UNIT- II

2 Static Modeling

Syllabus

Study of classes (analysis level and design level classes). ***Methods for identification of classes :*** *RUP (Rational Unified Process), CRC (Class, Responsibilities and Collaboration), Use of Noun Verb analysis (for identifying entity classes, controller classes and boundary classes).* ***Class Diagram :*** *Relationship between classes, Generalization/Specialization Hierarchy, Composition and Aggregation Hierarchies, Associations Classes, Constraints.*
Object diagram, Package diagram, Component diagram, Composite Structure diagram, Deployment Diagram.

Contents

2.1 Study of Classes

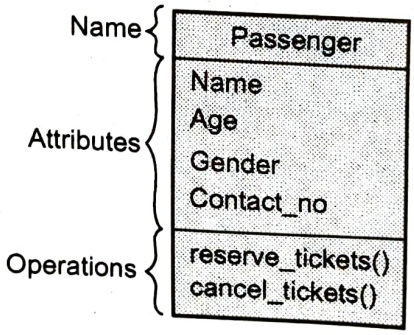
2.1.1 Analysis Classes

Definition : Analysis classes are the classes that i) represent the abstractive entity the problem domain and ii)which maps the real-world business concept.

The analysis class consists of following things -

1. **Name :** Represents the name of the analysis class
2. **Attribute :** Represents the data values for the analysis class.
3. **Operations :** In analysis class , operations represent the high level responsibiliti of the class. Operation parameters and return are also shown to understand t model in detail.

For example - Passenger is analysis class in the Railway Reservation system



2.1.2 Design Classes

Definition : Design classes are defined as the classes that describe some elements problem domain, focus on various aspects of problem from user's point of view.

Goals of Design Classes

The goal of design classes is :

1. To refine the analysis classes by providing the detail design, so that furthe implementation can be done easily.
2. To create new set of classes for implementing the infrastructure of the software.

There are five different types of design classes

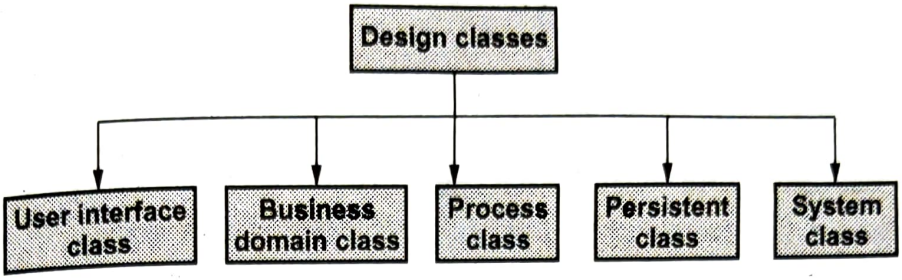


Fig. 2.1.1 Design classes

1. User Interface Class

The user interface class defines all the abstractions that are necessary for Human Computer Interface(HCI). The user interface classes is basically a visual representation of the HCI.

2. Business Domain Class

Business domain classes are the refinement of analysis classes. These classes identify the attributes and services that are needed to implement some elements of business domain.

3. Process Class

Process class implement lower level business abstractions used by the business domain.

4. Persistent Class

These classes represent the data store such as databases which will be retained as it is even after the execution of the software.

5. System Class

These classes are responsible for software management and control functions that are used for system operation.

Characteristics of Design Classes

Each class must be well formed design class. Following are some characteristics of well formed design classes -

1. Complete and Efficient

A design class must be properly encapsulated with corresponding attributes and methods. Design class must contain all those methods that are sufficient to achieve the intent of the class.

2. Primitiveness

Methods associated with one particular class must perform unique service and class should not provide another way to implement the same service.

3. High Cohesion

A cohesive designed class must posses small, focused set of responsibilities and th responsibilities must be associated with all the attributes of that class.

4. Low Coupling

Design classes must be collaborated with manageable number of classes. If one desi class is collaborated with all other design classes then the implementation, testing a maintenance of the system becomes complicated. The law of Demeter suggests that th one particular design class should send messages to the methods of neighboring desig classes only.

2.2 Rational Unified Process

The unified process is a framework for object oriented models. This model is als called as **Rational Unified Process model (RUP)**. It is proposed by Ivar Jacobson, Grad Bootch and James Rumbaugh. This model is iterative and incremental by nature. Let u discuss various phases of unified process.

There are 5 phases of unified process model and those are -

- Inception
- Elaboration
- Construction
- Transition
- Production.

Let us understand each of these phases in detail. (Refer Fig. 2.2.1)

Inception

In this phase there are **two major activities** that are conducted : ***Communication*** and ***planning***. By having customer communication business requirements can be identified. Then a rough architecture of the system is proposed. Using this rough architecture it then becomes easy to make a plan for the project. **Use cases** are created which elaborates the user scenario. Using use cases the sequence of actions can be identified. Thus use cases help to identify the scope of the project which ultimately proves to be the basis for the plan.

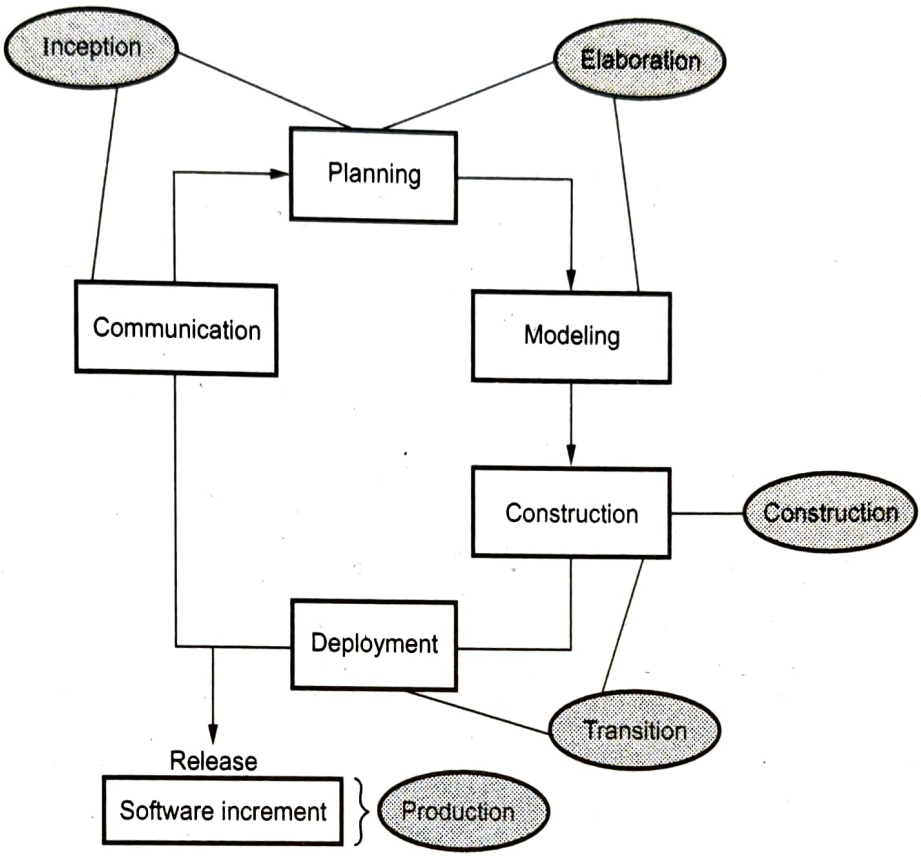


Fig. 2.2.1 Unified process model

Elaboration : Elaboration can be done using **two activities** : ***Planning*** and ***Modelling***. In this phase the use cases are redefined. And an architectural representation is created using five models such as use-case model, analysis model, design model, implementation model and deployment model. Thus **executable baseline** gets created. Then a plan is created carefully to check whether scope, risks and delivery dates are reasonable.

Construction The main activity in this phase is to make the use cases **operational**. The analysis and design activities that are started in elaboration phase are completed in this phase and a **source code** is developed which implements all desired functionalities. Then unit testing is conducted and acceptance testing is carried out on the use cases.

Transition : In the transition phase all the activities that are required at the time of deployment of the software product are carried out. **Beta testing** is conducted when software is delivered to the end user. **User feedback** report is used to remove defects from the created system. Finally software team prepares user manuals, installation guides and trouble shooting procedures. This makes the software more usable at the time of release.

Production : This is the final phase of this model. In this phase mainly the maintenance activities are conducted in order to support the user in operational environment.

Various **work products** that may get generated in every phase are shown Table 2.2.1.

Inception phase	Elaboration phase	Construction phase	Transition phase
• Initial use case model • Initial risk assessment • Project plan	• Use case model • Requirements analysis model • Architecture model • Preliminary design model • Risk list • Iterative project plan • Preliminary user manual	• Design model • Software components • Test plan • Test cases • User manual • Installation manual	• Delivered software increments • Beta test report • User feedback report.

Table 2.2.1

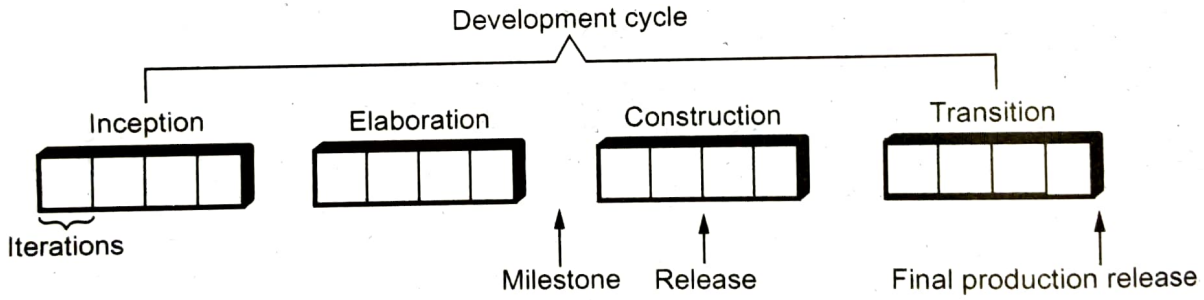


Fig. 2.2.2 Phases of UP

Thus the generic software process models are applied for many years in software development process in order to reduce chaos in the process of development. Each of these models suggest different process flow but they insist on performing the same set of generic framework activities.

2.2.1 UP Disciplines

UP disciplines represent the set of activities in some specific subject area. For instance - UP disciplines can be software design, which represents the set of activities that are required for designing the software system. Fig. 2.2.3 represents the UP discipline. (See Fig. 2.2.3 on next page.)

Following are some important UP disciplines -

1. **Business modeling :** A model or graphical representation is created to represent the concept in application domain.

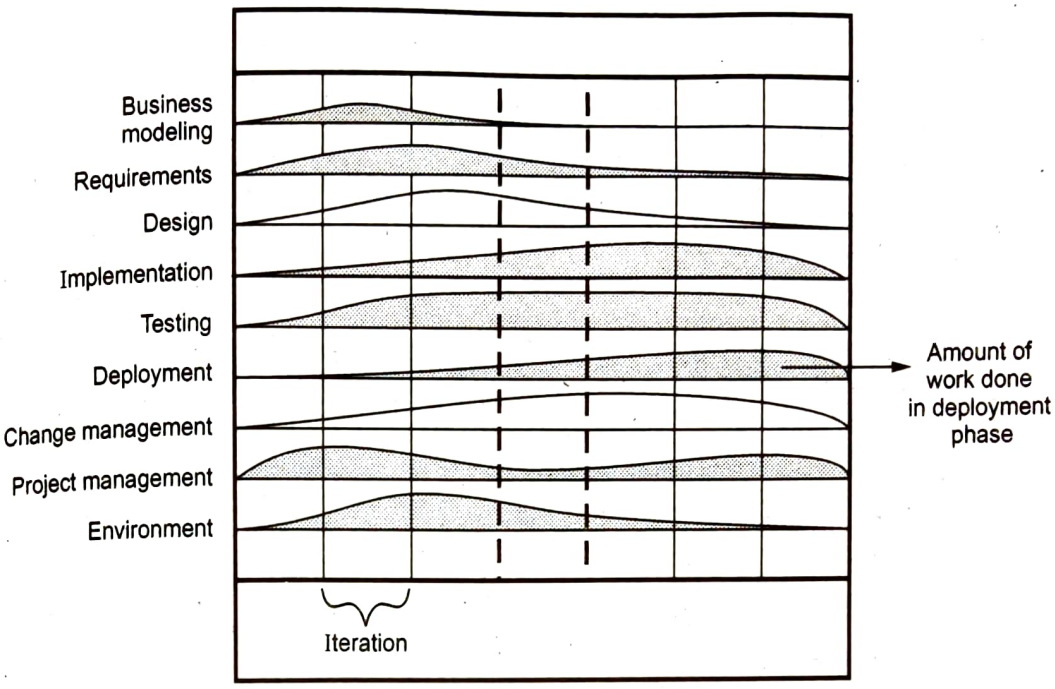


Fig. 2.2.3 UP Disciplines

2. **Requirements analysis and gathering :** The functional and non functional requirements are captured in this discipline by creating some specific type of models. For example use cases can be created to represent the functionality of the system.

3. **Design :** The software objects can be represented using the design model. There are several approaches that can be used to represent the software design.

4. **Implementation :** This discipline contains the source code that is created for implementing the system. After implementation, it is equally important to deploy that code in the targeted system environment.

2.3 Methods for Identification of Classes

There are two methods for finding the analysis classes -

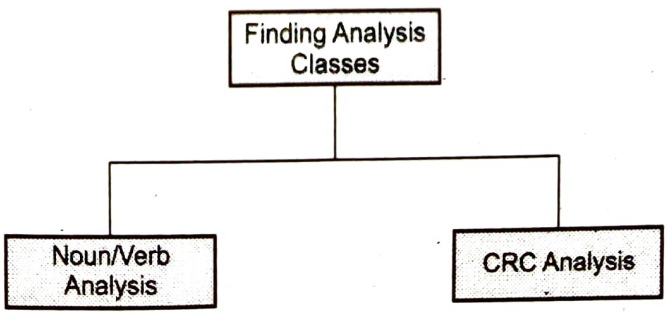


Fig. 2.3.1 Techniques for finding analysis classes

1. Noun Verb Analysis

- This is a very simple technique of finding names, attributes and operations.
- The noun and noun phrases in the problem statement indicate classes or attribu of the classes.
- The verb and verb phrases indicate responsibilities or operations of a class.
- Take care of synonyms and homonyms as these can give rise to spurious classes.

Procedure for Noun/Verb Analysis

Step 1 : Collect the as much relevant information as possible. The sources information can be -

i) Software Requirements Specification(SRS)

ii) Use Cases

iii) Project Glossary

iv) Any other useful documents/manuals

Step 2 : After collecting the information, analyse it and highlight following things -

i) Nouns, ii) Coun phrases,

iii) Verbs, iv) Verb phrases.

