



Object Oriented Modeling and Design ∞ Laboratory

DEPARTMENT OF COMPUTER SCIENCE

Session: Aug - Dec 2015

Case Study for all Lab Exercises



City College of Engineering (CCE) would like to computerize their library operations with the help of a Library Information and Management System (LIMS). Currently library operations are manual, and CCE would like to use a state-of-the-art LIMS.

StarUML Tool



☞ StarUML is a software modeling platform that supports UML (Unified Modeling Language). It is based on UML version 1.4 and provides eleven different types of diagram, and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept.

Unified Modeling Language



Introduction to UML

- **UML** – Unified Modeling Language diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction.
- One reason UML has become a standard modeling language is that it is programming-language independent.
- Since UML is not a methodology, it does not require any formal work products.
- In an effort to promote Object Oriented designs, three leading object oriented programming researchers joined ranks to combine their languages:
 - i. Grady Booch (BOOCH)
 - ii. James Rumbaugh (OML: object modeling technique)
 - iii. Ivar Jacobsen (OOSE: object oriented software eng) and come up with an industry standard [mid 1990's].

Types of UML Diagram



Structural Diagrams

1. Class Diagram
2. Component Diagram
3. Deployment Diagram

Behavioural Diagrams

1. Use case Diagram
2. Activity Diagram
3. Sequence Diagram
4. Collaboration Diagram
5. State Diagram

Types of UML Diagram



Structural Diagrams

- Structure diagrams emphasize on the things that must be present in the system being modeled.
- Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.

Behavioural Diagrams

- Behavior diagrams emphasize on what must happen in the system being modeled.
- Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

LIST OF EXPERIMENTS



- ❧ 1: Prepare Use Case Diagrams for the Case Study given.
- ❧ 2: Prepare Class Diagrams for the Case Study given.
- ❧ 3: Prepare Activity Diagrams for the Case Study given.
- ❧ 4: Prepare Sequence Diagrams for the Case Study given.
- ❧ 5: Prepare Collaboration Diagrams for the Case Study given.
- ❧ 6: Prepare State Diagrams for the Case Study given.
- ❧ 7: Prepare Component Diagrams for the Case Study given.
- ❧ 8: Prepare Deployment Diagrams for the Case Study given.

Experiment 1 – Use Case Diagram



❧ **Problem:** Identify the Actors and the corresponding Use Cases and come up with Use Case Diagram and also show the inheritance for the same

❧ **Solution:**

1. Identify the Nouns as the Actors and Verbs as the Use Cases.
2. Identify associations like
 <<include>>, <<extend>> and generalizations.
3. Identify the actor hierarchy, implying what use cases an actor can invoke additionally.
4. Draw the corresponding use case/hierarchy.

Solution – Use Case Diagram



Introduction:

- A use case diagram describes how a system interacts with outside actors.
- It is a graphical representation of the interaction among the elements and system.
- Each use case representation a piece of functionality that a system provides to its user.
- Use case identifies the functionality of a system.
- Use case diagram allows for the specification of higher level user goals that the system must carry out.
- These goals are not necessarily to tasks or actions, but can be more general required functionality of the system.
- You can apply use case to capture the intended behavior of the system you are developing, without having to specify how that behavior is implemented.
- A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case.
- A use case diagram contains four components.
 - i. The boundary, which defines the system of interest in relation to the world around it.
 - ii. The actors, usually individuals involved with the system defined according to their roles.
 - iii. The use cases, which the specific roles are played by the actors within and around the system.
 - iv. The relationships between and among the actors and the use cases.

Solution – Use Case Diagram



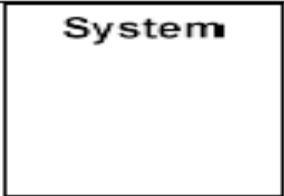
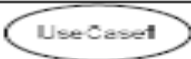




Purpose:



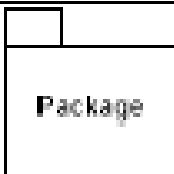



- The main purpose of the use case diagram is to capture the dynamic aspect of a system.
- Use case diagram shows, what software is suppose to do from user point of view.
- It describes the behavior of system from user's point.
- It provides functional description of system and its major processes.
- Use case diagram defines the scope of the system you are building.

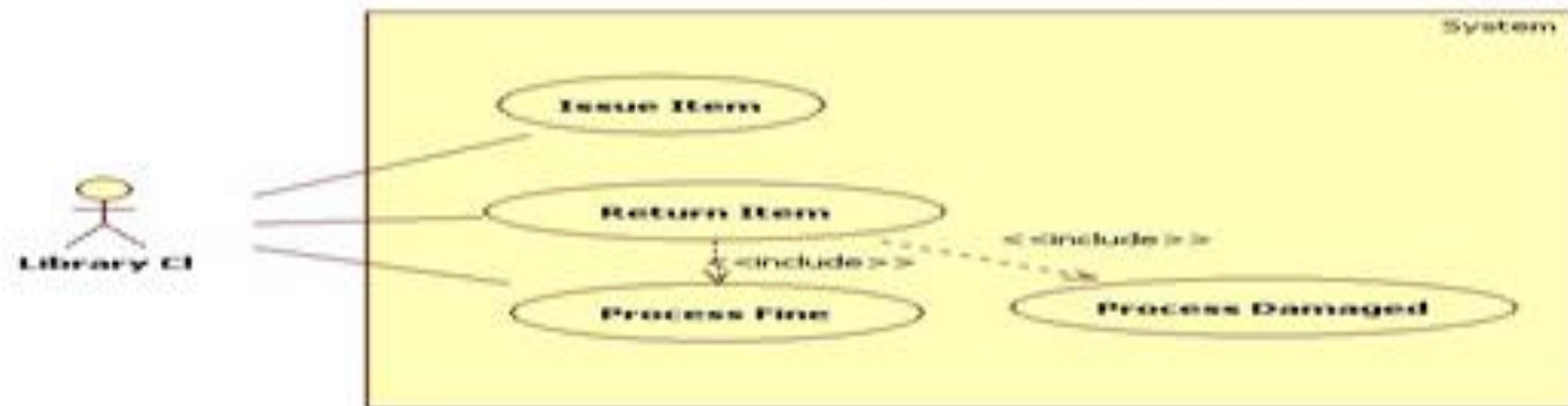
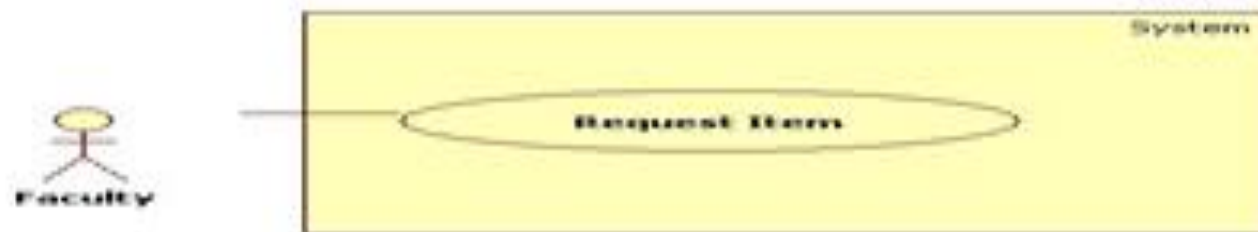
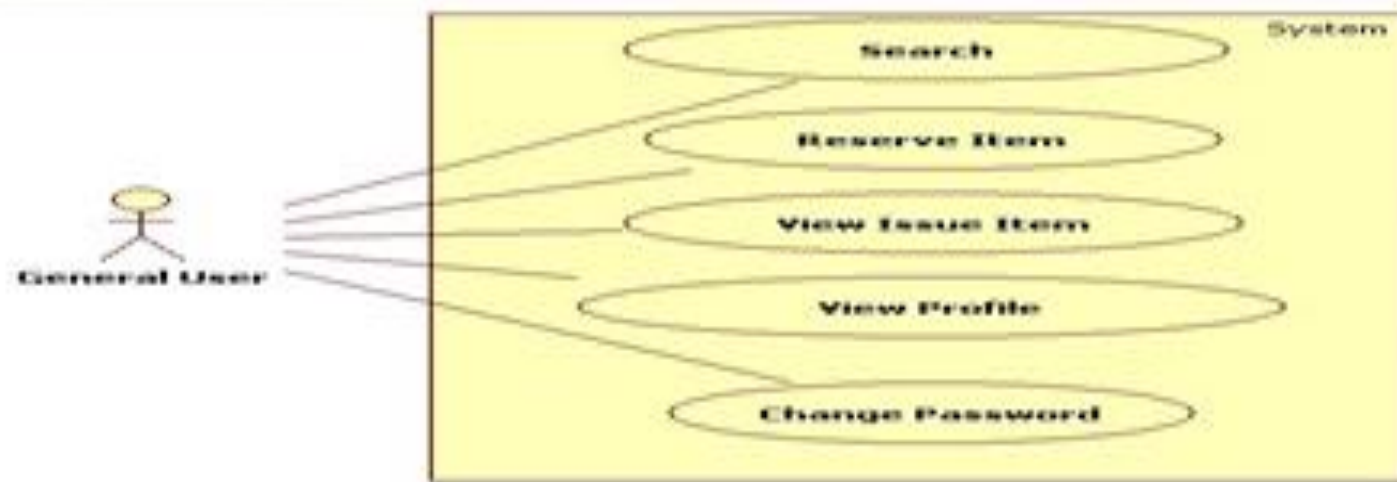
When to Use: Use Cases Diagrams

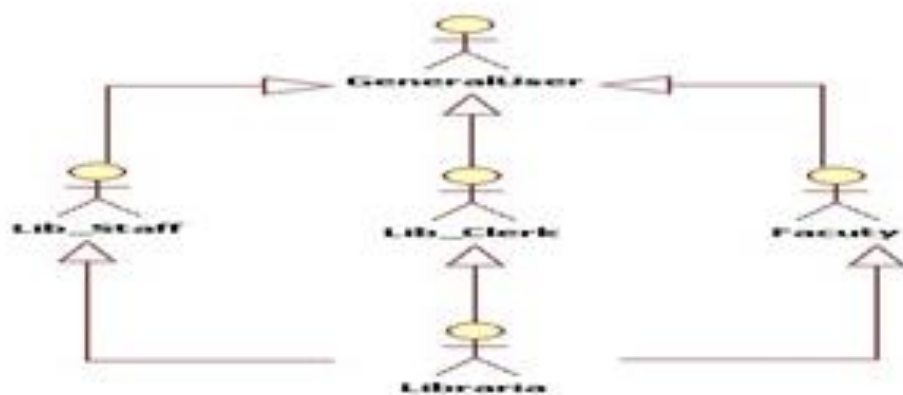
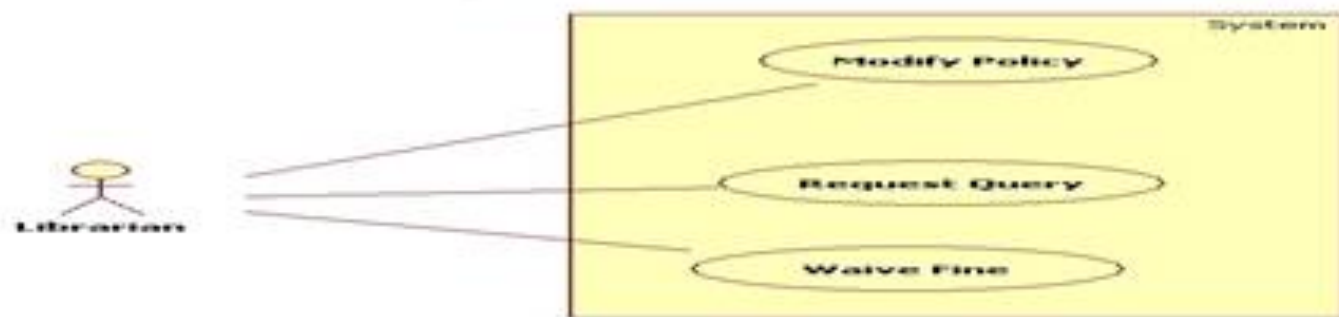
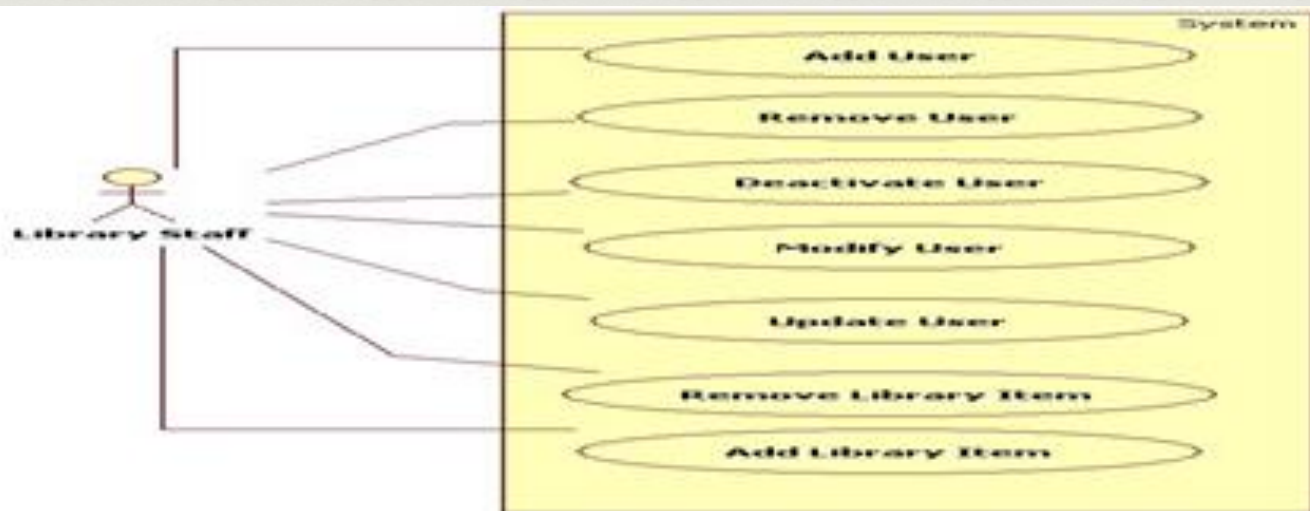
- Use cases are used in almost every project.
- They are helpful in exposing requirements and planning the project.
- During the initial stage of a project most use cases should be defined.

