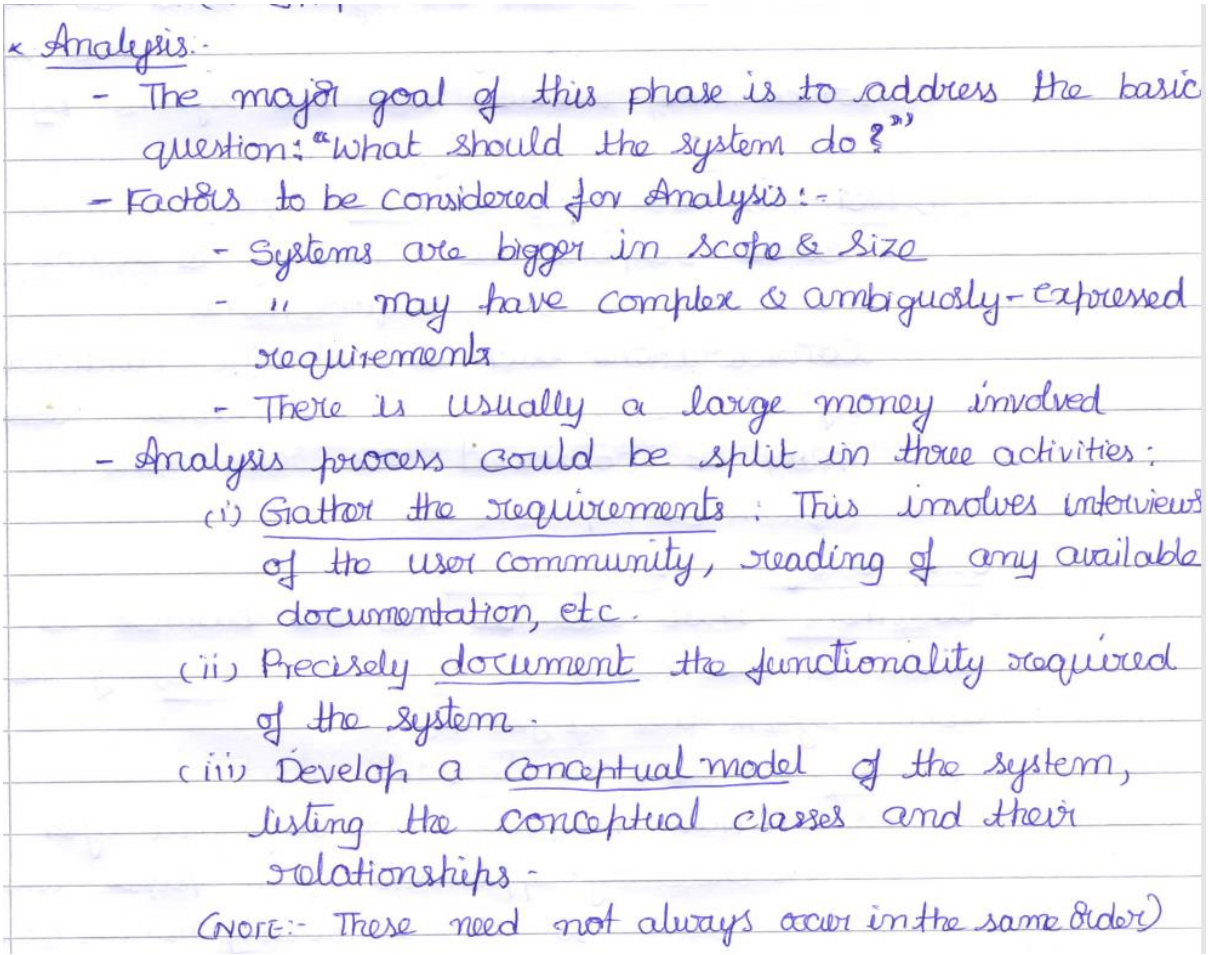
Solution

Internal Assessment Test 2 – October.2019

Sub:	Software Architecture & Design Patterns							Code:	17IS72
Date:	12/10/2019	Duration:	90mins	Max Marks:	50	Sem:	VII	Branch:	ISE

Note: Answer Any Five Questions

Explain the major steps in analysis phase



The major step is gathering the requirements.

Stage 1:- Gathering the requirements.

* Requirements for a new system are determined by a team of analysts by interacting with teams from the company paying for the development (clients) and the user community, who ultimately uses the system on a day-to-day basis.
* This interaction can be in the form of interviews, surveys, observations, study of existing manuals, etc.,

existing manuals, etc.,

* Requirements can be classified in two categories:
  (i) Functional requirements:- These describe the interaction between the system and its users, and between the system and any other systems, which may interact with the system by supplying & receiving data.
  (ii) Non-Functional requirements:- Requirements that include response-time, usability & accuracy are non-functional requirements.
  considerations such as, Specific hardware & software, budget & time constraints may be considered as restrictions on system development.

* Case study:- Library system

- Document that describes how the business is conducted also known as business processes of the library system are as follows:

(i) Register new members :-

- New members are registered after receiving applications from people who want to become library members (users).
- Details such as name, contact no., address

are provided to the library for membership.
- The library assigns a unique ID for the transactions.

(ii) Add books to the collection:-
- For each book, a unique ID is maintained, along with title, author's name.

(iii) Issue a book to a member (user)
- A book is issued to a user, by registering the details of the book with its unique ID along with user's ID, in a single transaction

(iv) Record the return of a book
- A book that is returned, updates the status of the book and the user.
- If there is a hold on the book, the system should remind the clerks to set the book aside so that the hold can be processed.

(v) Remove books from the collection:-
- From time to time, the library may remove books from its collection, as they are worn-out, or are no longer of interest to the users, etc.,

(vi) Print out a user's transaction:-
- print out the interactions between a specific user and the library on a certain date.

(vii) place/remove a hold on a book:-
- To do so, book's ID & user's ID is supplied to the clerks and things are done accordingly.

(viii) Renew books issued to a member:-
- Users may request for renewal of the books issued.
- The system must display the relevant books, allow the user to make a selection, and inform the user of the result.

(ix) Notify member of book's availability :-
- Users who had placed a hold on a book are notified when the book is returned. This process is done once at the end of each day. The clerk enters the ID of each book that was set aside, and the system returns the name and phone number of the user who is next in line to get the books.