The noun and noun phrases indicate classes or class attributes. The verb and v phrases may indicate the responsibilities of classes.

Analysis classes can be identified using the category list. Some commonly u categories are as shown in the following table -

Category	Example
Physical Objects	Ticket, Die, Board, Register, Item, Book.
Place of service	Store, Airport,ATM.
Catalog	ProductCatalog, BookCatalog.
Role	Passanger, Accountant, Student, Customer, Librarian, Administrator.
Financial Items	Credit Card, Cash
Schedules	DailySchedule, RepairSchedule.
Record keeping	Database, Register, Ledger.

Step 3 : Once you have this list of candidate classes, attributes, and responsibilities, you make a tentative allocation of the attributes and responsibilities to the classes. Find out the relationships among these classes.

Step 4 : Thus the first cut class model can be prepared which can be refined by further analysis.

Example : ATM System

"Consider an ATM System in which the customer with an ATM card can establish session with the system. First of all he inserts the card into the system and enters the PIN. The console of the system reads the card and authenticates the customer. The system has console containing display and keyboard.

The session with the system can be accomplished by the transaction. Using the withdrawal transaction the customer can get the requested amount through cash dispenser.

In above problem statement underlined words denote objects of the system. Hence -

1) ATM system is controller object.

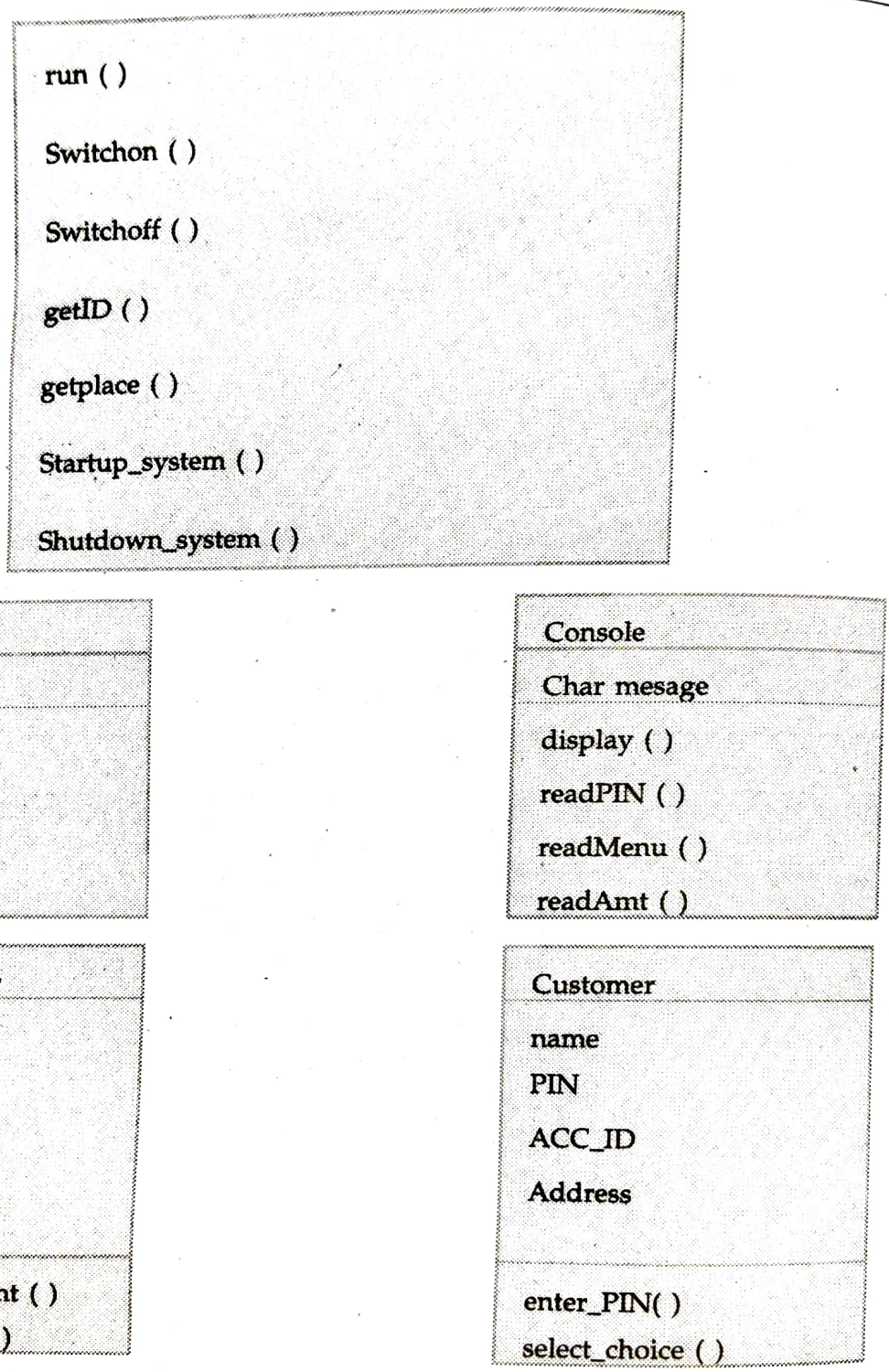
2) Individual components can be - Card reader, Console, Cash dispenser

3) The initiating object is customer.

4) The controlling objects are - Session and Transaction

Various objects can then be as shown below.

ATM System
int id
CharBankname[20]
Char BankAdd[20]
int State
int Switch_ON
int Switch_OFF
int Customer_State



2. CRC Analysis

- The Class-Responsibility-Collaborator modelling is a technique which helps in identifying and organizing the classes that are relevant to the system requirements.
- Ambler has described the CRC modelling as : "The CRC model is a collection of standard index cards. This card is divided into three sections. The first section contains name of the class, type of the class and characteristics of the class. Then the second section is described at the left side and third section is described at the

right side of CRC card. The second section describes the responsibilities and the third section describes the Collaborations.

- The typical structure of CRC card is as shown by following Fig. 2.3.2.

Class Name :	
Class Type :	
Characteristics :	
Responsibilities :	Collaborations :

Fig. 2.3.2 Structure of CRC Index Card

Procedure for CRC Analysis

For complex system analysis, the CRC analysis is used in conjunction with noun/phrase analysis of uses cases, SRS, glossary and other relevant documents.

The CRC analysis is performed in two steps - information gathering and analysis

Step 1 : Brainstorm : Information Gathering phase

1. Arrange a brainstorming session
2. Ask the team members to name the things that operate in business domain.
3. Ask the team members to identify the responsibilities for the **things.** Ask them to note down those responsibilities in the CRC card Compartment for responsibilities.
4. By communicating with other team members, identify the classes that might work together. Record these collaborations in the CRC index card.

Step 2 : Information Analysis Phase

1. The domain experts and OO analysts must take part in this analysis phase.
2. If there are some key **things** that clearly represent the business concepts then those things must surely be the classes.
3. If a note logically seems to be a part of another note, then that entity becomes the attribute.

4. There can be some very less important thing or there can be the things that sho very little interesting behavior, then such things can be eliminated and can not b considered as classes.

5. This model can be refined later.

Example 2.3.1 *Write a CRC for the class 'Email system user'.*

Solution :

Class Name : Email System User	
Class Type : user_id, password	
Characteristics :	
Responsibilities :	**Collaborations :**
Enters the user id and password for logging in.	Authentication sub-system
Reads the mail Sends the mail	Mailing sub-system
View/delete address-book entries.	address book sub-system
Chatting	messenger subsystem

2.4 Class Diagram

SPPU : Aug.-15, Dec.-15, 18, 19, May-18, April-19, March-20, Marks 8

- Class is the most important concept used in object oriented system. It describes th objects that have the common set of attributes, operations, relationships an semantics.
- Graphically graph is represented by a rectangle -

For example :

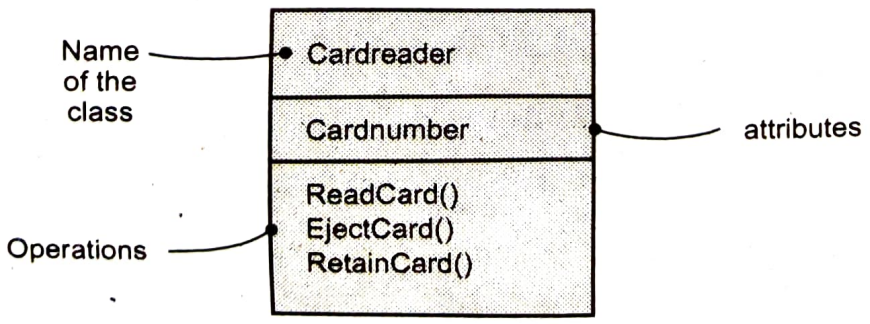
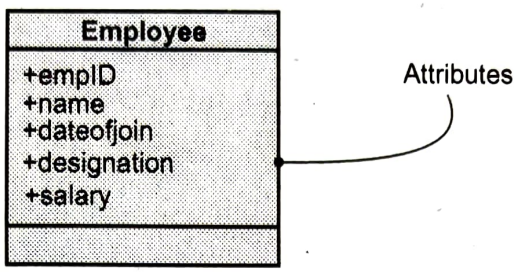


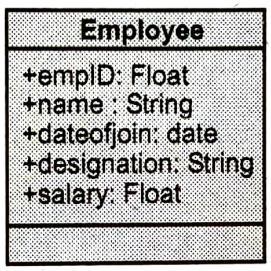
Fig. 2.4.1

Attributes

- Attributes define the **data values** for the class.
- It is an abstraction for the kind of data or the state of the object.
- Normally the **short nouns** or the **noun phrases** represent the attributes.
- Following figure shows the attributes of the class -

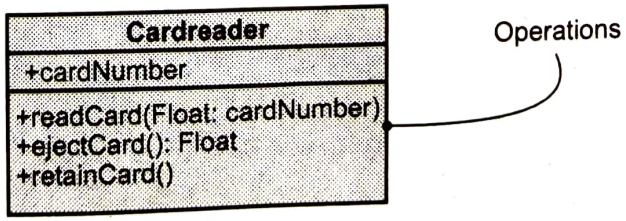


- The attributes can be represented along with the data types. It is as shown below



Operations

- Operation means **implementation** of the service which is demanded by the other objects from the belonging object/class.
- For example : The ***CardReader*** performs the operations such as ejection of card, retaining of the card, reading the card etc.
- The operation names are verbs or verb phrases.
- Typically capitalize the first letter of every word in the operation except the first letter of the word. For instance : ***readCard()***.
- The operation can also be specified by using its signature that includes - Type, default value of parameters, and return type.



2.4.1 Relationships

- Relationships are used to define the collaborations among the classes. In UML ways that connect the things together either logically or physically are modeled relationships.
- In object oriented modeling, there are three kinds of relationships that are used
  1. Dependency
  2. Generalization
  3. Association

Let us discuss them in detail.

1. Dependencies :

The dependency is a relationship in which one thing uses the information and serv of another thing. Graphically dependency can be represented using the dashed direc line. The arrowhead points to the thing being dependent upon. Usually this kind relationship is shown when one class uses the operations of another class. Following an example that shows use of dependency relationship -

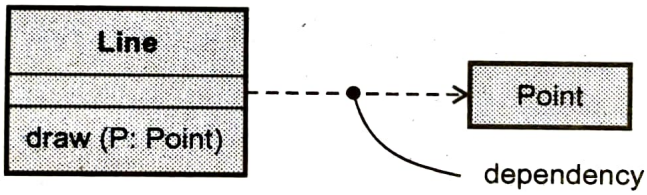


Fig. 2.4.2 Dependency relationship

2. Generalization

A generalization is a relationship between a general thing and a specific thing. T type of relationship is sometimes called as **"is-a-kind-of"** relationship. We u generalization relationship when we want to show parent-child relationships. The wo generalization means the child is substitutable for declaration of the parent. The ch makes use of the attributes and operations defined by parents but it can have its ow additional attributes and operations.

The child overrides the implementation operation of the parent. For example : T parent may define the method named **display** to display some message, whereas t child **Triangle** may use the same method name **display** to display the area of triang This mechanism of overriding the operation is called as **polymorphism.**

Graphically the generalization can be represented by a solid line with a large unfille triangular arrowhead. For example -

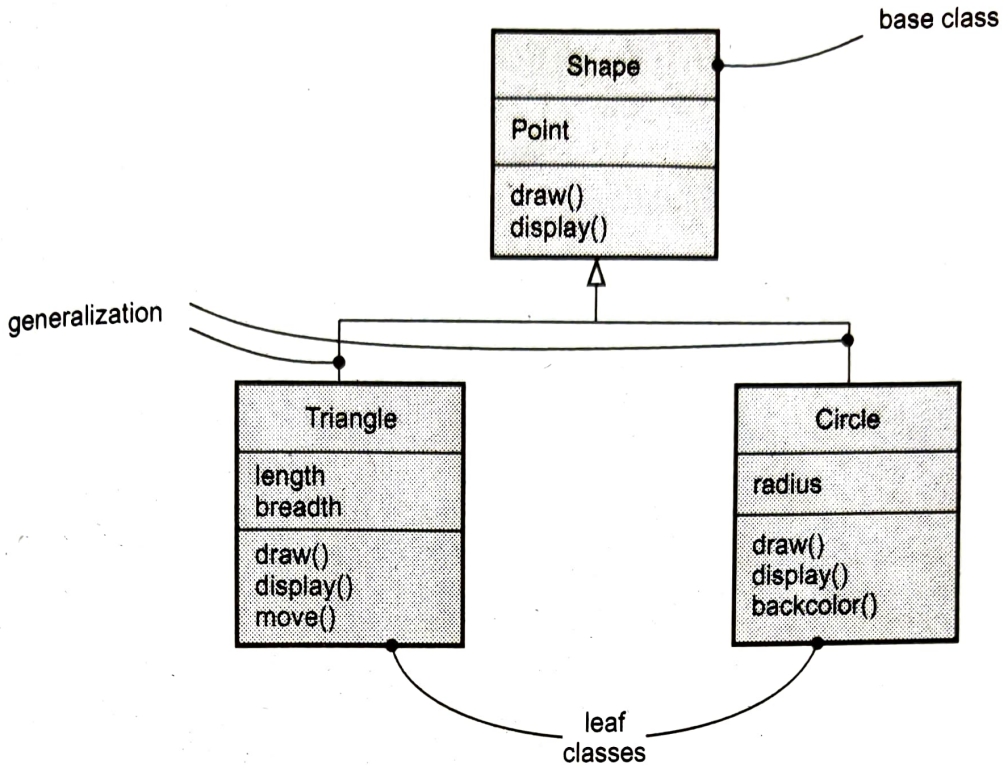


Fig. 2.4.3 Generalization

3. Association

Association is a structural relationship in which one object is get connected to another object. Using association it is possible to connect two objects of the same class. An association that connects the two classes is called **binary association.** However more than two classes can be connected using association relationship. This is called **n-ary association.**

Graphically association can be shown using the solid line connecting same or different classes. For showing the structural relationship the association relationship is used.