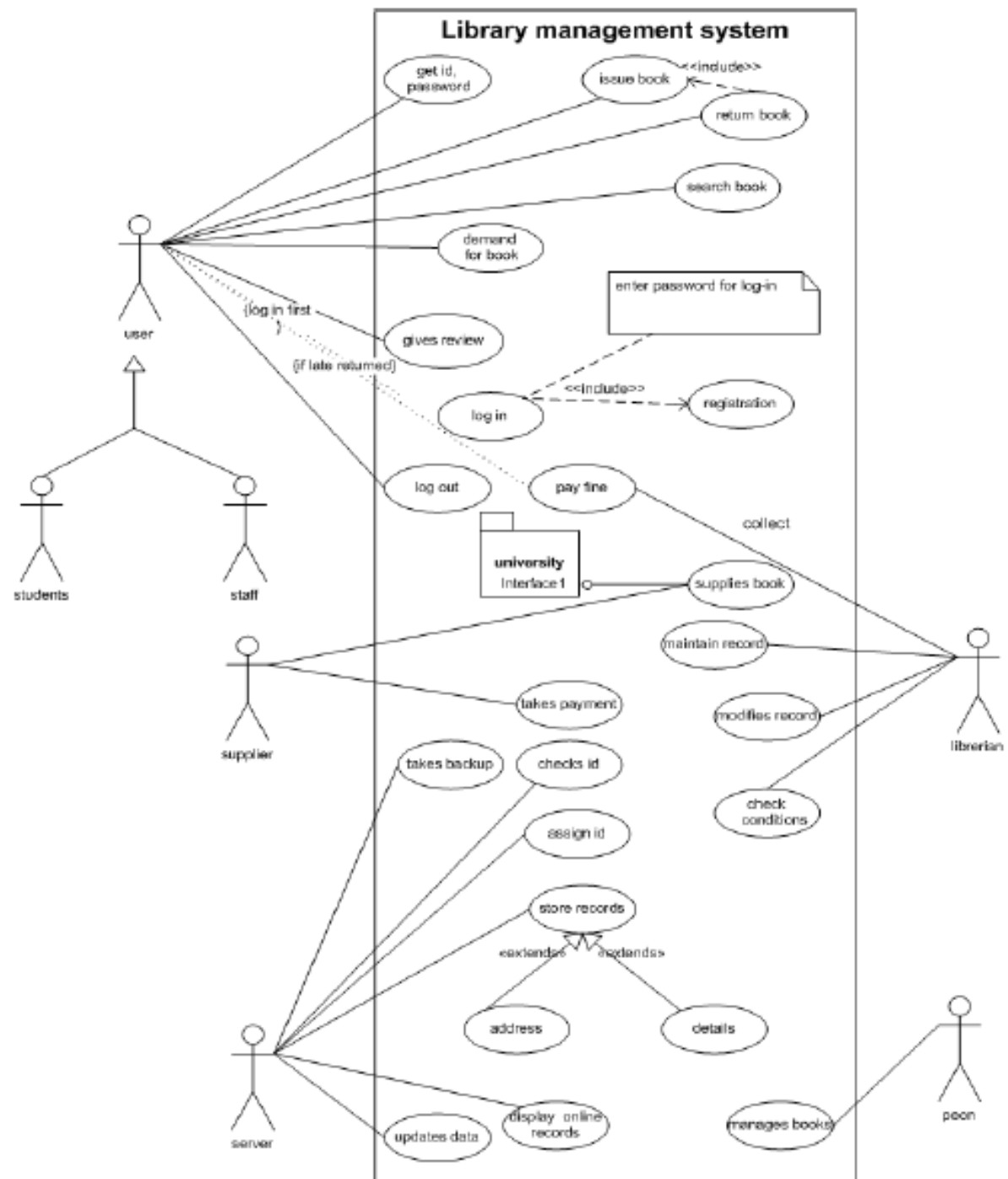
Use Case Notations

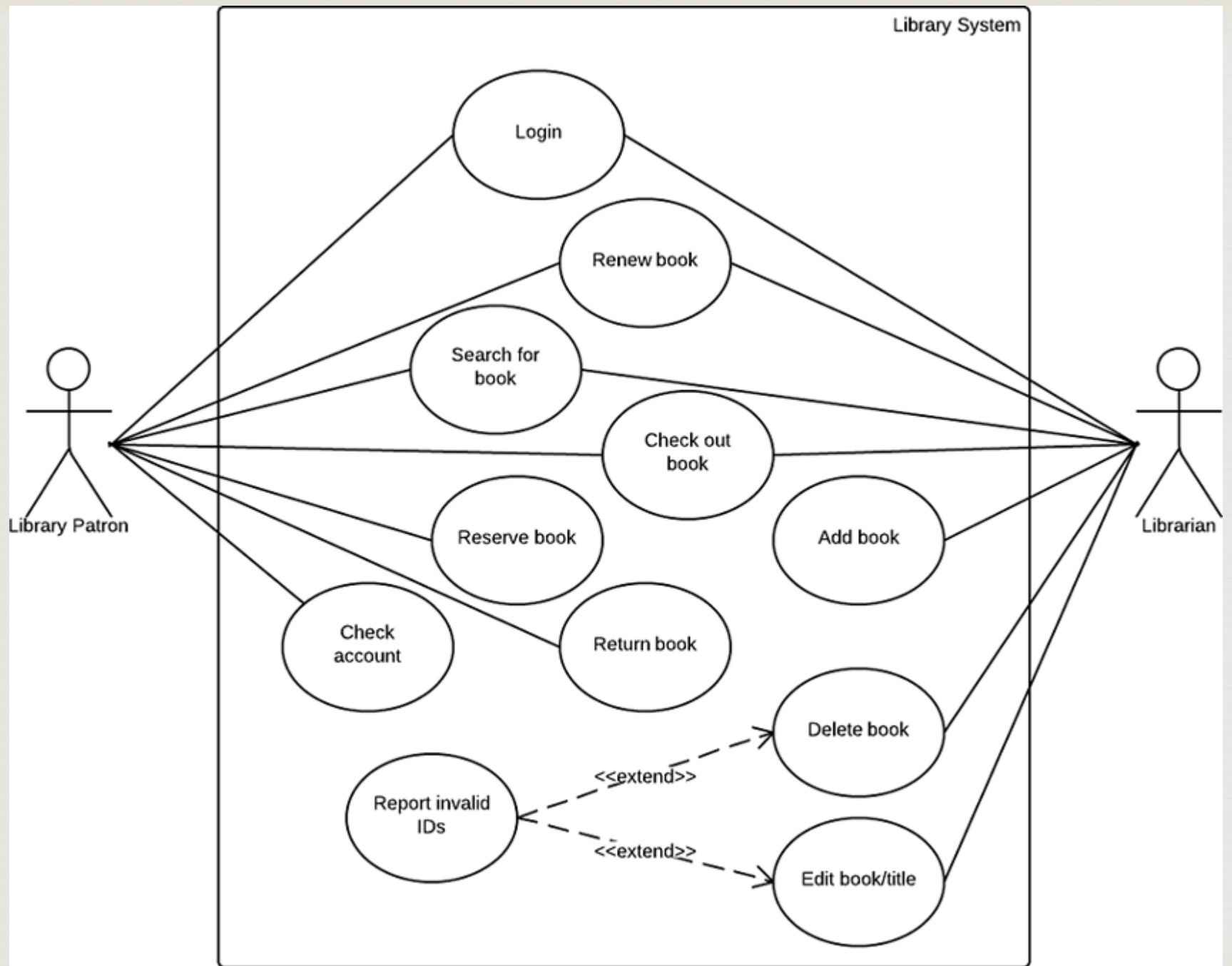
No.	Name	Notation	Description
1	System boundary		The scope of a system can be represented by a system boundary. The use cases of the system are placed inside the system boundary, while the actors who interact with the system are put outside the system. The use cases in the system make up the total requirements of the system.
2	Use case		A use case represents a user goal that can be achieved by accessing the system or software application.
3	Actor		Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So an actor may be people, computer hardware, other systems, etc. Note that actor represent a role that a user can play, but not a specific user.
4	Association		Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association corresponds to a sequence of actions between the actor and use case in achieving the use case.
5	Generalization		A generalization relationship is used to represent inheritance relationship between model elements of same type.
6	Include		An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

7	Extends		An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.
8	Constraint		Show condition exists between actors an activity.
9	Package		Package is defined as collection of classes. Classes are unified together using a package.
10	Interface		Interface is used to connect package and use-case. Head is linked with package and tail linked with use-case.
11	Note		Note is generally used to write comment in use-case diagram.
12	Anchor		Anchor is used to connect a note the use case in use case diagram









Experiment 2 - Class Diagram



∞ Problem:

Class Diagram - Identify the Classes, Attributes and Operations. Prepare the data dictionary for LIMS classes.

∞ Solution:

Identify the nouns and choose the ones that can become classes.

Identify the attributes of the classes based on description provided in Case Study.

Identify the operations each class can perform.

Solution – Class Diagram



Introduction

- The class diagram is a static diagram.
- A class model captures the static structure of a system by characterizing the objects in the system, the relationship between the objects, and the attributes and operations for each class of objects.
- The class diagram can be mapped directly with object oriented languages.
- The class model is the most important among the three models.
- Class diagram provide a graphical notation for modeling classes and their relationship.
- They are concise, easy to understand, and work well in practice.
- Class diagrams are the backbone of almost every object-oriented method including UML.
- They describe the static structure of a system.

Solution – Class Diagram



Purpose

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.

When to use : Class Diagram

- Useful for Forward and Reverse engineering.
- Class diagrams are useful both for abstract modeling and for designing actual programs.
- Developer uses class diagram for implementation decision.
- Business analysts can use class diagrams to model systems from the business perspective.

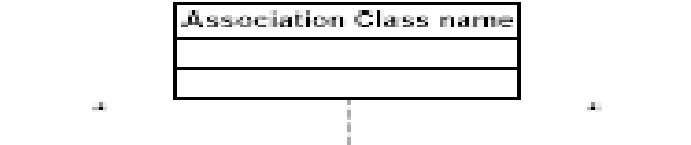
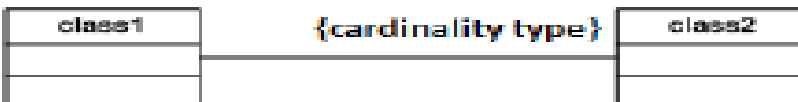
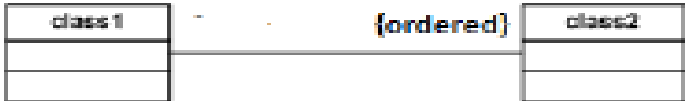

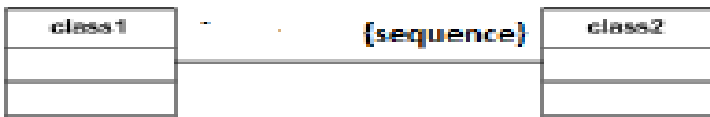
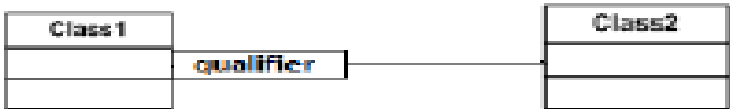
Solution – Class Diagram

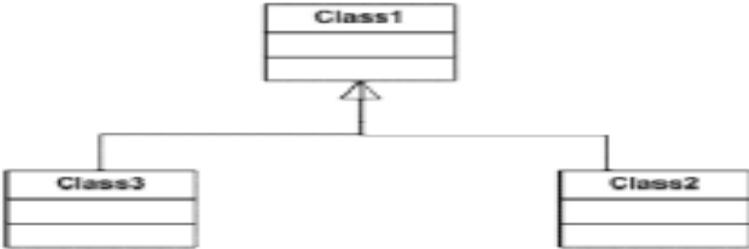
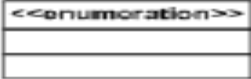



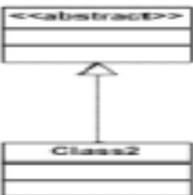



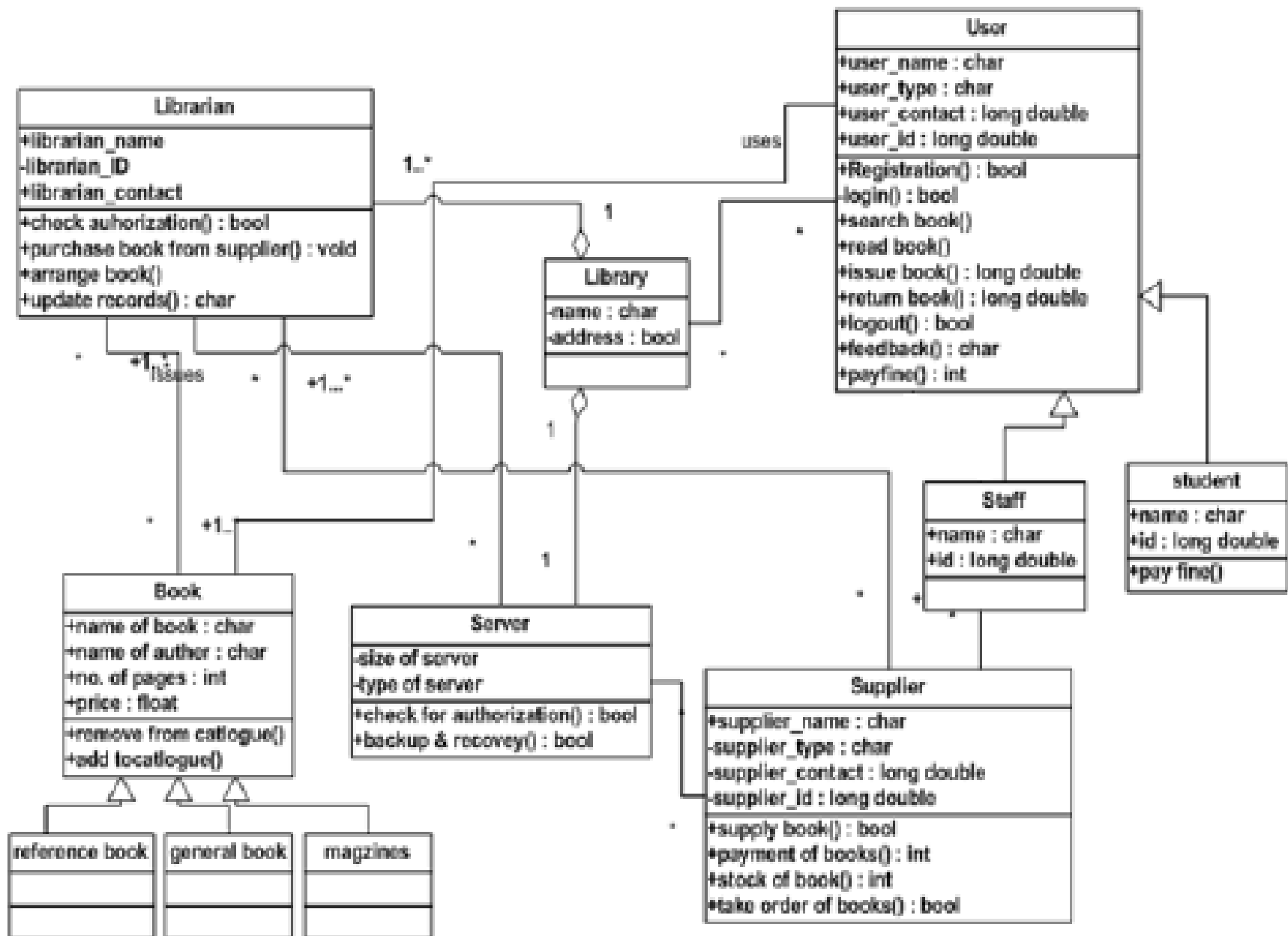
1. CRC (Class – Responsibilities – Collaborators) for significant classes.
2. Data Dictionary for all classes of Class Diagram.

Class Diagram Notations

Sr. No.	Name	Symbol	Meaning
1.	Class	<div><div>class name</div><div></div><div></div></div>	Class is an entity of the class diagram. It describes a group of objects with same properties & behavior.
2.	Object	<div>Object name : Class</div>	An object is an instance or occurrence of a class.
3.	Link	<div><div>Object1</div><div></div></div> <div></div> <div><div>Object2</div><div></div></div>	A link is a physical or conceptual connection among objects
4.	Association	<div><div>Class3</div><div></div><div></div></div> <div></div> <div><div>Class4</div><div></div><div></div></div>	An association is a description of a links with common structure & common semantics.
5.	Multiplicity	<div>Ex.<div><div>1to1</div><div>1to*</div><div>*to*</div><div>*to1</div><div>1to0...2</div></div><div><div>class1</div><div></div><div></div></div><div>*</div><div></div><div>*</div><div><div>class2</div><div></div><div></div></div></div> <td>Multiplicity specifies the number of instances of one class that may relate to a single instance of an associated class. It is a constraint on the cardinality of a set.</td>	Multiplicity specifies the number of instances of one class that may relate to a single instance of an associated class. It is a constraint on the cardinality of a set.

6.	Association class		It is an association that is a class which describes the association with attributes.
7.	cardinality		It describes the count of elements from collection.
8.	ordering		It is used to indicate an ordered set of objects with no duplication allowed.
9.	bag		A bag is a collection of unordered elements with duplicates allowed.
10.	sequence		A sequence is an ordered collection of elements with duplicates allowed.
11.	qualified association		Qualification increases the precision of a model. It is used to avoid many to many multiplicities and it converts into one to one multiplicity.

12.	generalization	 <pre> classDiagram Class3 -- > Class1 Class2 -- > Class1 </pre>	Generalization organizes classes by their super-class and sub-class relationship.
13.	enumeration	 <pre> classDiagram class Enum["<<enumeration>>"] </pre>	An enumeration is a data type that has a finite set of values.
14.	aggregation	 <pre> classDiagram Class1 o-- Class3 Class1 o-- Class2 </pre>	It is a strong form of association in which an aggregate object is made of constituent parts.
15.	composition	 <pre> classDiagram Class1 *-- Class2 </pre>	It is a form of aggregation. Composition implies ownership of the parts by the whole.
16.	Abstract class	 <pre> classDiagram class Abstract["<<abstract>>"] </pre>	It is a class that has no direct instances.
17.	Concrete class	 <pre> classDiagram class Abstract["<<abstract>>"] Class2 -- > Abstract </pre>	It is a class that is intangible; it can have direct instances. Class-2 is example of concrete class
18.	package	 <pre> classDiagram package Package["Package name"] </pre>	A package is a group of elements with common theme.



Data Dictionary of Class Diagram

Book : Represents a libray item which contains unique ISBN, Title, Author as its attributes.

Book Copy : Represents an instance of the item book

Checkout Duration : Represents the duration till which the user can hold a particular item. Attributes include Type of the User, type of the item and the duration.

Issue : A Library clerk can issue items to the user which contains an issue date, due date , return date, and fine.

Item : Item can be any library item which includes book, CD, DVD, Periodical, Bound Periodical etc..Each of these items must contain the following attributes, i.e the title, accession number, year of publication, publisher, classification number, number of copies, staus of each copy.

Item Catelog : Includes the functionalities that a user can perform. Functionalities include searching a library item, reserve an item, get a library item etc..

Library : Main class which is responsible for overall functionalities of LIMS.

Media : Represents a libray item which contains type of media, title & duration.

Media Copy : Represents an instance of an item media.

Data Dictionary of Class Diagram

Periodical : Represents a libray item which contains title, ISBN, Volume number, issue number.

Policy : Librarian can create various policies for the user and items. Policies include fine policy, duration policy, issue policy.

Reservation: Users can reserve library items.Reservation is done based on the proprity of the user. Only 3 reservations can be made.

Request : User (Libray Staff) can request an item.

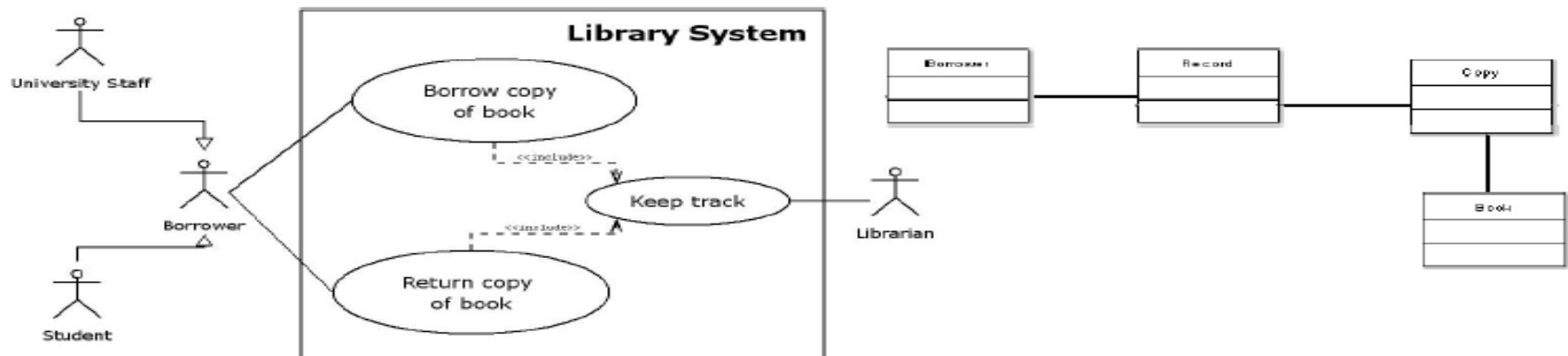
User: Are people who accesses the LIMS. Users are a General User or an Administrative user. General Users include students, staff. Administrative user are Libray staff, Library Clerk or an Librarian. Each user will have a unique user id, name, address, phone number, e-mail id, fine details.

User Catelog: User Catelog contains various functionalities that a Library can perform with respect to user on a LIMS. Functionalities include delete the user, add new user, deactivate the user account, re-activate account, invalid user.

CRC (Class – Responsibilities – Collaborators) for significant classes



A Design Example



- **A Library System**
- The library system must keep track of when books are borrowed and returned
- The system must support librarian work
- The library is open to **university** staff and students

Using CRC Cards

1. Choose a coherent set of **use cases**
2. Put a **card** on the table
3. Walk through the scenario, naming **cards** and **responsibilities**
4. Vary the **situations** (i.e., assumptions on the use case), to stress test the cards
5. Add cards, push cards to the side, to let the design evolve (that is, evaluate different **design alternatives**)
6. Write down the key **responsibility decisions** and **interactions**

A Design Example

Borrower	
Borrow book Return book	Record Record

Record	
Keep track	Copy

Copy	
Record: Borrowed or Returned Book copies	Librarian Book

Book	
Book information Number of available copies	

Experiment 3 : Activity Diagram



❧ Problem:

Identify the Activity Flow and prepare the Activity Diagram of Library Information Management System (LISM).

