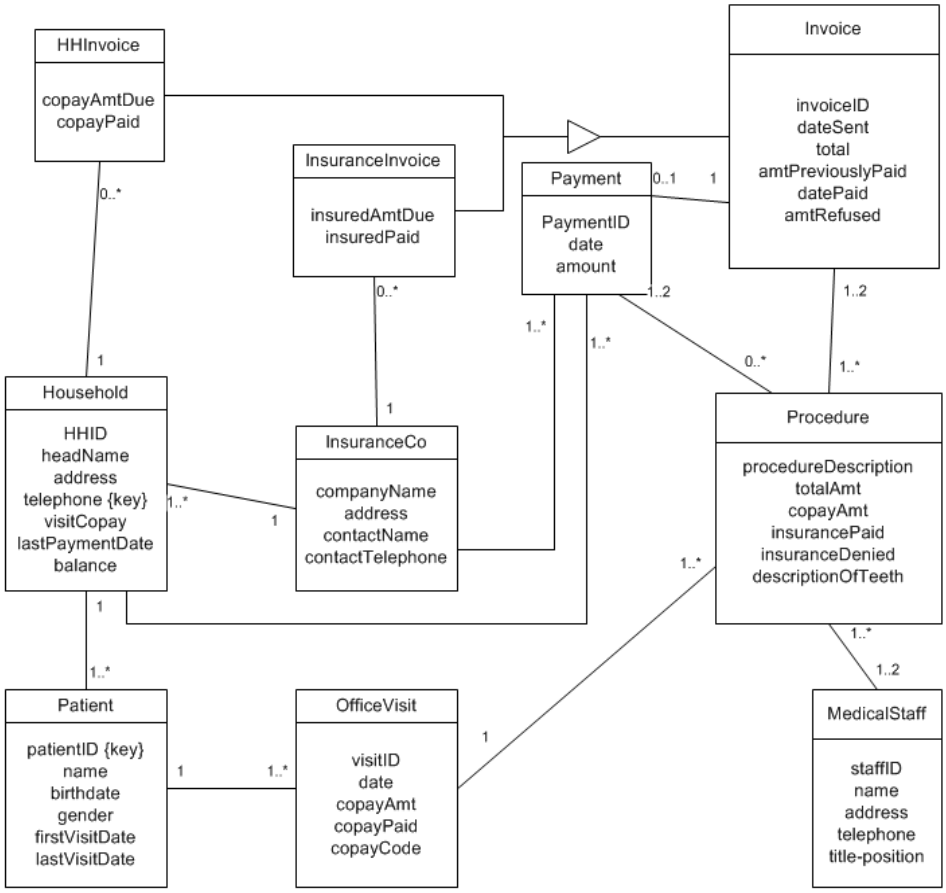
Finally, the system also keeps track of invoices. There are two types of invoices: invoices to insurance companies and invoices to heads of household. Both types of invoices are fairly similar, listing each visit, the procedures involved, the patient copay amount, and the total due. Obviously, the totals for the insurance company are different from the patient amounts owed. Even though an invoice is a report (when printed), it also maintains some information such as date sent, total amount, amount already paid, amount due and the total received, date received, and total denied. (Insurance companies do not always pay all they are billed.)

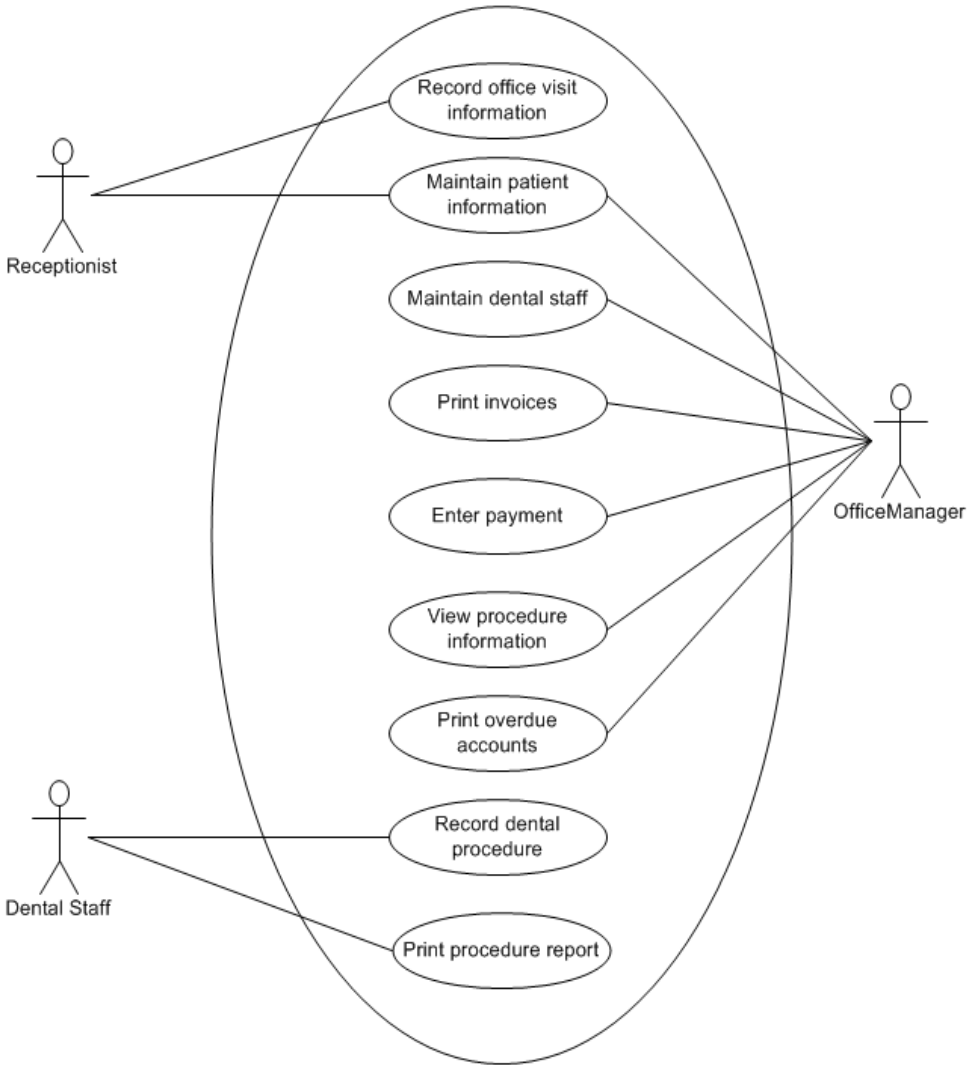
The receptionist keeps track of patient and head-of-household information, and will enter this information in the system. The receptionist will also keep track of office visits by the patients. Patient information is also entered and maintained by the office business manager. In addition, the business manager maintains the information about the dental staff.

The business manager also prints the invoices. Patient invoices are printed monthly and sent to the head of household. Insurance invoices are printed weekly. When the invoices are printed, the business manager double-checks a few invoices against information in the system to make sure it is being aggregated correctly. She also enters the payment information when it is received.

Dental staff are responsible for entering information about the dental procedures they perform. The business manager also prints an overdue invoice report that shows heads of household who are behind on their payments. Sometimes dentists like to see a list of the procedures they performed during a week or month, and they can request that report.

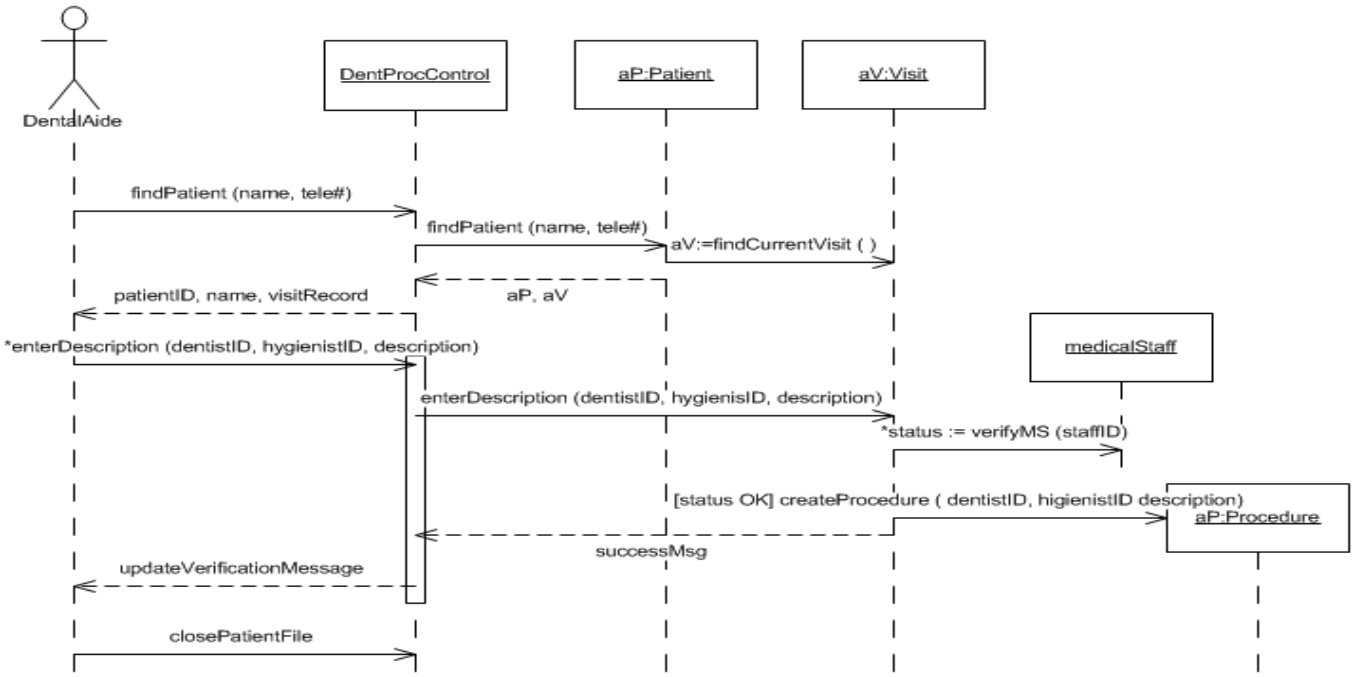
Class Diagram and Use Case Diagram for Dental Clinic System



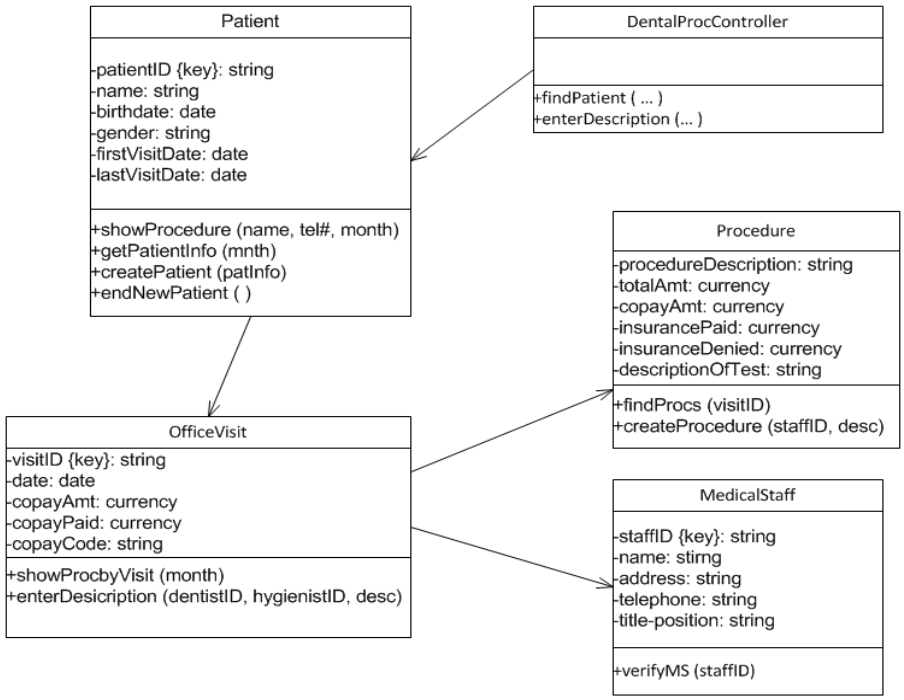

Record office visit information
Maintain patient information
Maintain dental staff
Print invoices
Enter payment
View procedure information
Print overdue accounts
Record dental procedure
Print procedure report
Receptionist
Dental Staff
OfficeManager

8. Figure 11-28 is an SSD for the use case *Record dental procedure* in the dental clinic system. Do the following:

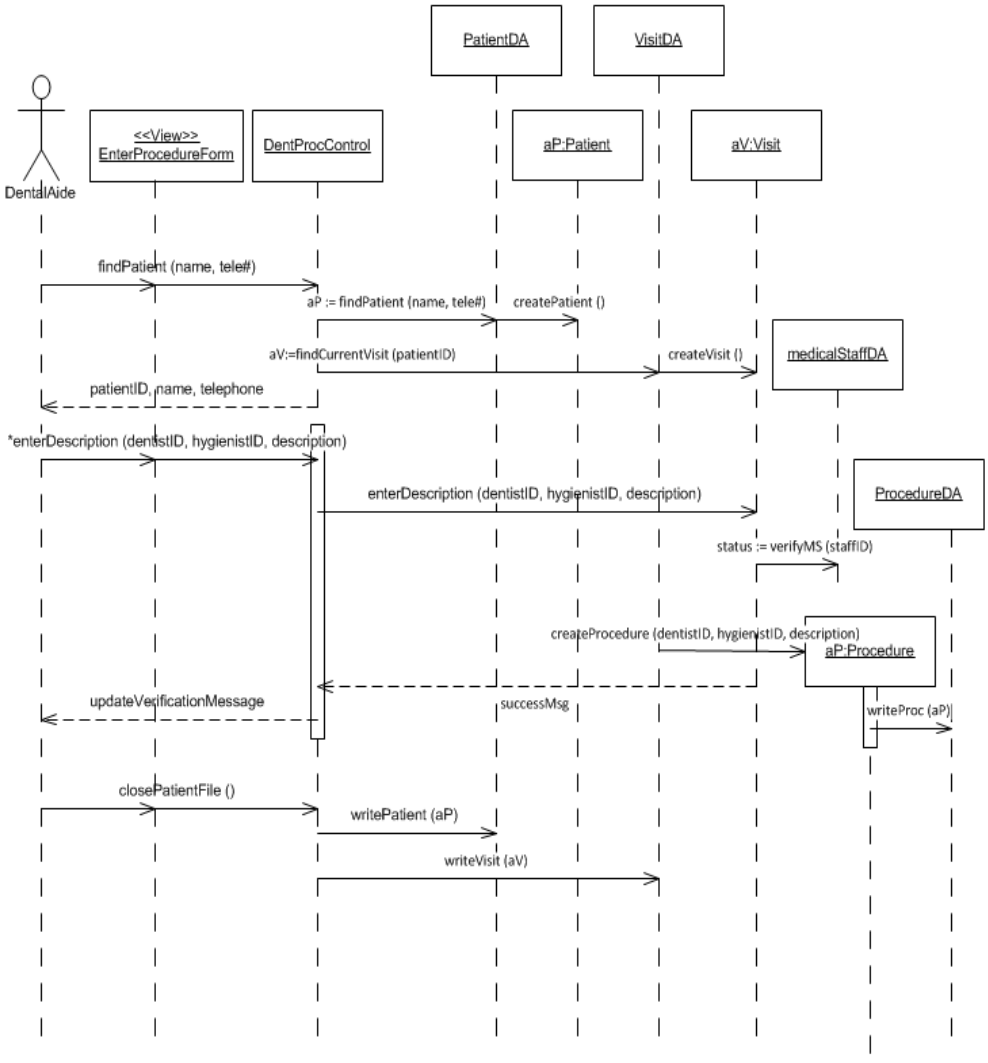
a. Develop a first-cut sequence diagram that only includes the actor and problem domain classes.