There are **four adornments** that are applied to **associations.**

1. Name

The name is used to describe the nature of relationship. One can also specify the direction to the association relationship. For example -

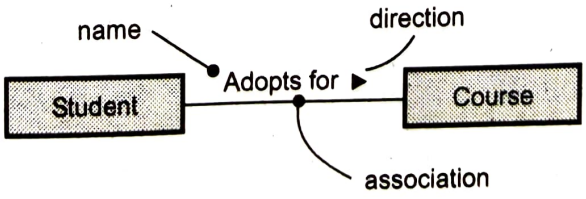


Fig. 2.4.4 Association Names

2. Role

When a class participates in an association then it has some specific role that plays in relationship. The role is specifies at the end of the relation. The roles played by an end of an association is called **end name.** Following Fig. 2.4.5 shows the use of role while specifying the association relations.

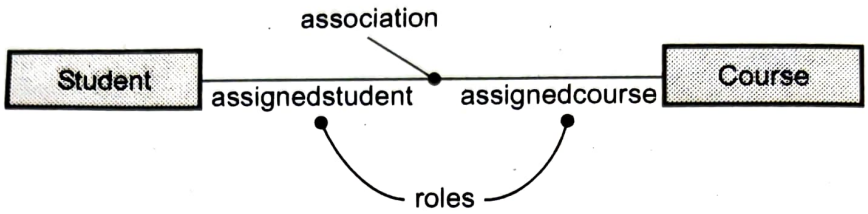


Fig. 2.4.5 Association role names

3. Multiplicity

Multiple objects can be associated with some object. This can be denoted b multiplicity. The multiplicity represents "how many" objects are connected. Th multiplicity is written as an expression with minimum and maximum values. The tw dots can be used to separate out the minimum and maximum values.

Following notations are used to denote the multiplicity of association

Notation	Description
1	Only one instance
0..1	Zero or one instance
*	Many instance
0..*	Zero or many instance
1..*	Zero or many instance
2..10	You can specify the integer range

For example -

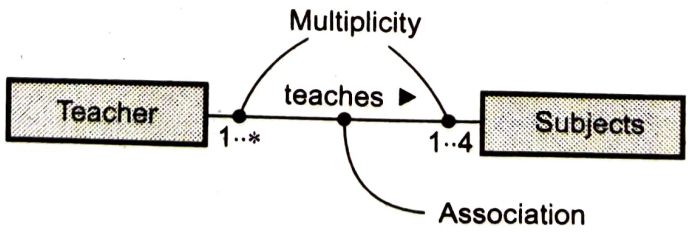


Fig. 2.4.6 Multiplicity

4. Aggregation

Aggregation is a kind of association used for representing the whole-part relationship. In this case, one class represent the larger entity(whole) which consists of smaller things(part). This kind of relationship represents "has-a" relationship. Graphically it is represented as a solid line having unfilled diamond at the whole end. It is denoted as shown in Fig. 2.4.7.

Fig. 2.4.7

For example -

DVD player and car are the two classes that can be associated by the aggregate relationship. Note that the DVD player can exist without car and car can exist without the DVD player. We can read the relationship as "DVD player is a part of car".

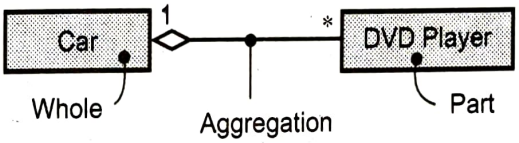


Fig. 2.4.8 Aggregation

5. Composition :

Composition is a special kind of aggregation which represents the whole-part relationship. To be more specific, a restricted aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. It is denoted as follows -

Fig. 2.4.9 Composition Relationship

For example - Steering and car are the two classes those can be associated by composite relationship. This is restricted relationship because the car can not exist without steering.

We can read the relationship as "Steering is a part of car"

Fig. 2.4.10 Example of composition

Difference between aggregation and composition

Both the composition and aggregation represent the **whole part** relationship but the composition is **more restricted** than the aggregation. The composed object can not exist without the other object. Hence it is a strong relationship.

This restriction is not there in aggregation. The existence of contained object is entirely optional in case of aggregation.

For example -

Books library contains books and students. The student and library share the aggregate relationship but the book and library share the composition relationship. As without books the library is not possible.

2.4.2 Commonly Used Notations

2.4.2.1 Notes, Constraints and Methods

The use of **note symbol** is very common in UML but this symbol can be specially used in class diagram to represent the **note** or **comment, constraint** and **method.**

Fig. 2.4.11 Note symbol

Note or Comment The extra information about the element used in the UML diagram can be provided by the Note. This does not have any semantic impact.

Constraints For representing the constraints on the system the Note symbol is used. But note that the constraints must be enclosed within {...}

Method For denoting the implementation of the operation the note symbol is used.

2.4.2.2 Keywords

Keywords are the special words that are associated with some meaning. These are used to categorize the model element. For example and interface in the class diagram can be represented by the keyword **interface** Most of the keywords are shown in guillemet.

<< >>
For example <<actor>>

Examples of using keywords

Keyword	Meaning
<<actor>>	To represent the actor in the class diagram, this keyword is used.

<<interface>>	The interface can be represented using this keyword.
{abstract}	The abstract class can be represented using this keyword. This notation is used in class diagram after the class name or after the operation name.
{ordered}	If the set of objects should appear in some specific order then this keyword must be used at the end of association link in the class diagram.

2.4.2.3 Stereotypes, Profiles and Tags

Stereotypes are used to extend the UML notational elements. A Stereotype provides the capability to create new kind of modeling element.

For example the stateless session can be represented by using the stereotype **<<stateless>>**

Profile is a collection of related stereotypes, tags and constraints that are normally used to specialize the specific domain using UML notations.

The stereotypes declare the set of **tags.** When stereotype is used in the model element then values of its properties are referred as **tagged values.** Tagged values are standard meta attribute values. Example -

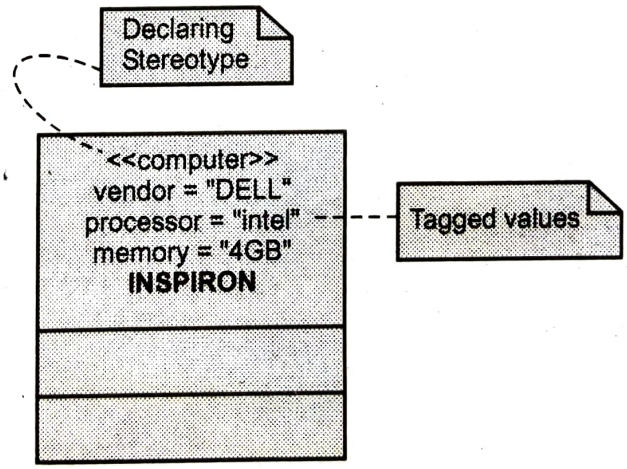


Fig. 2.4.12

2.4.2.4 Abstract Class

Abstract classes are those classes from which another class or an object cannot be instantiated. Abstract class exists only for other classes to inherit from and to support reuse of the features declared by it. The name of abstract class is shown in italics. Sometimes abstract classes and operations can be shown by the keyword {abstract}.

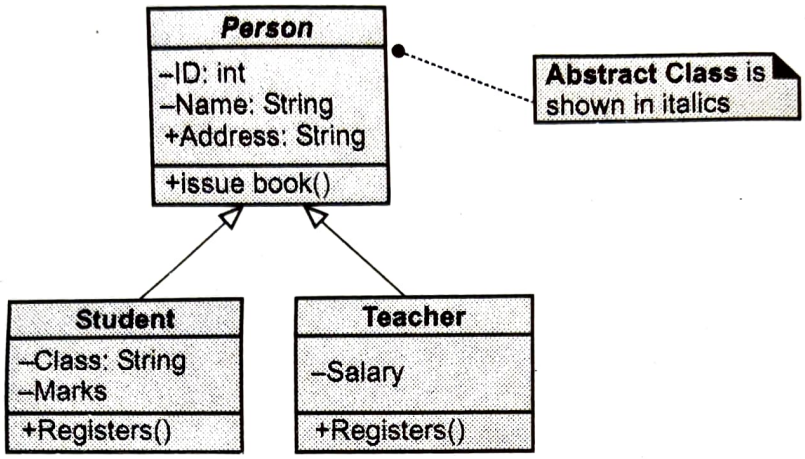


Fig. 2.4.13 Abstract class

The final classes and operations can not be overridden in sub-classes. It can b represented using the {leaf} tag.

2.4.2.5 Interfaces

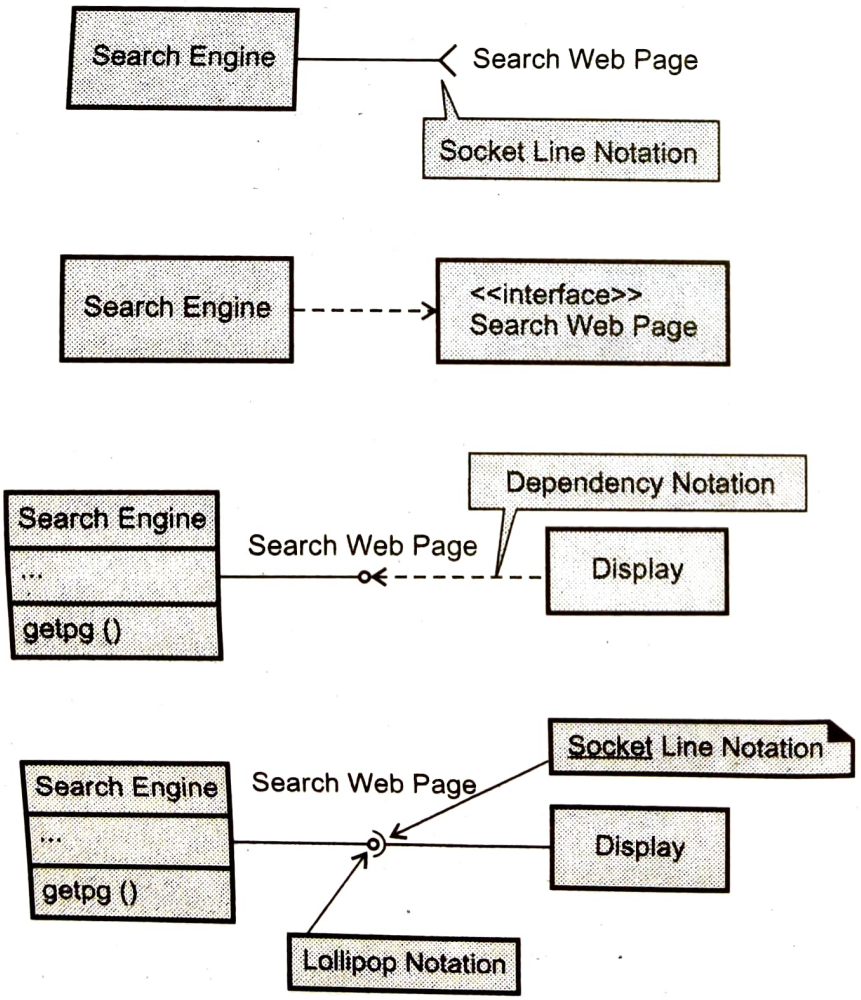


Fig. 2.4.14 Different Notations for interface (Search web page uses Search Engine)

An interface is a classifier that declares of a set of coherent public **features and obligations.**

The **required interface** specifies services that a classifier needs in order to perform its function. An interface specifies a **contract.**

2.4.2.6 Constraints

Constraints are commonly used in class diagram or package diagram. In UML the constraint is a condition or restriction on UML element. Constraint is a **condition** (a Boolean expression) which must evaluate to a Boolean value - true or false. Constraint must be satisfied by a correct design of the system. A constraint is shown as a text string **in curly braces**. The constraints on the operations can be specified by precondition or postcondition. For example -

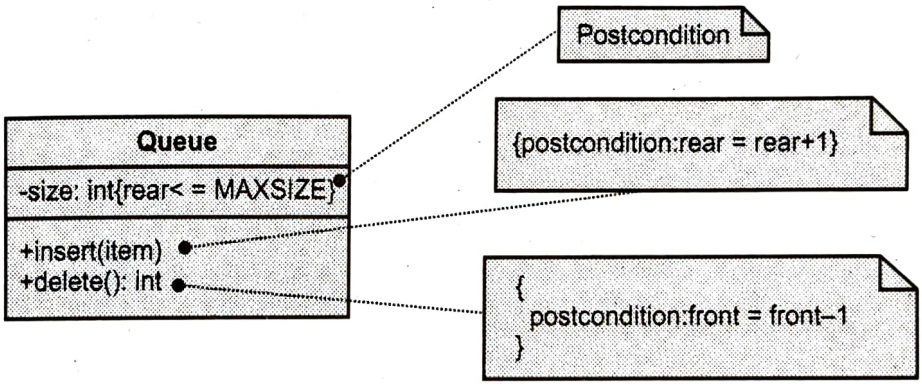


Fig. 2.4.15 Three ways to use constraints

OCL (Object Constraint Language) is a constraint language predefined in UML but UML specification does not restrict languages which could be used to describe constraints. Hence various examples of constraint languages are -

1. Java
2. Machine readable Language
3. Natural Language.

2.4.2.7 Qualified Association

The qualified association has **qualifier** which is used to select particular object from the set of objects. For example For selection of particular **Student** from the **Course** the qualifier will be **StudentID**

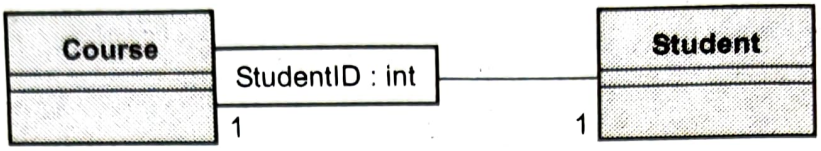


Fig. 2.4.16 Qualified Association

A qualifier is a property which defines **selection key.** UML allows having a qualifie on each end of a single association. The multiplicity can be associated with th association link.