Solution – Activity Diagram



Introduction

- An activity diagram is a type of flow chart with additional support for parallel behavior.
- This diagram explains overall flow of control.
- Activity diagram is another important diagram in UML to describe dynamic aspects of the system.
- Activity diagram is basically a flow chart to represent the flow from one activity to another activity
- The activity can be described as an operation of the system.
- The control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. This distinction is important for a distributed system.
- Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

Purpose

- Contrary to use case diagrams, in activity diagrams it is obvious whether actors can perform business use cases together or independently from one another.
- Activity diagrams allow you to think functionally.

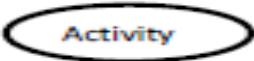

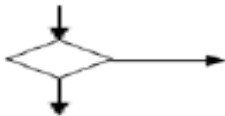



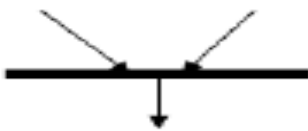
Solution – Activity Diagram



When to use : Activity Diagrams

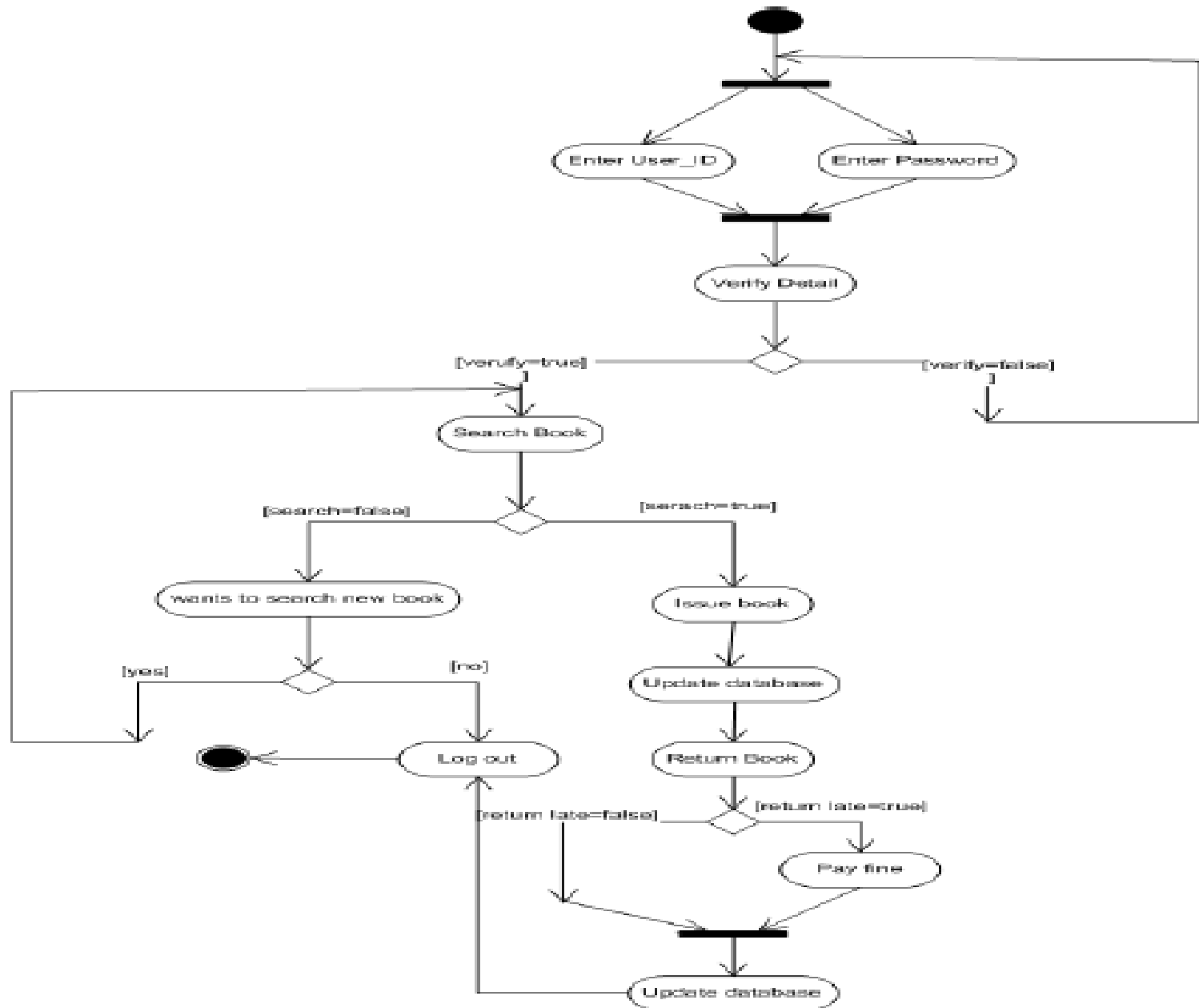
- Activity diagrams are most useful when modeling the parallel behavior of a multithreaded system or when documenting the logic of a business process.
- Because it is possible to explicitly describe parallel events, the activity diagram is well suited for the illustration of business processes, since business processes rarely occur in a linear manner and often exhibit parallelisms.
- This diagram is useful to investigate business requirements at a later stage.
- An activity diagram is drawn from a very high level. So it gives high level view of a system. This high level view is mainly for business users or any other person who is not a technical person.
- This diagram is used to model the activities which are nothing but business requirements.
- So the diagram has more impact on business understanding rather *implementation details*.

Activity Diagram Notations

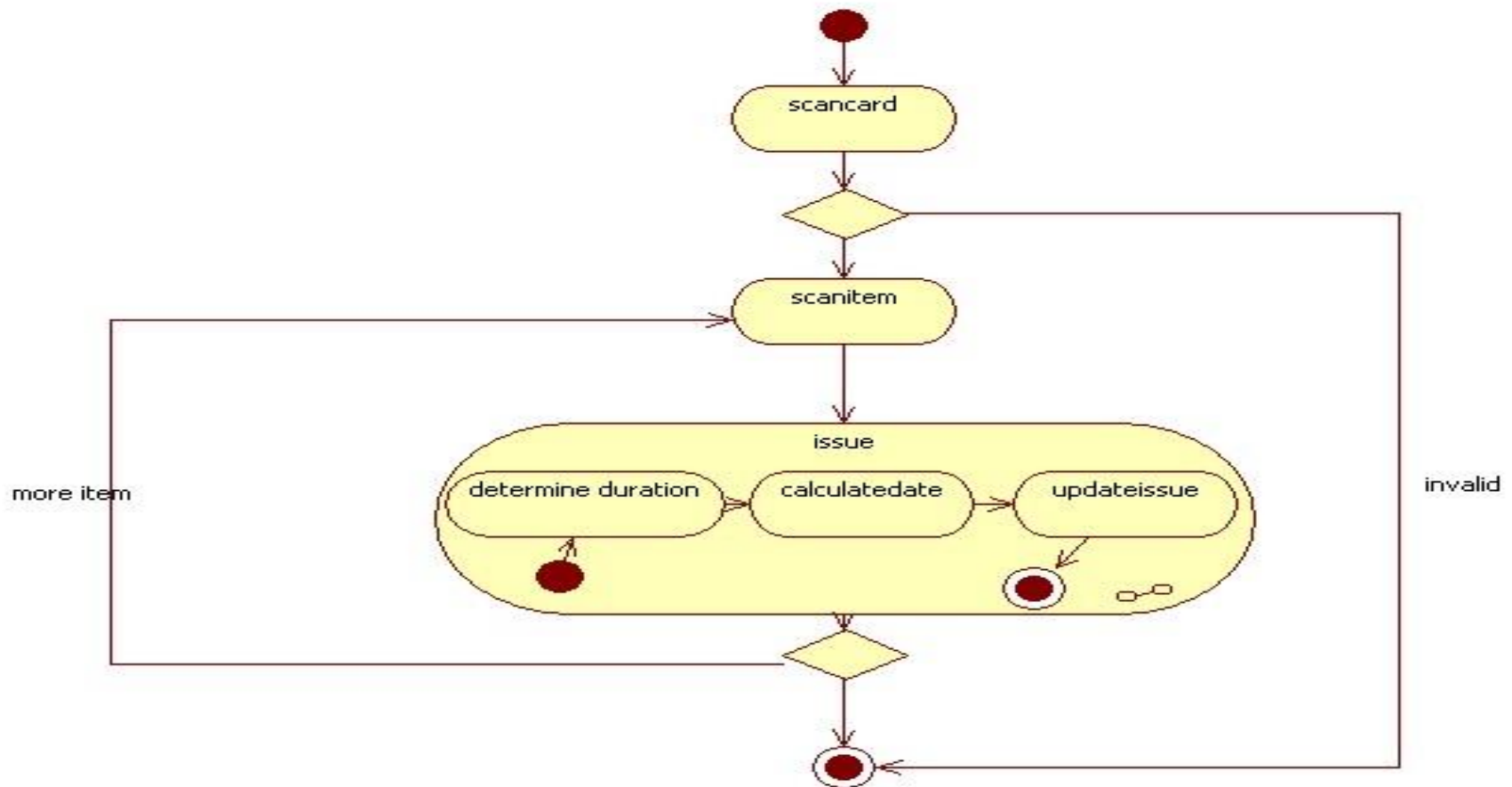
No.	Name	Symbol	Description
1.	Activity		Represent individual activity of system.
2.	Transition		Represents flow of data from one activity to another.
3.	Decision		Decision node is a control node that accepts tokens on one or more incoming edges and selects outgoing edge from two or more outgoing flows. The notation for a decision node is a diamond-shaped symbol.
4.	Initial activity		Initial node is a control node at which flow starts when the activity is invoked. Activity may have more than one initial node. Initial nodes are shown as a small solid circle.
5.	Final activity		Final node is a control final node that stops all flows in an activity. Activity final nodes are shown as a solid circle with a hollow circle inside. It can be thought of as a goal notated as "bull's eye," or target.
6.	Fork		A fork in the activity diagram has a single incoming transition and multiple outgoing transitions exhibiting parallel behavior. The incoming transition triggers the parallel outgoing transitions.
7.	Join		A join in the activity diagram synchronizes the parallel behavior started at a fork. Join ascertains that all the parallel sets of activities (irrespective of the order) are completed before the next activity starts. It is a synchronization point in the diagram. Each fork in an activity diagram has a corresponding join where the parallel behavior terminates.

Activity Diagram for Library Management System

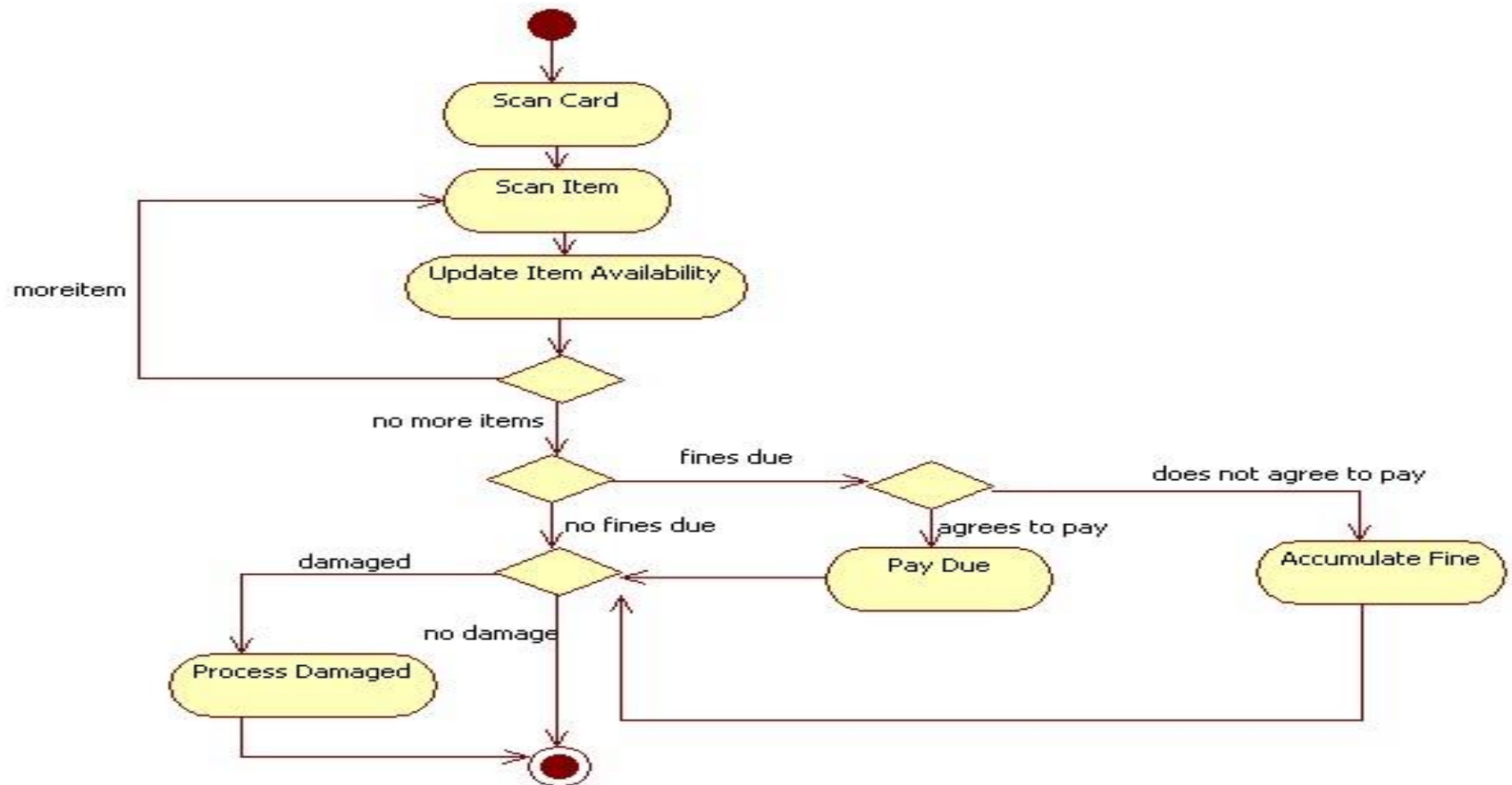
Issue and return book



Solution - Activity Diagram : Issue of Book



Solution - Activity Diagram : Return of Book



Experiment 4 - Sequence Diagram



Problem:

Prepare the Sequence Diagram for the significant Use Cases of Library Information Management System (LISM).

Solution – Sequence Diagram



Introduction

- Sequence diagrams model the dynamic aspects of a software system.
- The emphasis is on the “sequence” of messages rather than relationship between objects.
- A sequence diagram maps the flow of logic or flow of control within a usage scenario into a visual diagram enabling the software architect to both document and validate the logic during the analysis and design stages.
- Sequence diagrams provide more detail and show the message exchanged among a set of objects over time.
- Sequence diagrams are good for showing the behavior sequences seen by users of a diagram shows only the sequence of messages not their exact timing.
- Sequence diagrams can show concurrent signals.

Solution – Sequence Diagram



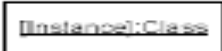
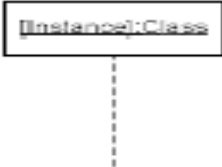
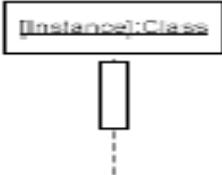
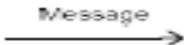
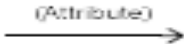
Purpose



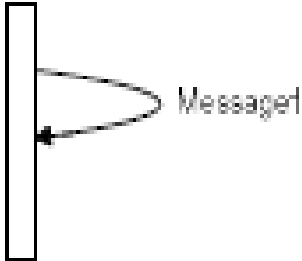
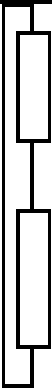
- The main purpose of this diagram is to represent how different business objects interact.
- A sequence diagram shows object interactions arranged in time sequence.
- It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

When to use : Sequence Diagram

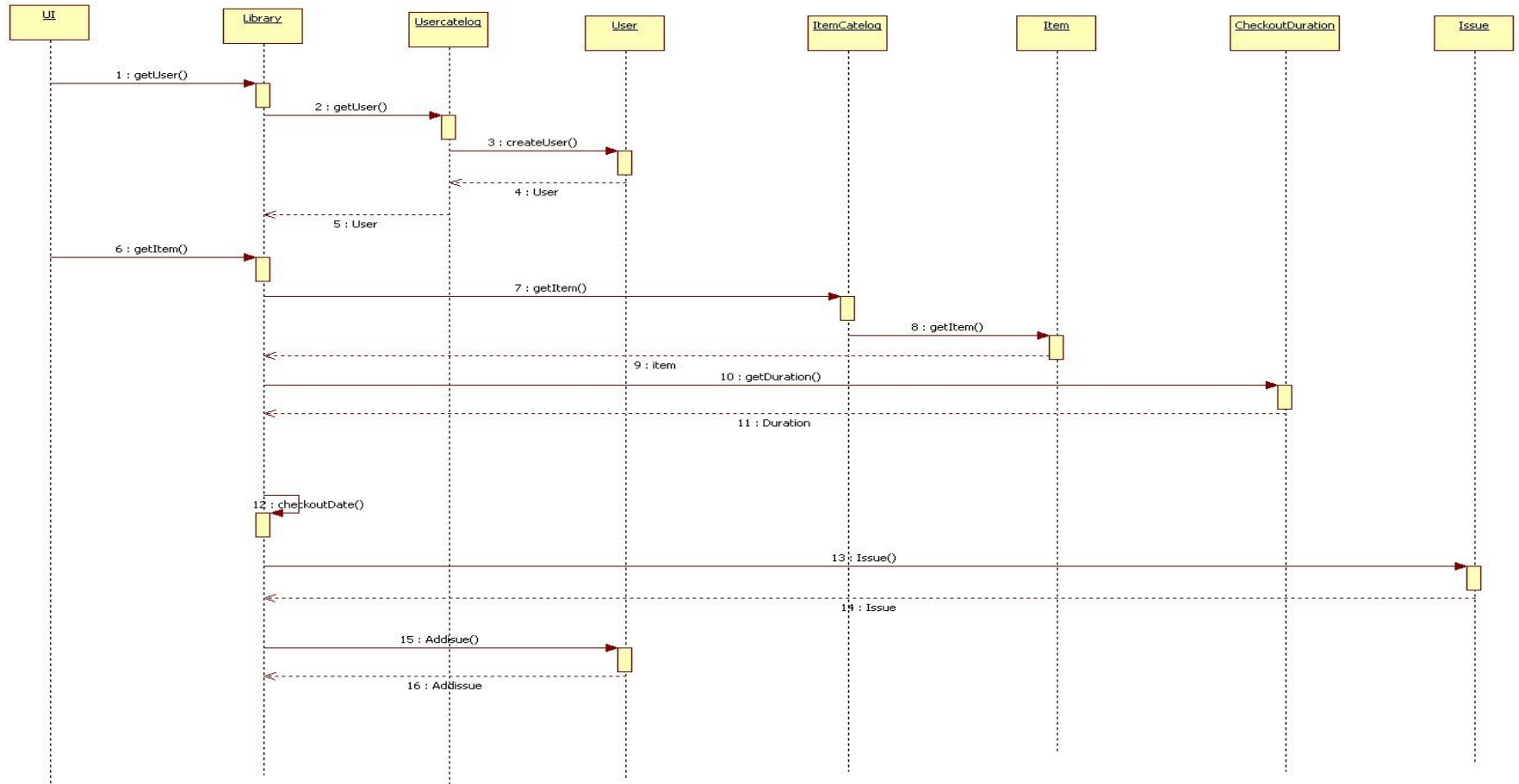
- Sequence diagram can be a helpful modeling tool when the dynamic behavior of objects needs to be observed in a particular use case or when there is a need for visualizing the “big picture of message flow”.
- A company’s technical staff could utilize sequence diagrams in order to document the behavior of a future system.
- It is during the design period that developers and architects utilize the diagram to showcase the system’s object interactions, thereby putting out a more fleshed out overall system design.