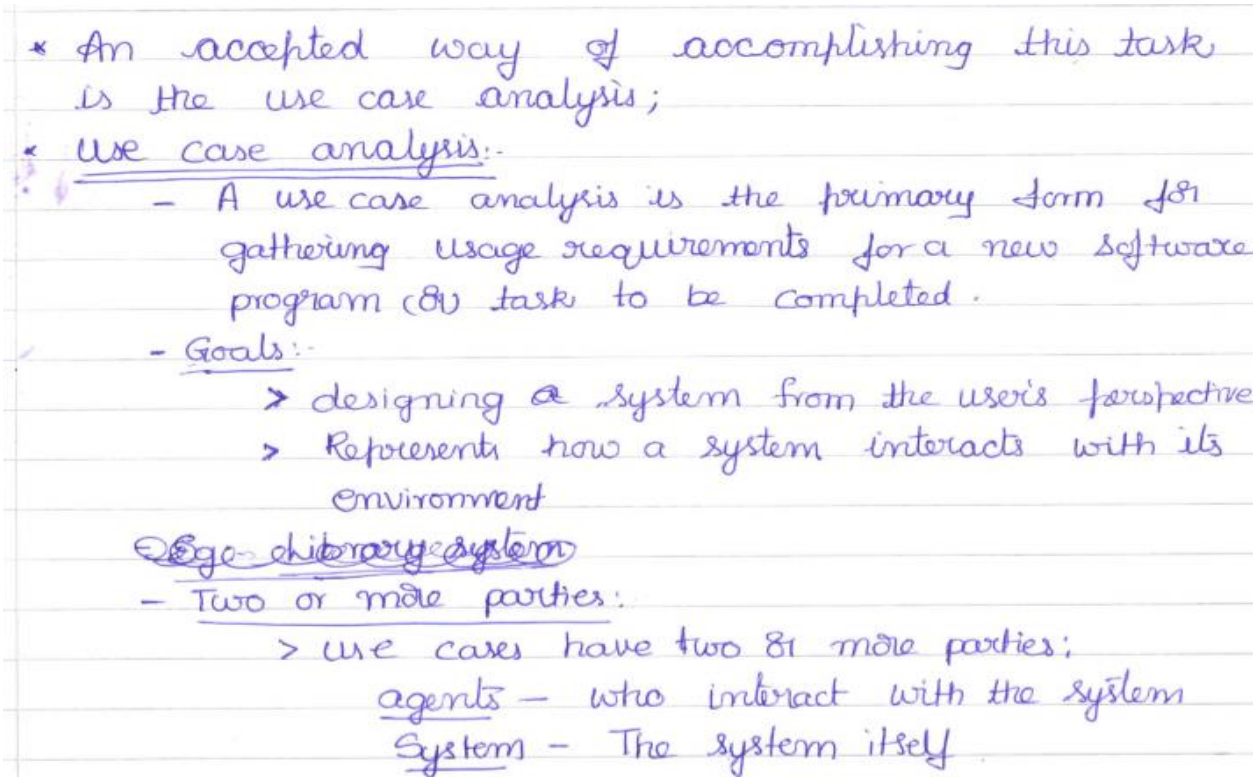
Additional requirements :
(i) A command to save the data on a long-term basis
(ii) A command to load data from a long-term storage device
(iii) A command to quit the application.
(iv) Generate reports of various kinds.
(v) Allow users to check out books themselves.

So, from these requirements, we need to document the functional requirements of the application and determine the system's major entities & relationships

2) a) Explain functional requirement specification
b) List and explain the guidelines to remember while writing Use cases.

Functional Requirements Specification

- The functional requirements specification documents the operations and activities that a system must be able to perform.
- Various functional requirements includes:
  (i) data descriptions to be entered into the system
  (ii) operations descriptions
  (iii) work-flows performed
  (iv) System reports or other outputs
  (v) who can enter data, etc.,



b) guidelines to remember when writing Use cases

A use case describes a certain piece of desired functionality of an application system. It is constructed during the analysis stage. It shows the interaction between an actor, which could be human or a piece of software or hardware and the system. It doesn't specify how system carries out the task.

The elements of a use case diagram include:

use cases - specific pieces of the system's functionality
actors - people or things that interact with the system's use cases
associations - used to link actors with the use cases they interact with
system boundary - defines what functionality is included within the system

In general use case diagrams are used for:

- Analyzing the requirements of a system
- High-level visual software designing
- Capturing the functionalities of a system
- Modeling the basic idea behind the system
- Forward and reverse engineering of a system using various test cases.

3) Explain use case analysis with an example.

1.

- Eg:- Library System

> The following diagram shows the Use case for the library system.

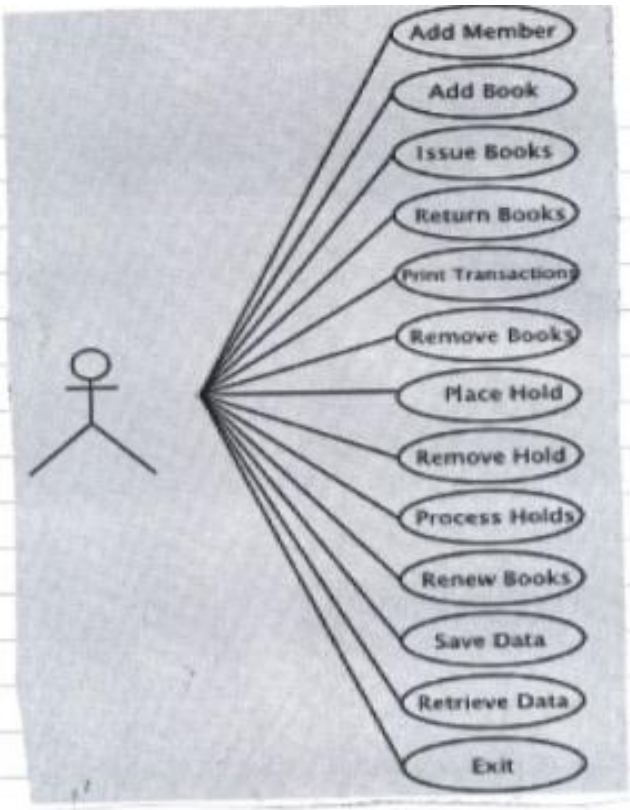
> The actors in our system are members of the library staff who manage the daily operations.