2.4.2.8 Association Class

The abstract class is a class that allows the association to be a class itself. That mean when two classes are related with each other by an association link, then the associatio itself can have attributes and operations. Hence this association can be represented by class. For example- in the following illustration the **Enrollment** is an association clas for the classes **Student** and **Course**

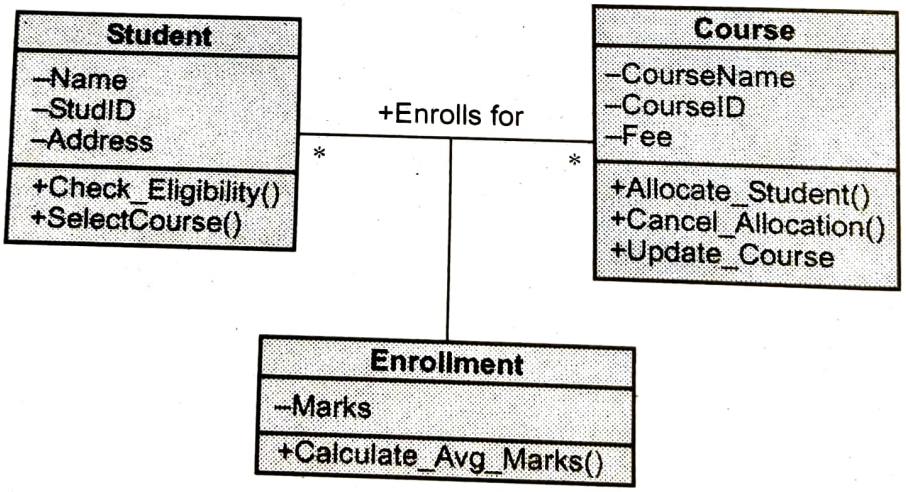


Fig. 2.4.17 Association class

Example 2.4.1 *Draw a class diagram for ATM.*

Solution :

Fig. 2.4.18 Class diagram for ATM system

Example 2.4.2 *Draw a class diagram for online shopping system.*

Solution :

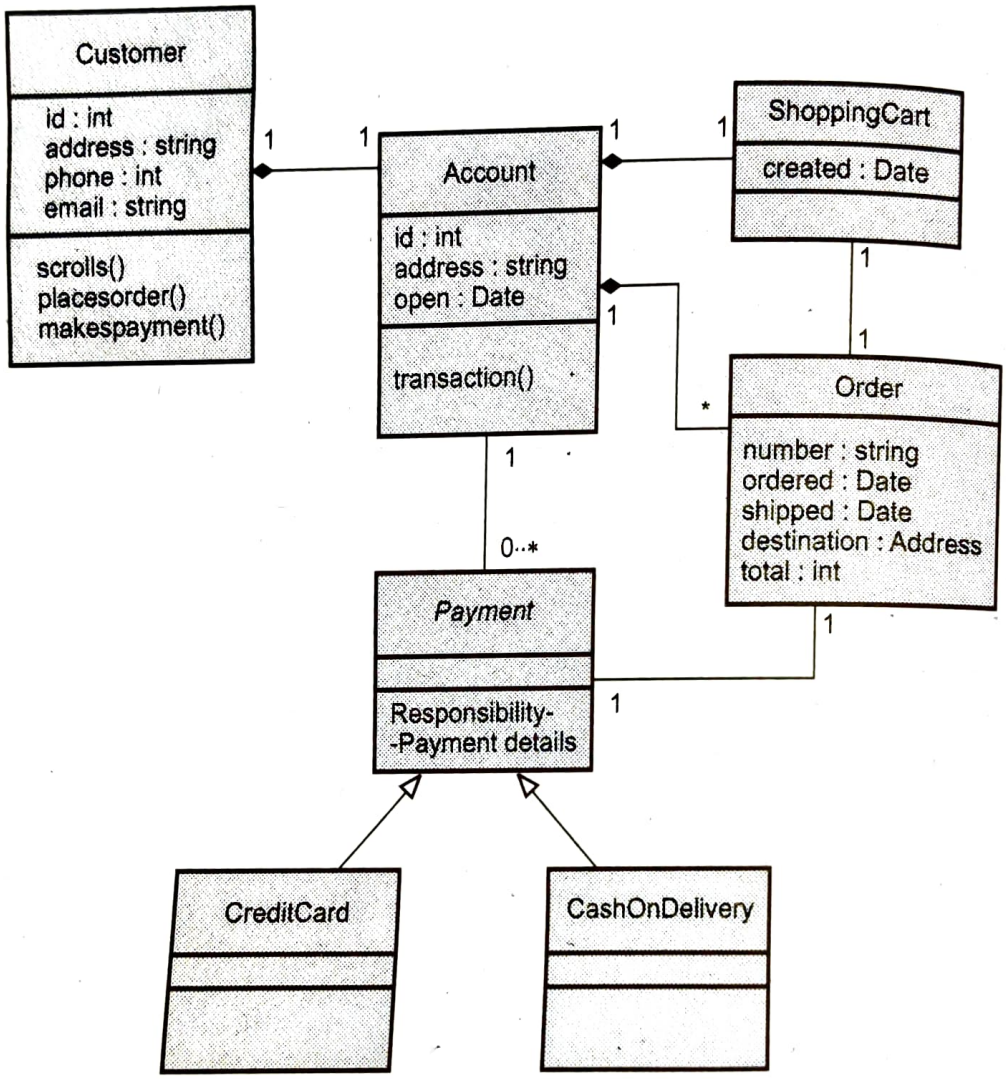
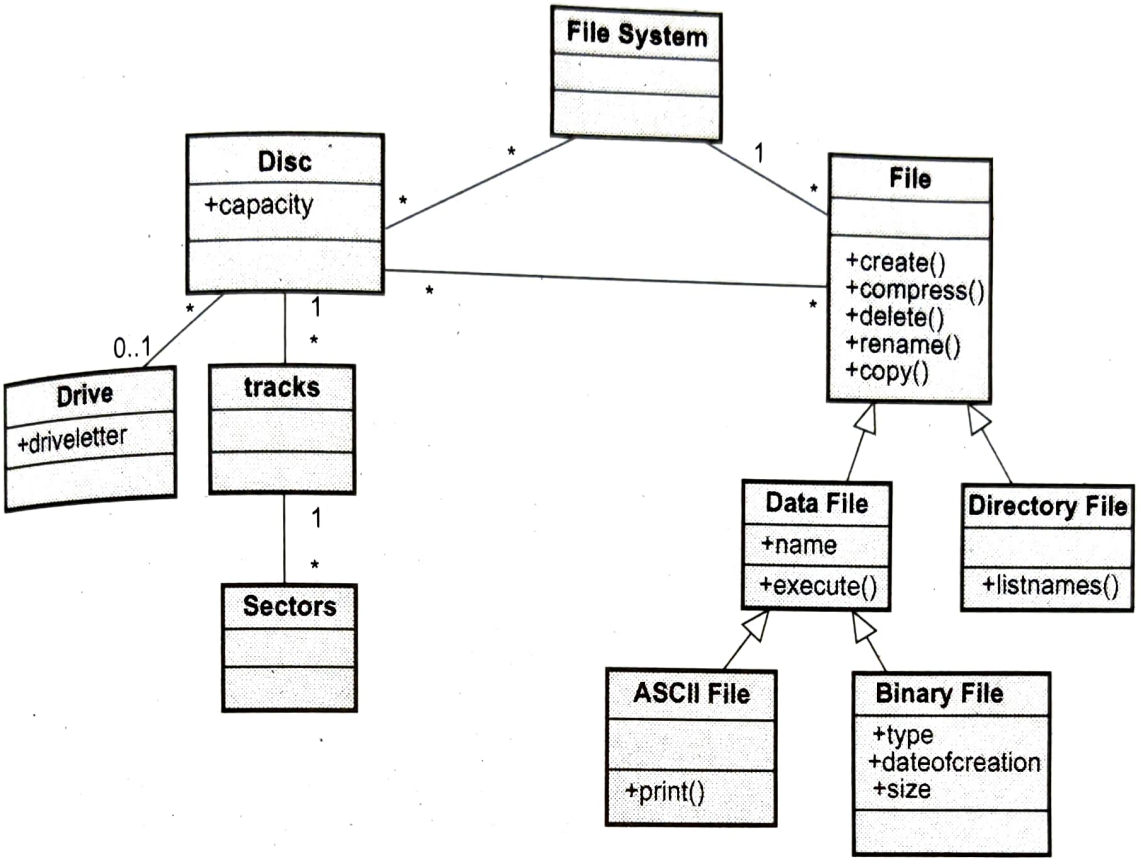


Fig. 2.4.19 Modeling Simple Collaborations for Online Shopping

Example 2.4.3 *Prepare a class diagram for each group of classes. Add at least 10 relationshi (association and generalizations) to each diagram.*

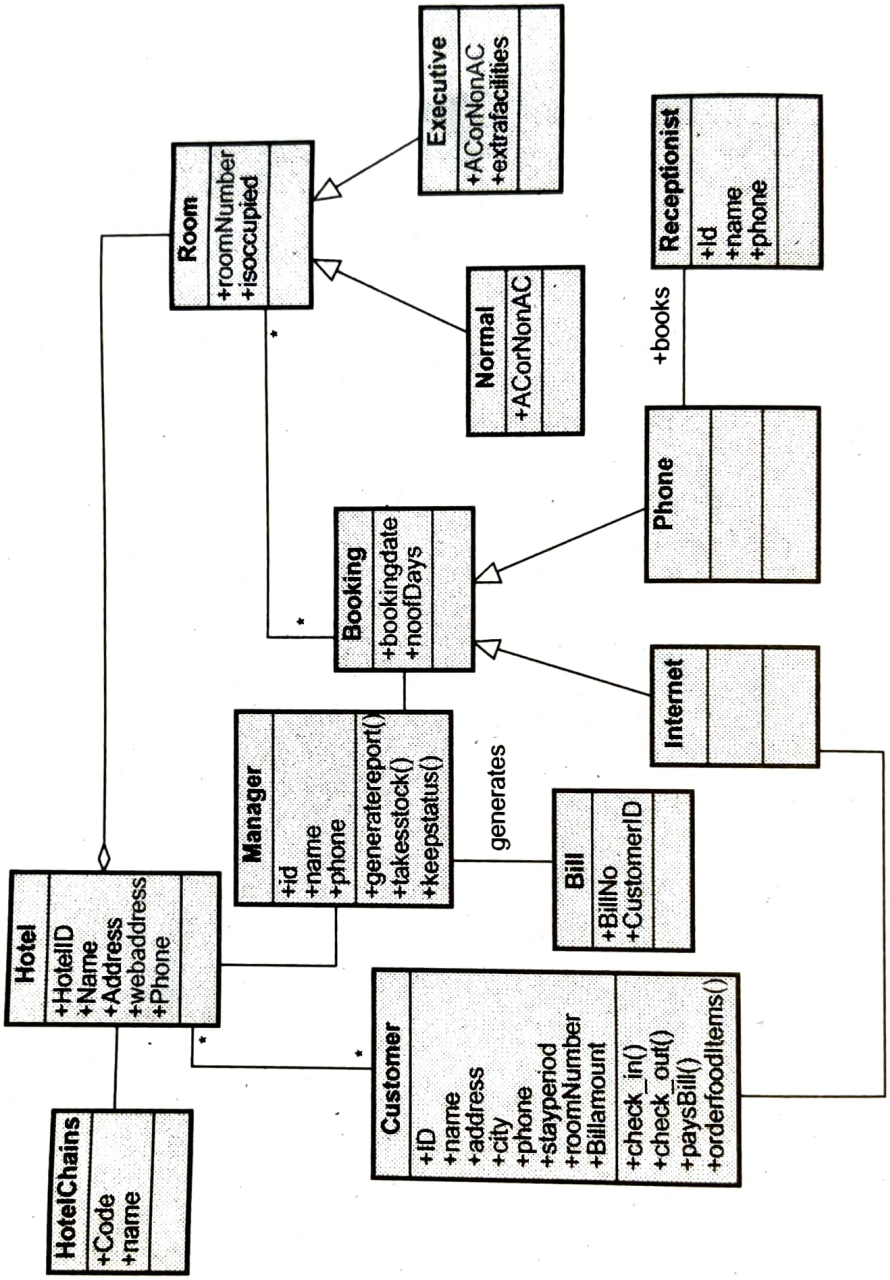
File system, file, ASCII file, binary file, directory file, disc, drive, track, sector

Solution : The required class diagram is given below -



Example 2.4.4 *Prepare a class model for the hotel management system. The system should support chain of hotels. A hotel contains two categories of rooms : executive and normal, both AC and non-AC. The customers of executive rooms can avail extra facilities like games, swimming, food service in rooms, etc. The booking is possible by Internet or by phone. If the booking is through phone, process is done by receptionist and if booking is done through Internet the process is carried out by customer through hotel website. Depending on the number of days customer stays, appropriate bill is generated. The bill also contains amount for transport, food and other facilities enjoyed by the customer along with necessary taxes. The manager should be able to generate reports like list of customers staying in the hotel, list of rooms empty, monthly/yearly income, etc.*

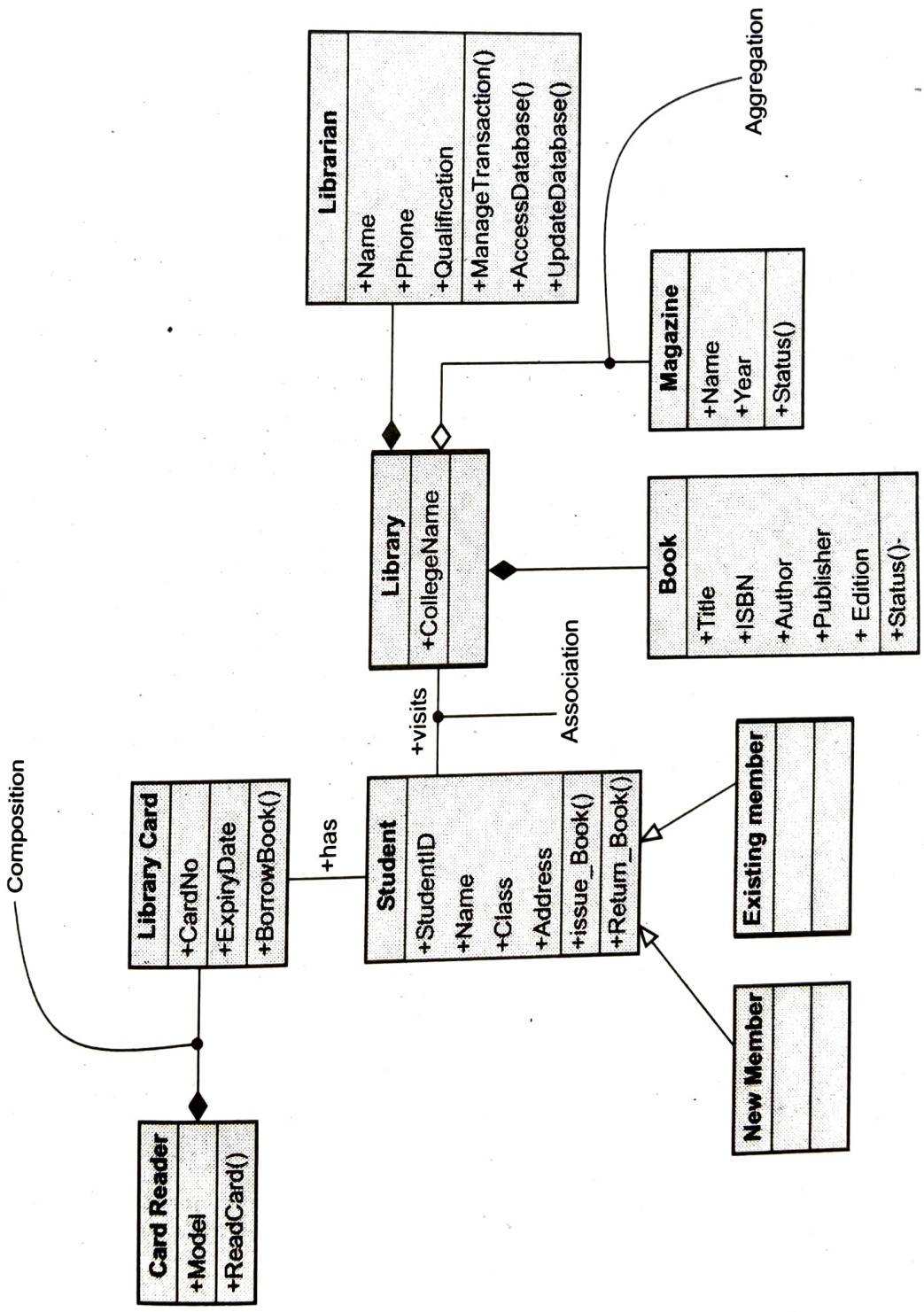
Solution :