Sequence Diagram Notations

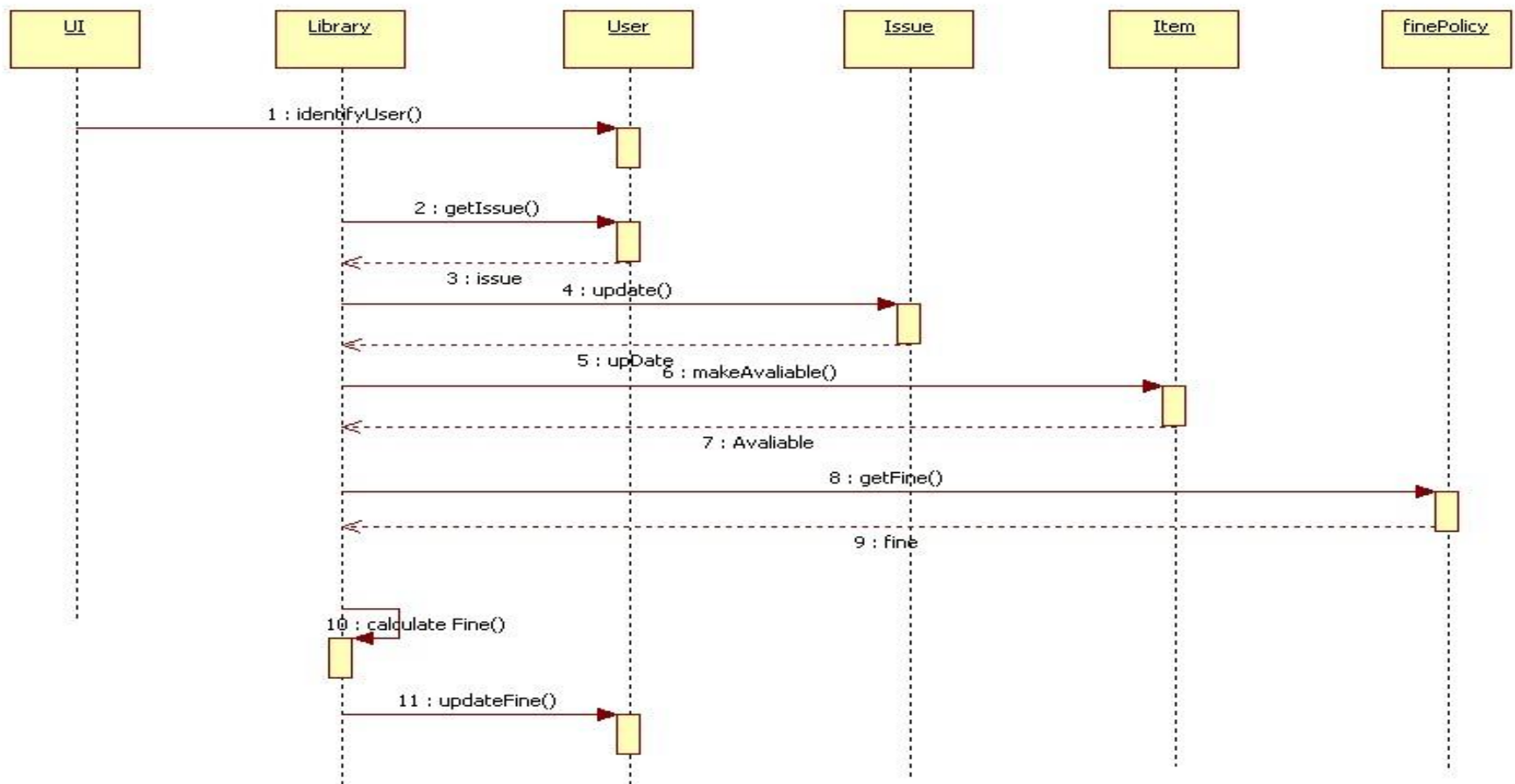
Sr. No.	Name	Notation	Description
1	Object		It represents the existence of an object of a particular time.
2	Life line		Lifeline represents the duration during which an object is alive and interacting with other objects in the system. It is represented by dashed lines.
3	Scope		It shows the time period during which an object or actor is performing an action.
4	Message transition		To send message from one object to another.
5	Message with attribute		To send message with some particular attribute

6	Message with constraint		To send message from one object to other by some constraint.
7	Acknowledgement		It represents communication between objects conveys acknowledgement.
8	Self message		Self message occurs when an object sends a message to itself.
9	Recursive message		Self message occurs when an object sends a message to itself within recursive scope.

Solution – Sequence Diagram : Issue of Book



Solution – Sequence Diagram : Return of Book



Experiment 5 : State Diagram



Problem:

Prepare the State Diagram for the significant Classes of Library Information Management System (LISM).

Solution – State Diagram



Introduction

- *A state diagram is a graph in which nodes correspond to states and directed arcs correspond to transitions labeled with event names.*
- A state diagram combines states and events in the form of a network to model all possible object states during its life cycle, helping to visualize how an object responds to different stimuli.
- A state diagram is a graph whose nodes are states and whose directed arcs are transitions between states.
- A state diagram specifies the state sequence caused by event sequence.
- State names must be unique within the scope of a state diagram.
- All objects in a class execute the state diagram for that class, which models their common behavior.
- We can implement state diagrams by direct interpretation or by converting the semantics into equivalent programming code.

Solution – State Diagram






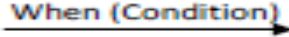
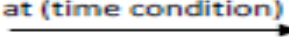

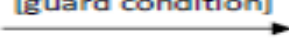

Purpose




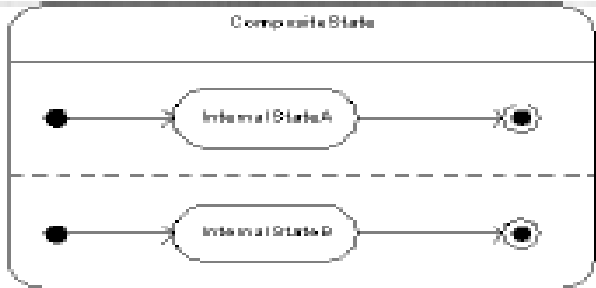



- The state model describes those aspects of objects concerned with time and the sequencing of operations events that mark changes, states that define the context for events, and the organization of events and states.
- They are used to give an abstract description of the behavior of a system.
- It provides direction and guidance to the individual counties within the states.
- It specifies the possible states, what transitions are allowed between states.
- It describes the common behavior for the objects in a class and each object changes its behavior from one state to another.
- It is used to describe the dependence of the functionality on the state of the system that is how the functionality of an object depends on its state and how its state changes as a result of the events that it receives.
- It describes dynamic behavior of the objects of the system.

When to use: State Diagram

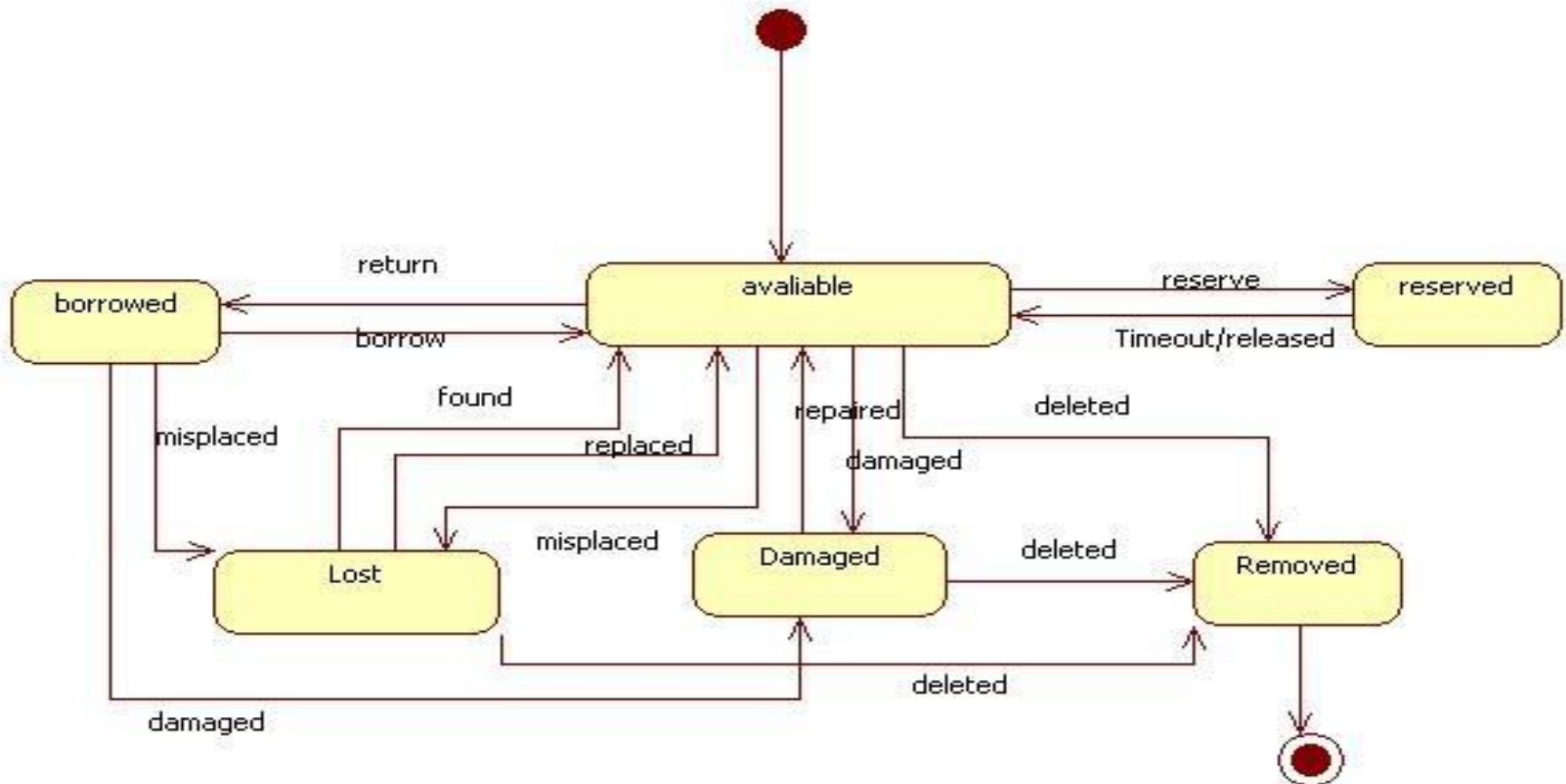
- They are perfectly useful to model behavior in real time system.
- Each state represents a named condition during the life of an object during which it satisfies some condition or waits for some event.
- It determines how objects of that class react to events.

State Diagram Notations

No.	Name	Notation	Description
1	State		A state is an abstraction of the values and links of an object. State models a situation during which some (usually implicit) invariant condition holds.
2	Transition		A transition is a directed relationship between a source state and a target state. It may be part of a compound transition, which takes the state machine from one state configuration to another
3	Event		A transition is an instantaneous change from one to another state
4	Change Event		A change in value of a Boolean expression
5	Time Event		The arrival of an absolute time or the passage of a relative amount of time
6	Signal Event		Receipt of an explicit, named, asynchronous communication among objects.
7	Guarded transition		A guard condition is a Boolean expression that must be true in order for a transition to occur.
8	Do activity		A do activity an activity that continuous for extended time within state.

9	Entry activity		An state is entered by any incoming transition the entry activity is performed
10	Exit activity		When the state is exited by any outgoing transition the exit activity is performed
11	Nested State Diagram Sub machine Diagram		A submachine state specifies the insertion of the specification of a submachine. The state machine that contains the submachine state is called the containing state machine.
12	Composite State		A state can be refined hierarchically by composite states.
13	Activity effect		An activity is actual behavior that can be invoked by any number of effects
14	Initial state point		It shows the starting state of object.
15	Final state point		It shows the terminating state of object.