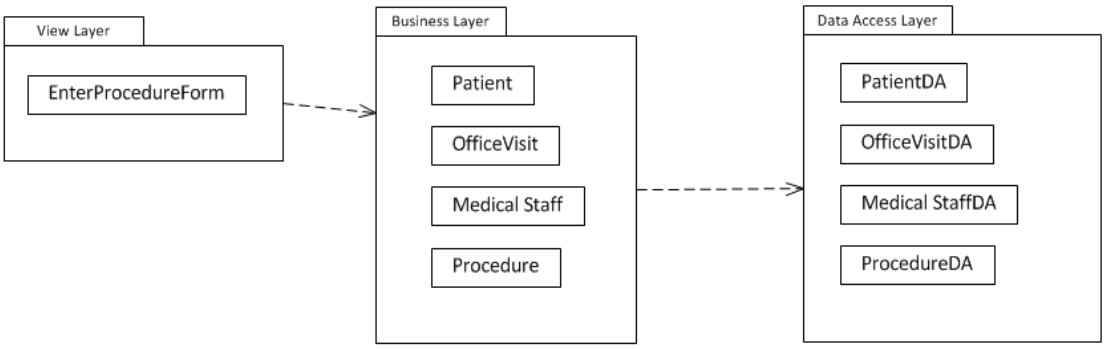
b. Develop a design class diagram based on the domain class diagram.



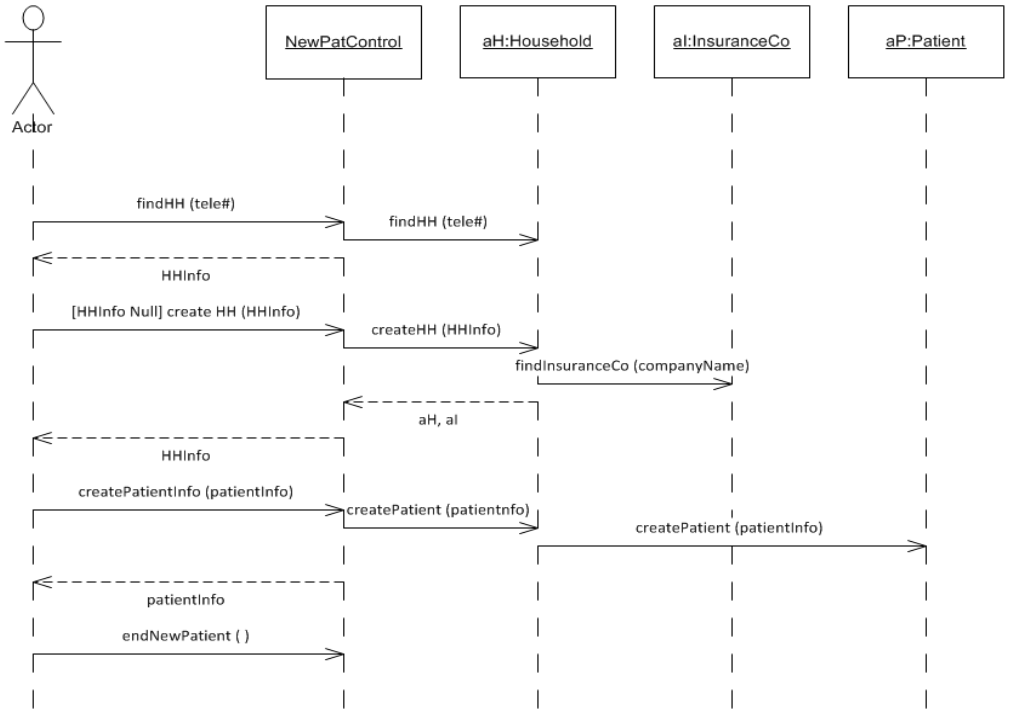
9. Using your solution to problem 8, do the following:
a. Add the view layer classes and the data access classes to your diagram.



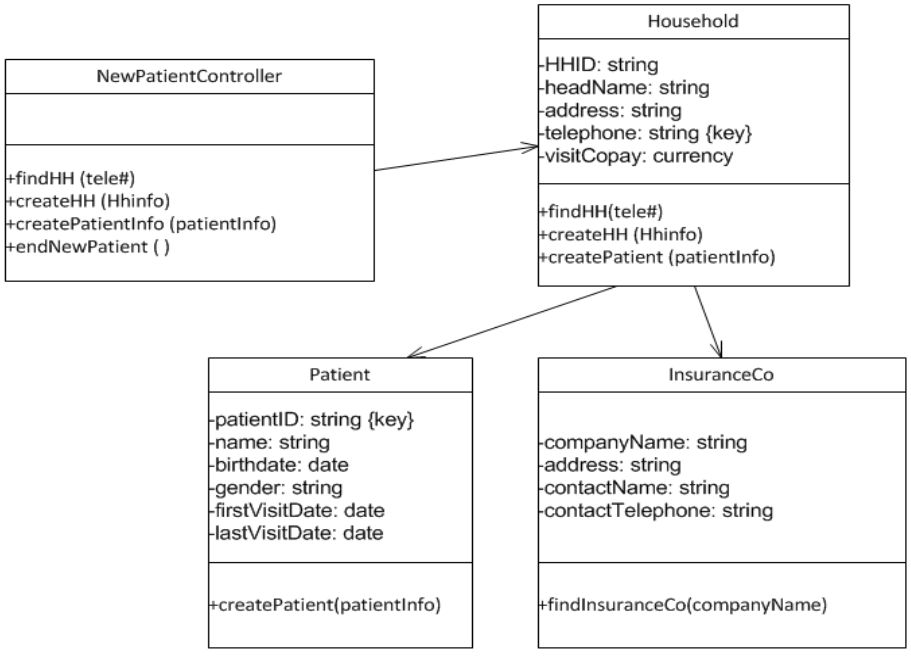
b. Develop a package diagram showing a three layer solution with view layer, domain layer, and data access layer packages.



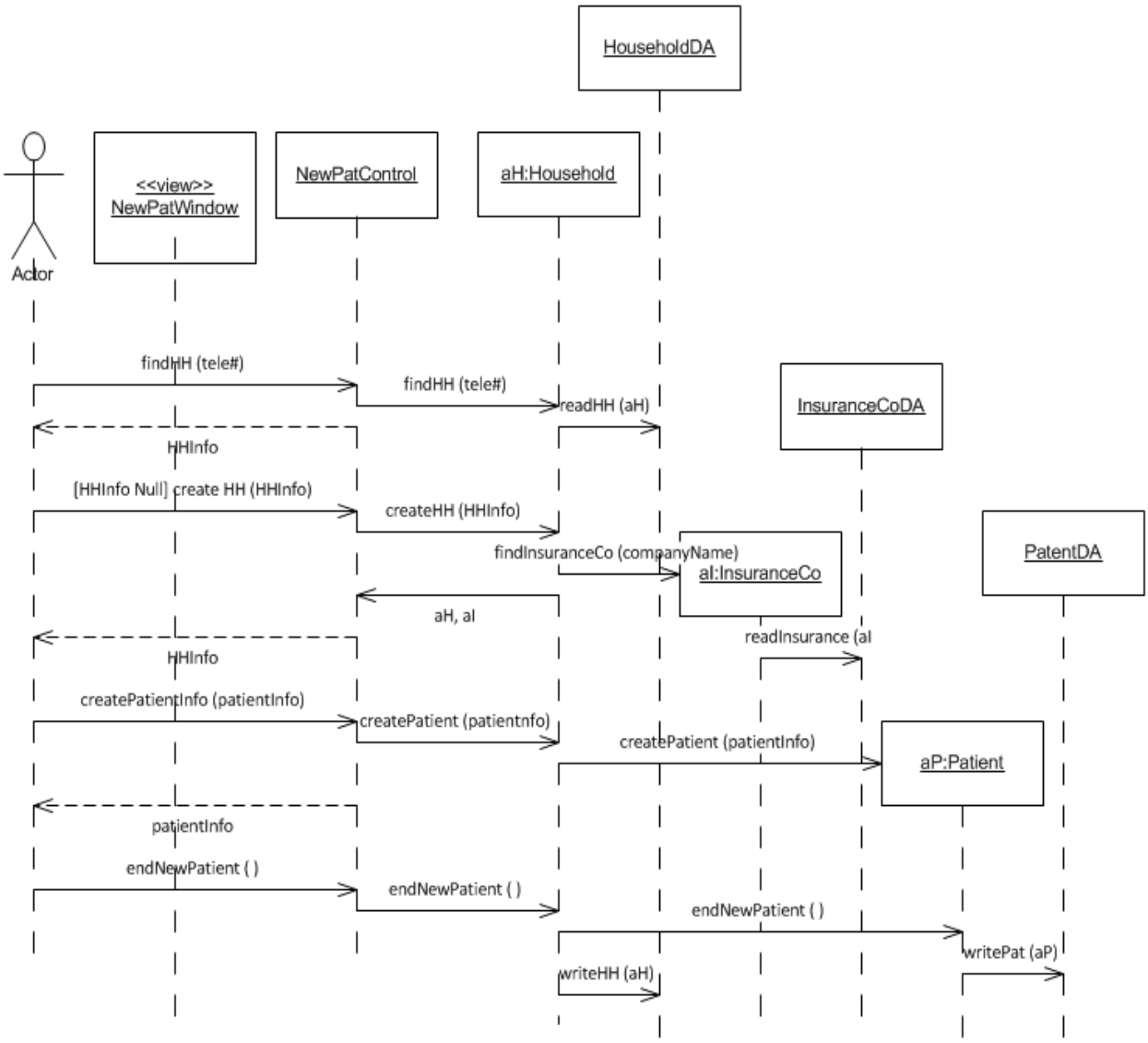
10. Figure 11-29 is an activity diagram for the use case *Enter new patient* information in the dental clinic system. Do the following:
a. Develop a first-cut sequence diagram that only includes the actor and problem domain classes.



b. Develop a design class diagram based on the domain class diagram.

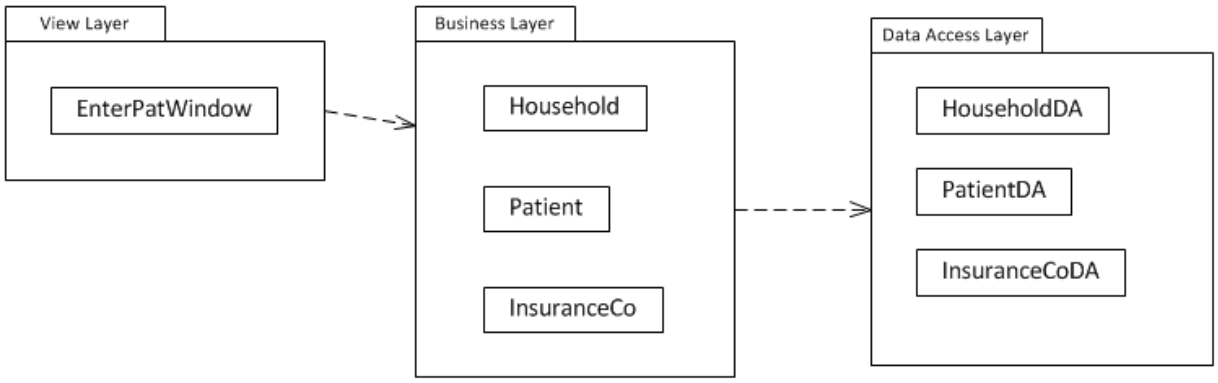


11. Using your solution to problem 10, do the following:
a. Add the view layer classes and the data access classes to your diagram.

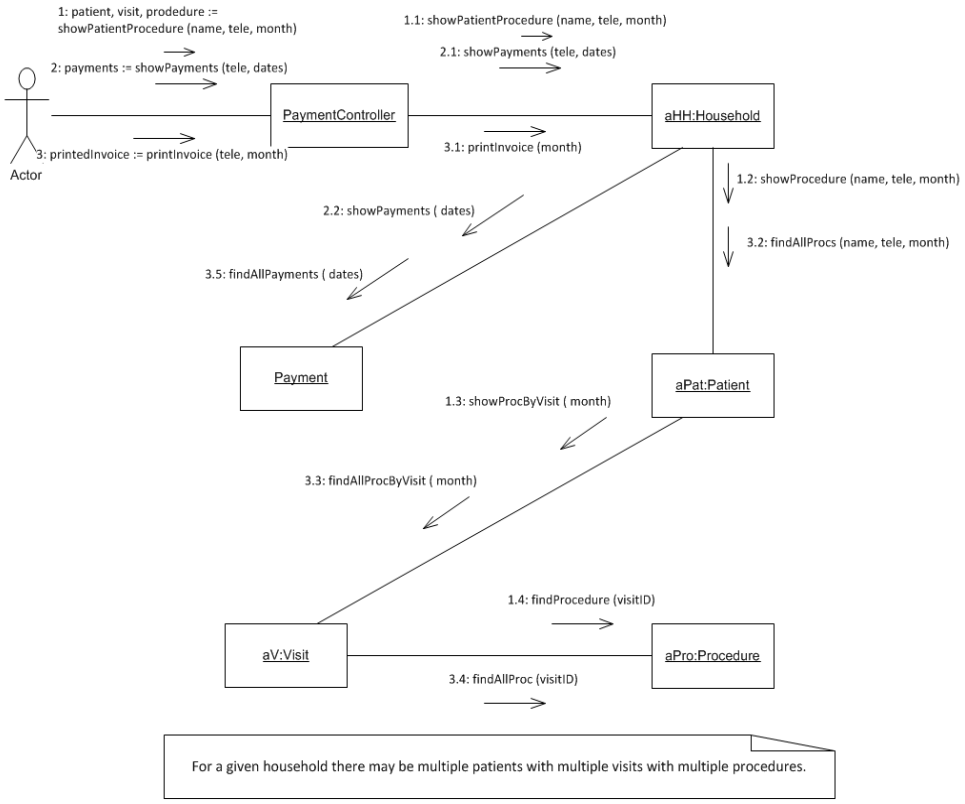


There are two ways to use the DA objects and business objects together.
1. As shown in here, the first message goes to the object, which creates an empty object. The object then sends a message to the DA object to fill in all lthe attributes.
2. As shown in the previous use case (Record dental procedure), the first message goes to the DA object, which reads the db and creates a new object.
Either solution works and is correct.