> The system's usage requirements are also depicted in the use-case diagram

P.T.O

> NOTE1:- Even in case of issuing books, the functionality is invoked by a library staff member, who performs the actions on behalf of a member.

> NOTE2:- Left-side states the actor & his/her actions, right-side states what the system does.



* Use case for registering a user:

The use case for registering a new user is as follows:

Table 6.1 Use case Register New Member

Actions performed by the actor	Responses from the system
1. The customer fills out an application form containing the customer's name, address, and phone number and gives this to the clerk	
2. The clerk issues a request to add a new member	
	3. The system asks for data about the new member
4. The clerk enters the data into the system	

	5. Reads in data, and if the member can be added, generates an identification number (which is not necessarily a number in the literal sense just as social security numbers and phone numbers are not actually numbers) for the member and remembers information about the member. Informs the clerk if the member was added and outputs the member's name, address, phone and id
6. The clerk gives the user his identification number	

* usecase for adding books :

Table 6.2 Use case Adding New Books

Actions performed by the actor	Responses from the system
1. Library receives a shipment of books from the publisher	
2. The clerk issues a request to add a new book	
	3. The system asks for the identifier, title, and author name of the book
4. The clerk generates the unique identifier, enters the identifier, title, and author name of a book	
	5. The system attempts to enter the information in the catalog and echoes to the clerk the title, author name, and id of the book. It then asks if the clerk wants to enter information about another book
6. The clerk answers in the affirmative or in the negative	
	7. If the answer is in the affirmative, the system goes to Step 3. Otherwise, it exits

* Use case for issuing books -

Table 6.3 Use case Book Checkout

Actions performed by the actor	Responses from the system
1. The member arrives at the check-out counter with a set of books and supplies the clerk with his/her identification number	
2. The clerk issues a request to check out books	
	3. The system asks for the user ID
4. The clerk inputs the user ID to the system	
	5. The system asks for the ID of the book
6. The clerk inputs the ID of a book that the user wants to check out	
	7. The system records the book as having been issued to the member; it also records the member as having possession of the book. It generates a due-date. The system displays the book title and due-date and asks if there are any more books
8. The clerk stamps the due-date on the book and replies in the affirmative or negative	
	9. If there are more books, the system moves to Step 5; otherwise it exits
10. The customer collects the books and leaves the counter	

— There are some drawbacks to the way this usecase is written.
- Eg:- It doesn't specify how due-dates are computed.
* Solution is applying Business rules - These rules may be applicable to one or more use cases.

*Eg2 - It doesn't state what to do in case things go wrong, like, person may not be member at all or clerk may have entered an invalid book id.

- These drawbacks are addressed and the use case for issuing the books is revised as follows:

Table 6.5 Use case Book Checkout revised

Actions performed by the actor	Responses from the system
1. The member arrives at the check-out counter with a set of books and supplies the clerk with his/her identification number	
2. Clerk issues a request to check out books	
	3. The system asks for the user ID
4. Clerk inputs the user ID to the system	
	5. If the ID is valid, the system asks for the ID of the book; otherwise it prints an appropriate message and exits the use case
6. The clerk inputs the identifier of a book that the user wants to check out	
	7. If the ID is valid and the book is issuable to the member, the system records the book as having been issued to the member; It records the member as having possession of the book and generates a due-date as in Rule 1. It then displays the book's title and due-date. If the book is not issuable as per Rule 2, the system displays a suitable error message. The system asks if there are more books
8. The clerk stamps the due-date, prints out the transaction (if needed) and replies positively or negatively	
	9. If there are more books for checking out, the system goes back to Step 5; otherwise it exits
10. The clerk stamps the due date and gives the user the books checked out. The customer leaves the counter	

*Eg2- It doesn't state what to do in case things go wrong, like, person may not be member at all or clerk may have entered an invalid book id.

- These drawbacks are addressed and the use case for issuing the books is revised as follows:

Table 6.5 Use case Book Checkout revised

Actions performed by the actor	Responses from the system
1. The member arrives at the check-out counter with a set of books and supplies the clerk with his/her identification number	
2. Clerk issues a request to check out books	
	3. The system asks for the user ID
4. Clerk inputs the user ID to the system	
	5. If the ID is valid, the system asks for the ID of the book; otherwise it prints an appropriate message and exits the use case
6. The clerk inputs the identifier of a book that the user wants to check out	
	7. If the ID is valid and the book is issuable to the member, the system records the book as having been issued to the member; It records the member as having possession of the book and generates a due-date as in Rule 1. It then displays the book's title and due-date. If the book is not issuable as per Rule 2, the system displays a suitable error message. The system asks if there are more books
8. The clerk stamps the due-date, prints out the transaction (if needed) and replies positively or negatively	
	9. If there are more books for checking out, the system goes back to Step 5; otherwise it exits
10. The clerk stamps the due date and gives the user the books checked out. The customer leaves the counter	

* Use case for removing (deleting) books, [illegible]

Table 6.7 Use case Removing Books

Actions performed by the actor	Responses from the system
1. Librarian identifies the books to be deleted	
2. The clerk issues a request to delete books	
	3. The system asks for the identifier of the book
4. The clerk enters the ID for the book	
	5. The system checks if the book can be removed using Rule 3. If the book can be removed, the system marks the book as no longer in the library's catalog. The system informs the clerk about the success of the deletion operation. It then asks if the clerk wants to delete another book
6. The clerk answers in the affirmative or in the negative	
	7. If the answer is in the affirmative, the system goes to Step 3. Otherwise, it exits

* Use case for printing member transactions

Actions performed by the actor	Responses from the system
1. The clerk issues a request to get member transactions	
	2. The system asks for the user ID of the member and the date for which the transactions are needed
3. The clerk enters the identity of the user and the date	
	4. If the ID is valid, the system outputs information about all transactions completed by the user on the given date. For each transaction, it shows the type of transaction (book borrowed, book returned or hold placed) and the title of the book
5. Clerk prints out the transactions and hands them to the user	

* use case for removing (deleting) books, [illegible]