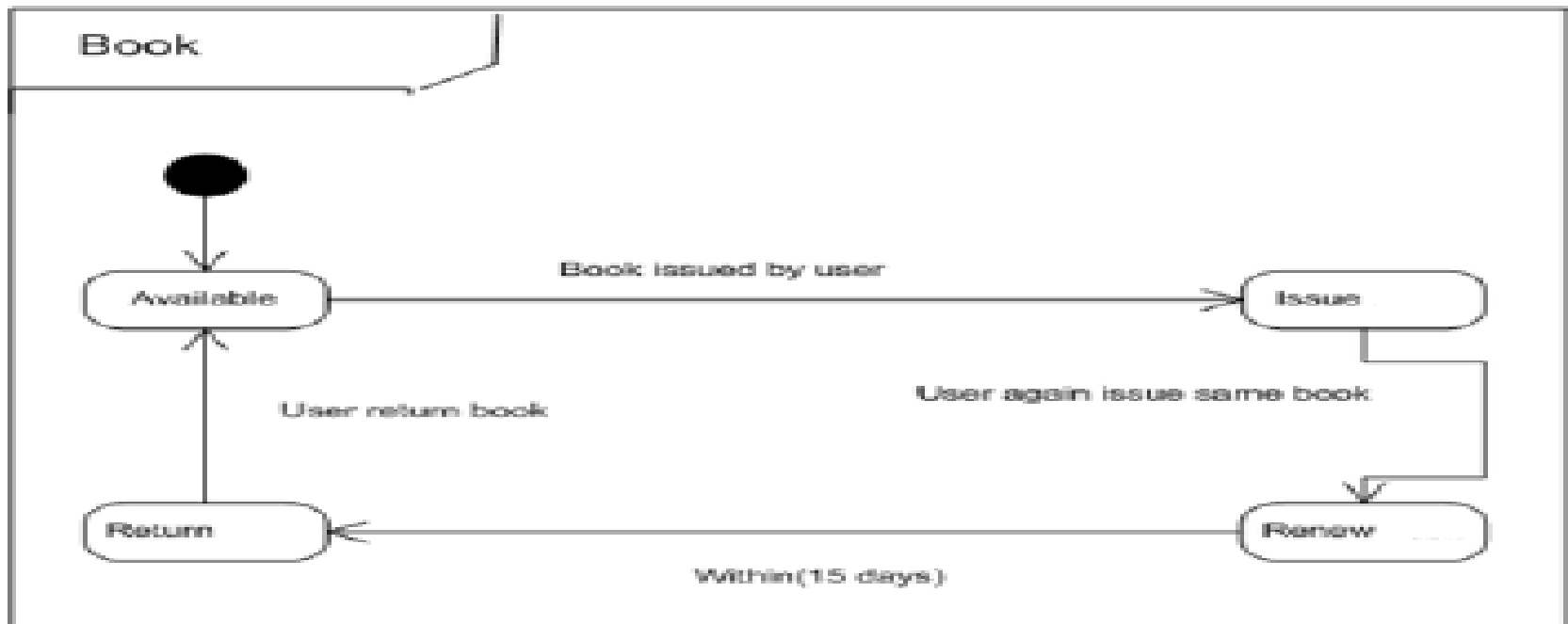
Solution - State Diagram : Book



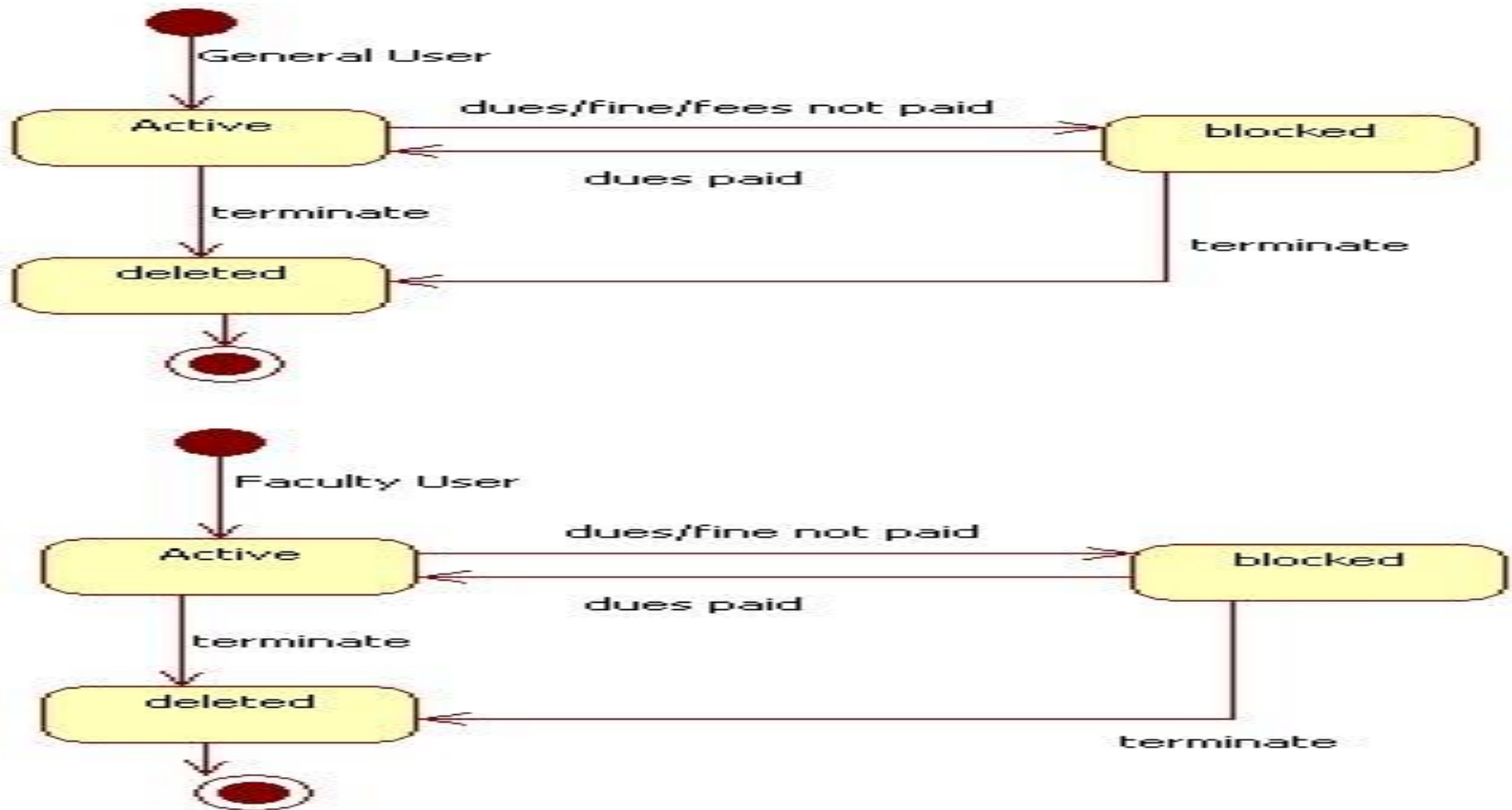
Solution - State Diagram : Book



State diagram for library management system

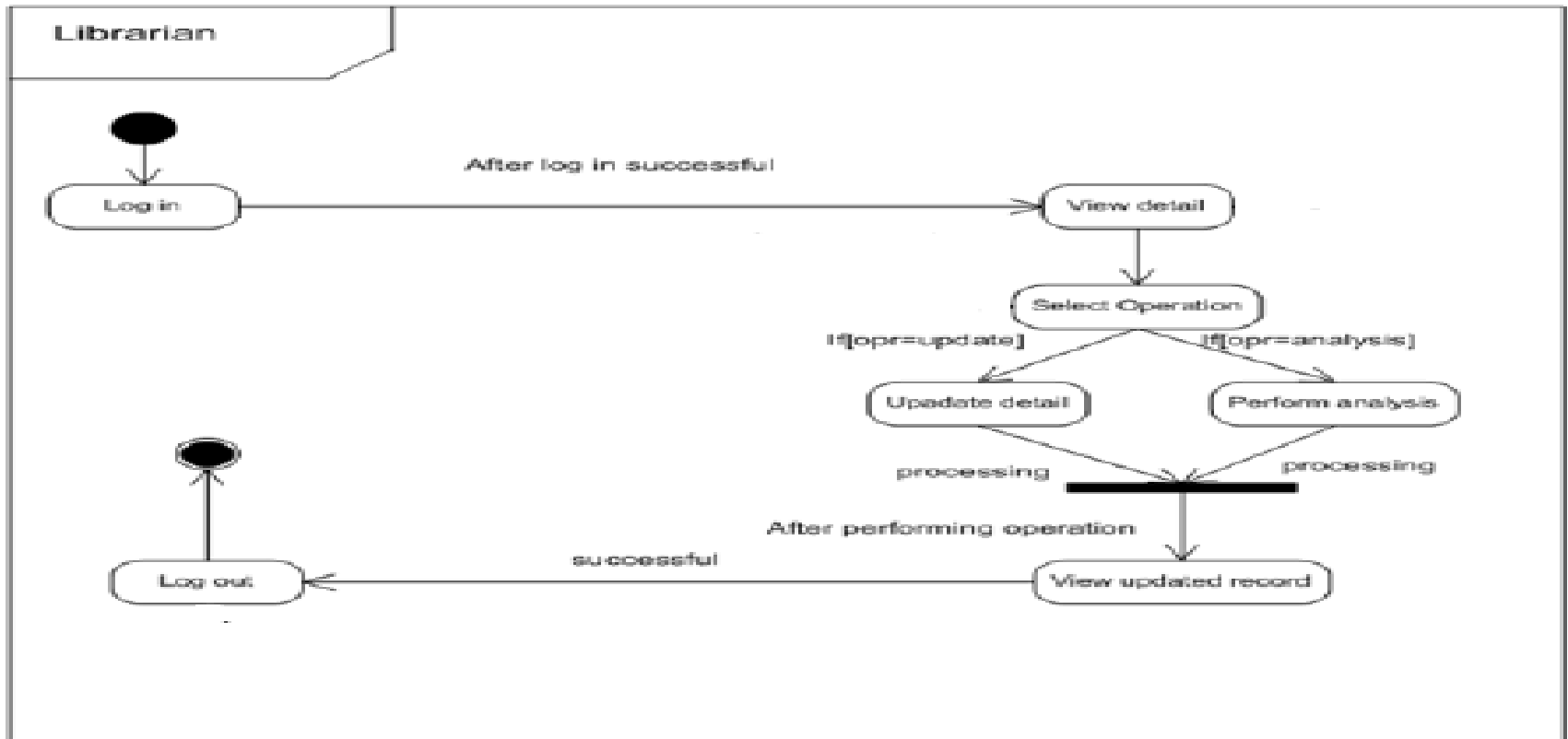


Solution - State Diagram : User

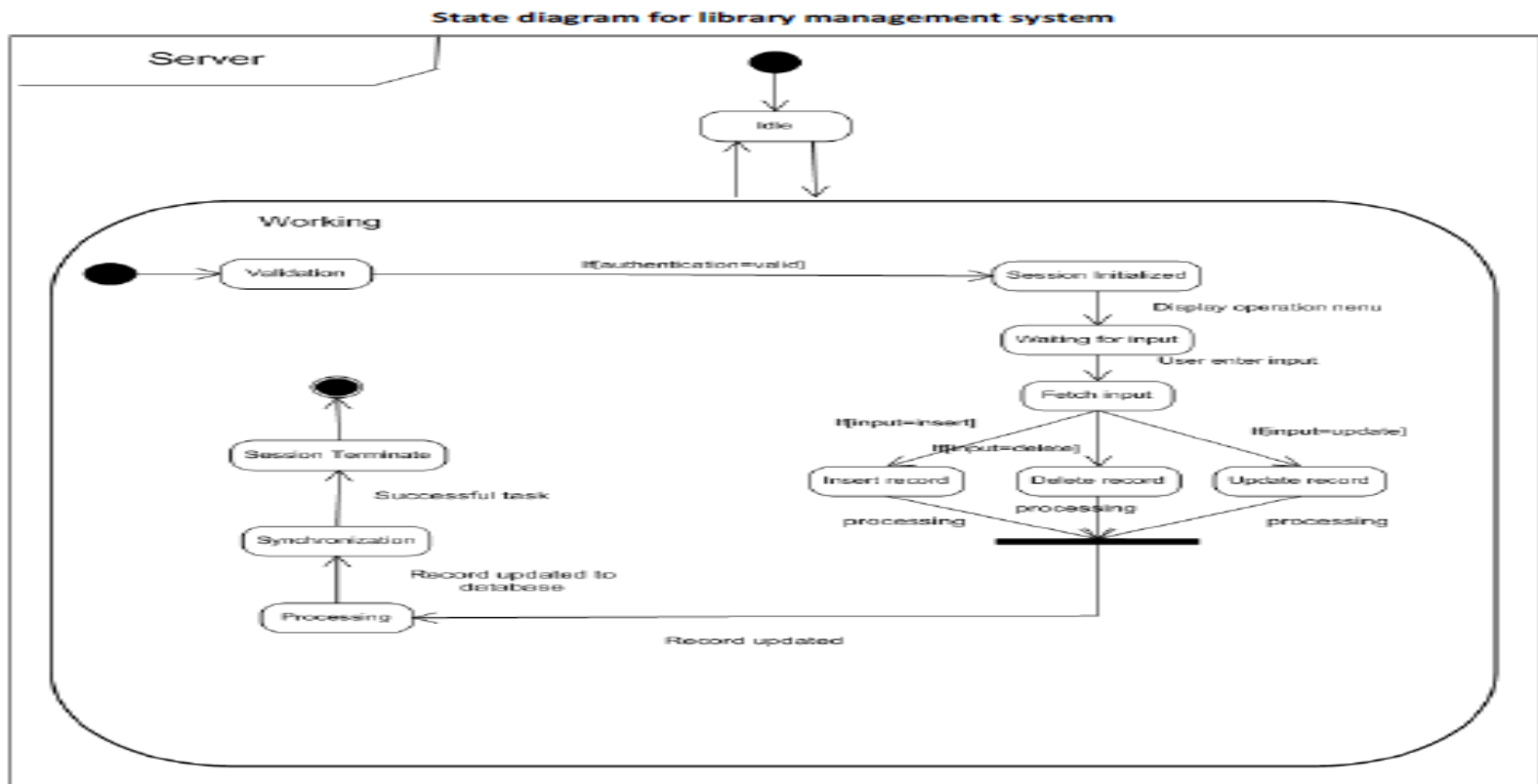


Solution - State Diagram : Librarian

State diagram for library management system



Solution - State Diagram : Server



Experiment 6 - Collaboration Diagram



Problem:

Prepare the Collaboration Diagram for the significant Use Cases of Library Information Management System (LISM).

Solution - Collaboration Diagram



Purpose

- ❧ A collaboration diagram shows the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

Compared with a sequence diagram

- ❧ A sequence diagram shows the objects and messages involved in an interaction. It shows the timing of the messages, but not the relationships among the objects.

Solution 6 - Collaboration Diagram



Solution 6 - Collaboration Diagram

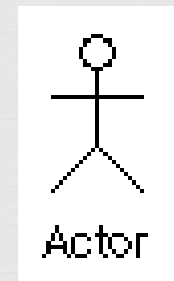


1. Actor

✧ An actor in a collaboration diagram represents the person, software, hardware, or other agent external to the system that is interacting with the system.

Label

✧ Name



Solution 6 - Collaboration Diagram



2. Instance

- ❧ An instance in a collaboration diagram represents an instantiation of a class in a class diagram or a use case in a use case diagram

Label

- ❧ [instance-name][:instance-type]
- ❧ instance-name is the name of the instance
- ❧ instance-type is a class or a use case



Charles:Customer

- ❧ **Note:** While both elements of the label are optional, you must provide one of them to avoid check errors.

Solution 6 - Collaboration Diagram

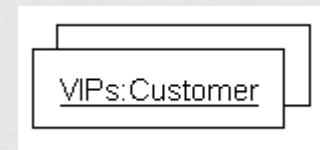


3. Multi object

- ❧ A multi object represents a set of instances at the many end of an association. If necessary, you can use the instance symbol to show one instance of the set and a composite link symbol to show that the instance is part of the multi object.

Label

- ❧ [name][:type]
- ❧ name is the name of the multi object
- ❧ type is a class or a use case



Note: While both elements of the label are optional, you must provide one of them to avoid check errors.

Solution 6 - Collaboration Diagram

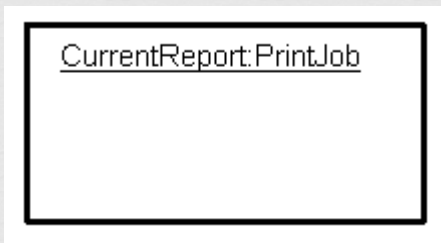


4. Active object

- ✧ An active object is an instance that owns a thread of control and can initiate control activity. For example, processes and tasks are active objects. Active objects can contain other symbols and links among them.

Label

- ✧ [name][:type]
- ✧ name is the name of the active object
- ✧ type is a class or a use case



Note: While both elements of the label are optional, you must provide one of them to avoid check errors.

Solution 6 - Collaboration Diagram



Messages

- ❧ A message flow carries a message from one object to another along any type of connector (link, aggregation link, and so on). Each message flow in a collaboration diagram is characterized by direction and type.

Creating and moving messages

- ❧ You must create a connector before you can create a message. You can place message symbols on or near a connector, moving them if necessary to prevent them from overlapping the connector. When you move a connector, its message symbols move with it.

Message direction

There are two message directions:

- ❧ Backward Messages
- ❧ Forward Messages

The direction in which the link was drawn determines the direction of the message.

Solution 6 - Collaboration Diagram



Message type

You can create these types of messages:

- ❧ Nested message
- ❧ Flat message
- ❧ Asynchronous message

Solution 6 - Collaboration Diagram



Nested message

- ❧ The nested message represents a procedure call or other nested flow of control. The nested sequence is completed before the outer level sequence resumes. The nested message symbol is represented by a filled solid arrowhead.



Solution 6 - Collaboration Diagram



Flat message

✧ The flat message shows the progression to the next step in a sequence. The flat message symbol is represented by a stick arrowhead.