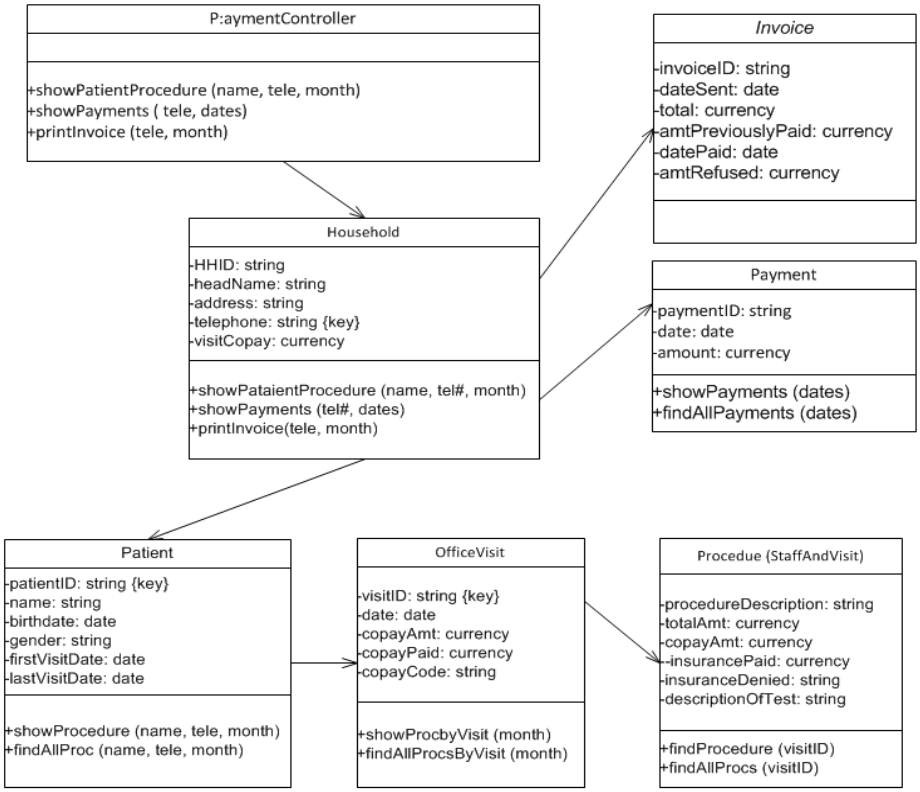
b. Develop a package diagram showing a three layer solution with view layer, domain layer, and data access layer packages.



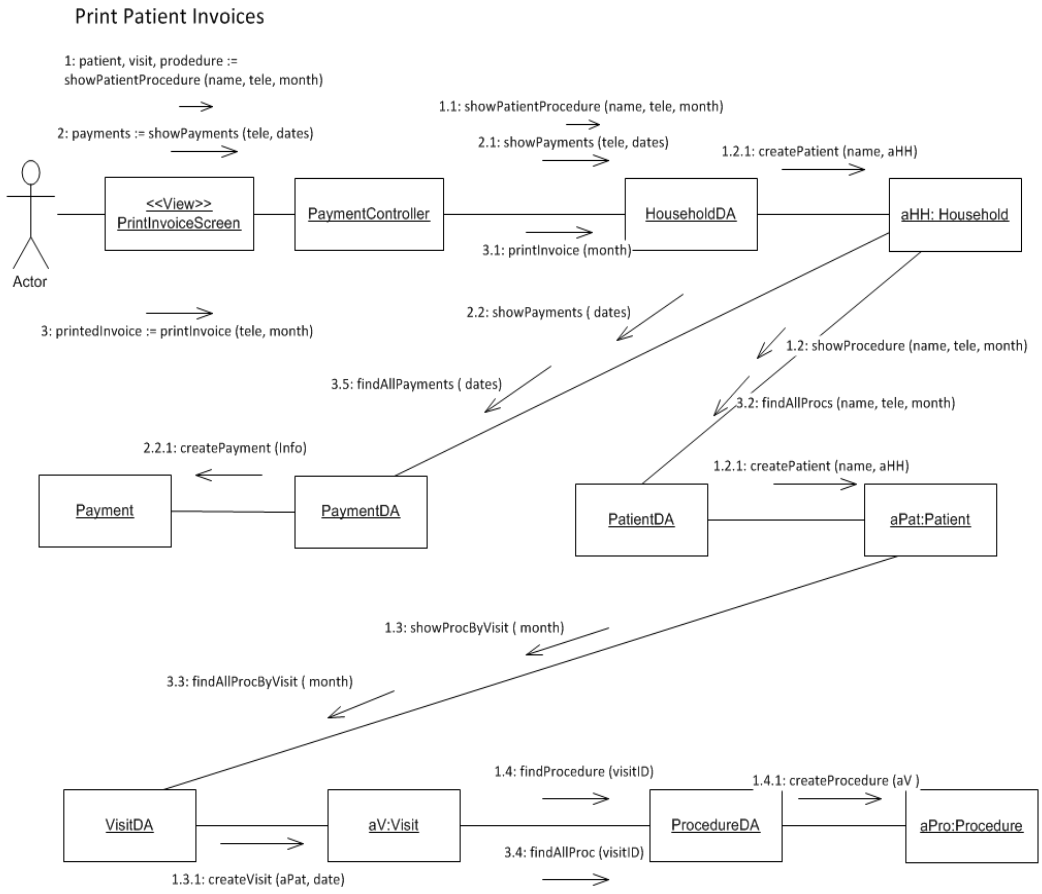
12. Figure 11-30 is a fully developed use case description for the use case *Print patient invoices* in the dental clinic system. Do the following:
a. Develop a first-cut communication diagram that only includes the actor and problem domain classes.



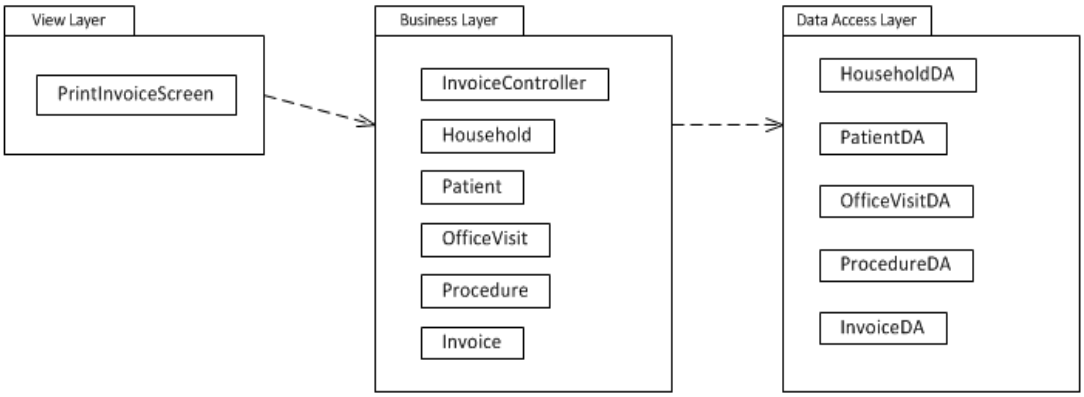
b. Develop a design class diagram based on the domain class diagram.



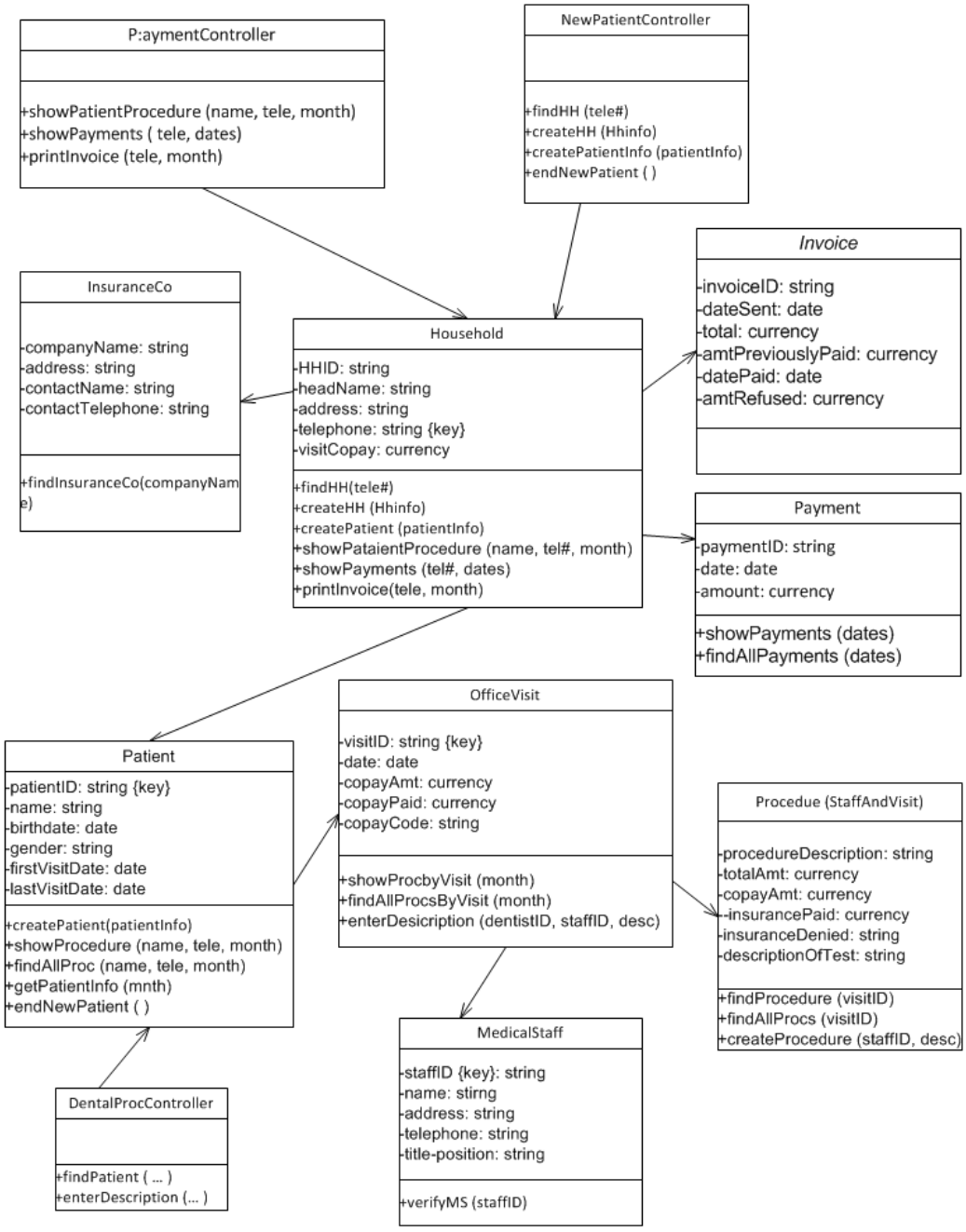
13. Using your solution to problem 12, do the following:
a. Add the view layer classes and the data access classes to your diagram.



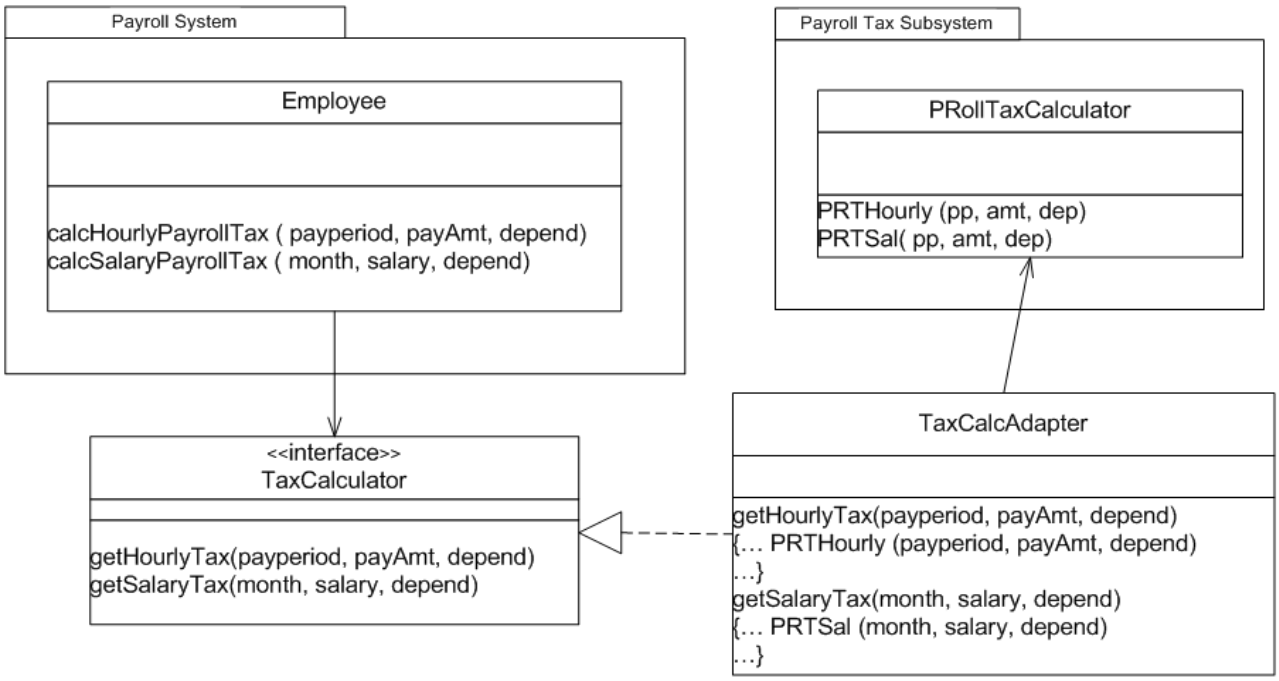
b. Develop a package diagram showing a three layer solution with view layer, domain layer, and data access layer packages.