Table 6.7 Use case Removing Books

Actions performed by the actor	Responses from the system
1. Librarian identifies the books to be deleted	
2. The clerk issues a request to delete books	
	3. The system asks for the identifier of the book
4. The clerk enters the ID for the book	
	5. The system checks if the book can be removed using Rule 3. If the book can be removed, the system marks the book as no longer in the library's catalog. The system informs the clerk about the success of the deletion operation. It then asks if the clerk wants to delete another book
6. The clerk answers in the affirmative or in the negative	
	7. If the answer is in the affirmative, the system goes to Step 3. Otherwise, it exits

* use case for printing member transactions

Actions performed by the actor	Responses from the system
1. The clerk issues a request to get member transactions	
	2. The system asks for the user ID of the member and the date for which the transactions are needed
3. The clerk enters the identity of the user and the date	
	4. If the ID is valid, the system outputs information about all transactions completed by the user on the given date. For each transaction, it shows the type of transaction (book borrowed, book returned or hold placed) and the title of the book
5. Clerk prints out the transactions and hands them to the user	

* Use caser for Renew books

Table 6.12 Use case Renew Books

Actions performed by the actor	Responses from the system
1. Member makes a request to renew several of the books that he/she has currently checked out	
2. Clerk issues a request to renew books	
	3. System asks for the member's ID
4. The clerk enters the ID into the system	
	5. System checks the member's record to find out which books the member has checked out. If there are none, the system prints an appropriate message and exits; otherwise it moves to Step 6
	6. The system displays the title of the next book checked out to the member and asks whether the book should be renewed
7. The clerk replies yes or no	
	8. The system attempts to renew the book using Rule 4 and reports the result. If the system has displayed all checked-out books, it reports that and exits; otherwise the system goes to Step 6