Executive
+ACorNonAC
+extrafacilities
Room
+roomNumber
+isoccupied
Receptionist
+id
+name
+phone
Normal
+ACorNonAC
+books
Phone
Booking
+bookingdate
+noofDays
Internet
Manager
+id
+name
+phone
+generatereport()
+takesstock()
+keepstatus()
generates
Bill
+BillNo
+CustomerID
Hotel
+HotelID
+Name
+Address
+webaddress
+Phone
Customer
+ID
+name
+address
+city
+phone
+stayperiod
+roomNumber
+Billamount
+check_in()
+check_out()
+paysBill()
+orderfoodItems()
HotelChains
+Code
+name

Example 2.4.5 *Construct design for Library Information system which comprises and following notations*

(i) Aggregation (ii) Composition (iii) Association

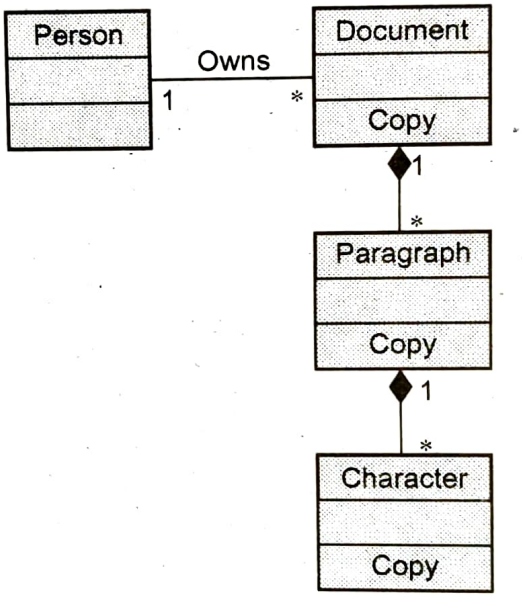
Solution :



Example 2.4.6 *Draw a class diagram for the given description with appropriate relationshi relationship names and multiplicity.*

A person owns multiple documents. Each document is composed of paragraphs that are, turn, composed of characters. The copy operation propagates from document to paragra to characters. Copying a paragraph copies all the characters in it. The operation does propagate in reverse direction; a paragraph can be copied without copying the wh document.

Solution :



Example 2.4.7 *For the description given below, draw the class diagram.*

When a book has isbn number, price, title and one or more authors. When it is bought the library it gets 'purchased' state, when it is added to a catalogue it goes to 'catalogue state. When the cataloguing is complete it goes to 'Available on stack' state. When it borrowed by a member it goes to 'borrowed' state. When it is returned by a member goes to 'available on stack' state.

SPPU : Dec.-15, End Sem, Marks

Solution :

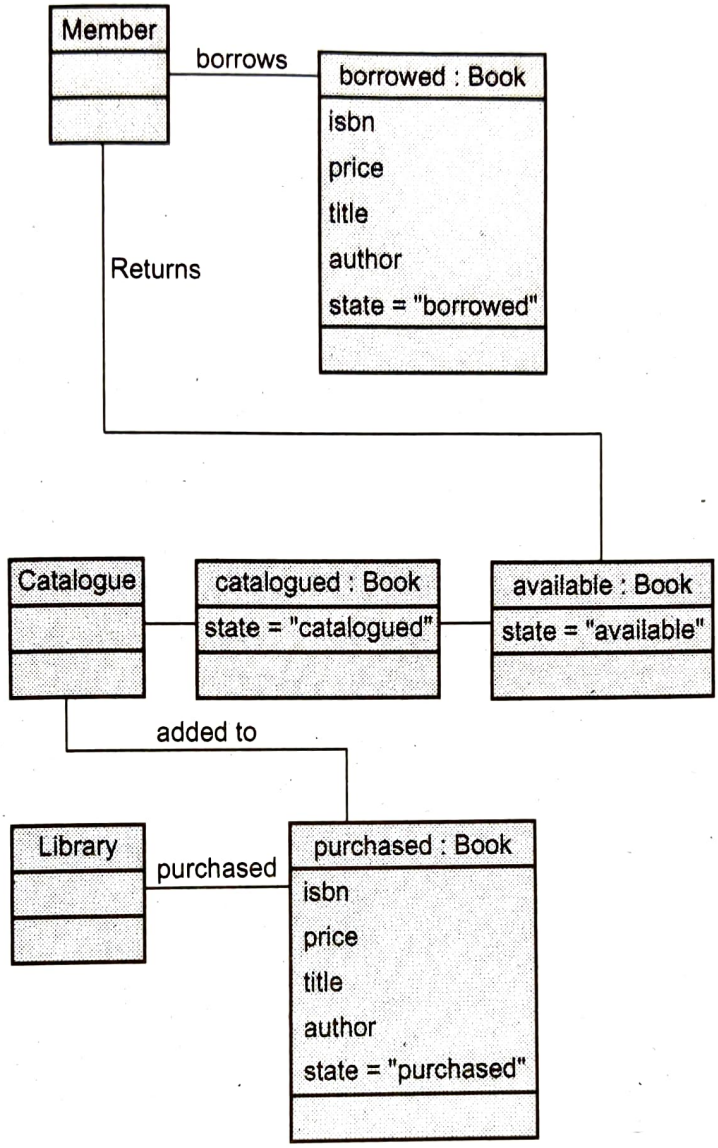


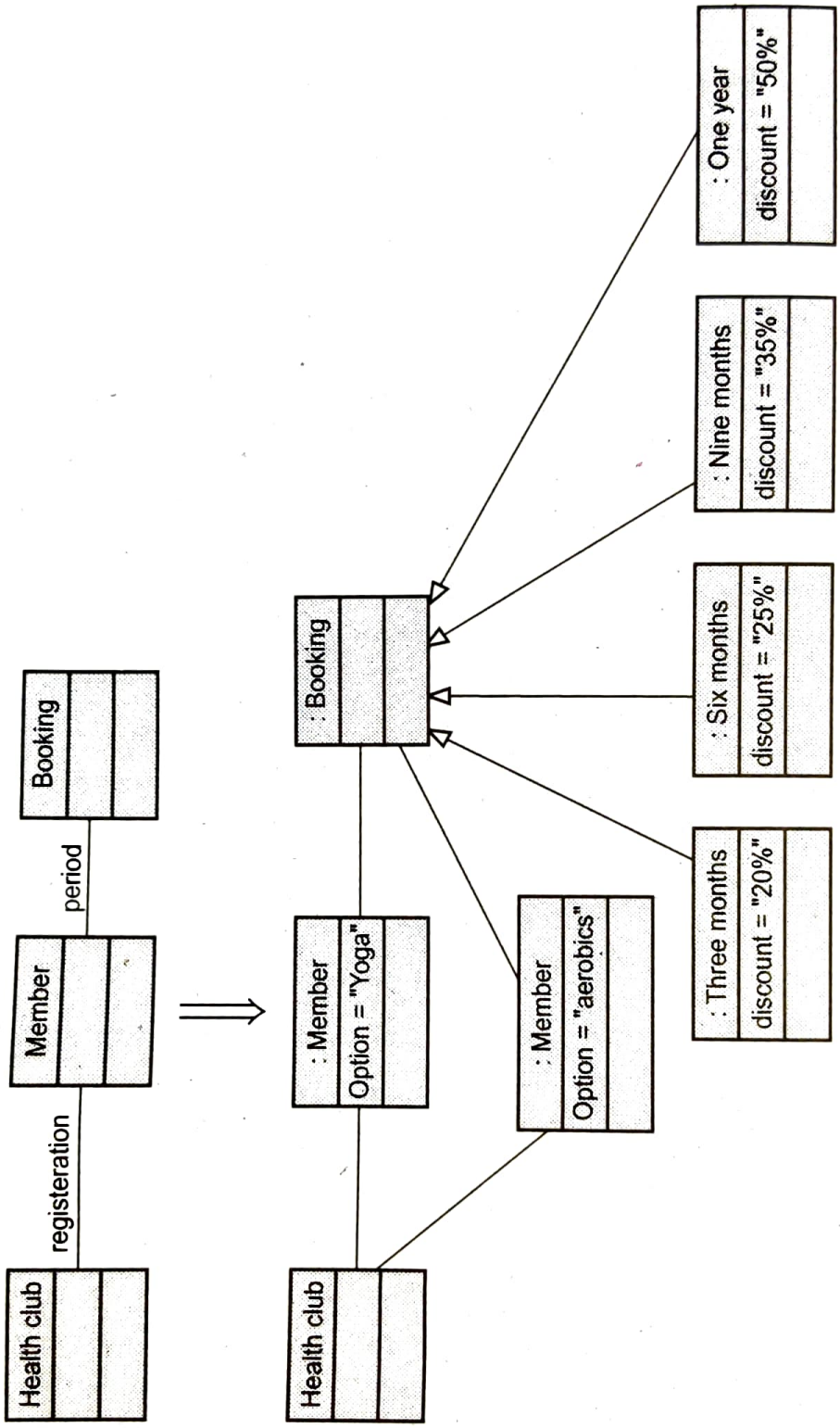
Fig. 2.4.20

Example 2.4.8 *For the description given below, draw the class diagram*

Grand health club offers a scheme for membership of the health club. The options available for registering are 'yoga' and 'aerobics'. The monthly charges for aerobics membership are 2000.00. The monthly charges for yoga membership are 1000.00. The members can avail a single option out of the two options. If a person books for three months, he gets 20 % discount. If he books for six months, he gets 25 percent discount. If he books for nine months, he gets 35 % discount. If he books for one year, he gets 50 % discount.

SPPU : Dec.-15, End Sem, Marks 8

Solution :


Booking
period
Member
registeration
Health club
: Booking
: Member
Option = "Yoga"
: Member
Option = "aerobics"
Health club
: One year
discount = "50%"
: Nine months
discount = "35%"
: Six months
discount = "25%"
: Three months
discount = "20%"

Example 2.4.9 *Convert the following into a class diagram with appropriate classes, relationships, multiplicity.*

A computer program has many statements. An expression is a statement. A function is statement. An expression contains a variable, a constant and operator. A relational operator is an operator. An arithmetic operator is an operator. A function has an argument list, a return type.

SPPU : Aug.-15, In Sem, Marks 3

Solution : Refer Fig. 2.4.21.

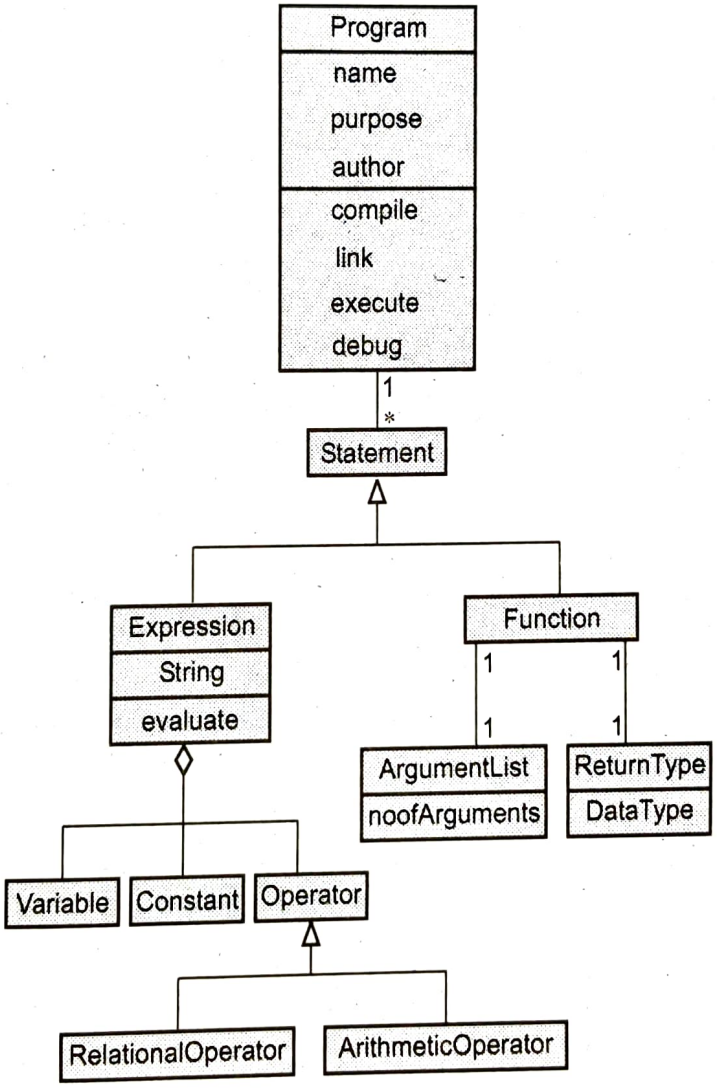


Fig. 2.4.21

Example 2.4.10 *Draw class diagram for following description -*

Car dealer sales cars. A car is owned by an owner. The owner has an address. The owner can be a person, a company or a bank. A car loan may be involved in the purchase of a car. Bank provides the loan.