14. Integrate the design class diagram solutions that you developed for problems 8, 10, and 12 into a single design class diagram.



15. In Figure 11-31, the package on the left contains the classes in a payroll system. The package on the right is a payroll tax subsystem. What technique would you use to integrate the payroll tax subsystem into the payroll system? Show how you would solve the problem by modifying the existing classes (in either figure). What new classes would you add? Use UML notation.

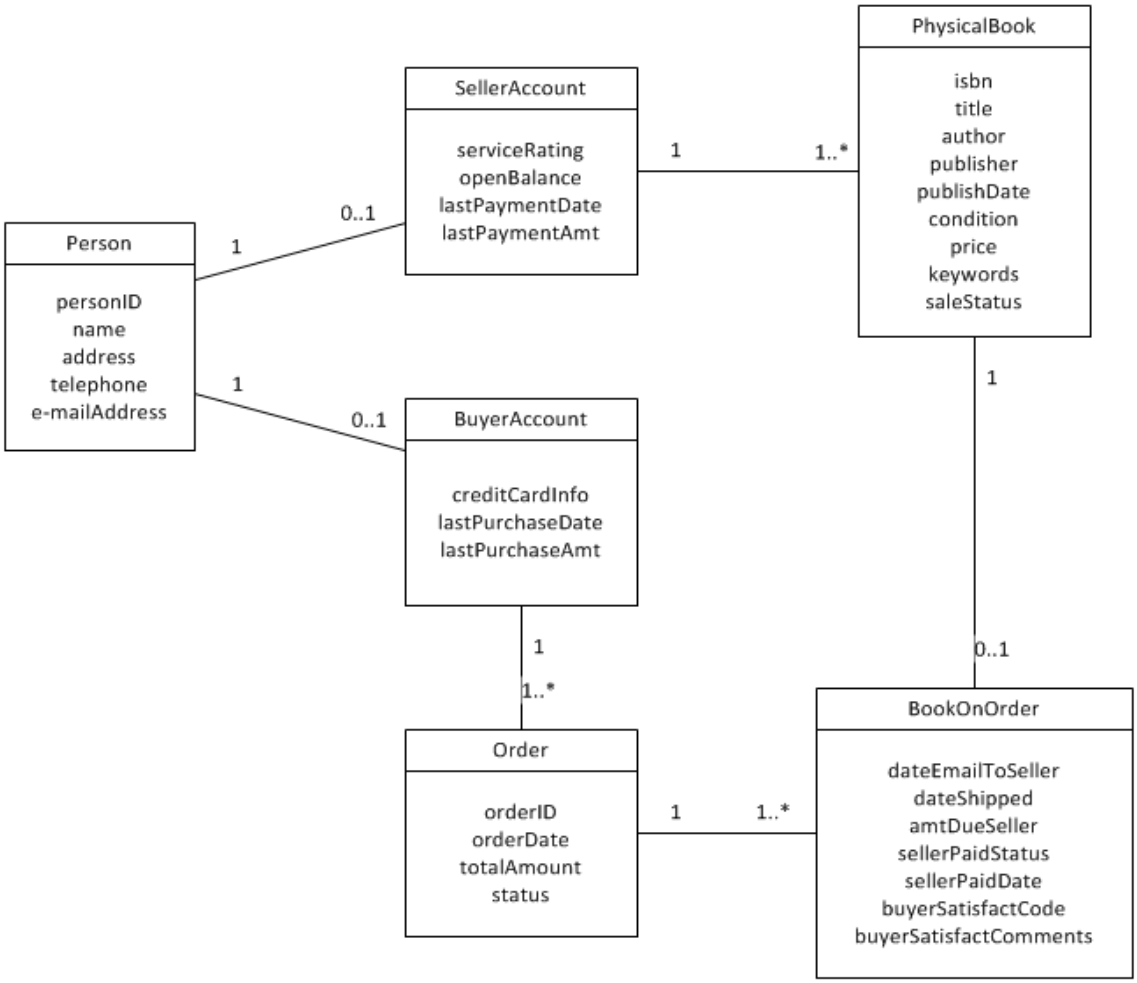


Solutions to End-of-Chapter Cases

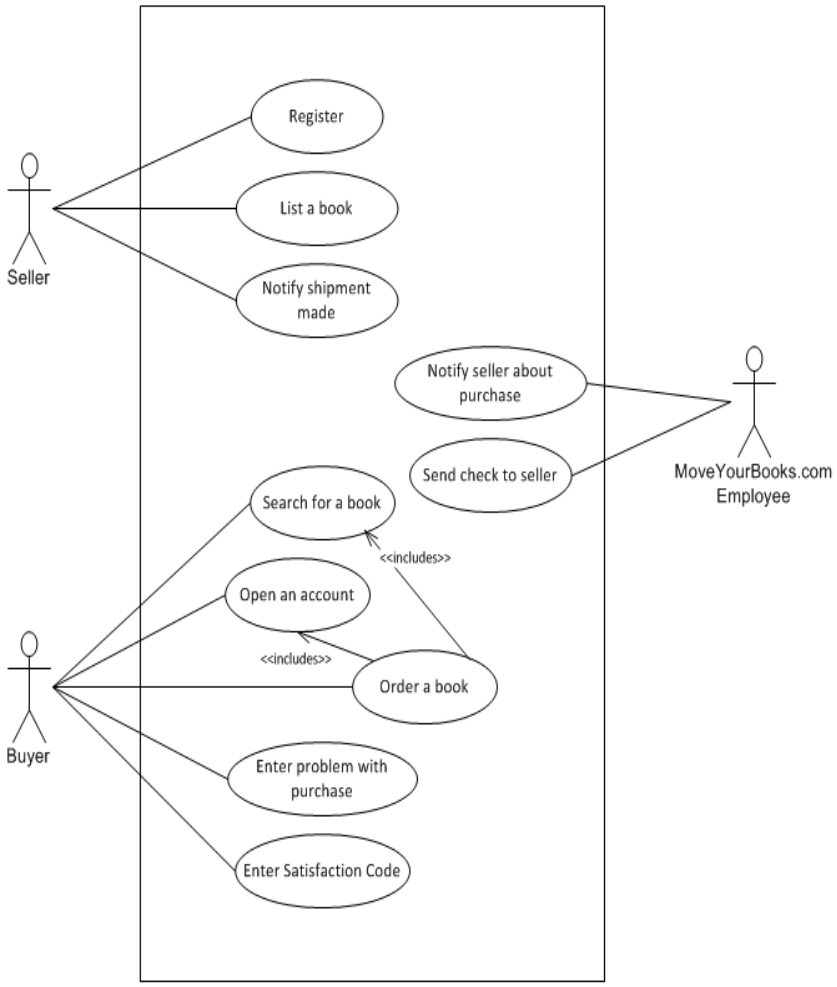
Case Study: MoveYourBooksNow.com Book Exchange

For this case, develop the following diagrams:

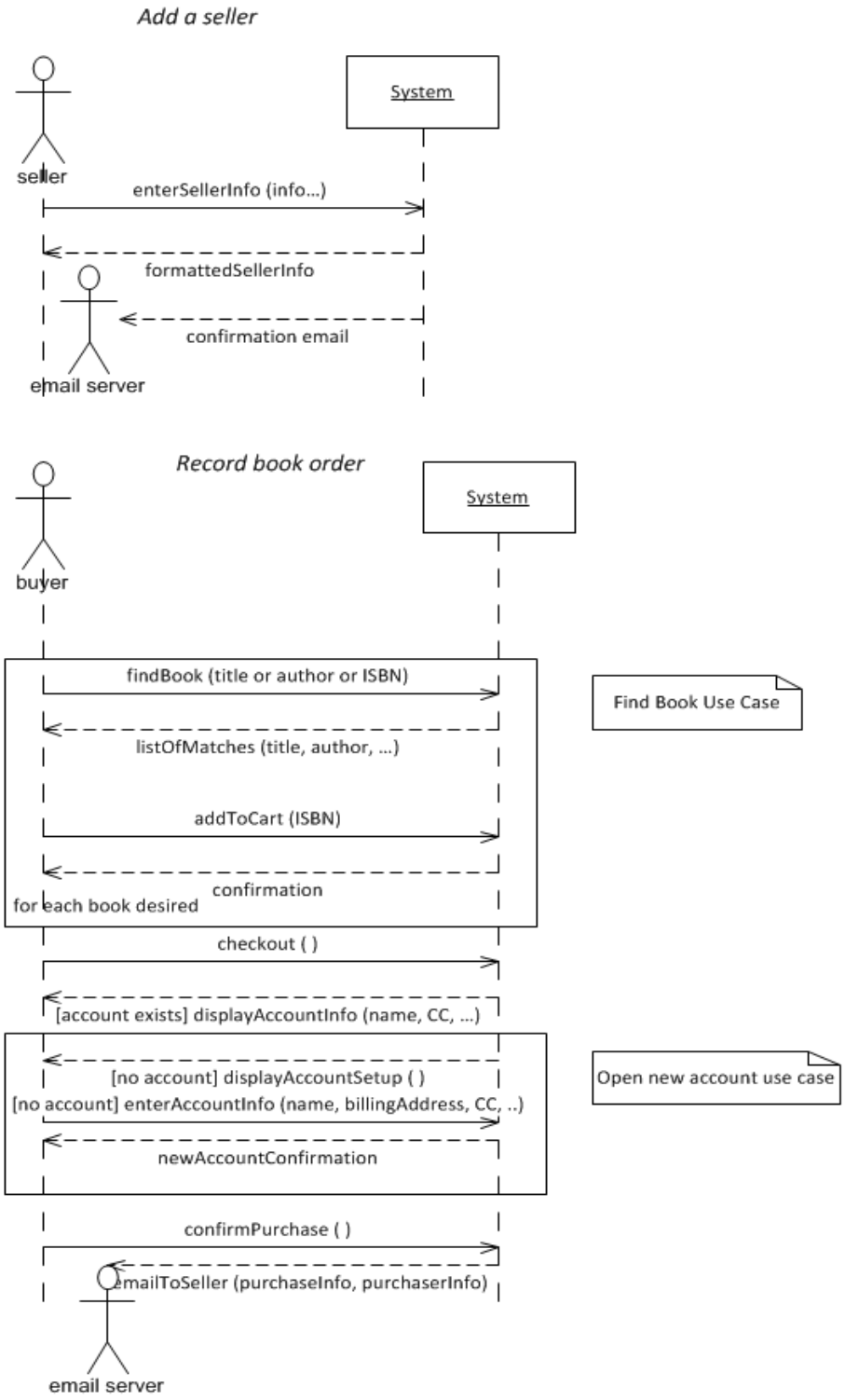
1. A domain model class diagram



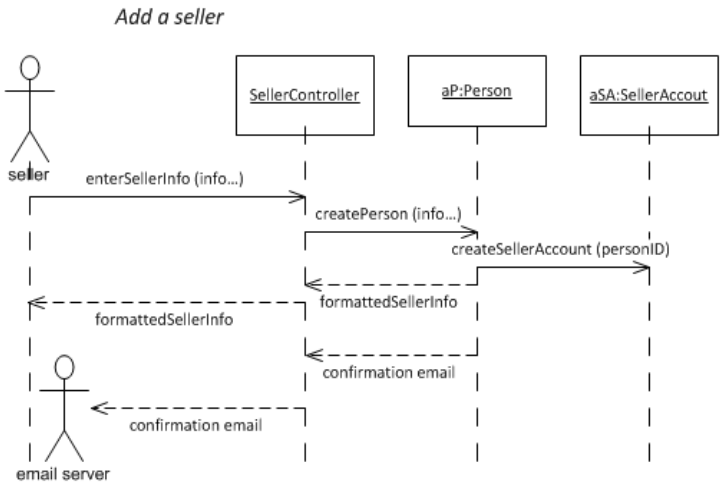
2. A use case diagram

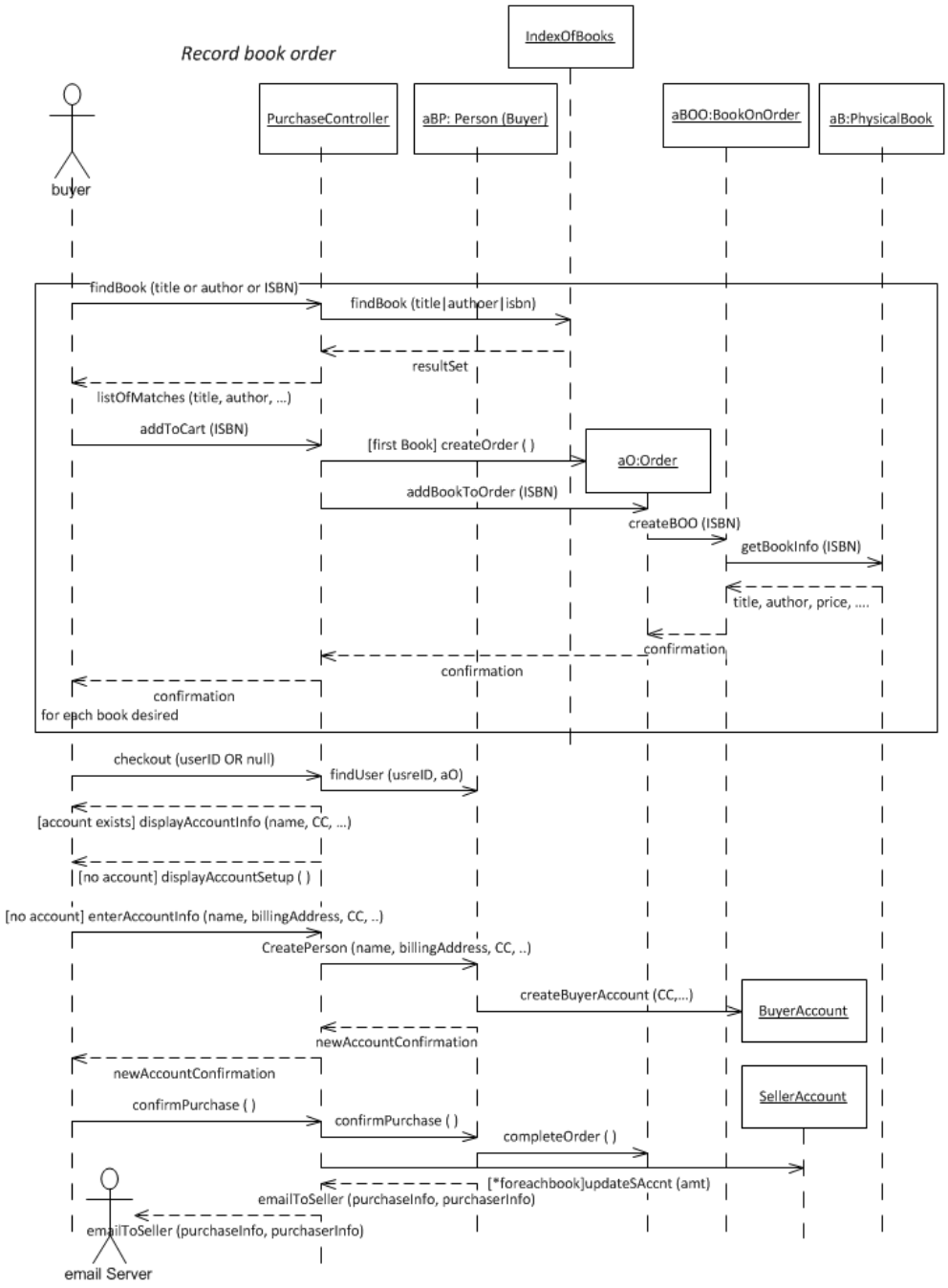


3. SSDs for two use cases, such as *Add a seller* and *Record a book order*

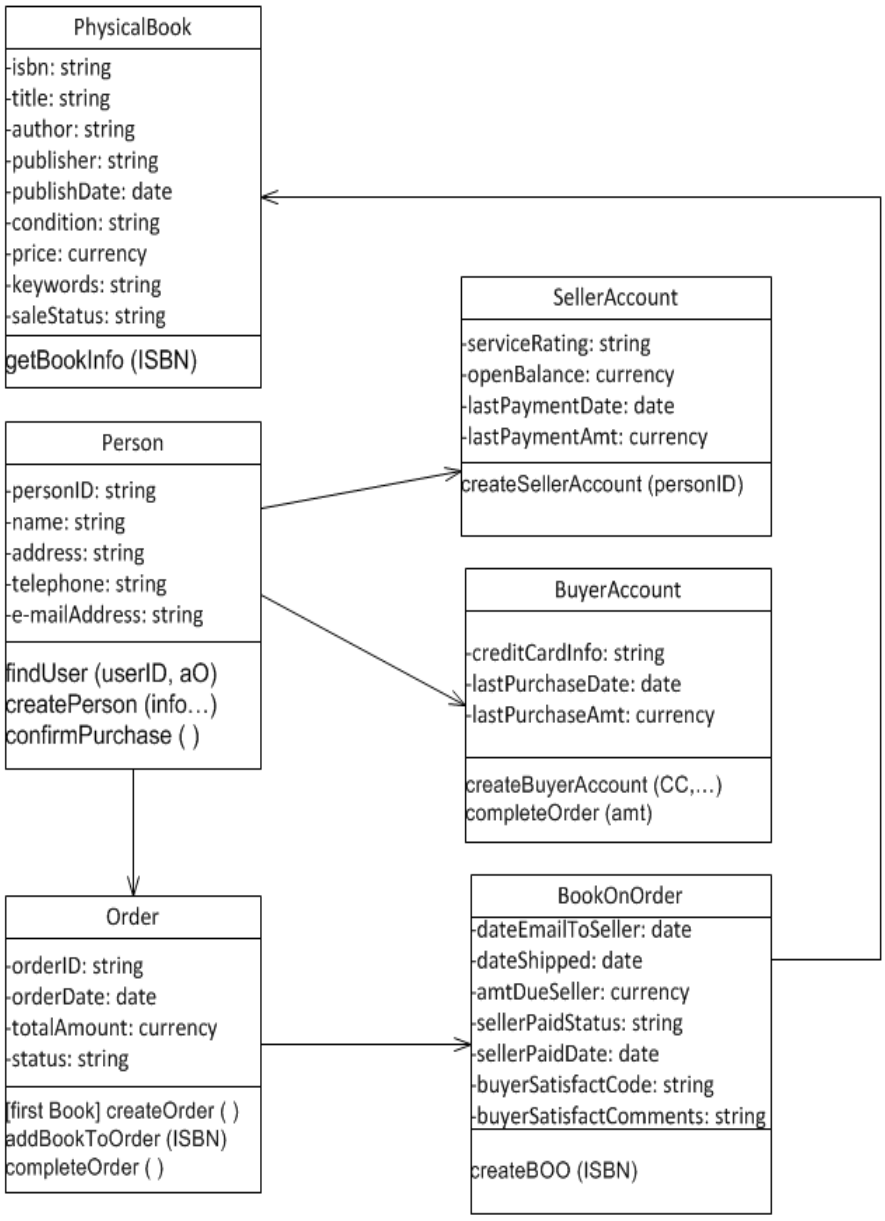


4. A first-cut sequence diagram for each of the above use cases




Record book order
IndexOfBooks
buyer
PurchaseController
aBP: Person (Buyer)
aBOO:BookOnOrder
aB:PhysicalBook
findBook (title or author or ISBN)
findBook (title|author|isbn)
resultSet
listOfMatches (title, author, ...)
addToCart (ISBN)
[first Book] createOrder ( )
aO:Order
addBookToOrder (ISBN)
createBOO (ISBN)
getBookInfo (ISBN)
title, author, price, ...
confirmation
confirmation
confirmation
for each book desired
checkout (userID OR null)
findUser (usreID, aO)
[account exists] displayAccountInfo (name, CC, ...)
[no account] displayAccountSetup ( )
[no account] enterAccountInfo (name, billingAddress, CC, ..)
CreatePerson (name, billingAddress, CC, ..)
createBuyerAccount (CC,...)
BuyerAccount
newAccountConfirmation
newAccountConfirmation
SellerAccount
confirmPurchase ( )
confirmPurchase ( )
completeOrder ( )
[*foreachbook]updateSAccnt (amt)
emailToSeller (purchaseInfo, purchaserInfo)
emailToSeller (purchaseInfo, purchaserInfo)
email Server

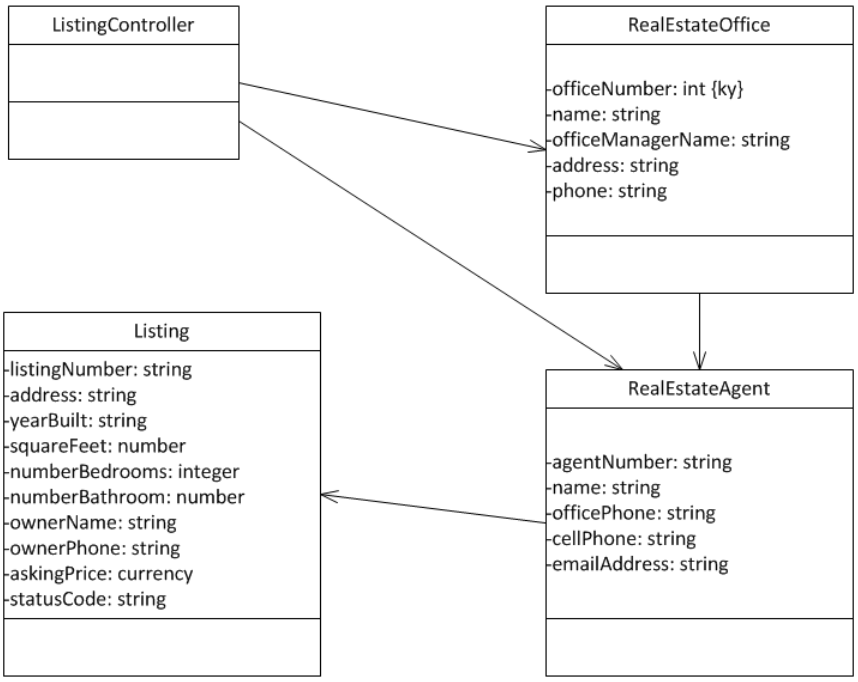
5. An integrated design class diagram that includes classes, methods, and navigation attributes



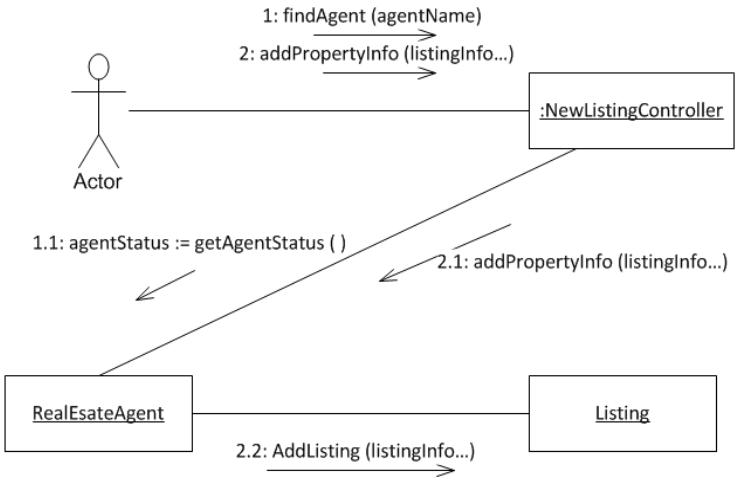
Running Cases: Community Board of Realtors

In Chapter 3, you identified use cases for the business events for the Community Board of Realtors. In Chapter 5, you elaborated on those use cases. In Chapter 4, you identified the classes associated with the business events. Using your solutions from those chapters, develop:

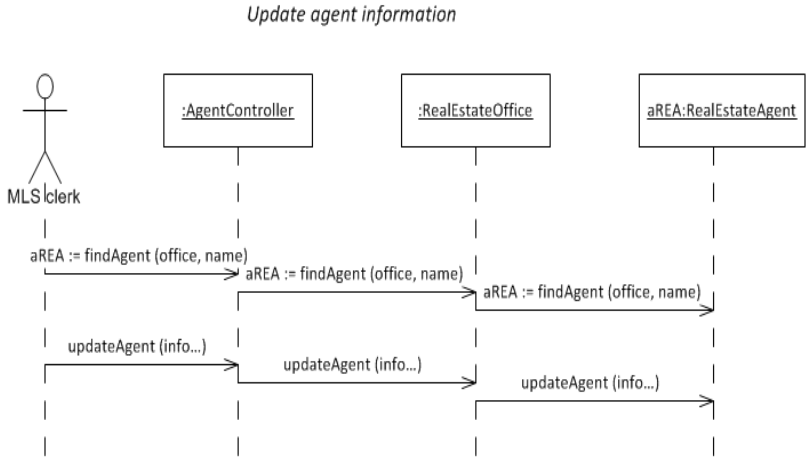
1. A first-cut DCD by using the problem domain classes that you identified in Chapter 4.



2. A first-cut communication diagram for the *Create new listing* use case (domain classes and controller class only).

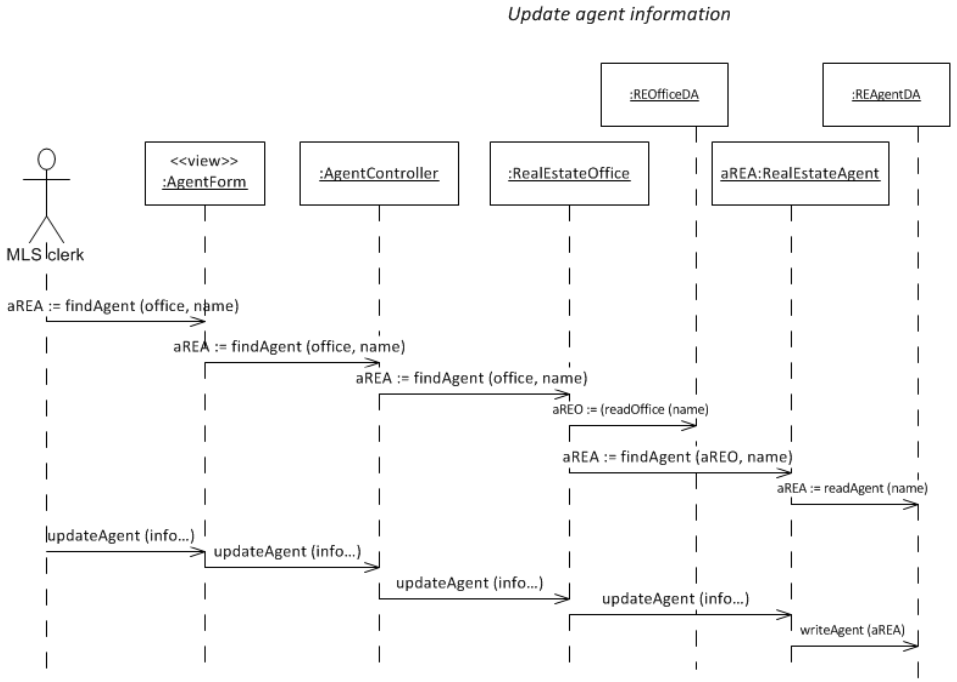


3. A first-cut sequence diagram for the *Update agent information* use case (domain classes and controller class only).

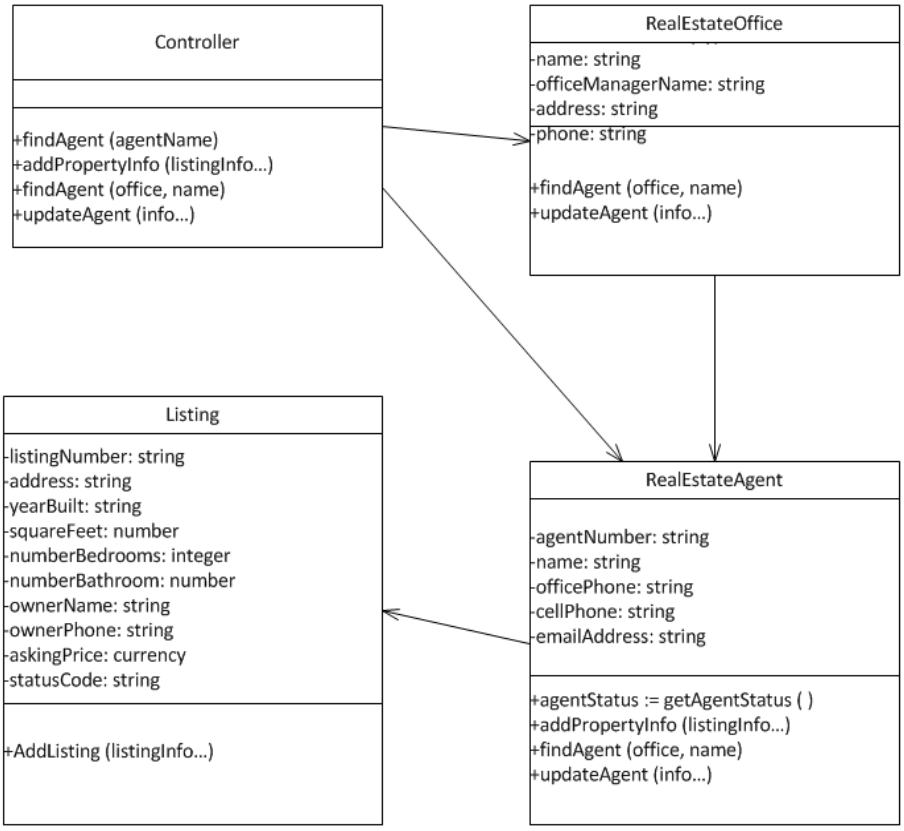


4. A multilayer sequence diagram for the *Update agent information* use case that includes domain classes and data access classes.