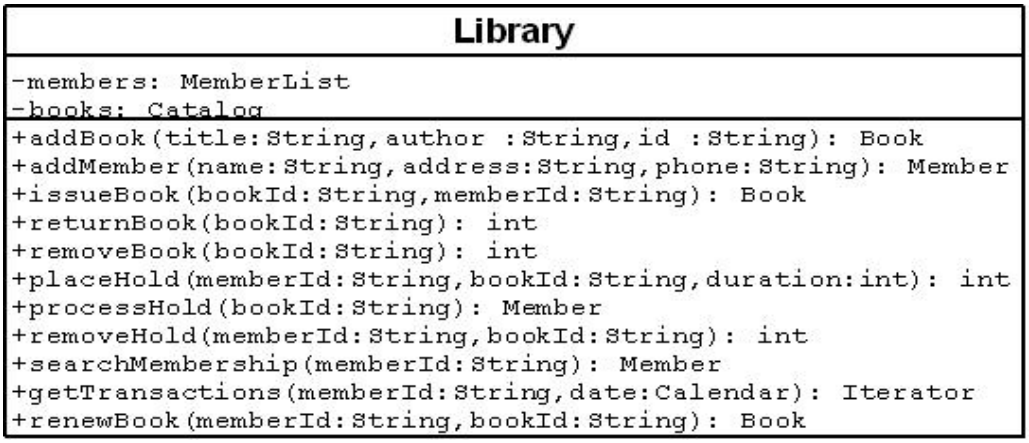
* Use case for processing holds

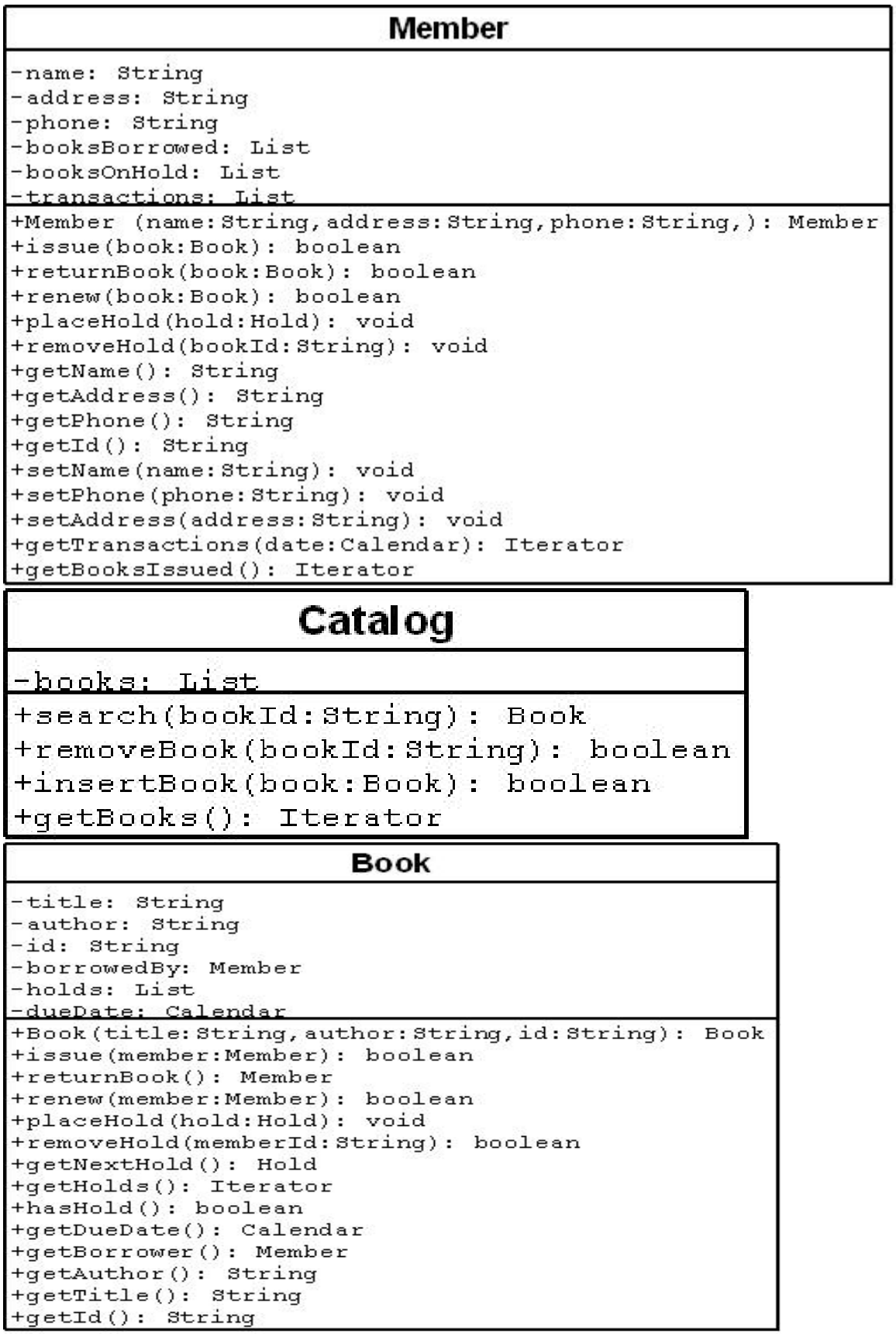
able 6.11 Use case Process Holds

Actions performed by the actor	Responses from the system
1. The clerk issues a request to process holds (so that Rule 5 can be satisfied)	
	2. The system asks for the book's ID
3. The clerk enters the ID of the book	
	4. The system returns the name and phone number of the first member with an unexpired hold on the book. If all holds have expired, the system responds that there is no hold. The system then asks if there are any more books to be processed
5. If there is no hold, the book is then shelved back to its designated location in the library. Otherwise, the clerk prints out the information, places it in the book and replies in the affirmative or negative	
	6. If the answer is yes, the system goes to Step 2; otherwise it exits

4) Draw class diagram for the following:

a) Library
b) Member of library
c) Book




Member
-name: String
-address: String
-phone: String
-booksBorrowed: List
-booksOnHold: List
-transactions: List
+Member (name:String,address:String,phone:String,): Member
+issue(book:Book): boolean
+returnBook(book:Book): boolean
+renew(book:Book): boolean
+placeHold(hold:Hold): void
+removeHold(bookId:String): void
+getName(): String
+getAddress(): String
+getPhone(): String
+getId(): String
+setName(name:String): void
+setPhone(phone:String): void
+setAddress(address:String): void
+getTransactions(date:Calendar): Iterator
+getBooksIssued(): Iterator
Catalog
-books: List
+search(bookId:String): Book
+removeBook(bookId:String): boolean
+insertBook(book:Book): boolean
+getBooks(): Iterator
Book
-title: String
-author: String
-id: String
-borrowedBy: Member
-holds: List
-dueDate: Calendar
+Book(title:String,author:String,id:String): Book
+issue(member:Member): boolean
+returnBook(): Member
+renew(member:Member): boolean
+placeHold(hold:Hold): void
+removeHold(memberId:String): boolean
+getNextHold(): Hold
+getHolds(): Iterator
+hasHold(): boolean
+getDueDate(): Calendar
+getBorrower(): Member
+getAuthor(): String
+getTitle(): String
+getId(): String

5) Compare functional and non-functional requirements

Functional requirements

a) Functional requirements specifies a function that a system or system component must be able to perform. It can be documented in various ways. The most common ones are written descriptions in documents, and use cases.
b) Use cases can be textual enumeration lists as well as diagrams, describing user actions. Each use case illustrates behavioural scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.
c) Functional requirements is what a system is **supposed to accomplish**. It may be
    a. Calculations
    b. Technical details
    c. Data manipulation
    d. Data processing
    e. Other specific functionality
d) A typical functional requirement will contain a unique name and number, a brief summary, and a rationale. This information is used to help the reader understand why the requirement is needed, and to track the requirement through the development of the system.

Non-functional requirements

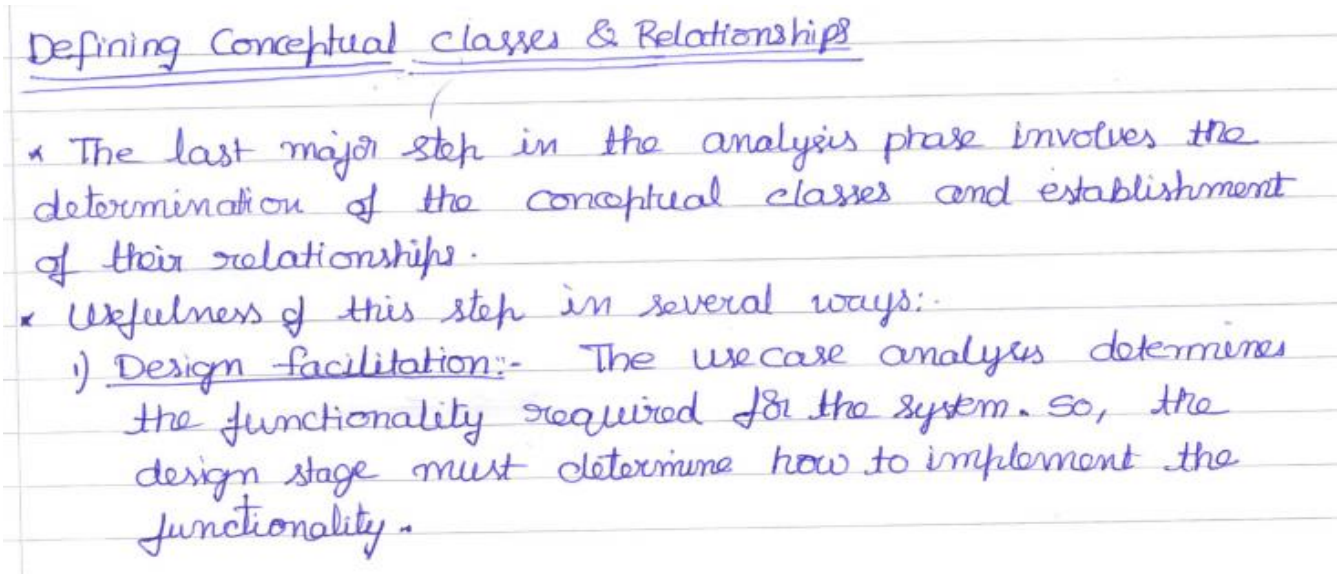
a) Non-functional requirements are the requirements that specifies criteria that can be used to **judge the operation of a system, rather than specific behaviours**.
b) Non-functional requirements are in the form of **"system shall be "**, an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed.
c) Non-functional requirements - can be divided into two main categories:
    a. **Execution qualities**, such as security and usability, which are observable at run time.
    b. **Evolution qualities**, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

d) Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet.