Solution 6 - Collaboration Diagram

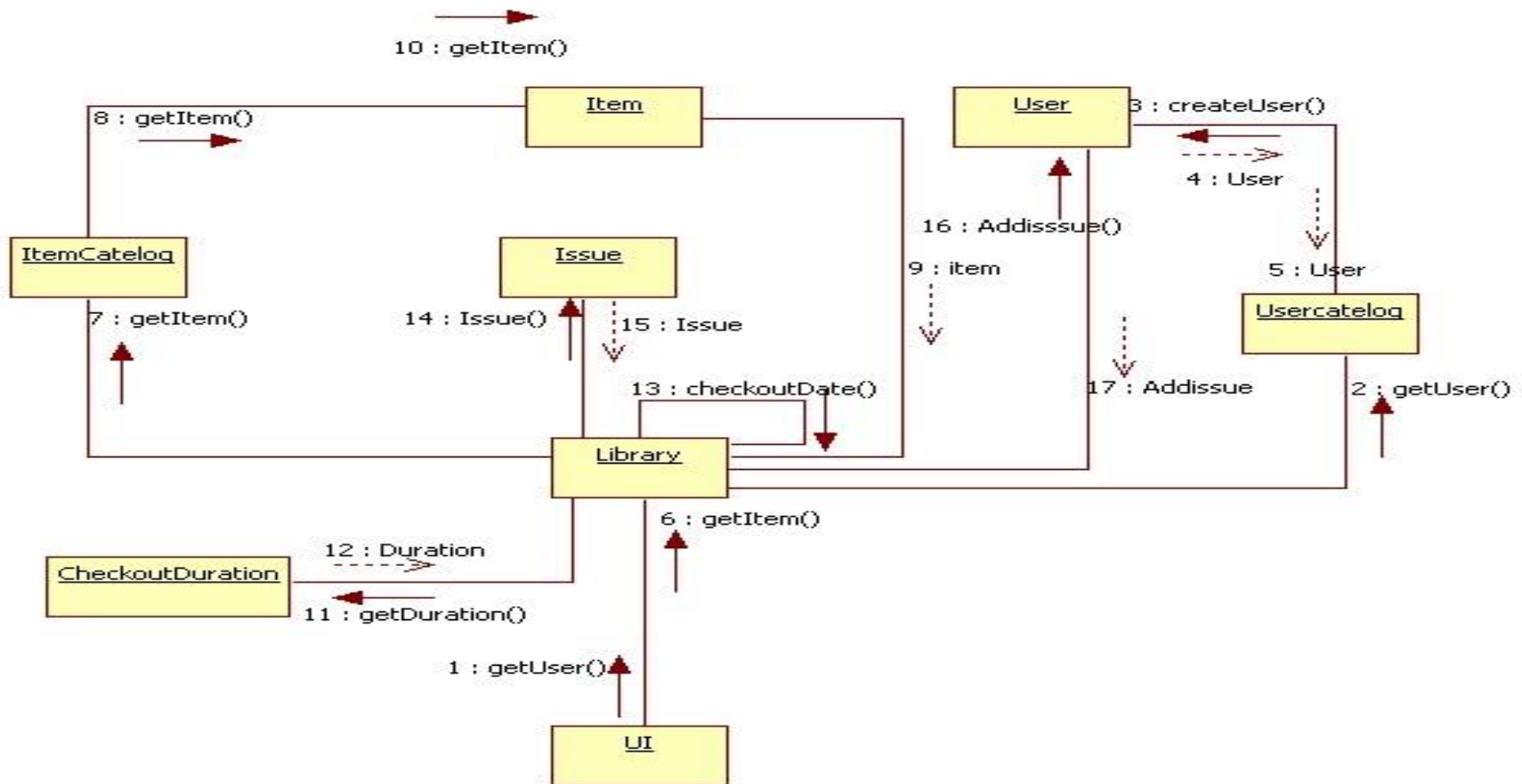


Asynchronous message

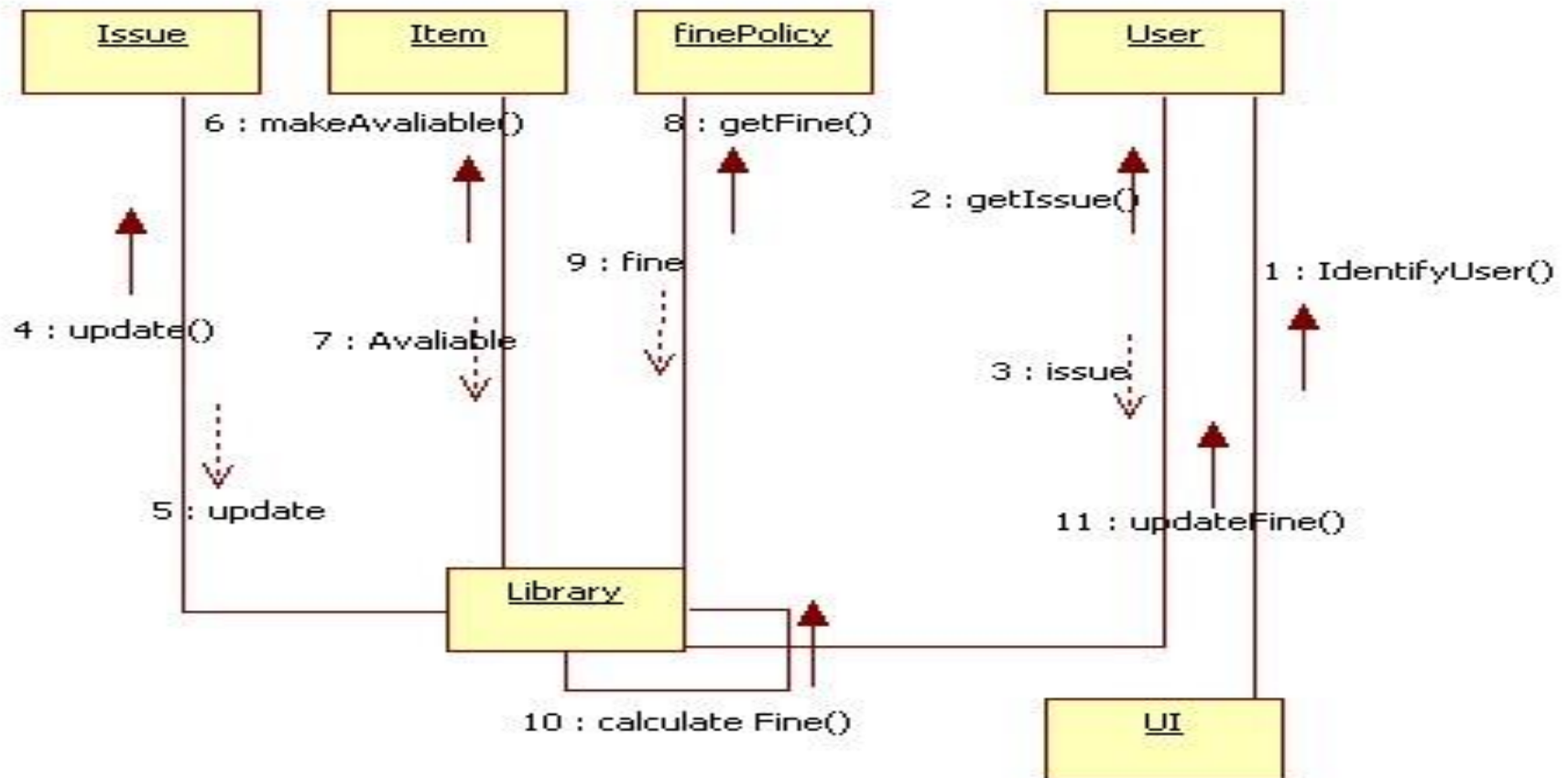
✧ The asynchronous message symbol shows an asynchronous message between two objects. The asynchronous message symbol is represented by a half stick arrowhead.



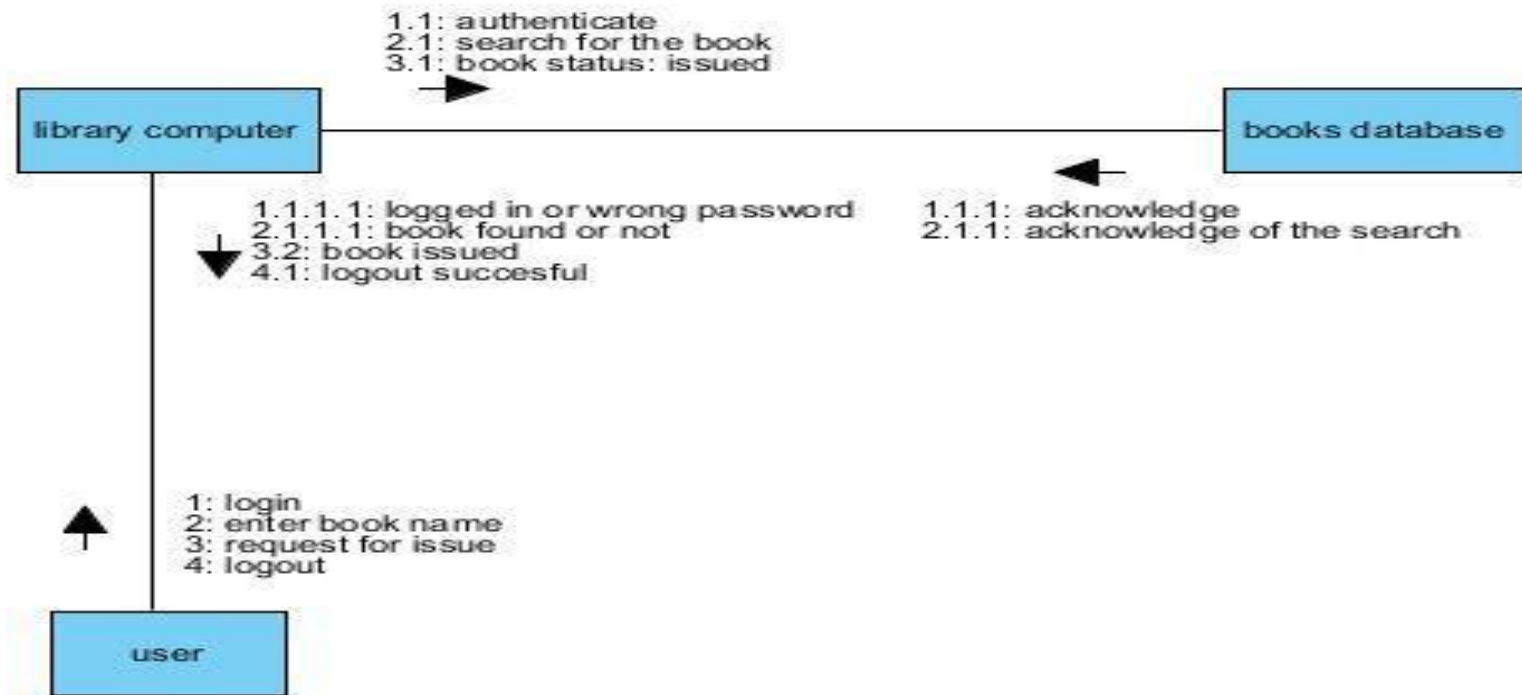
Solution - Collaboration Diagram : Issue of Book



Solution - Collaboration Diagram : Return of Book



Collaboration Diagram for searching a book and issuing it as per the request by the user from the librarian



Experiment 7 - Component Diagram



❧ Problem:

Prepare the Component Diagram of Library Information Management System (LISM).

Experiment 7 - Component Diagram

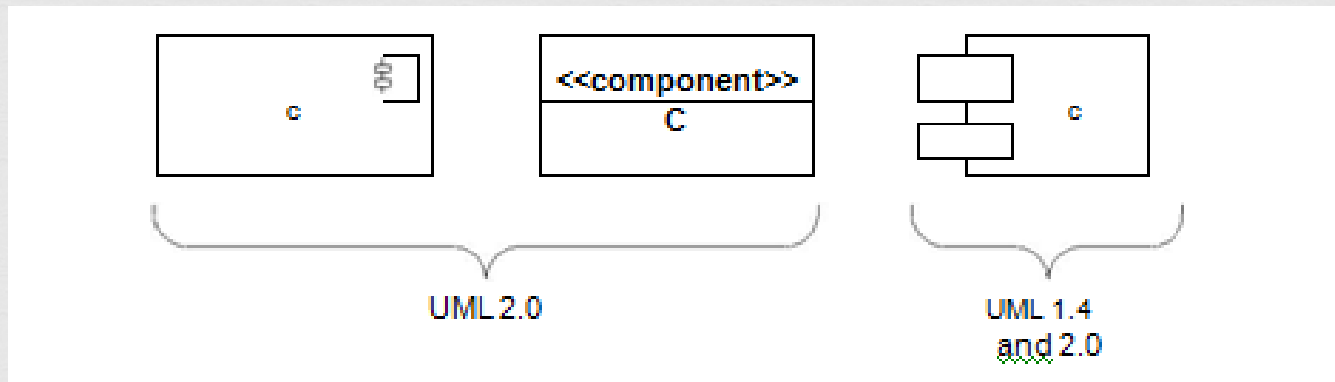


- ❧ The Component Diagram helps to model the physical aspect of an Object-Oriented software system.
- ❧ It illustrates the architectures of the software components and the dependencies between them.
- ❧ The software components includes run-time components, executable components also the source code components.

Solution - Collaboration Diagram



The major elements of UML component diagram - component, provided interface, required interface, port, connectors.



Three Ways to Represent Components

Solution - Collaboration Diagram



- ❧ A *port* represents a group of messages or operation calls that pass either into or out of a component. The group is described by an interface, which defines the port's type. A port can either provide an interface or require an interface.
- ❧ Specifies a distinct interaction point
 - ❧ Between that component and its environment
 - ❧ Between that component and its internal parts

Solution - Collaboration Diagram - Sample Example

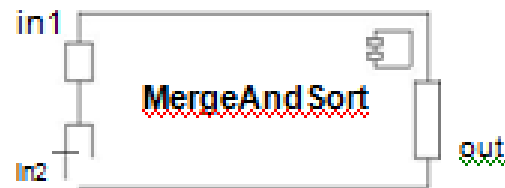


Figure 9: External View of a Component with Ports

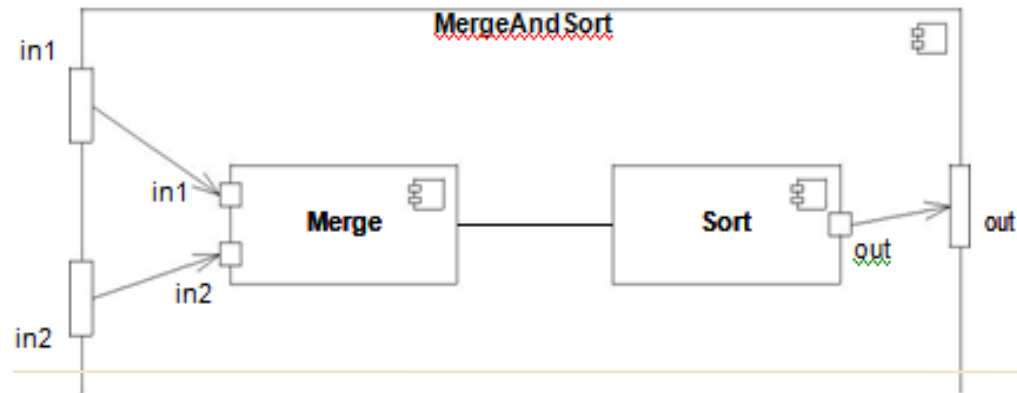


Figure 10: Internal View of a Component with Ports

Solution - Collaboration Diagram – Sample Example

- ❧ An **interface** is the definition of a collection of one or more operations.
- ❧ Provides only the operations but not the implementation
- ❧ Implementation is normally provided by a component

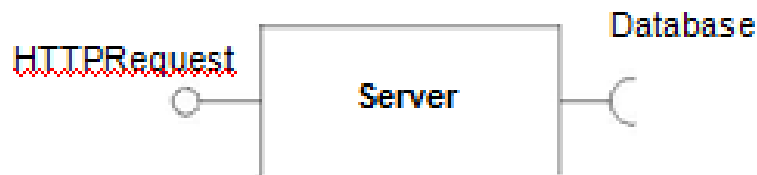
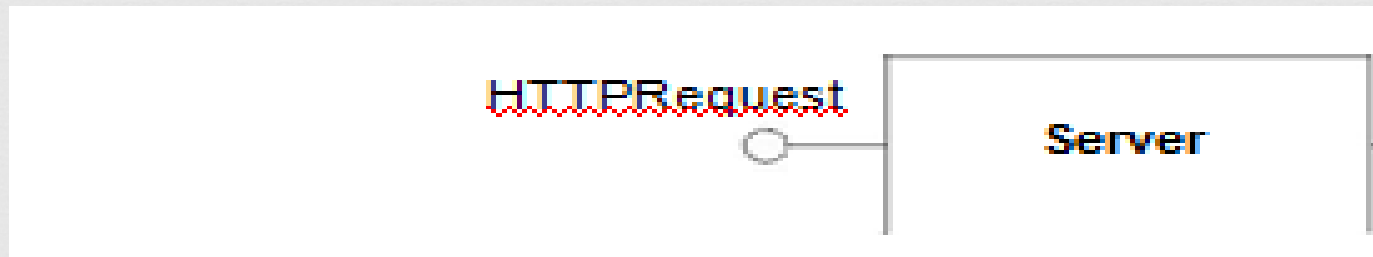


Figure 3: Interfaces Using Ball-and-Socket Notation

Solution - Collaboration Diagram - Sample Example

- ❧ A port with a *provided interface* supplies operations that are implemented by the component, and that can be used by other components.
- ❧ Examples include a user interface, a Web service, a .NET interface, or a collection of functions in any programming language.



Solution - Collaboration Diagram - Sample Example

- ❧ A port with a *required interface* represents a component's requirement for a group of operations or services to be provided by other components or external systems.
- ❧ For example, a Web browser requires Web servers, or an application add-in requires services from the application.
- ❧ A component can have any number of ports.

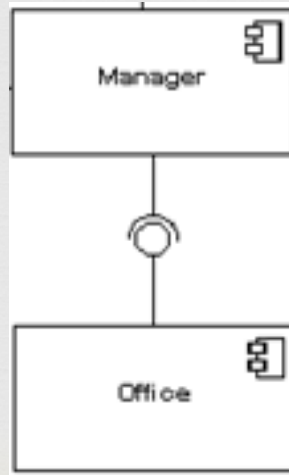


Solution - Collaboration Diagram

- ❧ A **connector** represents a communication link between two or more instances
- ❧ Two kinds of connectors:
 - Delegation
 - Assembly

Solution - Collaboration Diagram - Sample Example

- ❧ An **assembly connector** between 2 components defines that one component provides the services that another component requires
- ❧ It must only be defined from a required interface to a provided interface
- ❧ An assembly connector is notated by a “ball-and-socket” connection



Solution - Collaboration Diagram - Sample Example

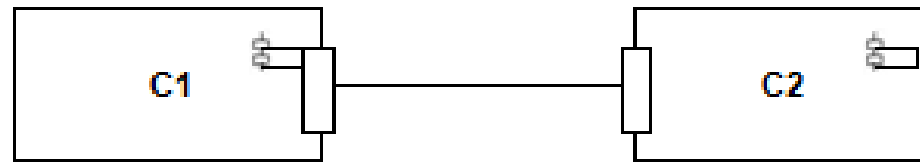


Figure 11: An Assembly Connector Between Ports

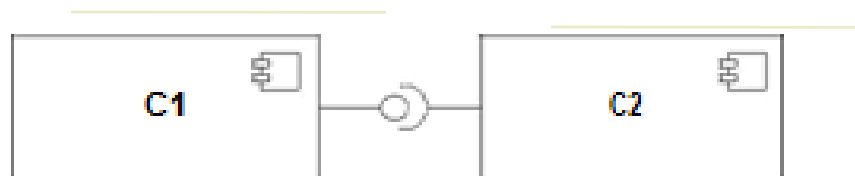


Figure 12: An Assembly Connector Between Interfaces

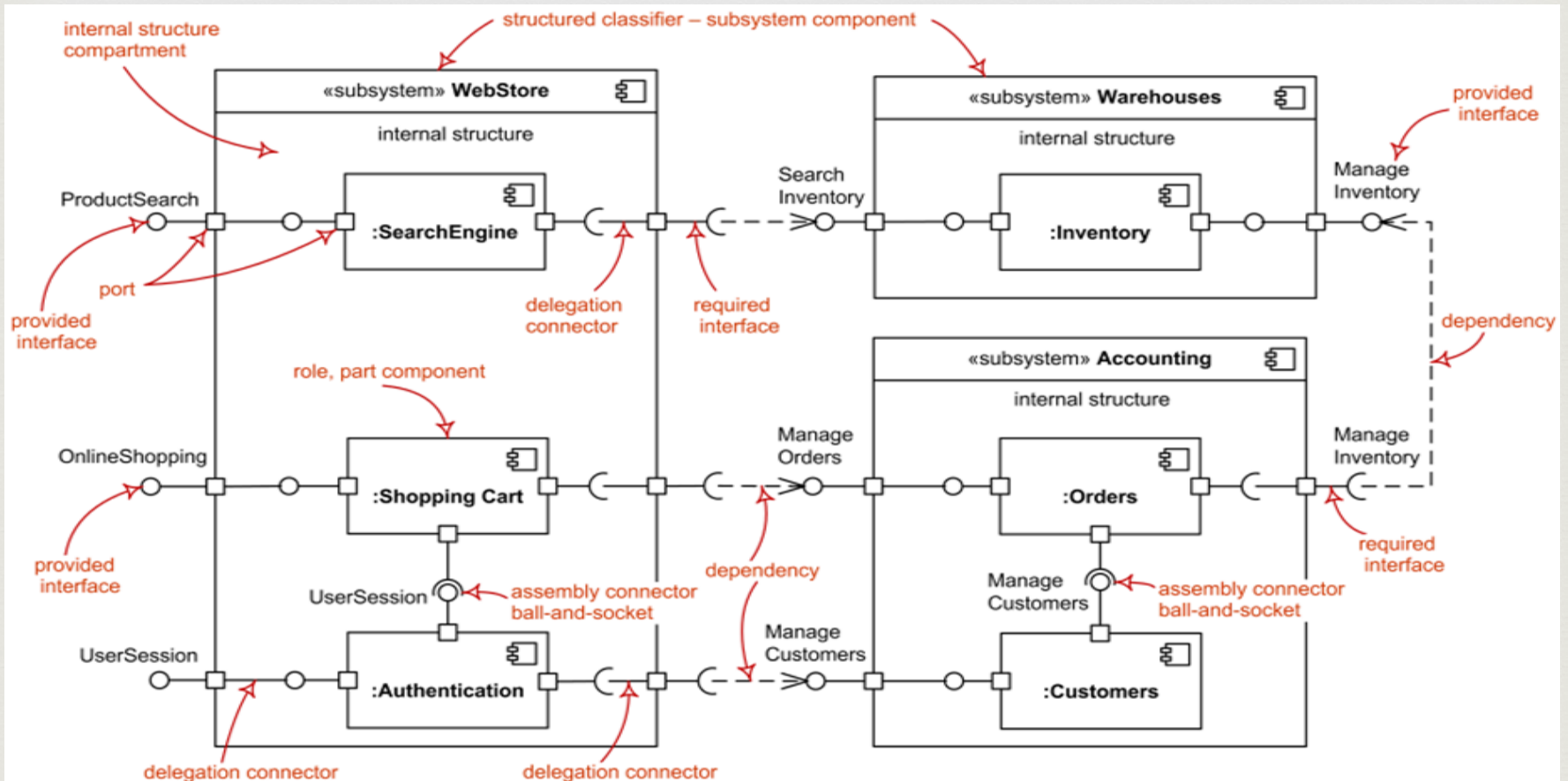
Solution - Collaboration Diagram - Sample Example

- ❧ **Delegation connector** links the external contract of a component to the internal realization
- ❧ Represents the forwarding of signals
- ❧ It must only be defined between used interfaces or ports of the same kind.