5. A separate multilayer sequence diagram for the *Update agent information* use case that includes the domain classes and the view layer classes.



6. A final design class diagram that includes the classes from both use cases. Include elaborated attributes, navigation arrows, and all the method signatures from both use cases.

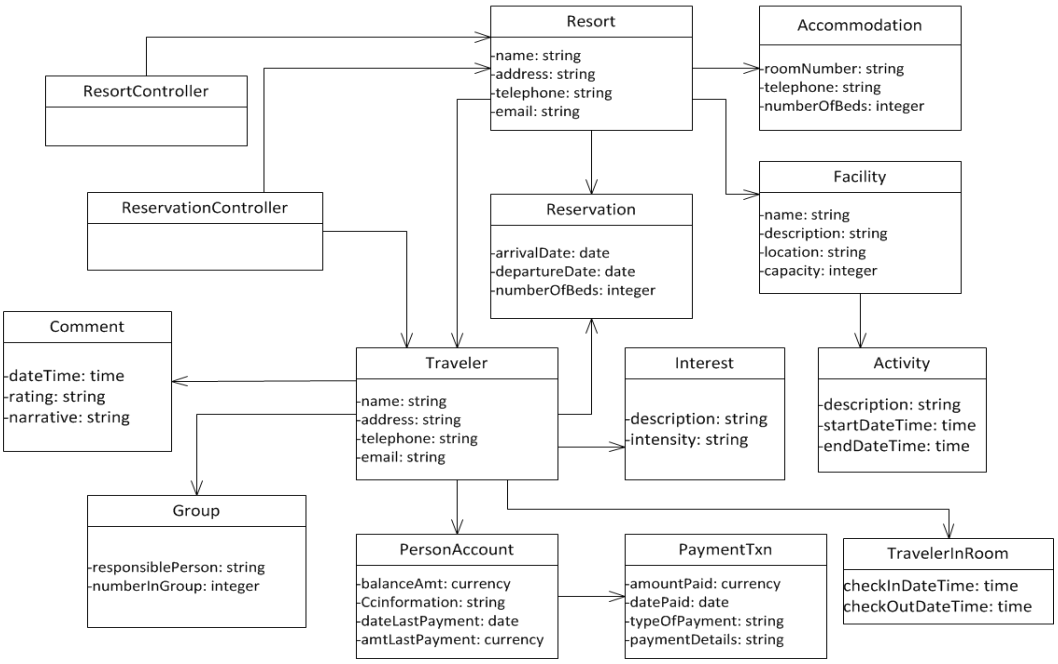

Controller
+findAgent (agentName)
+addPropertyInfo (listingInfo...)
+findAgent (office, name)
+updateAgent (info...)
RealEstateOffice
-name: string
-officeManagerName: string
-address: string
-phone: string
+findAgent (office, name)
+updateAgent (info...)
Listing
-listingNumber: string
-address: string
-yearBuilt: string
-squareFeet: number
-numberBedrooms: integer
-numberBathroom: number
-ownerName: string
-ownerPhone: string
-askingPrice: currency
-statusCode: string
+AddListing (listingInfo...)
RealEstateAgent
-agentNumber: string
-name: string
-officePhone: string
-cellPhone: string
-emailAddress: string
+agentStatus := getAgentStatus ( )
+addPropertyInfo (listingInfo...)
+findAgent (office, name)
+updateAgent (info...)

Running Cases: The Spring Breaks 'R' Us Travel Service

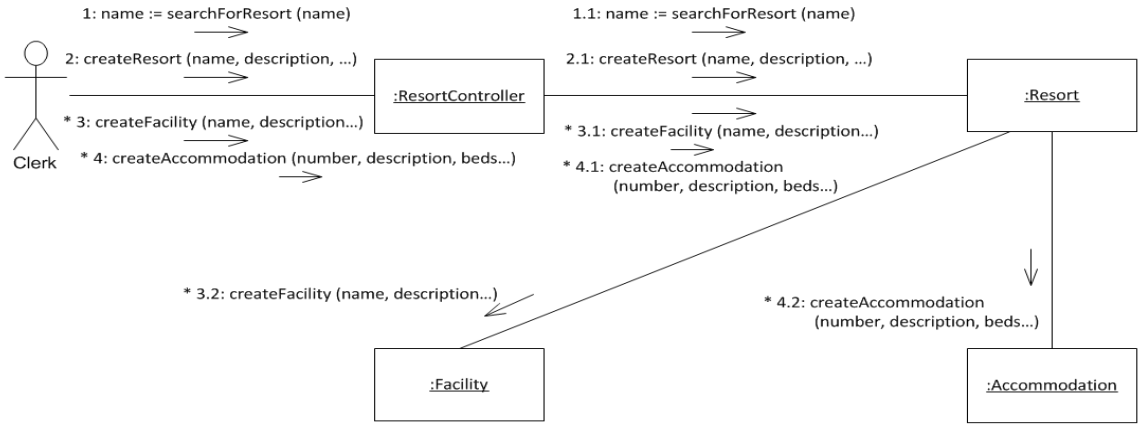
In Chapter 3, you identified use cases for the business events for the Spring Breaks 'R' Us Travel Service. In Chapter 5, you elaborated on those use cases. In Chapter 4, you identified the classes associated with the business events. Using your solutions from those chapters, develop:

1. A first-cut DCD by using the problem domain classes you identified in Chapter 4.

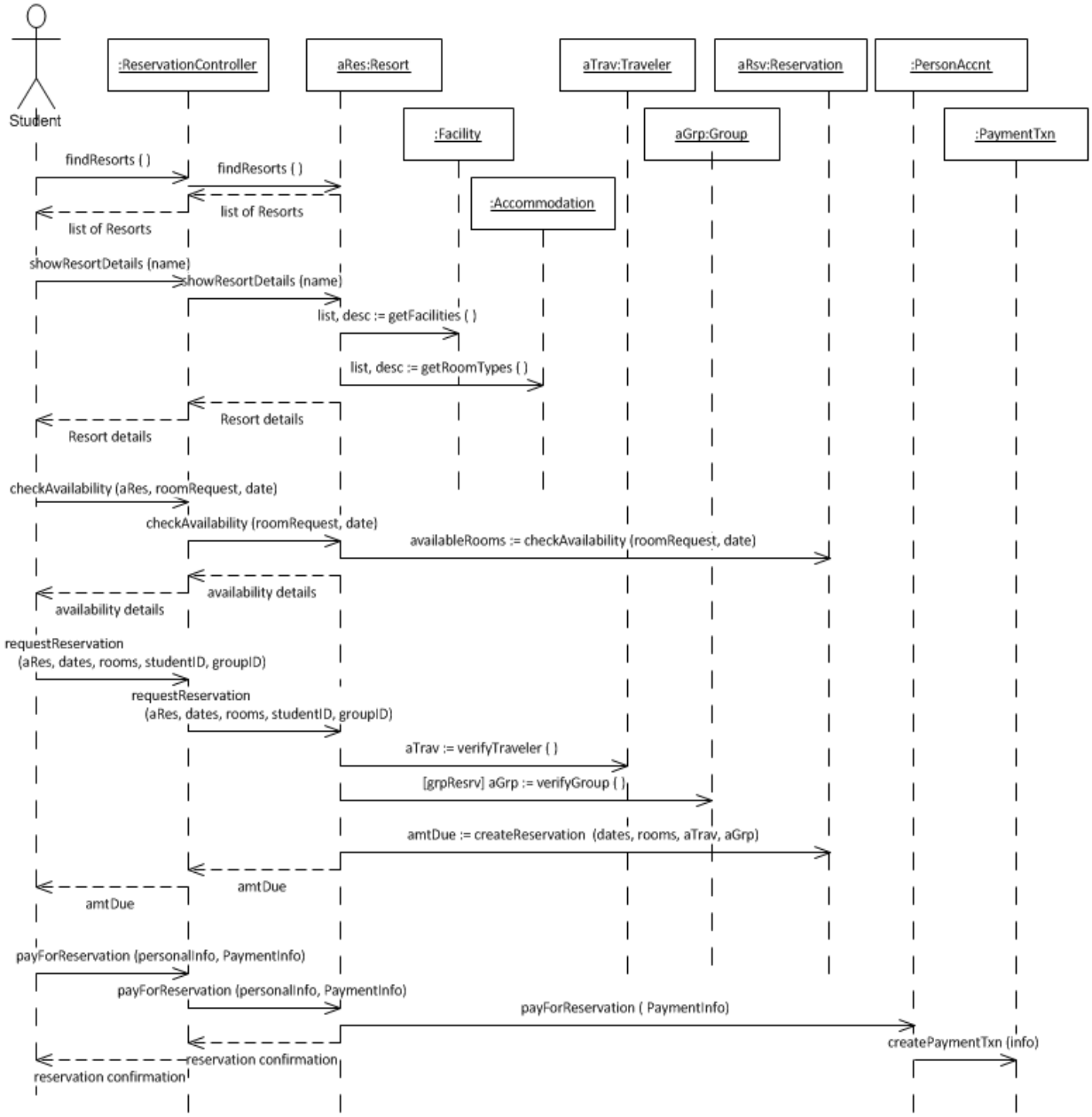
Note: This DCD is an expanded one, which also includes the classes that were added in Chapter 5 to support the use cases. Student answers will not have as many classes.



2. A first-cut communication diagram for the *Add a resort* use case (domain classes and controller class only).



3. A first-cut sequence diagram for the *Book a reservation* use case.
Note: Students can use the SSD given in Chapter 5 or the CRC cards from Chapter 10 as guidelines.

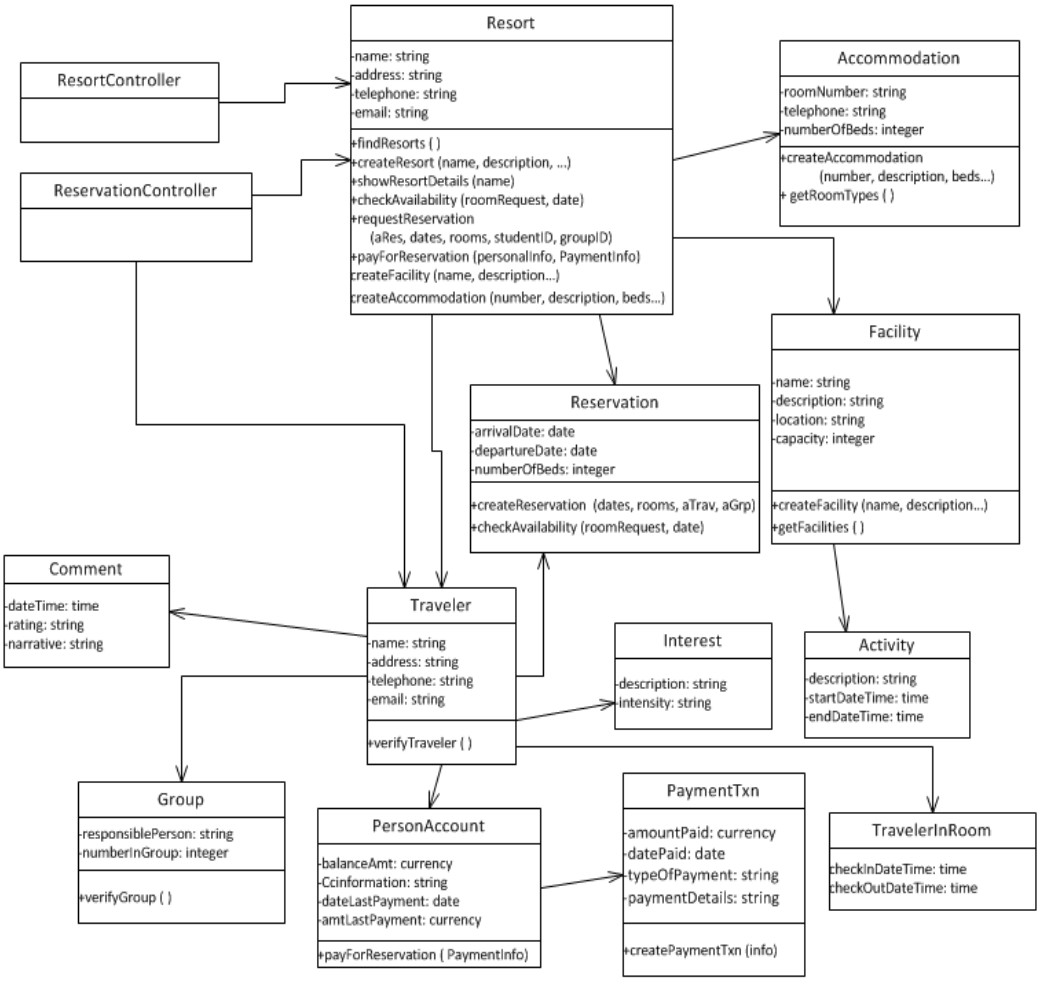


4. A multilayer sequence diagram for the *Book a reservation* use case that includes domain classes and data access classes.

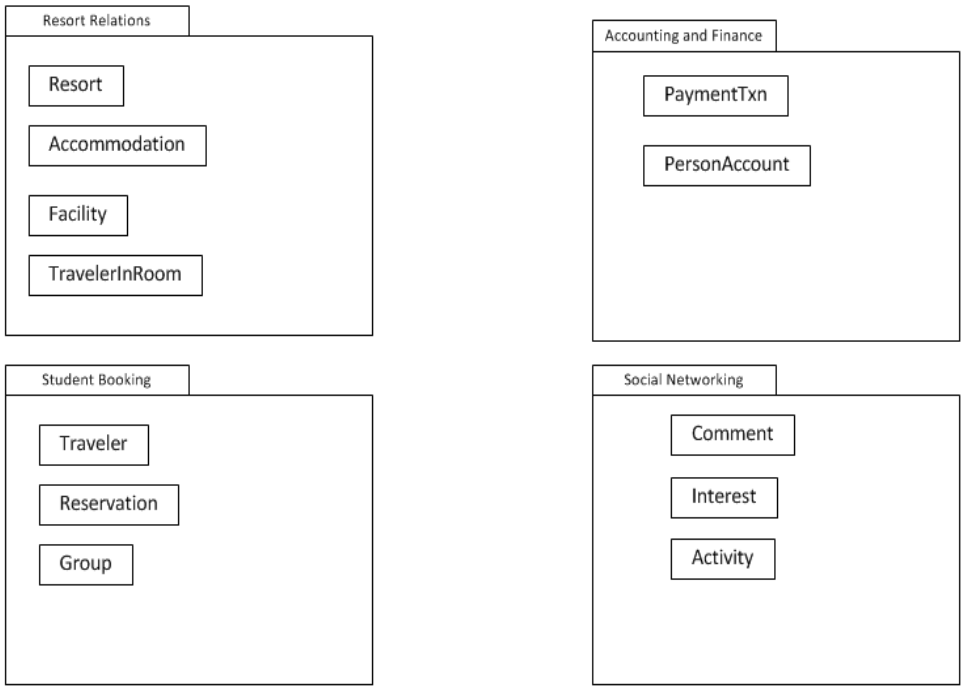
5. A separate multilayer sequence diagram for the *Book a reservation* use case that includes the domain classes and the view layer classes.