e) Some of them are:

i. Performance requirements
ii. Interface requirements
iii. Operational requirements
iv. Resource requirements
v. Verification requirements
vi. Acceptance requirements
vii. Documentation requirements
viii. Security requirements
ix. Portability requirements
x. Quality requirements
xi. Reliability requirements
xii. Maintainability requirements
xiii. Safety requirements

6) Describe conceptual classes and relationships

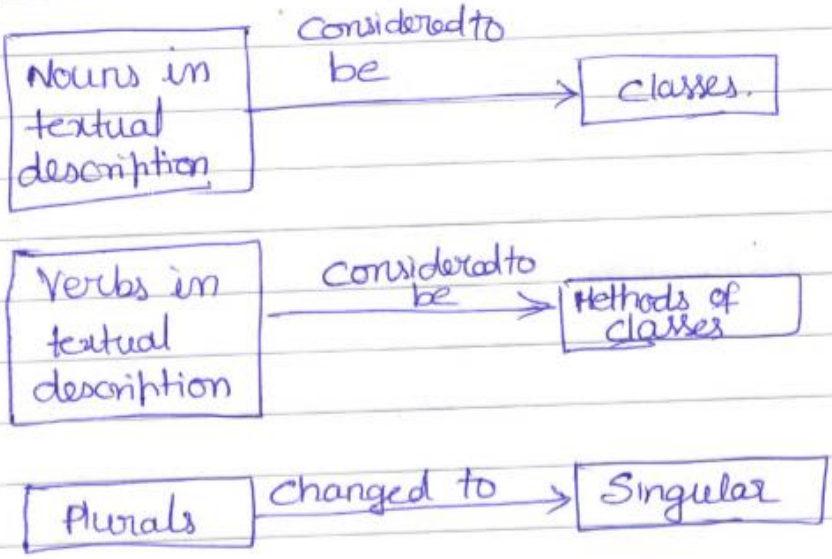


Defining Conceptual classes & Relationships

* The last major step in the analysis phase involves the determination of the conceptual classes and establishment of their relationships.
* Usefulness of this step in several ways:
  1) Design facilitation:- The use case analysis determines the functionality required for the system. So, the design stage must determine how to implement the functionality.

* Noun phrase Approach

- Conceptual classes, and methods can be identified by studying the use case, looking for relevant noun phrases.
  - i.e.,



- For this, the designers should be in a position to determine the classes that need to be defined, the objects to be created, and how the objects interact.
- This is better facilitated if the analysis phase classifies the entities in the application and determines their relationships

2) Added Knowledge - The use cases do not completely specify the system. Some of these missing details can be filled in by the class diagram.

3) End reduction :- The result can be shown to the client who can verify its correctness.

4) Useful documentation :- The classes & relationships provide a quick introduction to the system for someone who wants to learn it.

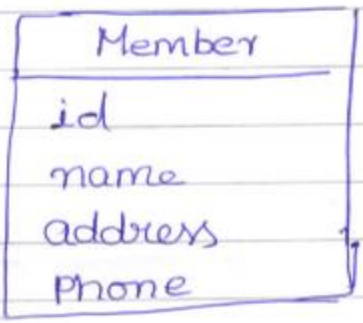
- For library instance, ~~following~~ nouns after eliminating duplicates are as follows:
  (i) Customer
  (ii) user
  (iii) Application form & request
  (iv) customer's name, Address, phone number
  (v) clerk
  (vi) Identification number
  (vii) data
  (viii) Information
  (ix) System.

  The noun 'System' implies a conceptual class that represents all of the software. We call this class 'Library'

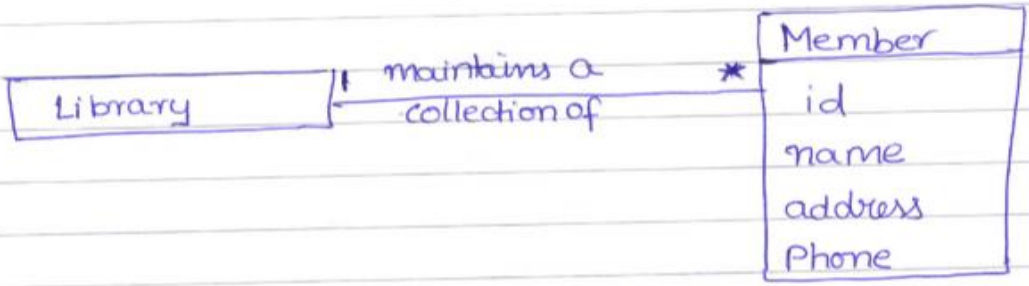
* UML convention:-
  The UML convention is to write the class name at the top with a line below it & the attributes listed just below the line.

- UML diagram for the class 'Library'

- UML diagram for the class 'Member'



- Association between classes
  - Determines relationship between two & more classes.
  - Eg:- The use case Register New member says that the system 'remembers information about the member'. This implies an association b/w the conceptual classes 'Library' and 'Member'. This is shown as:



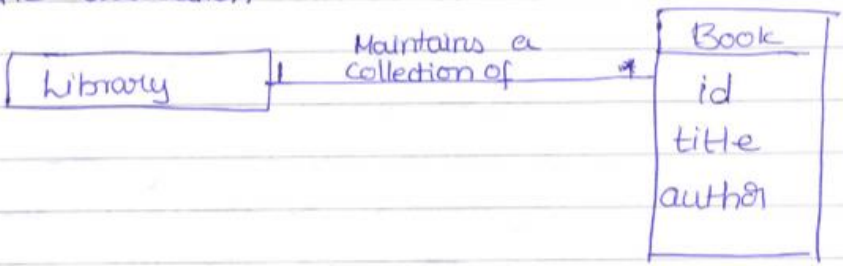
i.e, one instance of Library maintains a collection of zero or more members.

- Static & dynamic Associations:

Static associations are permanent, whereas, dynamic associations are those that change as a result of the transactions being recorded by the system. Such associations are typically associated with Verbs.