SPPU : Aug.-15, In Sem, Marks 3

Solution :

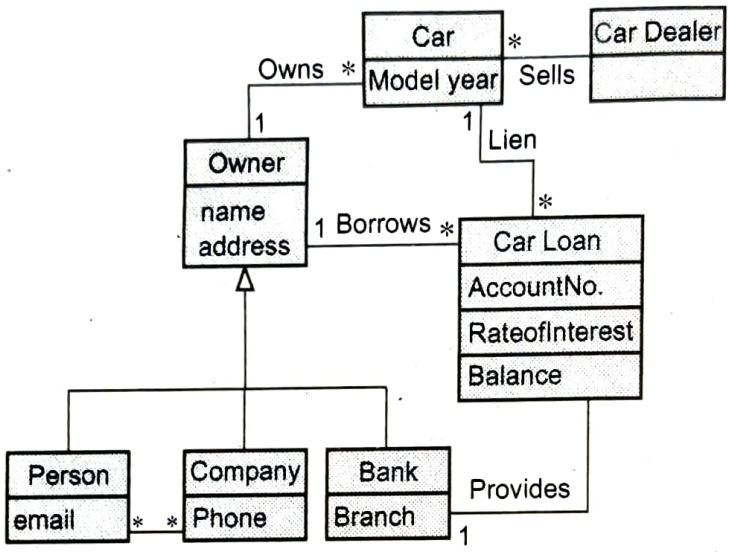


Fig. 2.4.22

Example 2.4.11 *Draw class diagram for following description -*

A drawing object is a text, a geometric object or a group. Group can contain at least two geometric objects. Point, polygon are geometrical objects. A polygon is composed of an ordered set of points. **SPPU : Aug.-15, In Sem, Marks 4**

Solution :

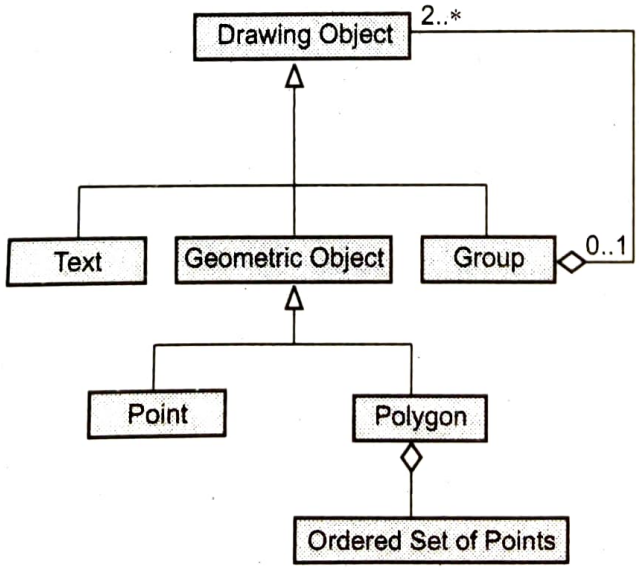


Fig. 2.4.23

Example 2.4.12 *Why class diagram is important in static modeling ? Explain different relationship used in class diagram.* **SPPU : Dec.-18, End Sem, Marks 5**

Solution : Importance of class diagram in static modeling : The class diagram in static modeling serves as a type of static structure diagram. It has following purposes -

1) The class diagram shows the static structure of classifiers in a system
2) The class diagram provides basic notation for other structure diagrams required to design the system using UML
3) The developers can make use of this static model to identify the dynamic behavior of the system and to develop the dynamic model.
4) Business analysts can use class diagrams to model systems from business perspective.

Relationships Used : Refer section 2.4.1.

Review Questions

1. *Explain the elements of a class diagram with an example.*
   SPPU : May-18, End Sem, March-20, In Sem, Marks 5
2. *Explain with an example the difference between aggregation and composition.*
   SPPU : May-18, Dec.-19, End Sem, Marks 5, March 20, In Sem, Marks 5
3. *What are the different degrees of multiplicity in association relationships ?*
   SPPU : April-19, In Sem, Marks 4
4. *State and explain below given concepts.*
   i) Class inheritance *ii) Overriding* *iii) Polymorphism*
   SPPU : April-19, In Sem, Marks 6
5. *Write short note on - Association.*
   SPPU : April-19, In Sem, Marks 5

2.5 Object Diagram

SPPU : Aug.-15, Marks 3

Object diagrams are the diagrams that model the **instances of things** that are present in the class diagram at some particular time.

The object diagrams are used to model the **static design view** of the system.

Graphically the object diagram is represented by vertices and edges.

Difference between Class Diagram and Object Diagram

Sr. No.	Class diagram	Object diagram
1.	Class diagram is a graph of classifier elements connected by various static relationships.	Object diagram is a graph of instances including objects and data values.

2.	It contains interfaces packages, relationships and even instances.	The use of object diagram is limited and it mainly represents data structures.
3.	Class diagram shows classes and their relationships.	The object diagram shows interaction between objects.

2.5.1 Contents

- Object diagram contains following things - 1) Objects 2) Links
- Object diagram also contains the notes and constraints to add more semantic the modeling.
- Sometimes the classes of the instances can be placed in the object diagram fo knowing the class behind the instance.

2.5.2 Uses of Object Diagram

- The object diagram is used to model the static design view or the static proces view of the system from the perspective of real world instances.
- For finding out the functional requirements of the system the object diagram i drawn.
- The object diagram is also used to model the static data structure.
- Using the static interaction view or static design view the object structure of the system is represented.
- Object diagram is used to visualize, specify, construct, and document the existence of certain instances and their relationships with other instances of the system.
- The object diagram is used to show the dynamic behaviour of the system.

2.5.3 Modeling Object Diagram

- Following steps are followed to model the object structure -

Step 1 : Identify some function or behavior of the system for modeling the interaction of the classes, interfaces and other things.

Step 2 : Create the collaboration to describe this behavior.

Step 3 : For each of this behavior, identify the classes, interfaces and other elements that participate in this collaboration. Identify the relationship among these elements.

Step 4 : Consider one scenario from this functionality, freeze that scenario at a moment of time. Represent the objects that are participating in this scenario.

Step 5 : Represent the attributes and state of these participating objects to understand the scenario.

Step 6 : Represent the links among these objects for representing the associations.

- **Example** - Following figure shows the object diagram for hospital management system which models the scenario that the tests/operations are conducted on the patient.

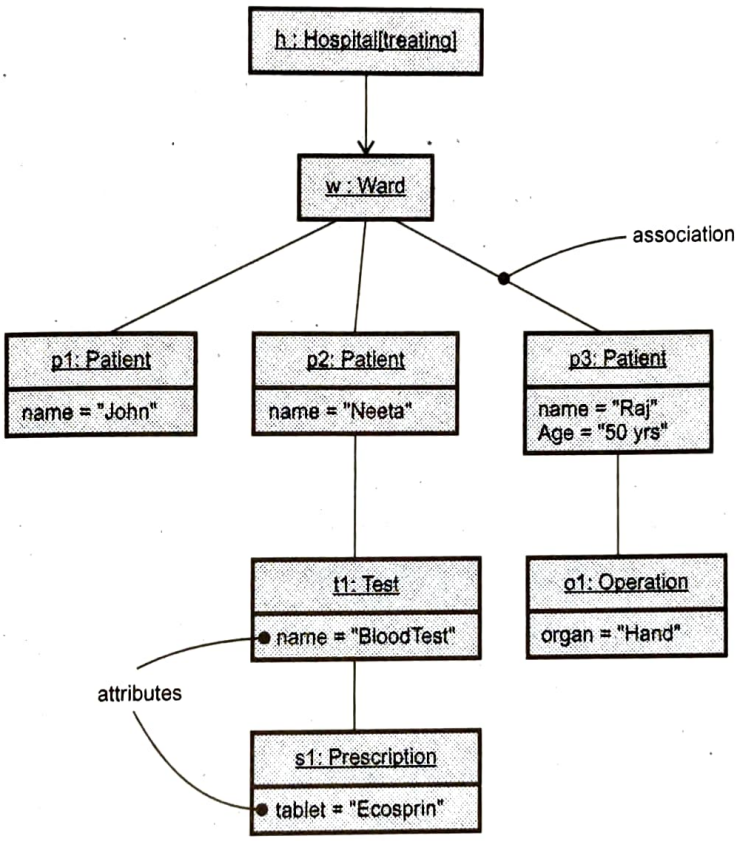


Fig. 2.5.1 Modeling Object Structure

In above Fig. 2.5.1 there are various objects participating in one scenario of the system. The patient gets treatment in the hospital and this scenario is captured by the figure.

The h is an instance of the class hospital, similarly p1,p2 and p3 are the instances of the class patient. The attributes are specified as quoted strings to add the information about the object diagram.

2.5.4 Examples

Example 2.5.1 *Draw a class and object diagram for company information system.*

Solution : (See Fig. 2.5.2 on next page.)

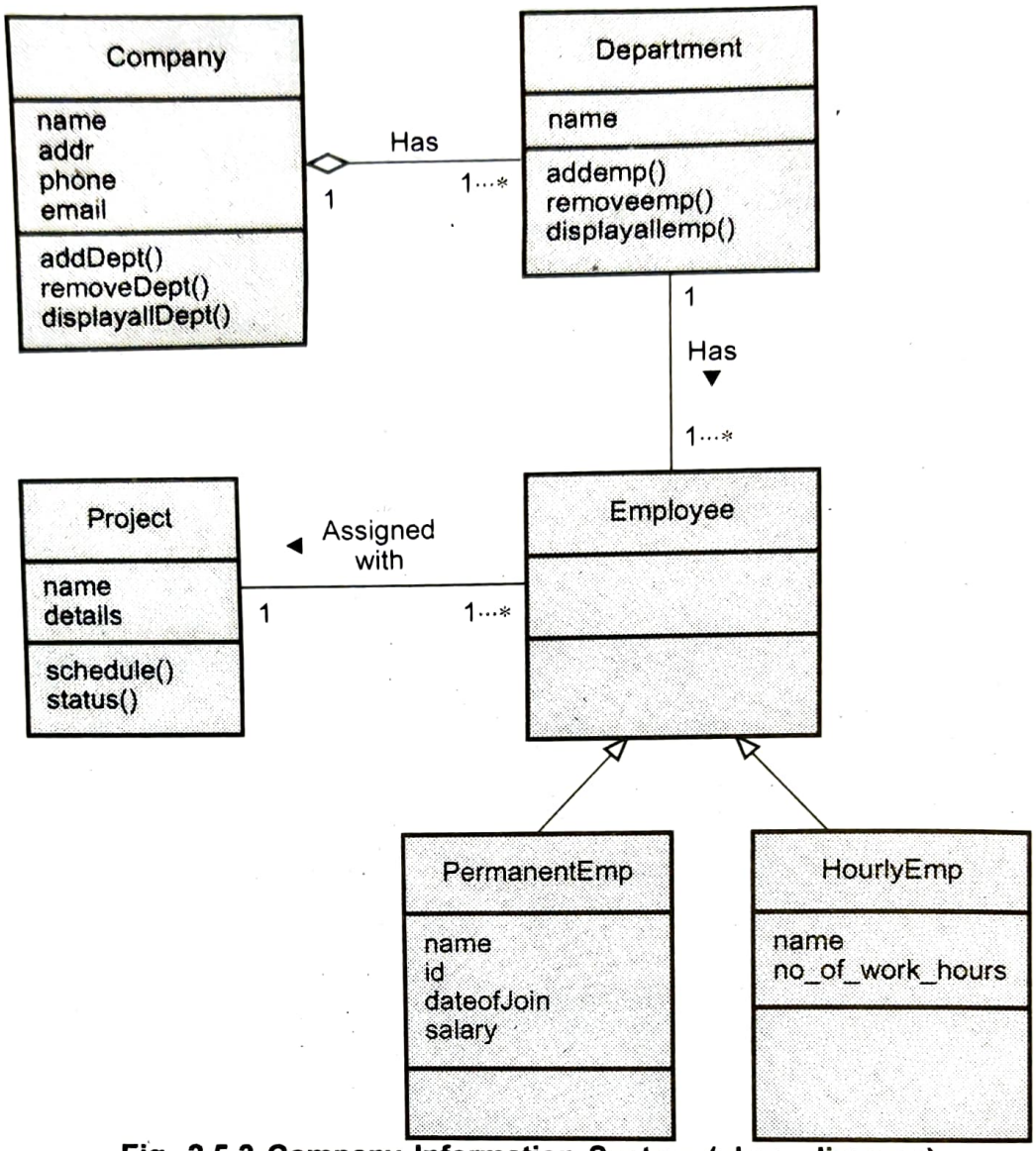


Fig. 2.5.2 Company Information System (class diagram)

The object diagram is as shown below-

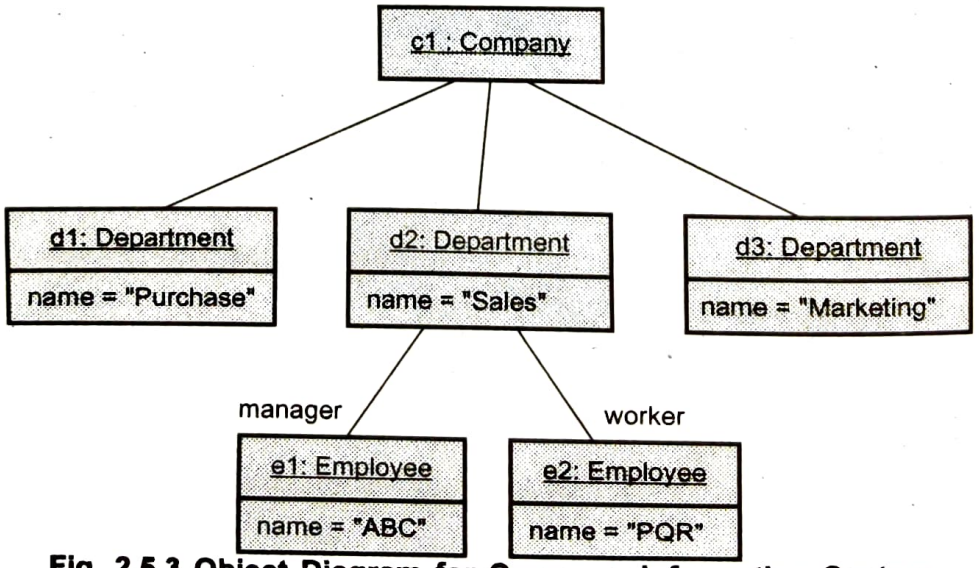


Fig. 2.5.3 Object Diagram for Company Information System

Example 2.5.2 *Design a class diagram for school information system. Specify clearly the classes, relationships among the classes, attributes, operations in each class.*

Solution :