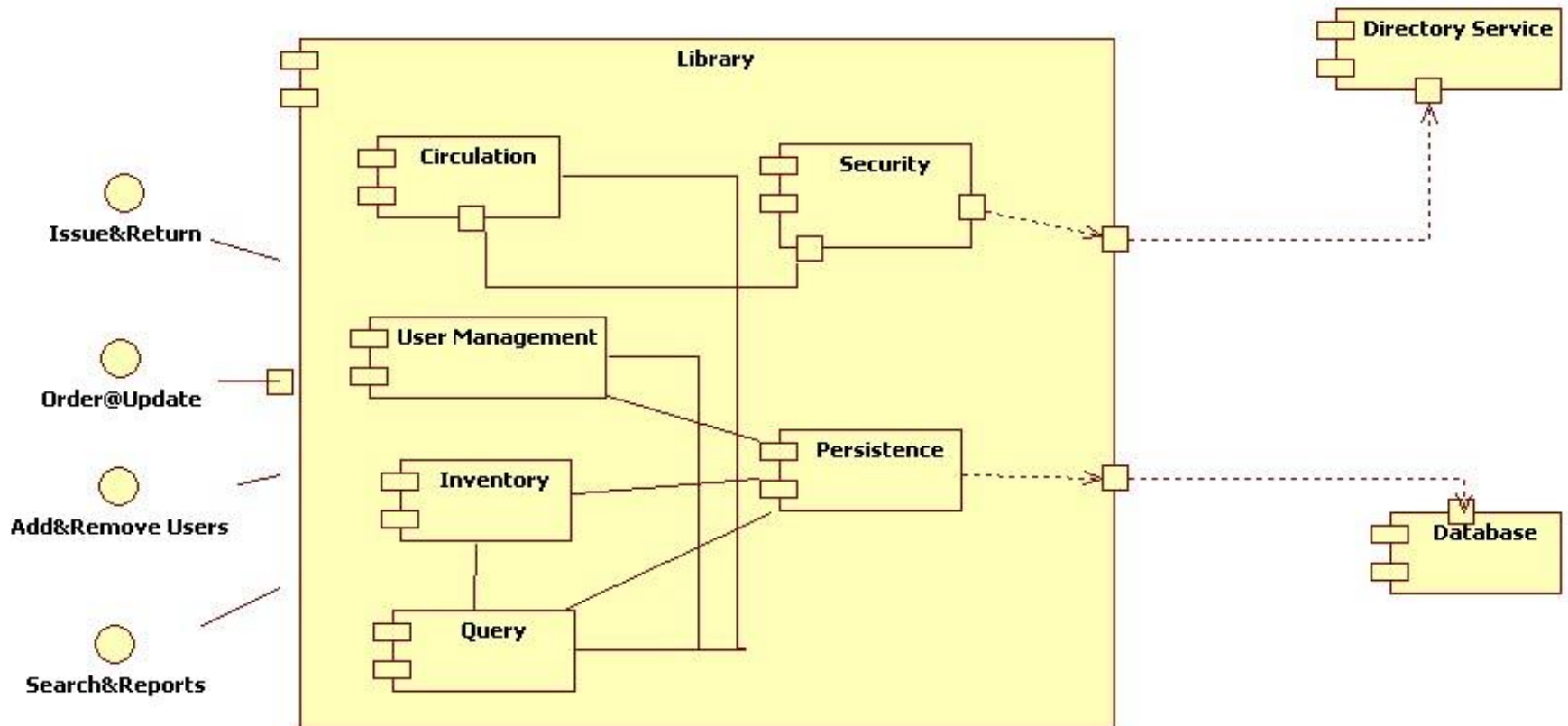


Figure 13: Delegation Connectors Between External and Internal Ports

Solution - Collaboration Diagram - Sample Example



Solution - Component Diagram for LISM



Experiment 8 : Deployment Diagram



Problem:

Prepare the Deployment Diagram of Library Information Management System (LISM).

Solution : Deployment Diagram



- ❧ A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).
- ❧ The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

Solution : Deployment Diagram

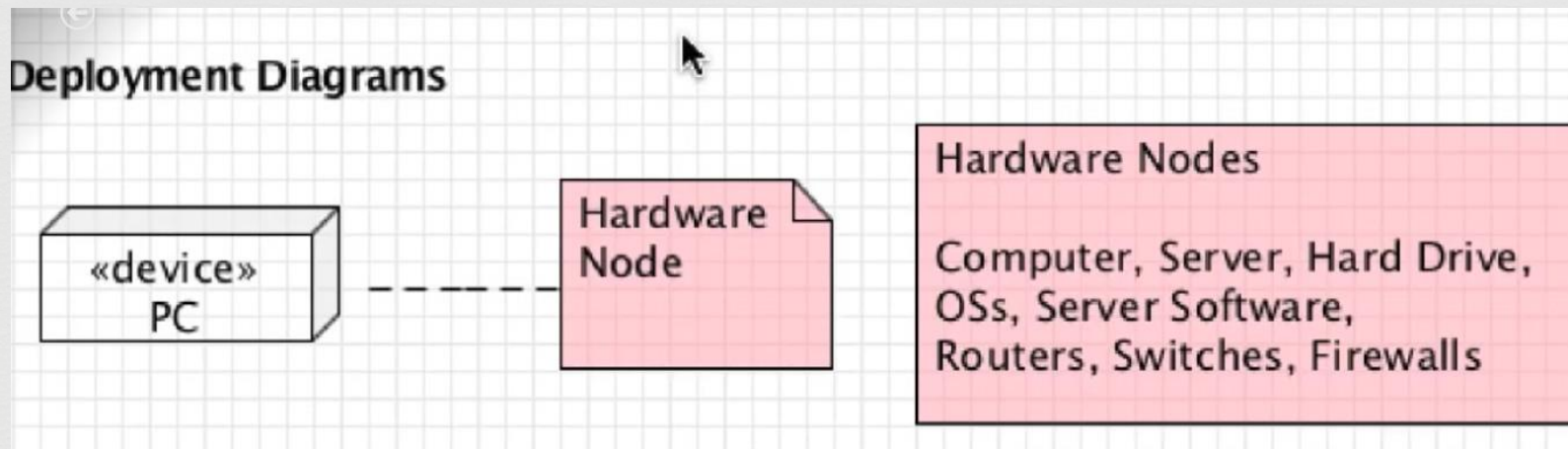


- ❧ There are two types of Nodes:
 - ❧ **Device Node**
 - ❧ **Execution Environment Node**

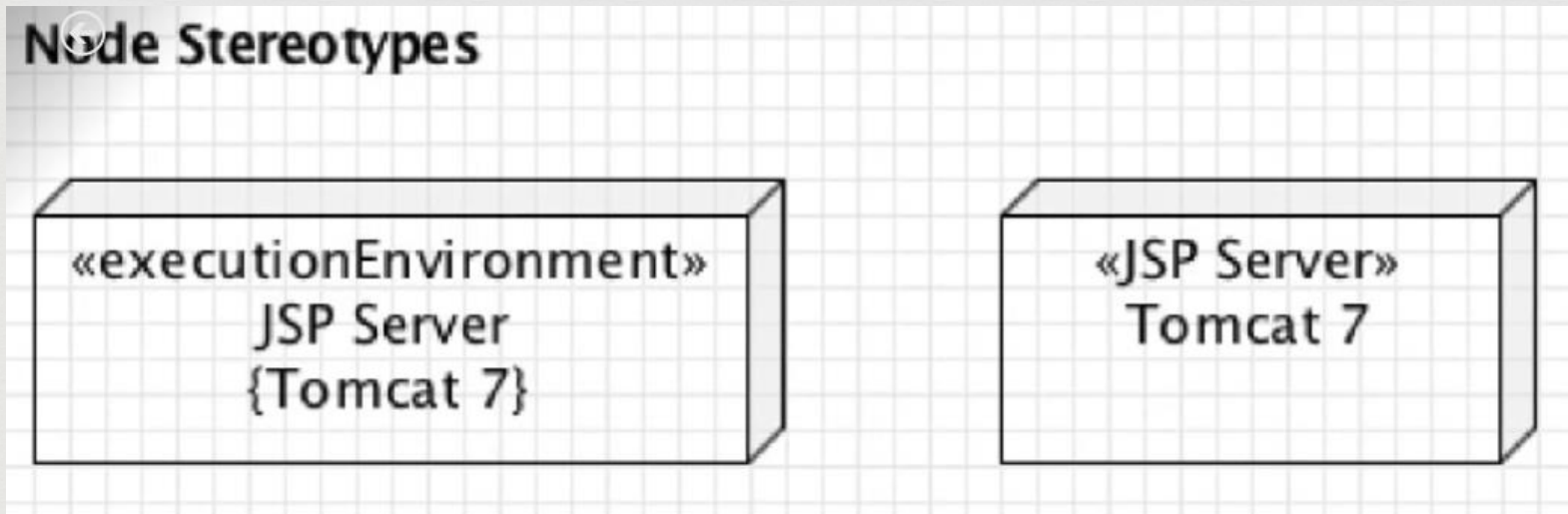
Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones.

An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements.

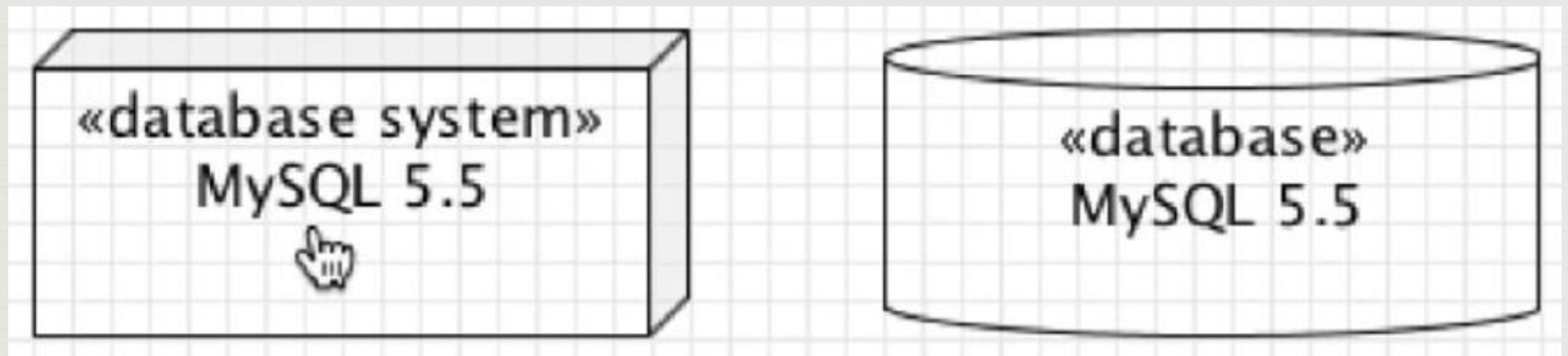
Solution : Deployment Diagram – Sample Example



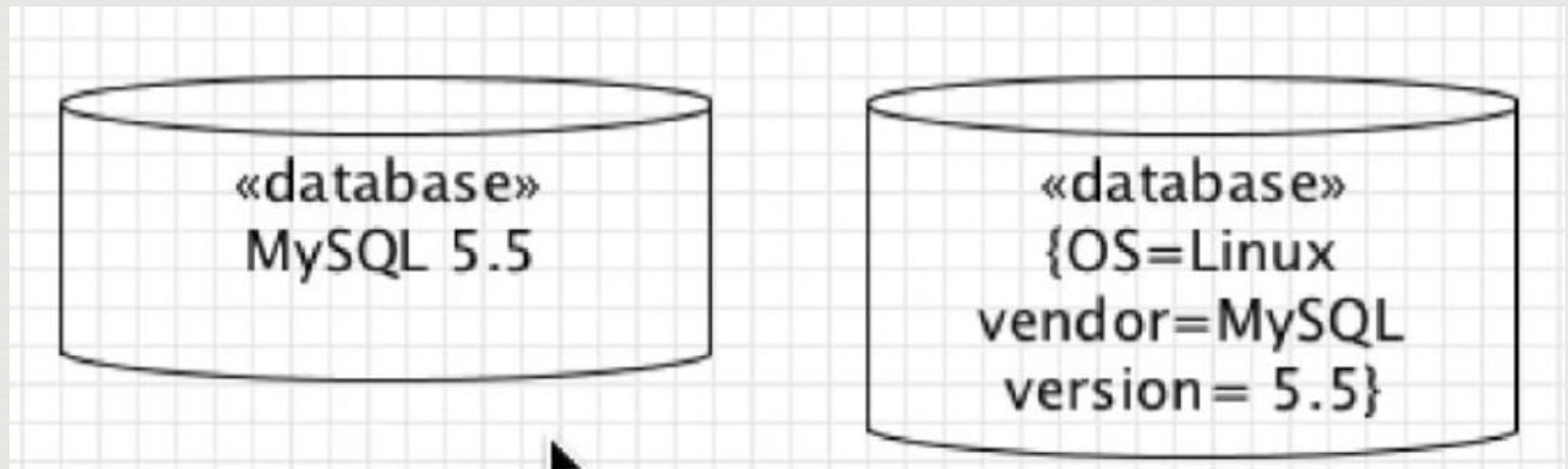
Solution : Deployment Diagram – Sample Example



Solution : Deployment Diagram – Sample Example

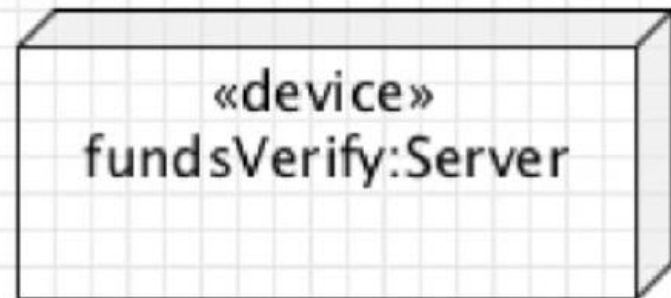
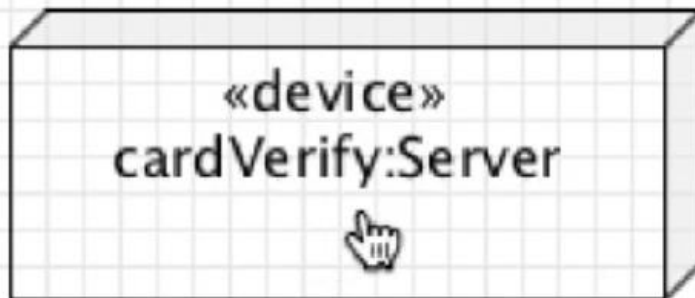