Note: Due to the complexity of this diagram, it will be best to divide this use case into two pages. A good dividing point is before checkAvailability ( ) message.

6. A final design class diagram that includes the classes from both use cases. Include elaborated attributes, navigation arrows, and all the method signatures from both use cases.



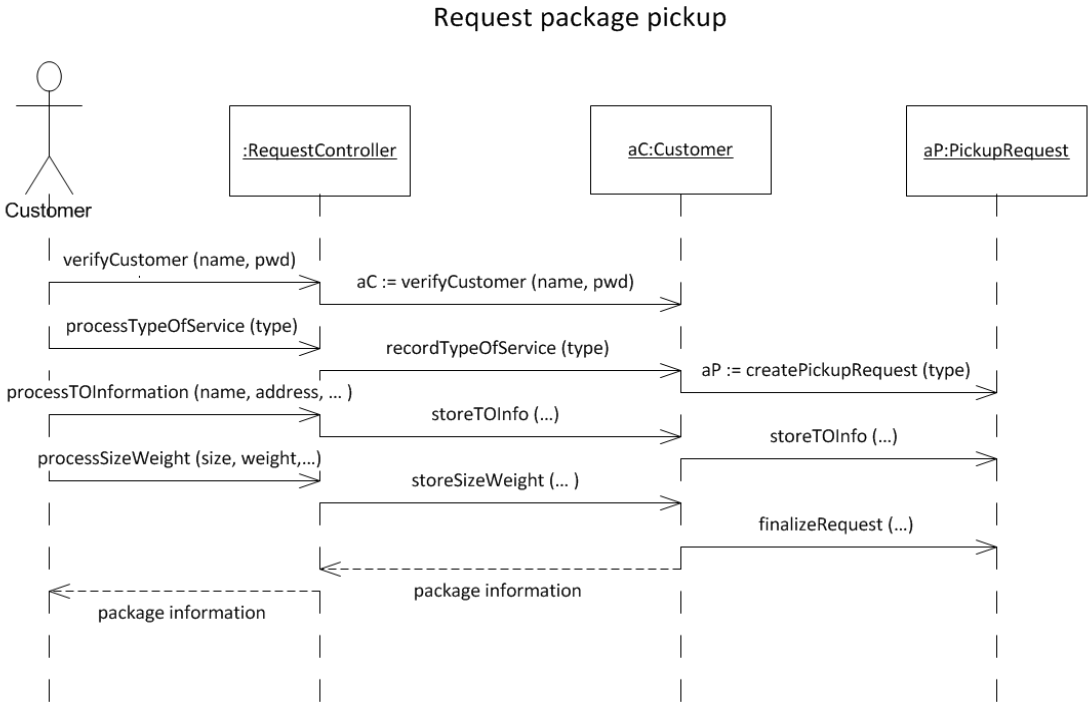
7. A package diagram of the four subsystems (Resort relations, Student booking, Accounting and finance, and Social networking) that includes all the problem domain classes.

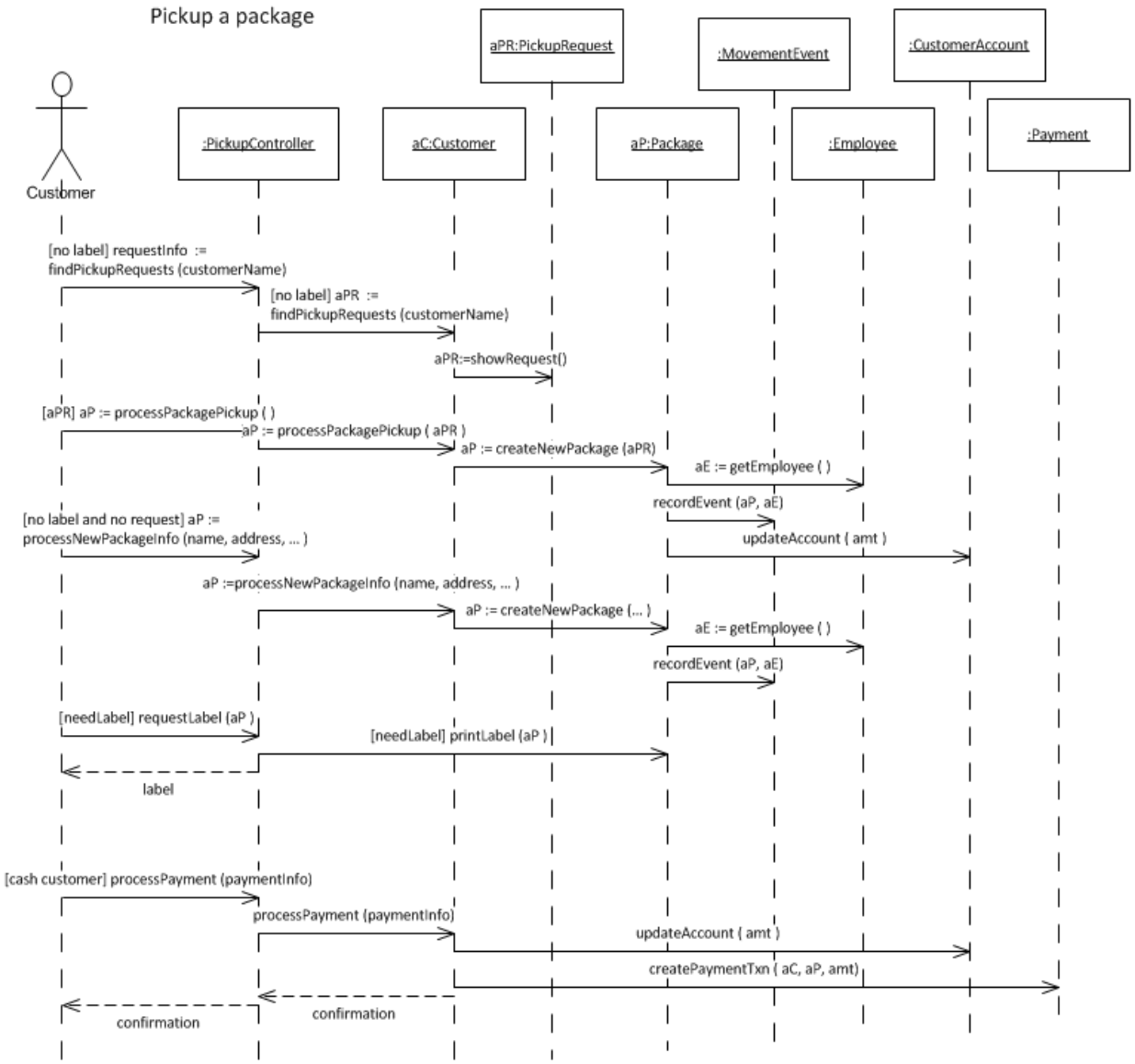


Running Cases: On the Spot Courier Services

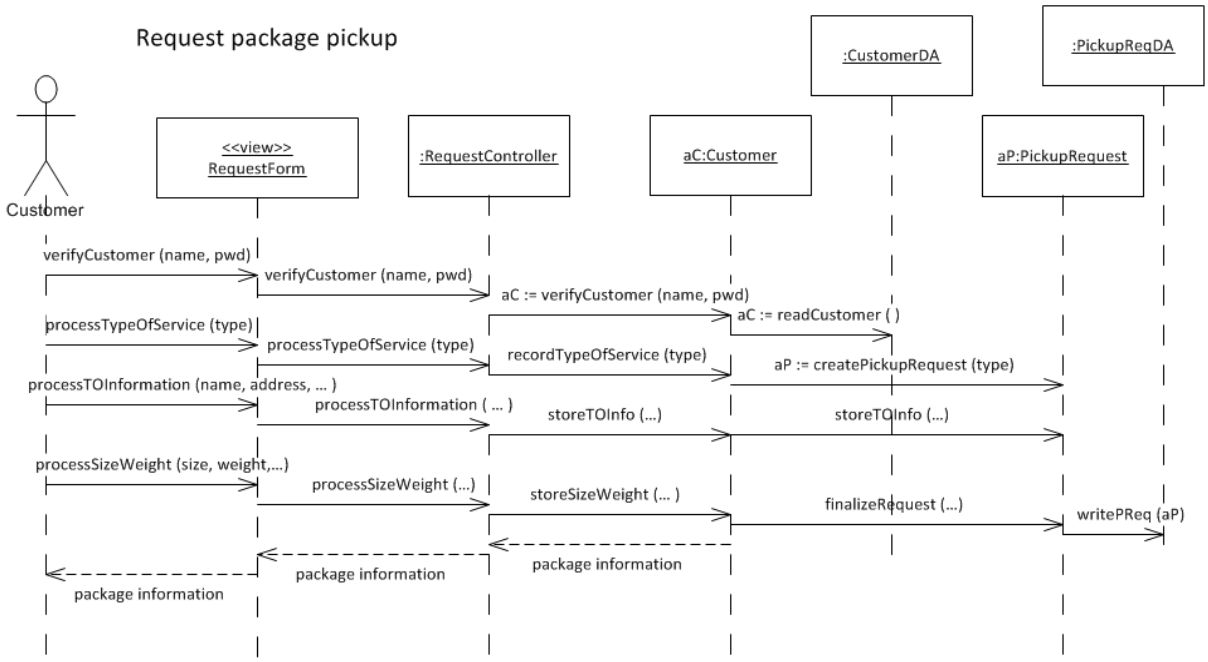
In Chapter 10, you developed a first-cut design class diagram and CRC card solutions for two use cases: *Request a package pickup* and *Pickup a package*. Let us extend your solution from that chapter by developing the following:

1. A first-cut sequence diagram for each use case (domain classes and controller classes only).

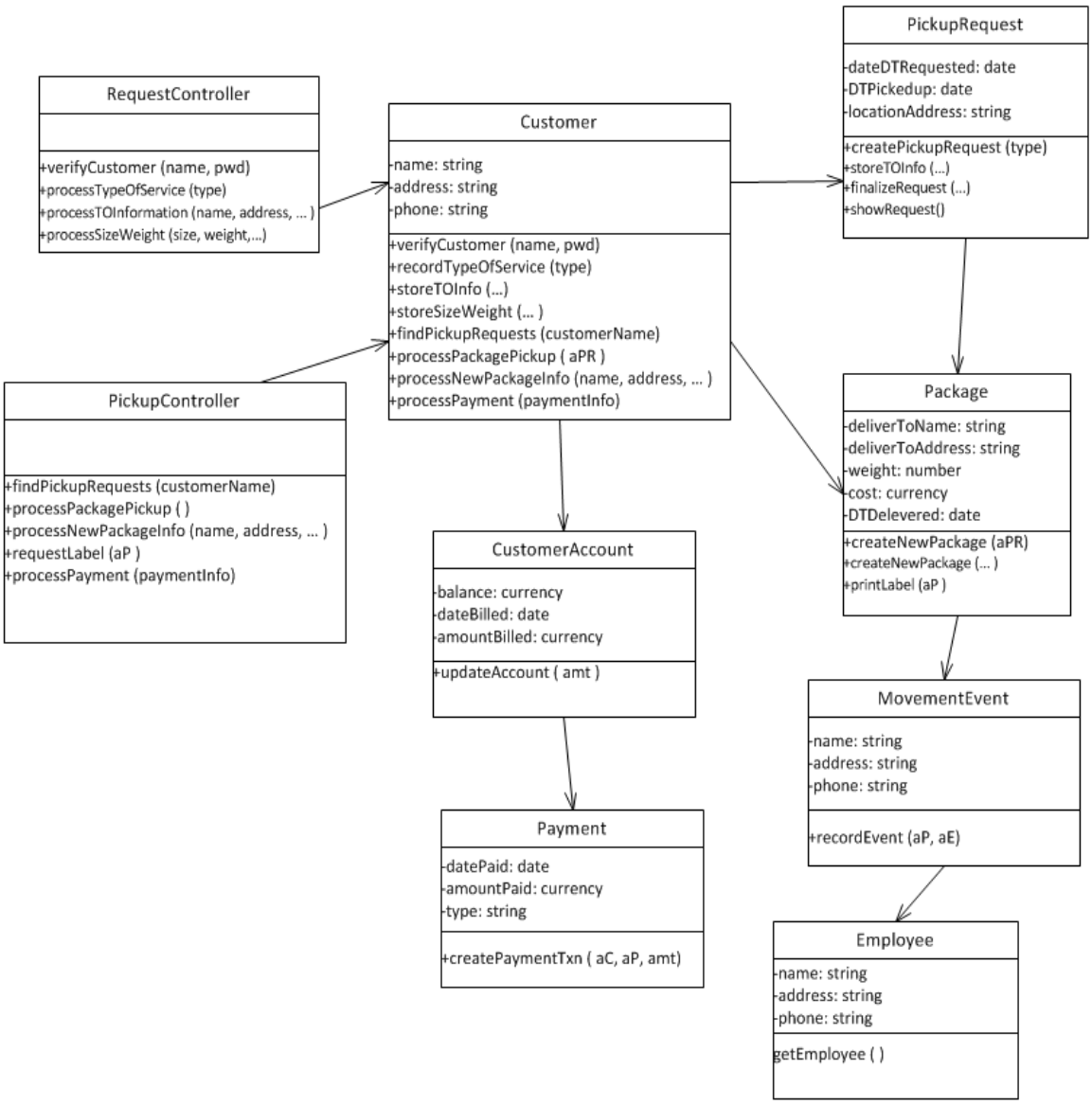



Pickup a package
aPR:PickupRequest
:MovementEvent
:CustomerAccount
:Payment
Customer
:PickupController
aC:Customer
aP:Package
:Employee
[no label] requestInfo :=
findPickupRequests (customerName)
[no label] aPR :=
findPickupRequests (customerName)
aPR:=showRequest()
[aPR] aP := processPackagePickup ( )
aP := processPackagePickup (aPR)
aP := createNewPackage (aPR)
aE := getEmployee ( )
recordEvent (aP, aE)
[no label and no request] aP :=
processNewPackageInfo (name, address, ...)
updateAccount (amt)
aP :=processNewPackageInfo (name, address, ...)
aP := createNewPackage (...)
aE := getEmployee ( )
recordEvent (aP, aE)
[needLabel] requestLabel (aP)
[needLabel] printLabel (aP)
label
[cash customer] processPayment (paymentInfo)
processPayment (paymentInfo)
updateAccount (amt)
createPaymentTxn (aC, aP, amt)
confirmation
confirmation

2. A multilayer sequence diagram for each use case that includes domain classes and data access classes.
3. A separate multilayer sequence diagram for each use case that includes the domain classes and the view layer classes. (We won't combine view and data access layers on the same drawing. It makes the drawing too complex.)