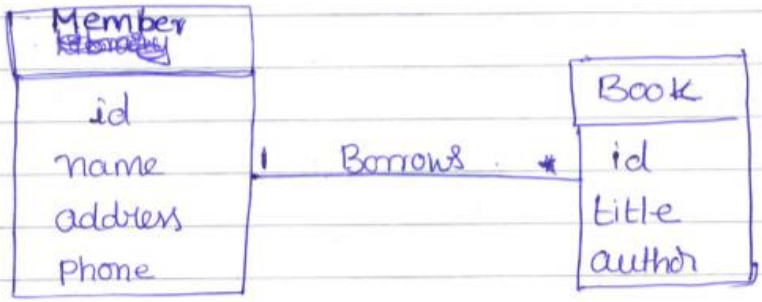
Eg:- UML diagram for the class Book

static association -



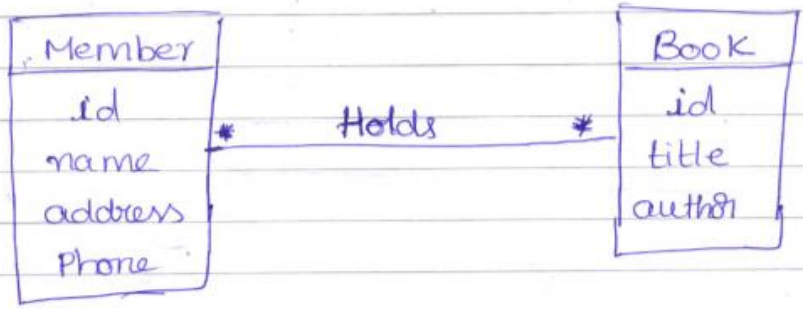
dynamic association.

At any instance, a book can be borrowed by one Member and a member may have borrowed any no. of books.



- Many-to-Many relationships: A member can place a hold on an arbitrary number of books.

Several users can have holds placed on a book, and a user may place holds on an arbitrary no. of books.

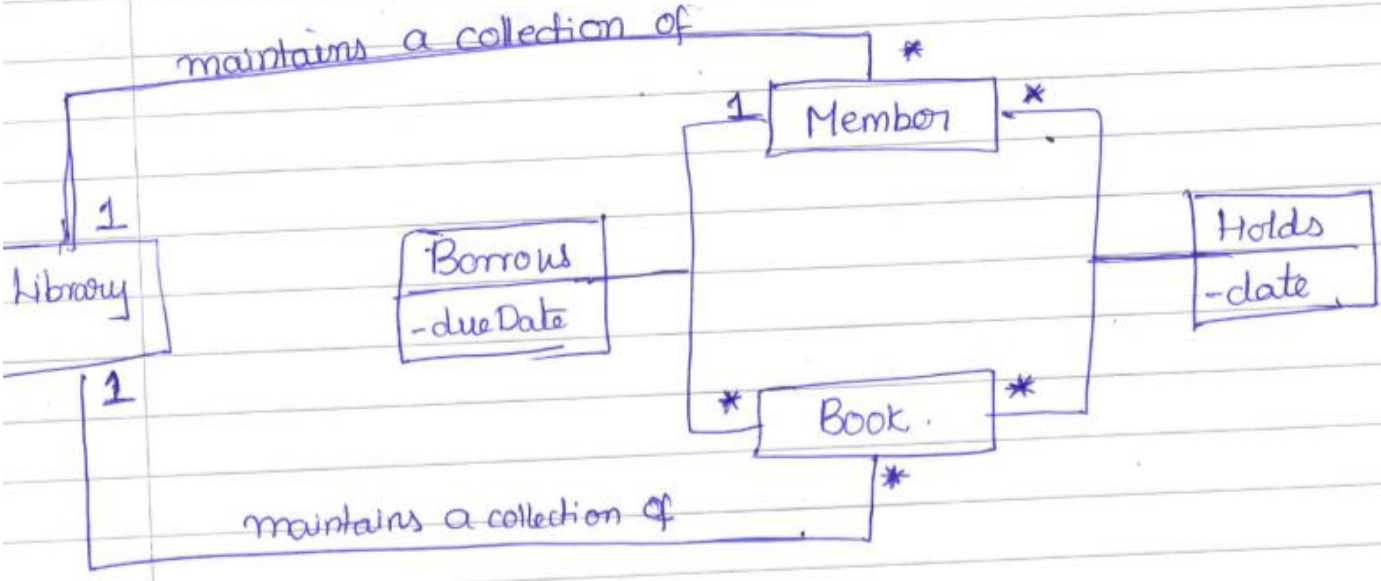


- Inter-class relationships: A relationship formed between two entities is sometimes accompanied by additional information which is relevant only in the context of the relationship.

  eg:- When a user borrows a book and when a user places a hold on a book. Borrowing introduces new information into the system ie, the date on which the book is due

to be returned. Likewise, placing a hold introduces some information, i.e., the date after which the book is not needed.

- The lines representing the association are augmented to represent the information that must be stored as part of the association.
- ~~For the Association 'Borrows' and the lines connecting members~~
  - For this, we come up with a Conceptual class named 'Borrows' having an attribute named 'dueDate'.
    Similarly, we create a conceptual class named 'Holds' with the attribute named 'date'
- The conceptual classes (or their representation) do not, in any way, tell us how the information is going to be stored (or accessed).
- The following diagram shows all of the Conceptual classes and their associations.