Solution : Deployment Diagram – Sample Example

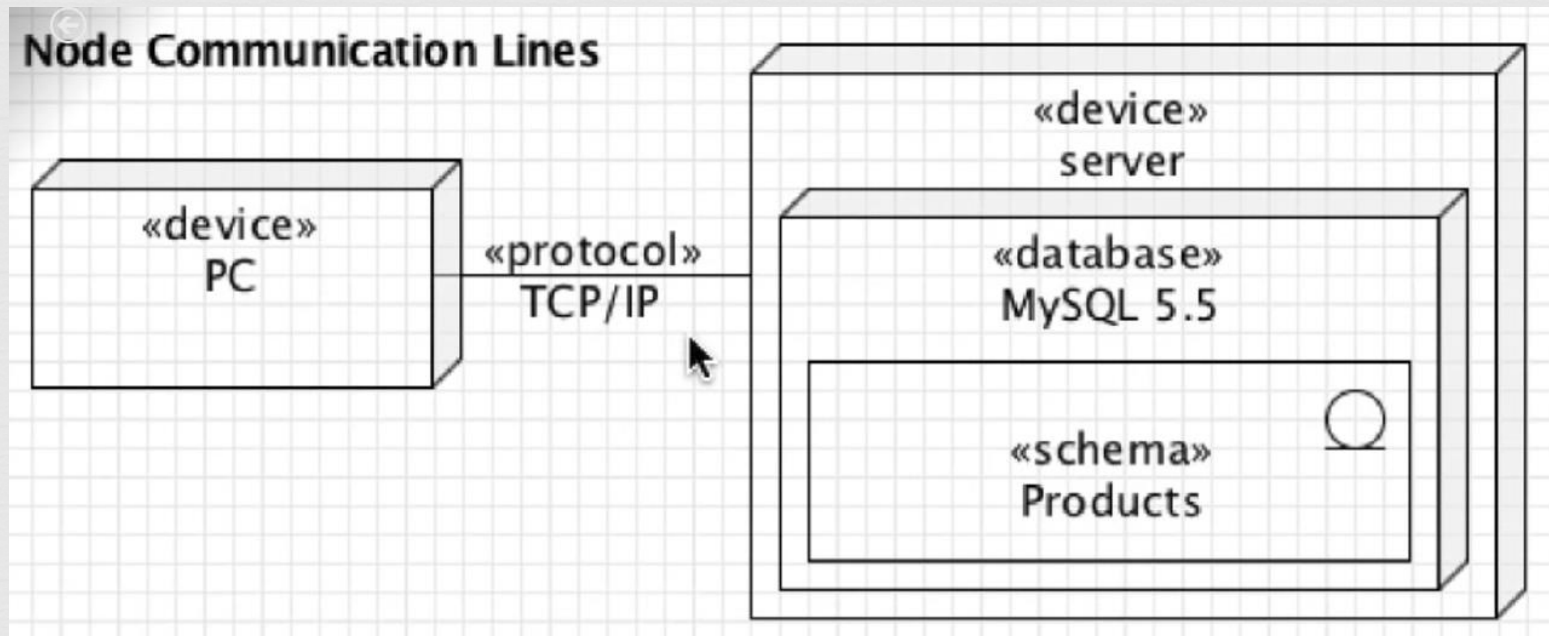


Solution : Deployment Diagram – Sample Example

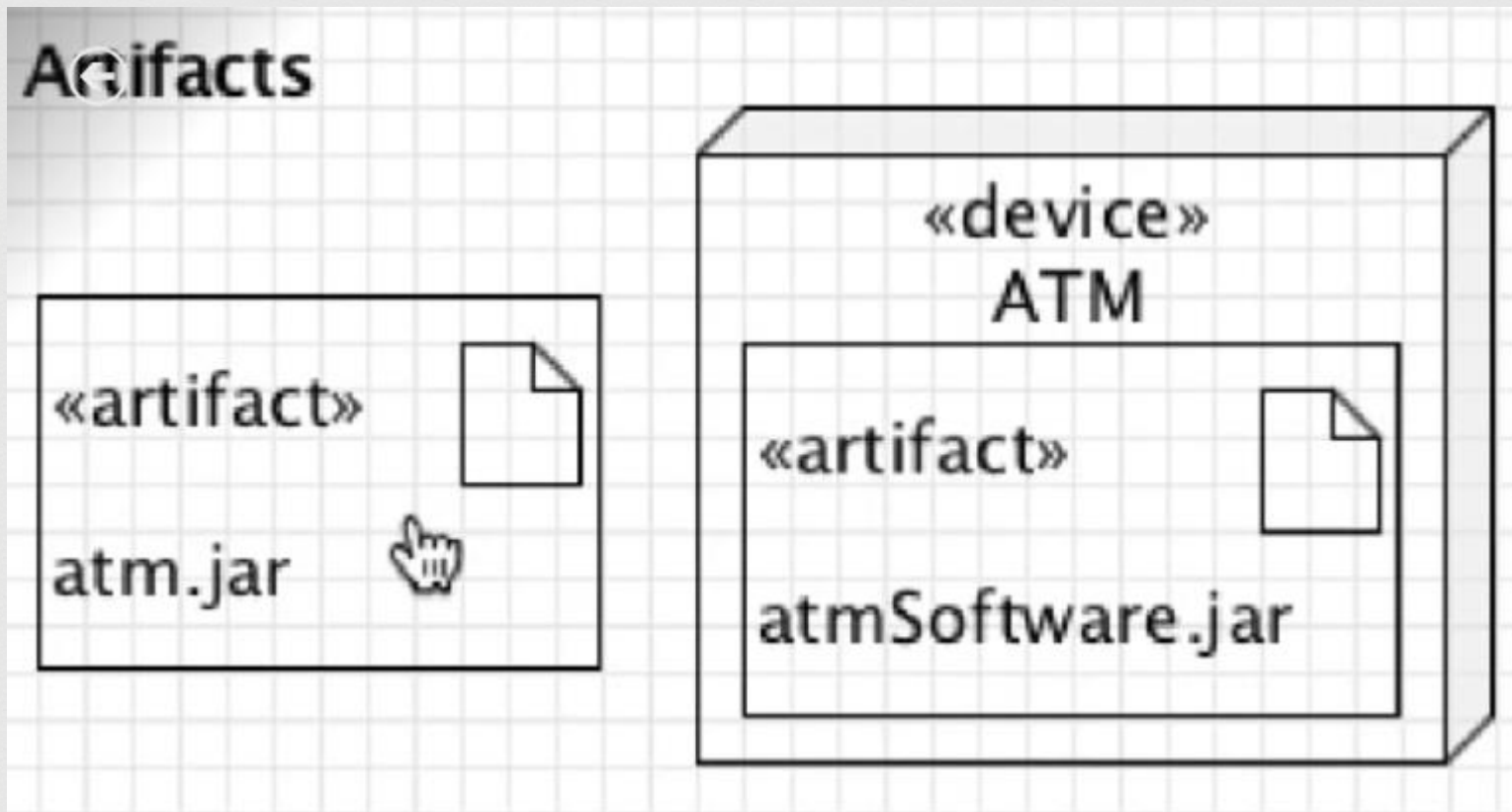
Named Nodes



Solution : Deployment Diagram – Sample Example



Solution : Deployment Diagram – Sample Example

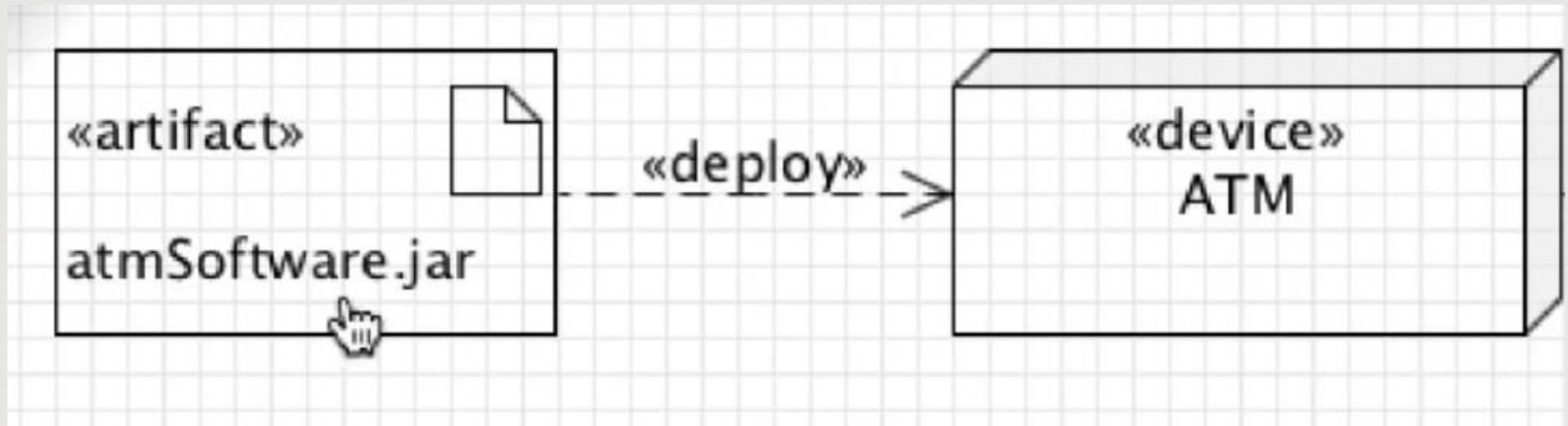


Solution : Deployment Diagram – Sample Example

Artifacts

txt, exes, jars, xml, drivers, src, config files or any other type of

Solution : Deployment Diagram - Sample Example



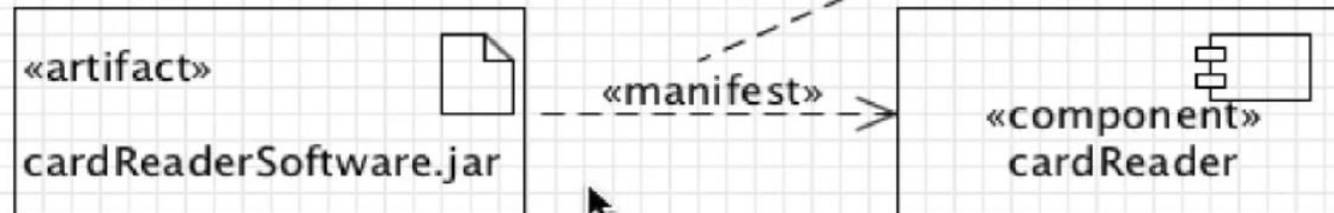
Solution : Deployment Diagram – Sample Example



Solution : Deployment Diagram - Sample Example

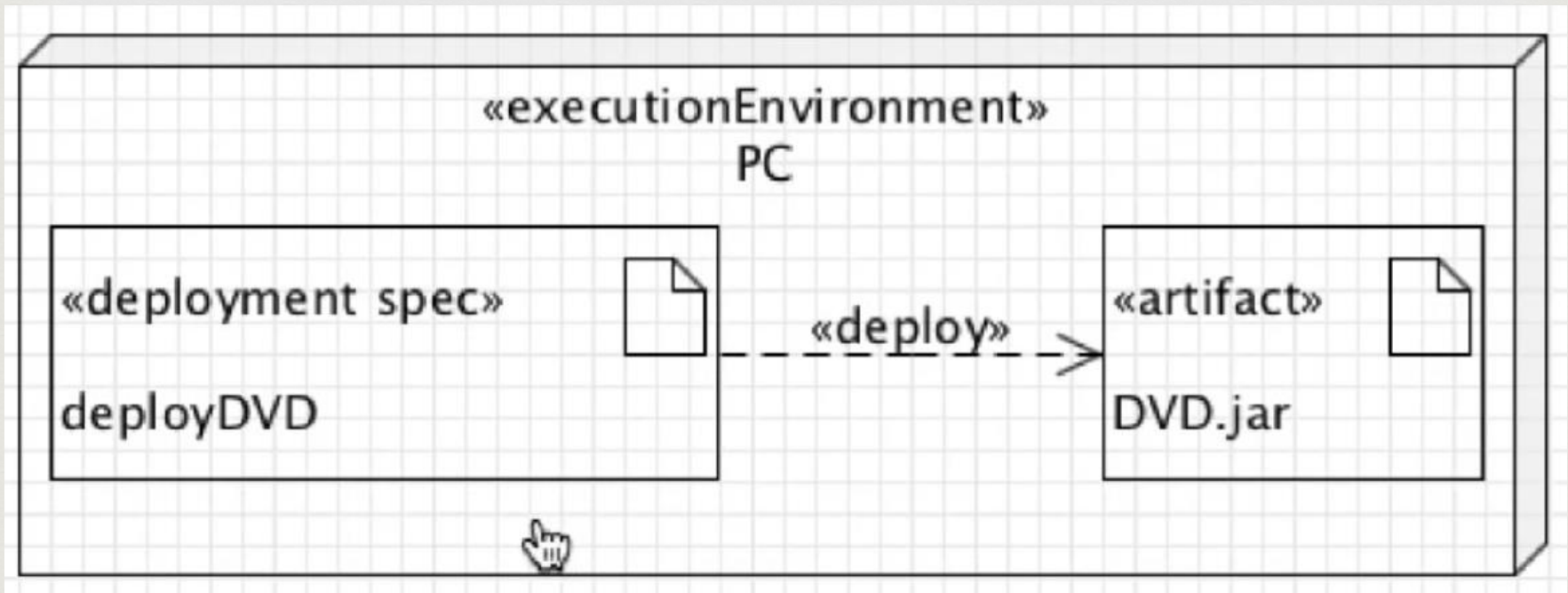
manifest means to implement a component

Artifacts & Components

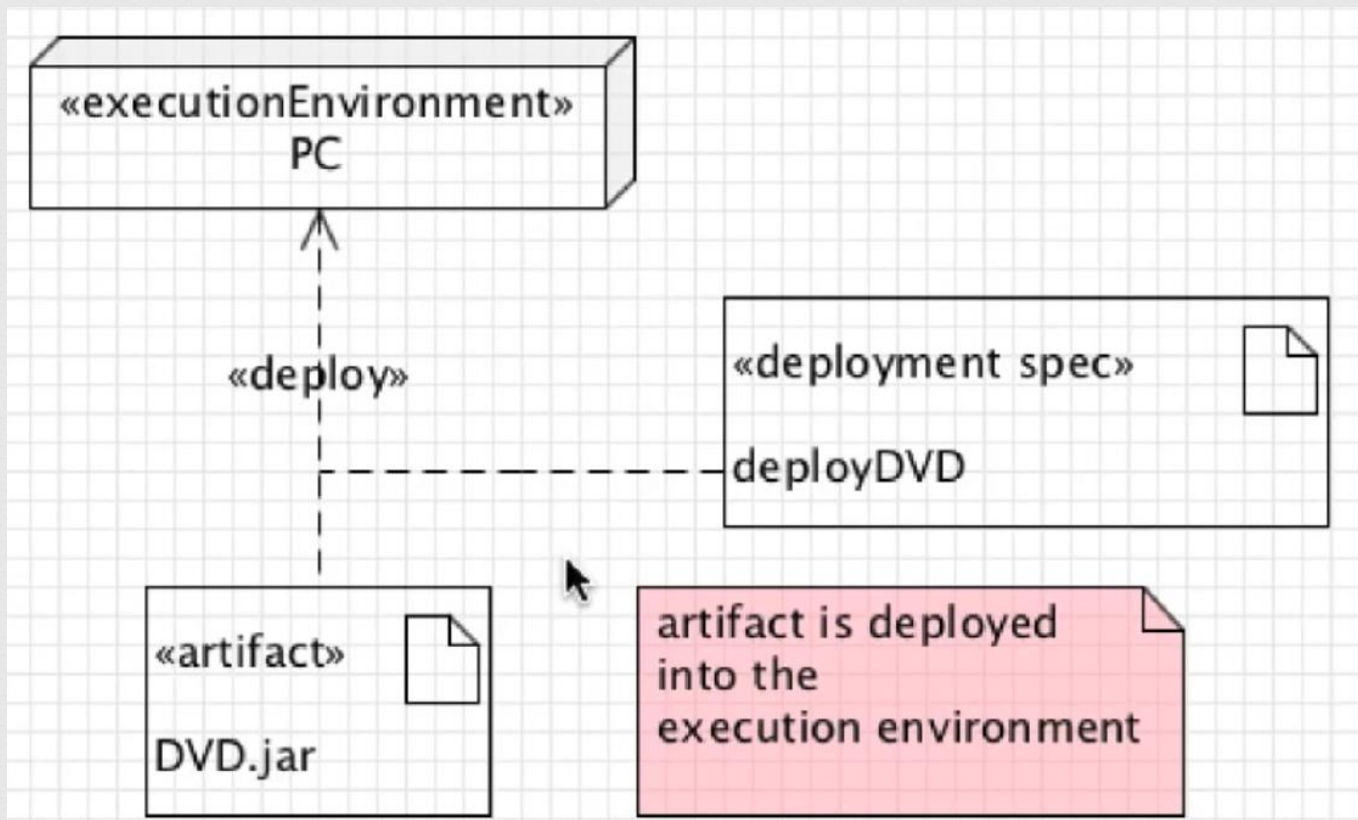


Components define requirements for software deployed to hardware

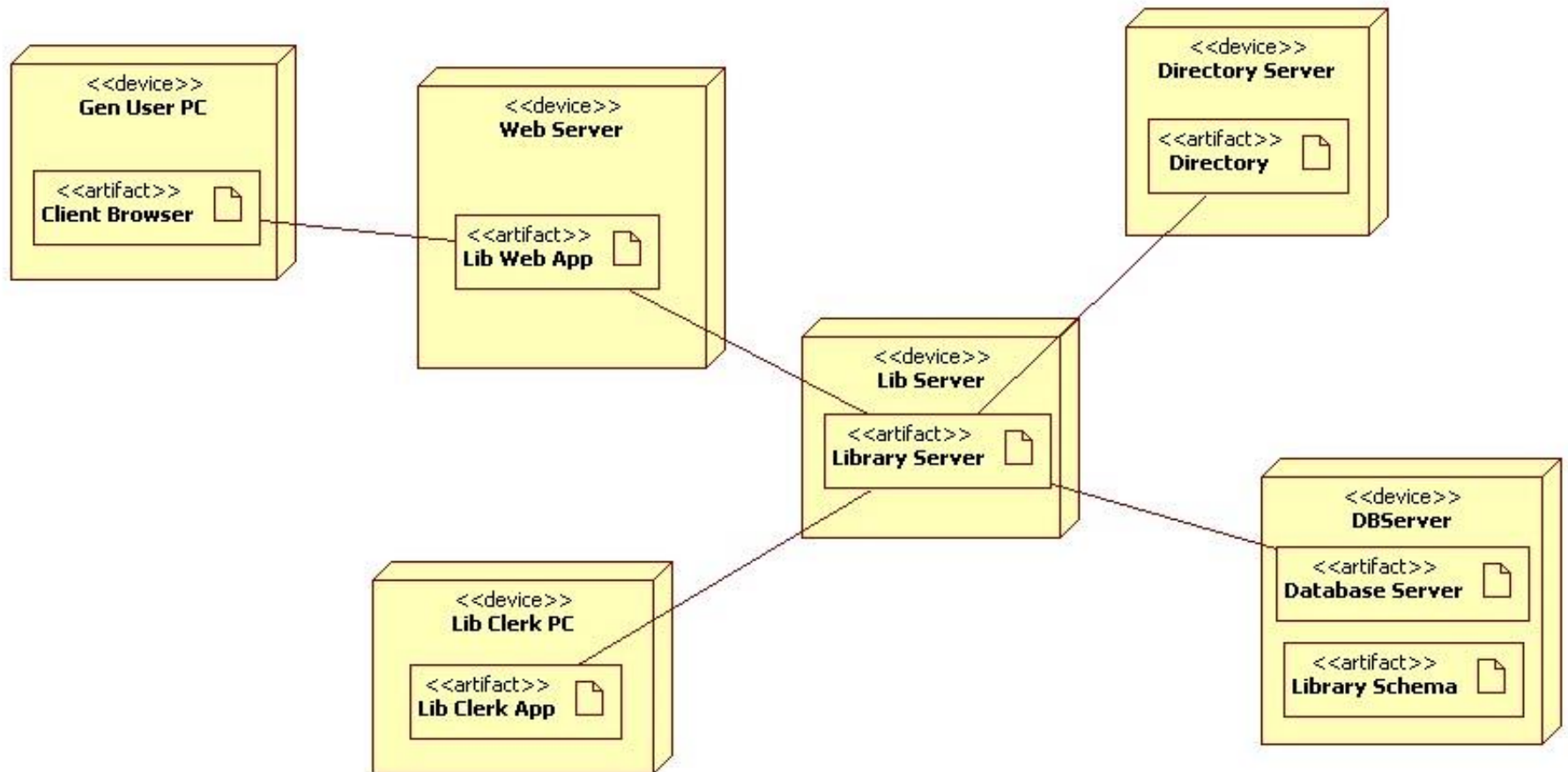
Solution : Deployment Diagram - Sample Example



Solution : Deployment Diagram - Sample Example



Solution – Deployment Diagram for LISM





Thank You !