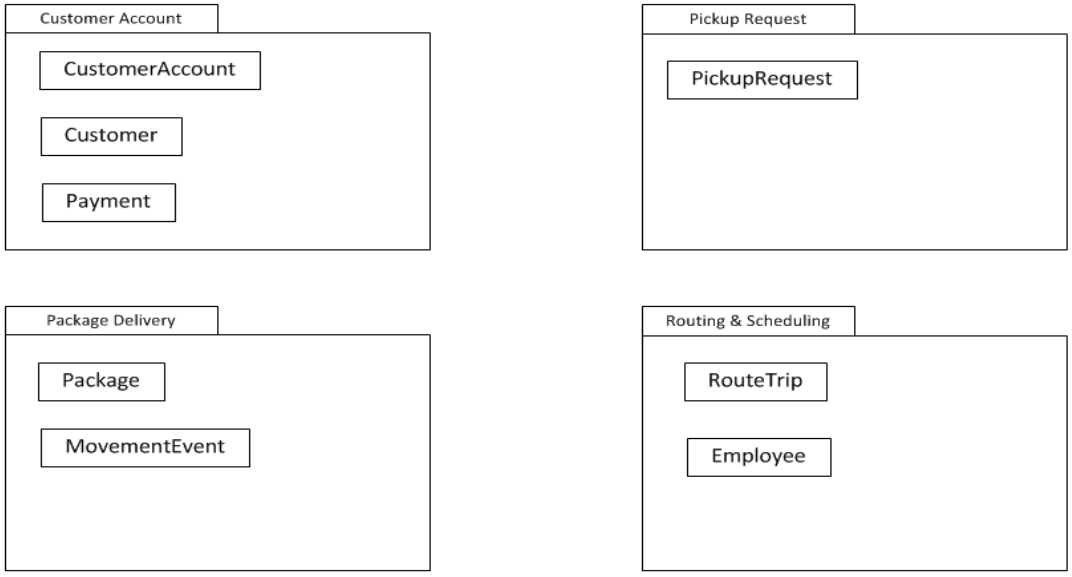
4. A final design class diagram that includes the classes from both use cases. Include elaborated attributes, navigation arrows, and all the method signatures from both use cases.



In Chapter 9, we defined four subsystems:

- **Customer account (like customer account)**
- **Pickup request (like sales)**
- **Package delivery (like order fulfillment)**
- **Routing and scheduling**

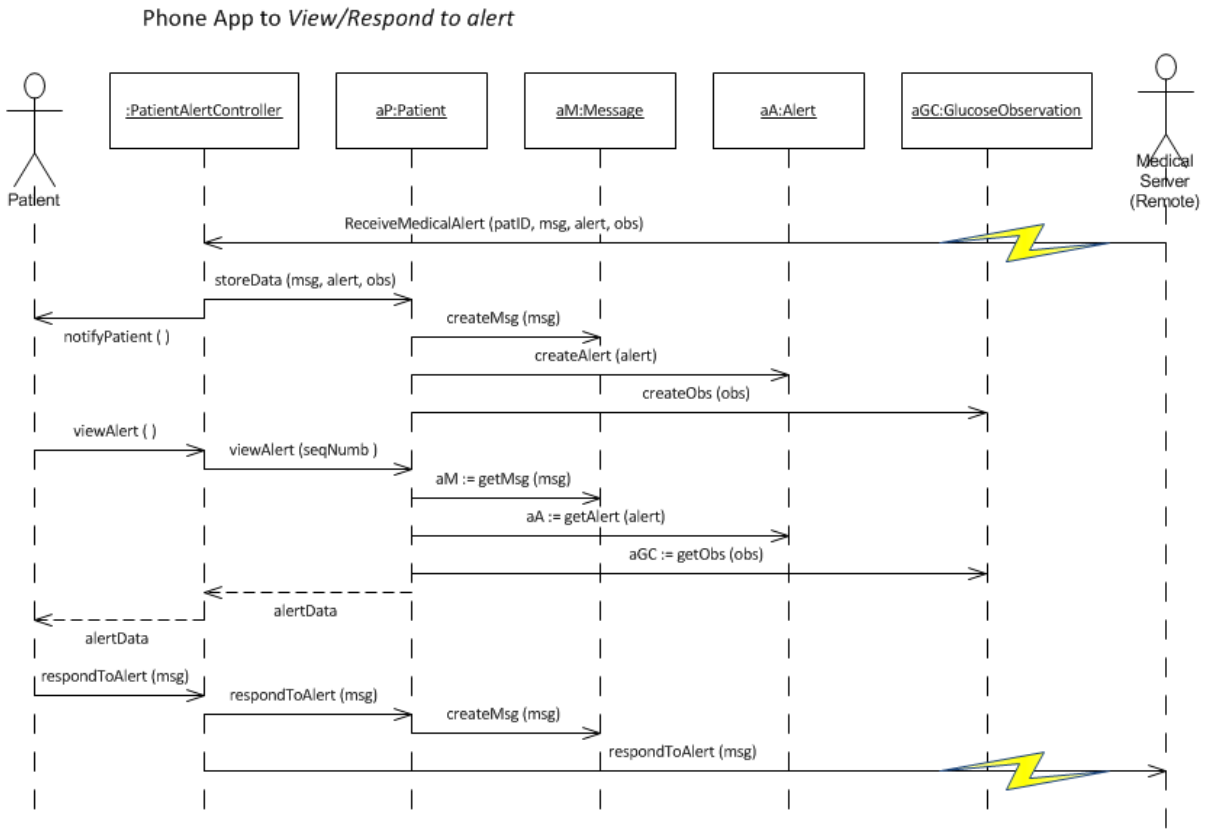
Even though these subsystems are somewhat arbitrary, we can treat each one as a separate package. Develop a package diagram for each of the four subsystems by assigning domain model classes to each package. A domain model class should belong to only one subsystem package. Normally, it is the subsystem that instantiates objects from that class. Also, show dependency relationships among the various packages and classes.



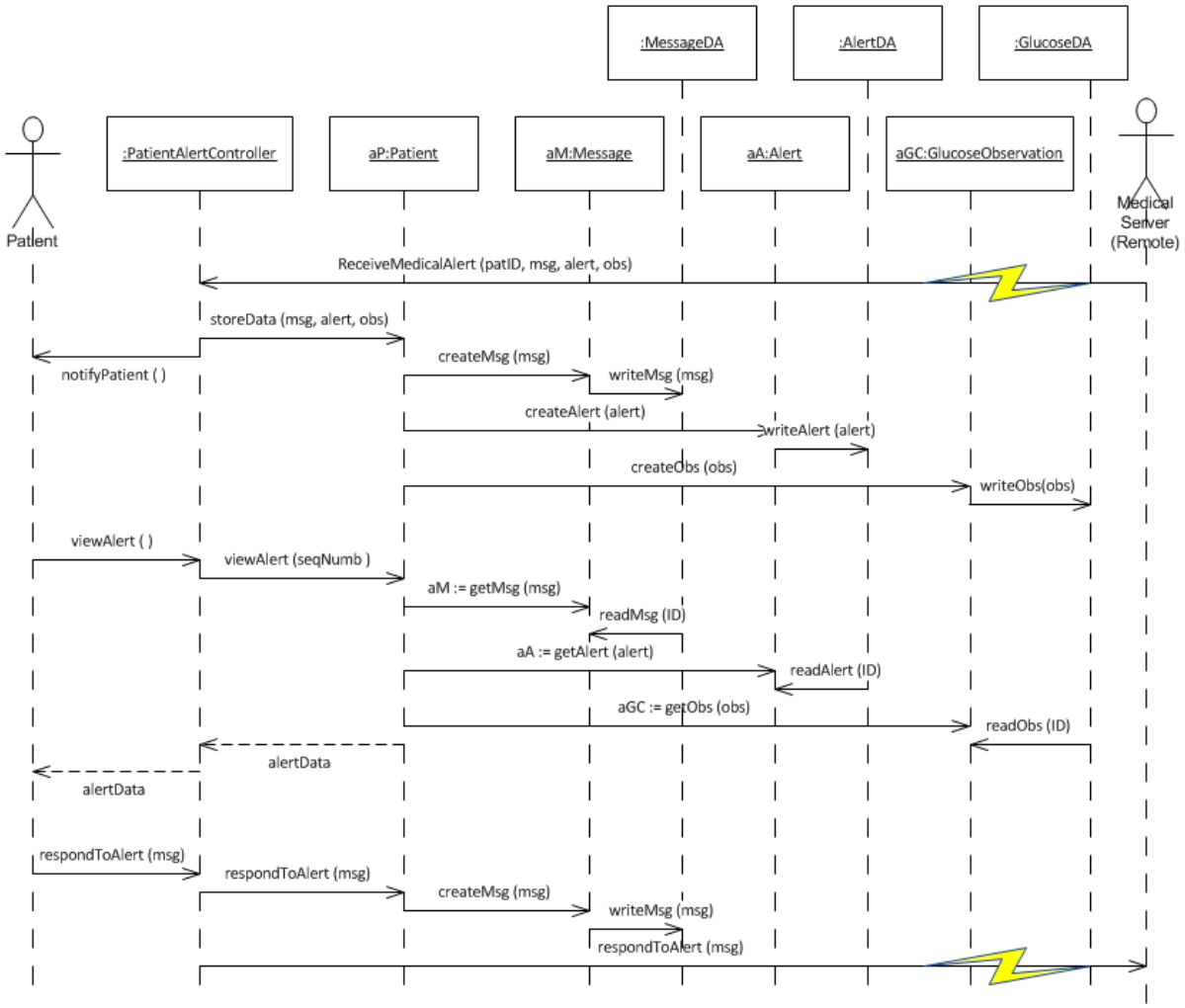
Running Cases: Sandia Medical Devices

Review your answers to the case-related questions in Chapter 10 and then do the following:

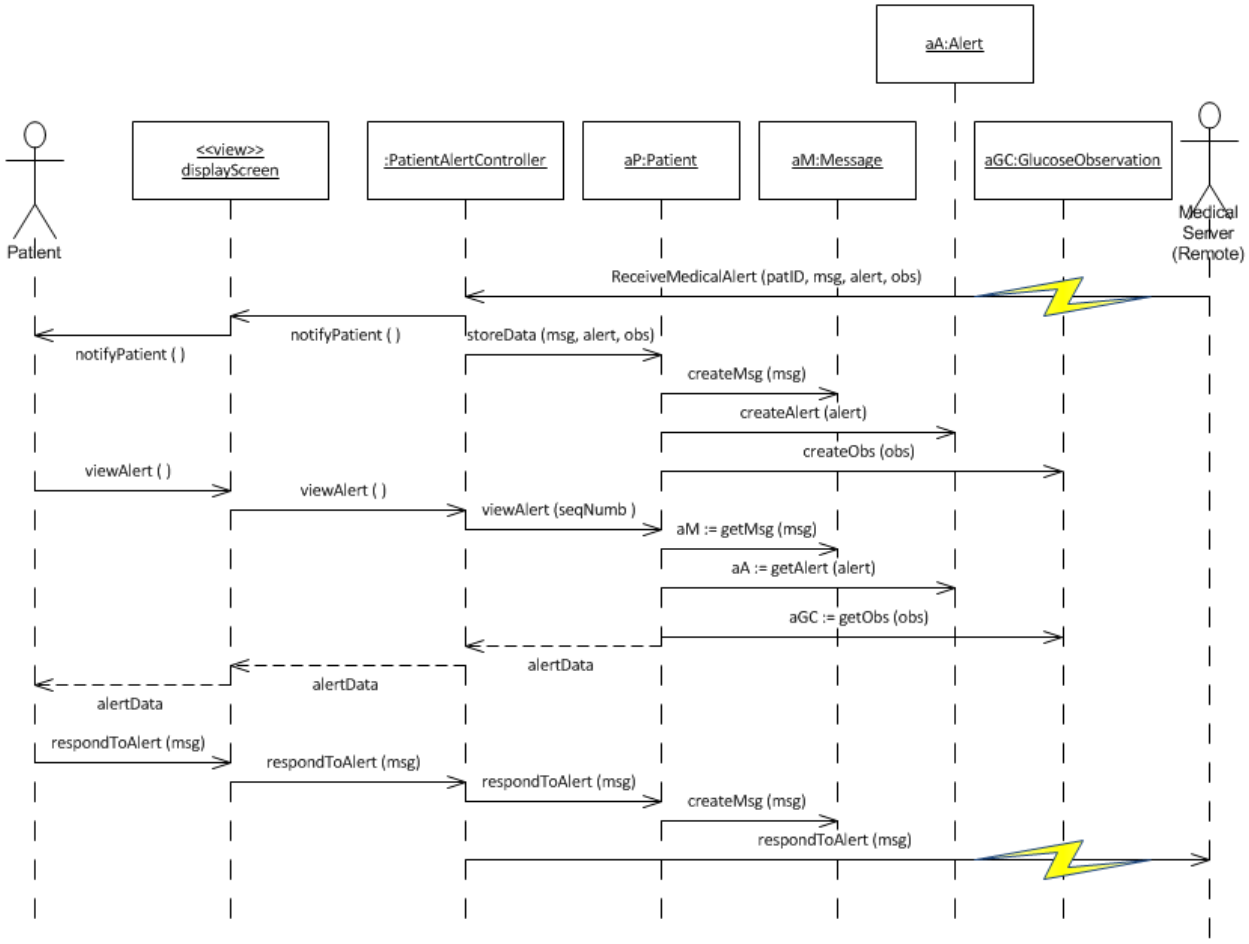
1. Develop a first-cut sequence diagram for the patient use case View/respond to alert.



2. Develop a multilayer sequence diagram that includes domain classes and data access classes.



3. Develop a separate multilayer sequence diagram that includes the domain classes and the view layer classes. (We won't combine view and data access layers on the same drawing. It makes the drawing too complex.)



4. Update your DCD from Chapter 10 to include the methods you have identified. Also, include any changes you may have made to navigation visibility and attribute details.