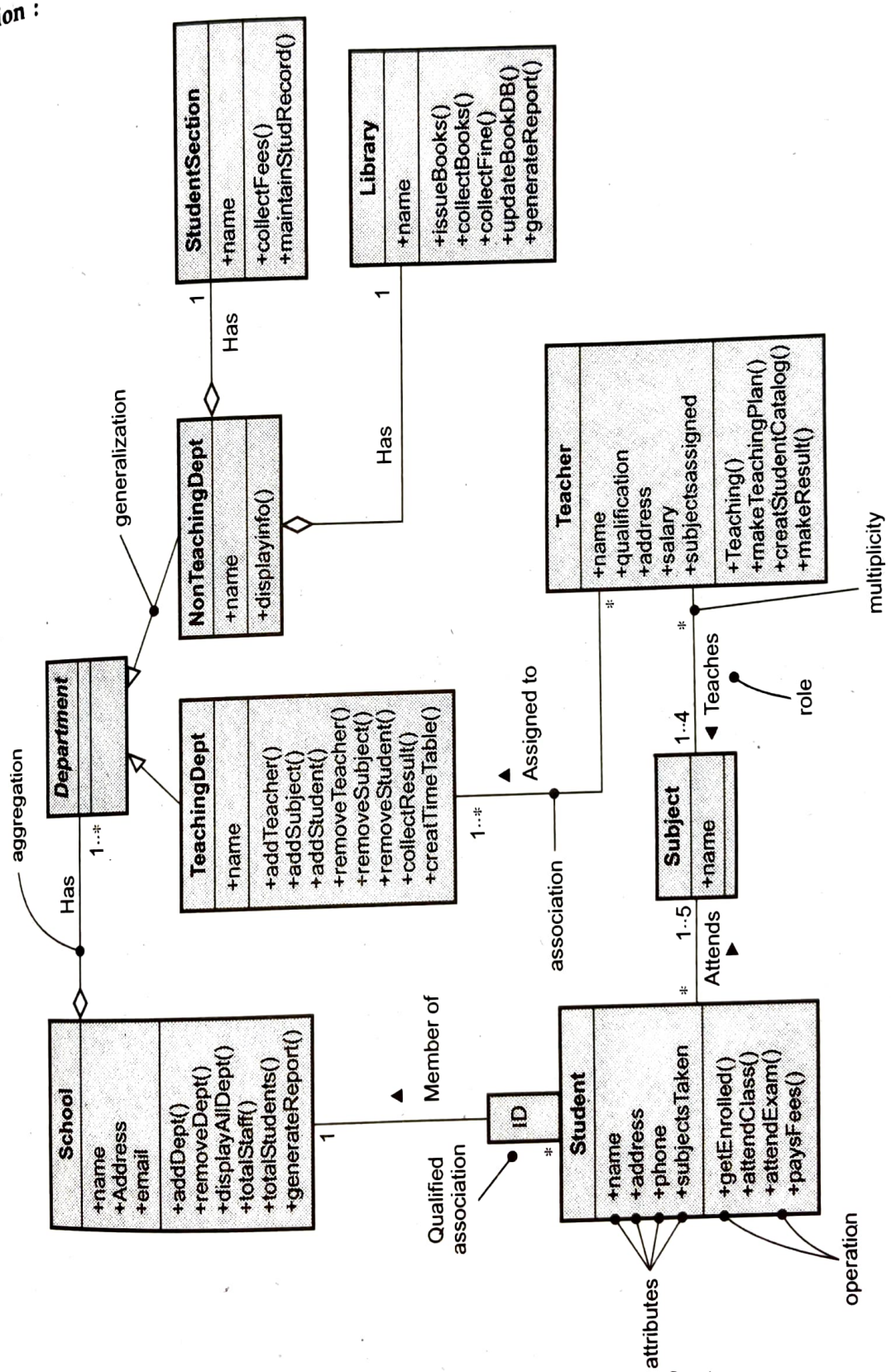


Fig. 2.5.4 Class Diagram for School Information System

Example 2.5.3 *How many object diagrams are possible from a class diagram ? Justify your answer with a suitable example.*

Solution : Any number of object diagrams are possible for a class diagram, becaus object diagram is drawn based on some scenario or behavior of the class diagram an there could be any number of instances running at a given time for defining som functionality. Following are two different object diagrams drawn for the class diagram **hospital management system.**

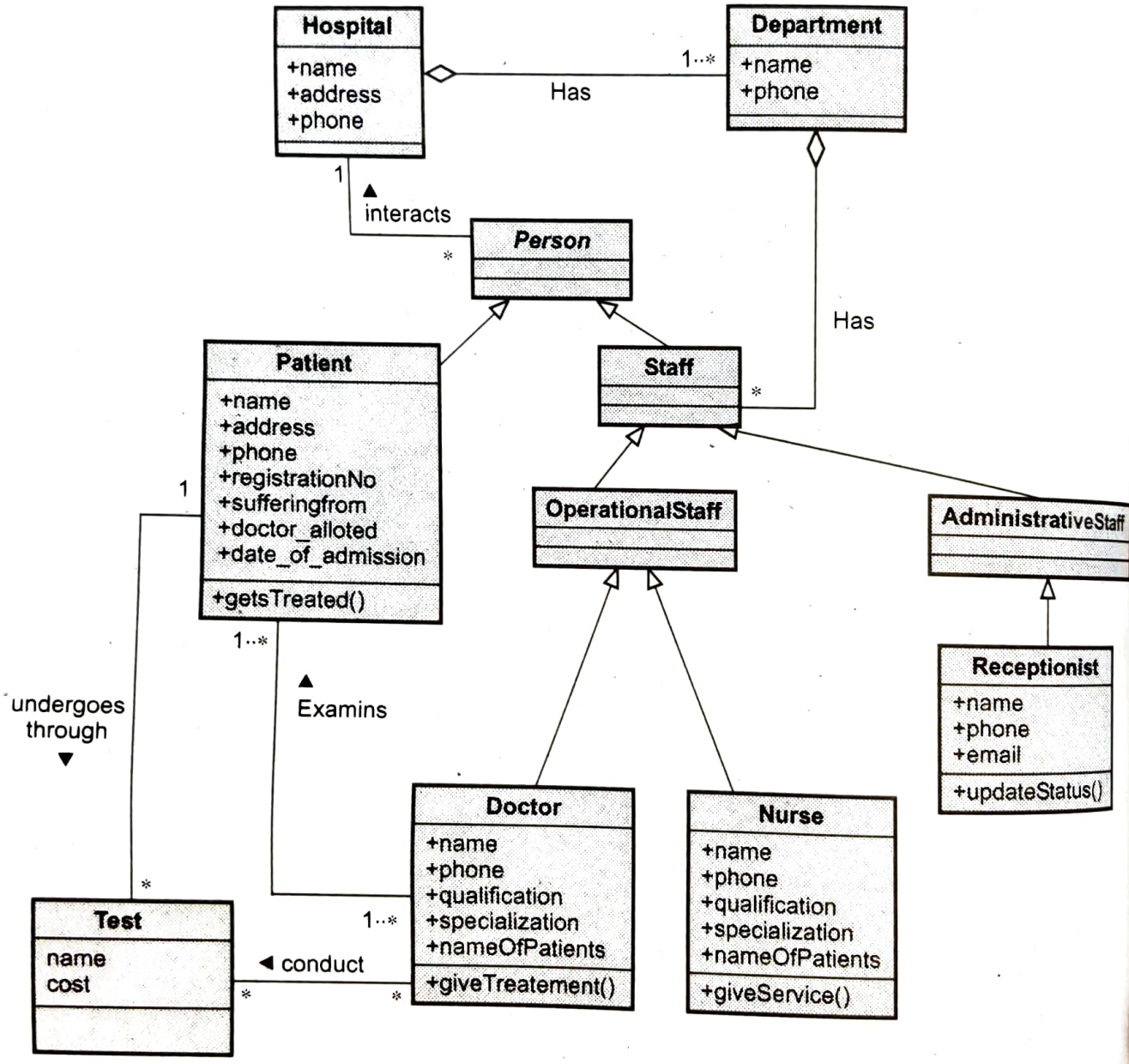


Fig. 2.5.5 Class Diagram for Hospital management system

We have drawn two object diagrams for the above class diagram.

Following are two object diagrams that focus two different mechanisms of instances of the **hospital management system**

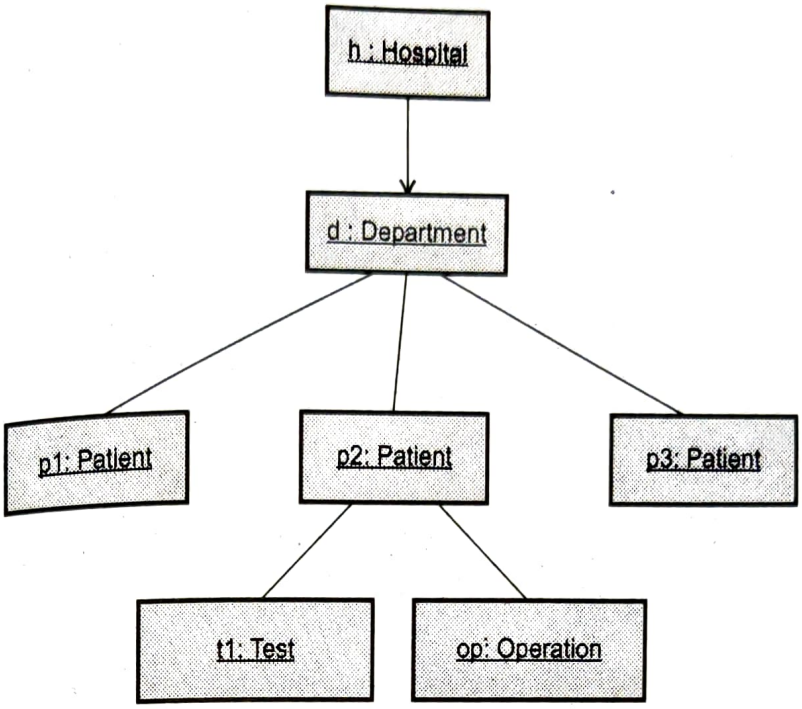


Fig. 2.5.6 Object Diagram1 for Hospital Management System

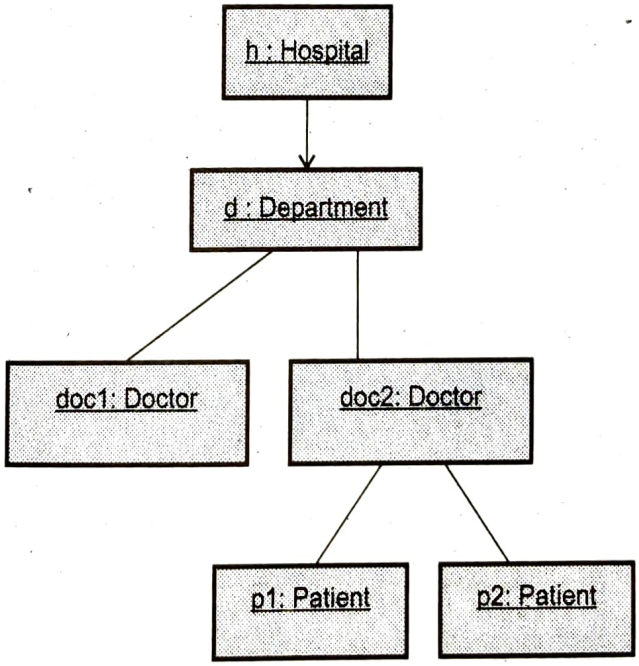


Fig. 2.5.7 Object Diagram2 for Hospital Management System

Example 2.5.4 *Draw a sample object diagram for railway reservation system.*

Soution : It is easy to draw the object diagram for a given class diagram, hence we will first draw the class diagram for railway reservation system. (See Fig. 2.5.8 on next page.)

The object diagram for the above class diagram is as follows - (See Fig. 2.5.9 on next page.)

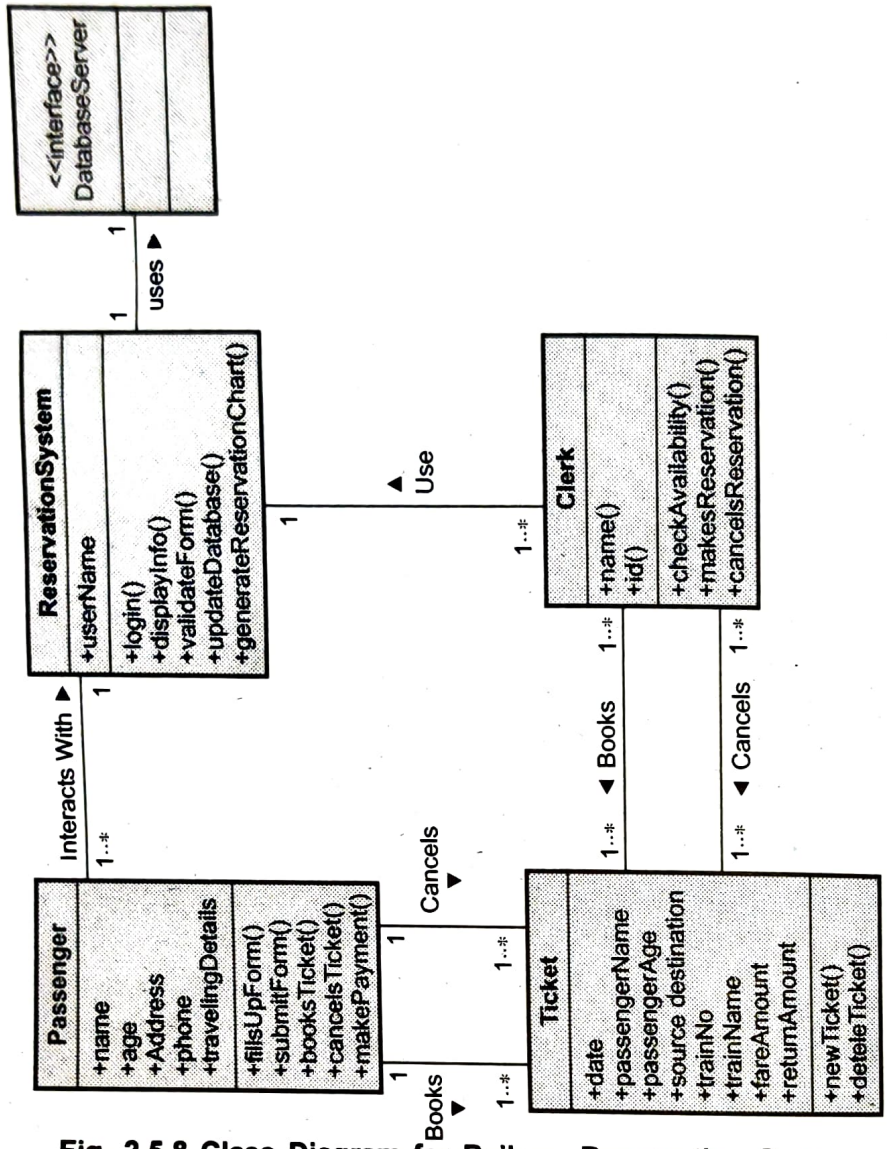


Fig. 2.5.8 Class Diagram for Railway Reservation System

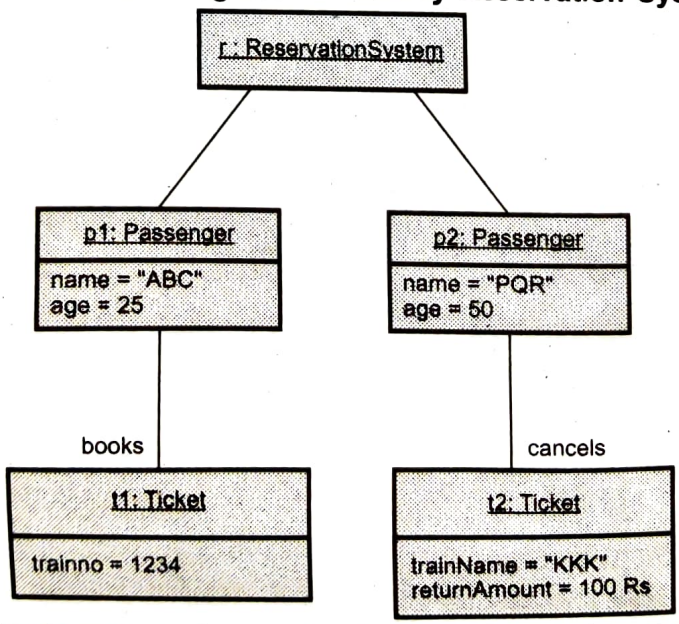


Fig. 2.5.9 Object Diagram for Railway Reservation System (Making/cancellation of reservation mechanisms)