7) Give the Use-Case diagrams for the following

a) Drawing a Line
b) Draw Label
c) To change font

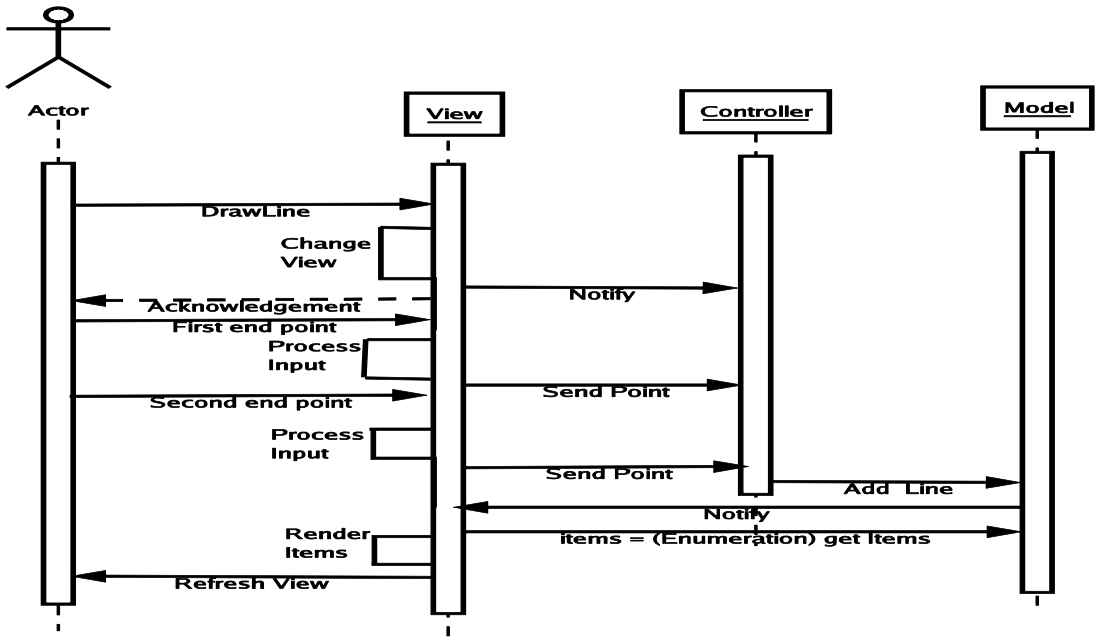
Actions performed by the actor	Responses from the system
1. The user clicks on the drawLine button in the command panel.	
	2. The system changes the cursor to a cross-hair
3. The user clicks first on one end point and then on the other end point of the line to be drawn.	
	4. The system adds a line segment with the two specified end points to the figure being created. The cursor changes to the default.

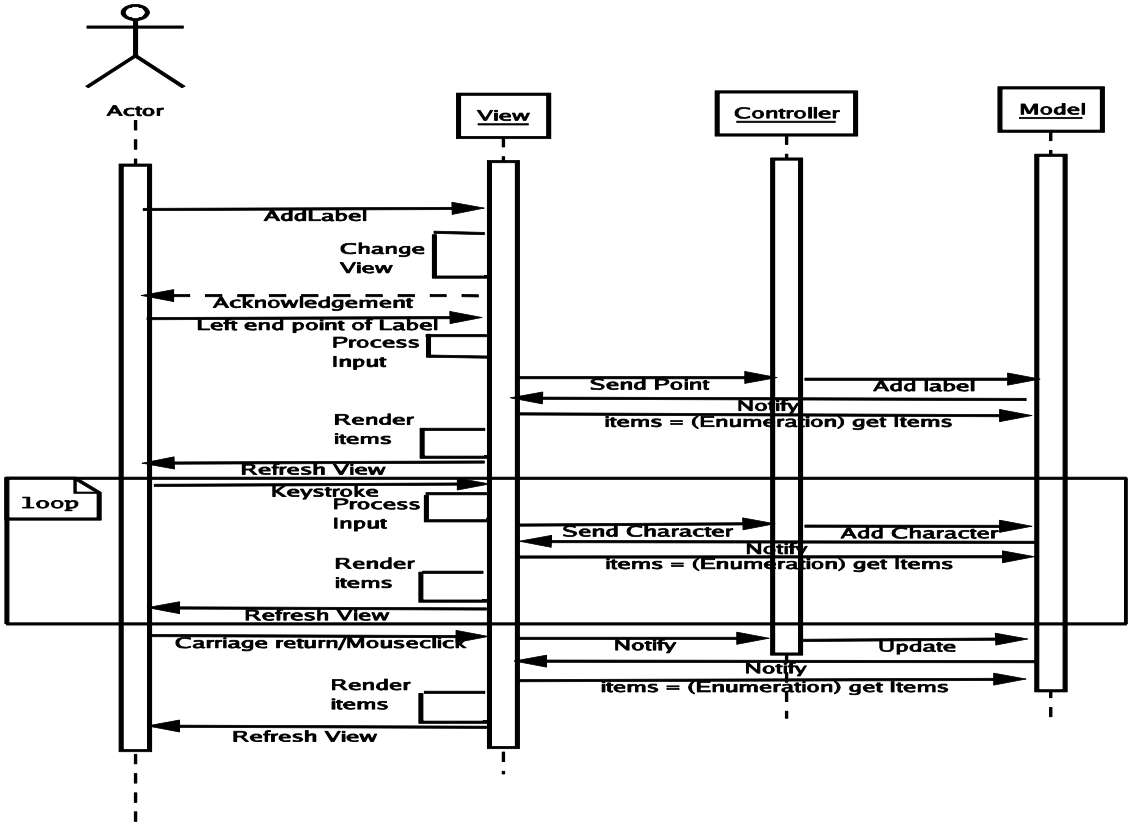
Actions performed by the actor	Responses from the system
1. The user clicks on the drawLine button in the command panel.	
	2. The system changes the cursor to a cross-hair
3. The user clicks first on one end point and then on the other end point of the line to be drawn.	
	4. The system adds a line segment with the two specified end points to the figure being created. The cursor changes to the default.

Actions performed by the actor	Responses from the system
1. The user clicks on the addLabel button in the command panel.	
	2. The system changes the cursor to a text cursor
3. The user clicks at the left end point of the intended label.	
	4. The system places a block cursor at the clicked location.
5. The user types a character or clicks the mouse at another location.	
	6. If the character is not a carriage return the system displays the typed character, moves the cursor forward and goes to Step 5; in case of a mouse-click, it goes to Step 4; otherwise it goes to the default state.

8) Give the Sequence diagrams for the following

a) Draw a Line
b) Add Label


Actor
View
Controller
Model
DrawLine
Change View
Notify
Acknowledgement
First end point
Process Input
Send Point
Second end point
Process Input
Send Point
Add Line
Notify
items = (Enumeration) get Items
Render Items
Refresh View


Actor
View
Controller
Model
AddLabel
Change View
Acknowledgement
Left end point of Label
Process Input
Send Point
Add label
Notify
items = (Enumeration) get Items
Render items
Refresh View
loop
Keystroke
Process Input
Send Character
Add Character
Notify
items = (Enumeration) get Items
Render items
Refresh Vlew
Carriage return/Mouseclick
Notify
Update
Notify
items = (Enumeration) get Items
Render items
Refresh View