Example 2.5.5 *Draw the possible class diagram of university automation system and also write corresponding object diagram of it.*

Solution :

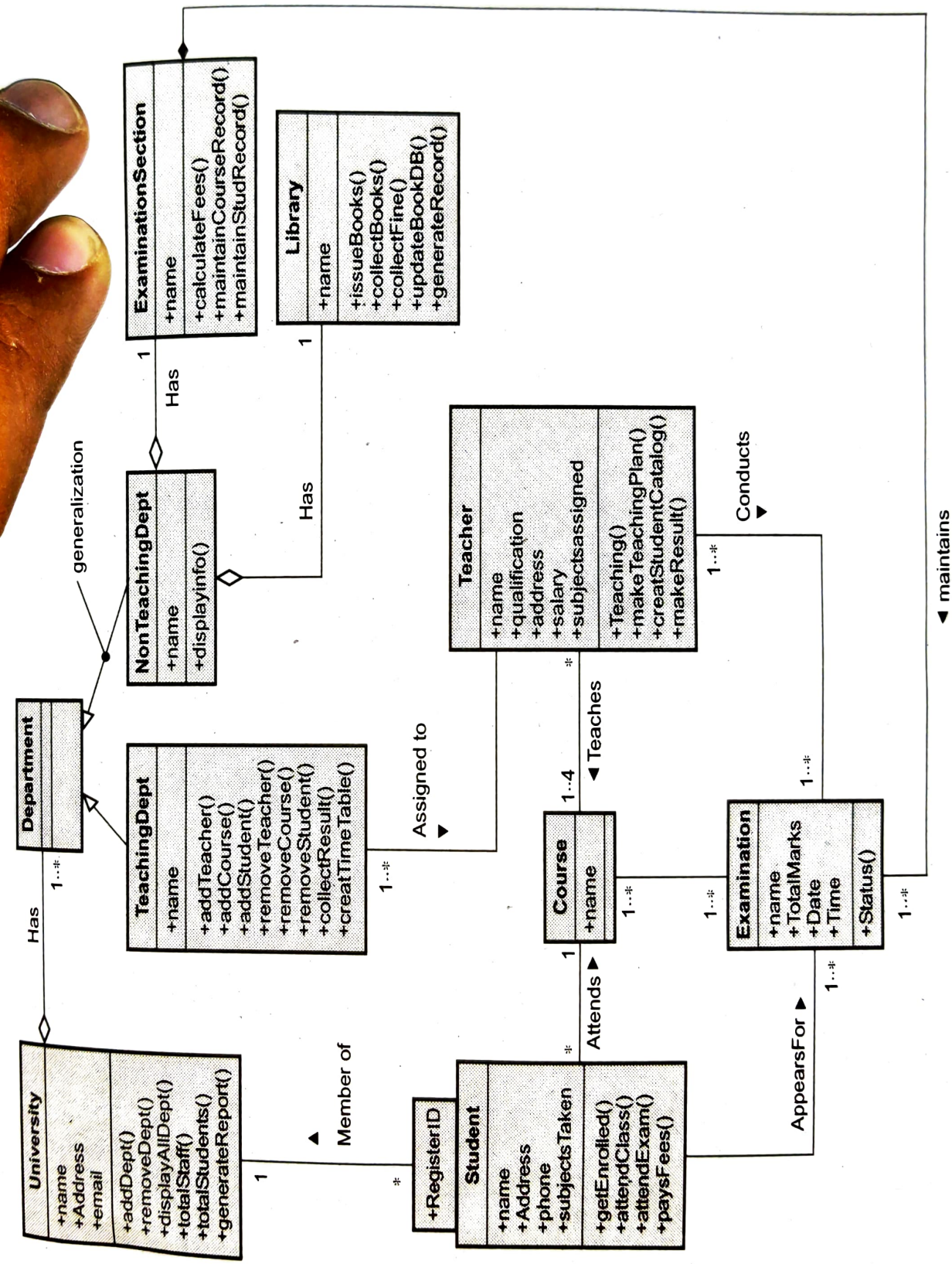


Fig. 2.5.10 Class Diagram for University Automation System

The corresponding object diagram will be -

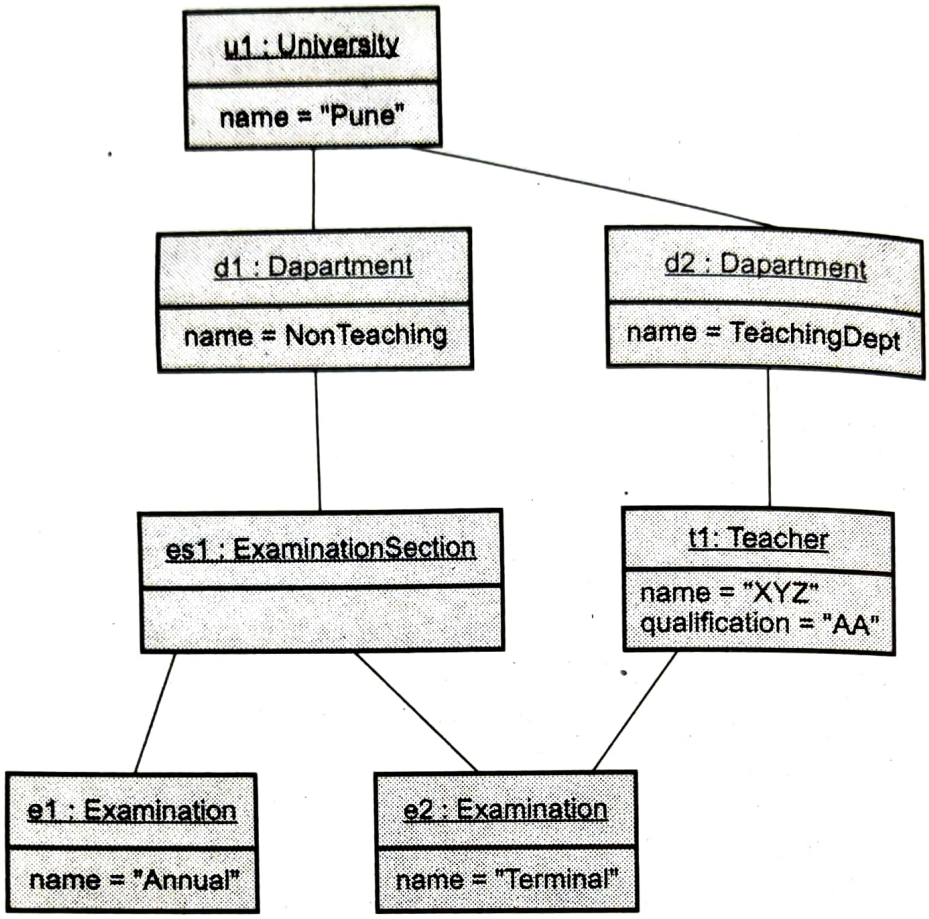


Fig. 2.5.11 Object Diagram for University Automation System (Exam Conduction Mechanism)

Example 2.5.6 *Draw object model for a library information system.*

Solution :

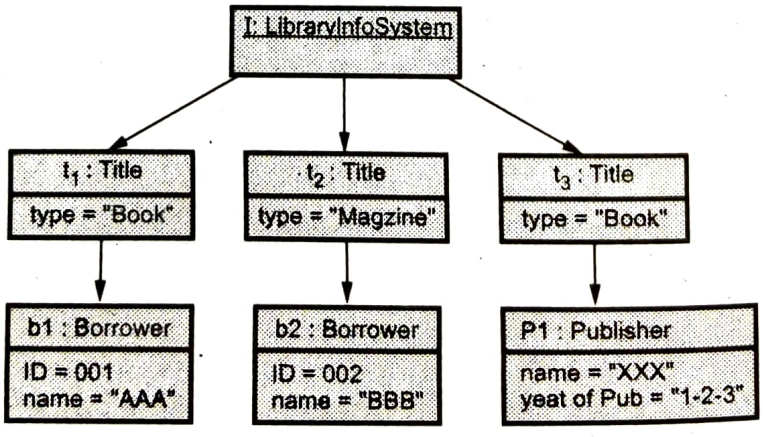


Fig. 2.5.12 Library information system

Example 2.5.7 *Prepare an object model to describe undirected graphs. An undirected graph consists of a set of vertices and a set of edges. Edges connect pairs of vertices. Your model should capture only the structure of graphs (i.e. connectivity)*

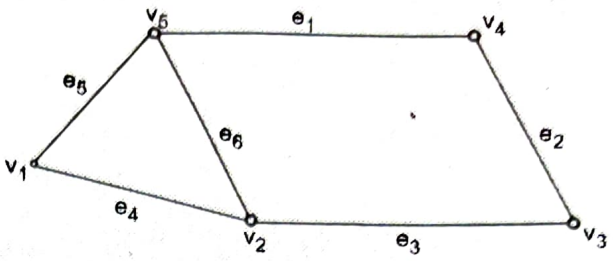


Fig. 2.5.13

Solution : The class diagram for given graph is -

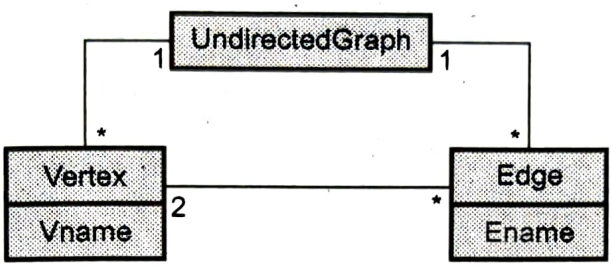


Fig. 2.5.13 (a) Class diagram

The object diagram for given graph is as given below -

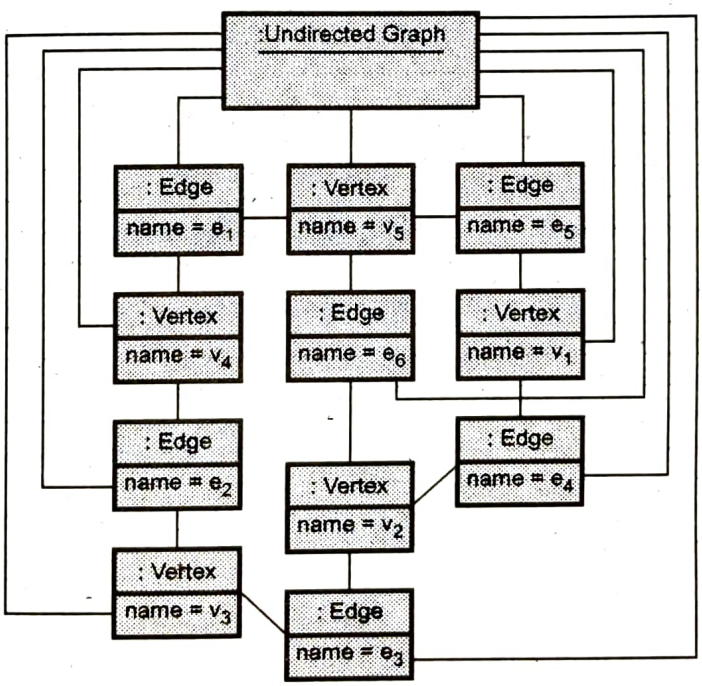


Fig. 2.5.13 (b) Object diagram

Example 2.5.8 *Draw object model for participant's registration system for seminar*

Solution : This model describes that static structure of objects in the system and relationships.

The object diagram for participant's registration system is Refer Fig. 2.5.14

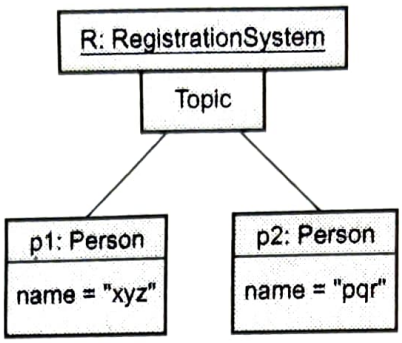


Fig. 2.5.14

Example 2.5.9 *Construct a class diagram using the following object diagram.*

SPPU : Aug.-15, In Sem, Marks 3

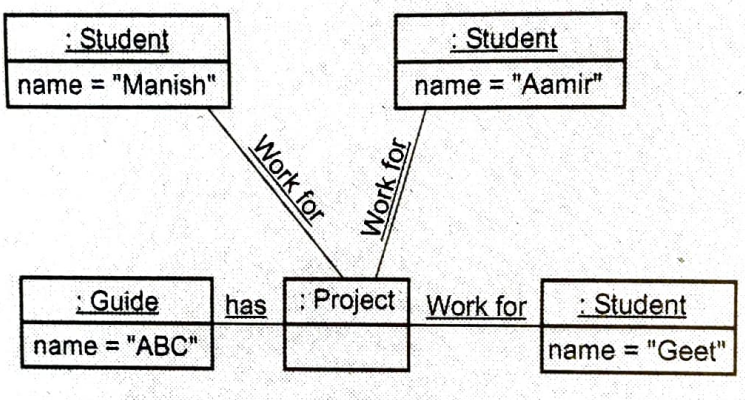


Fig. 2.5.15

Solution :

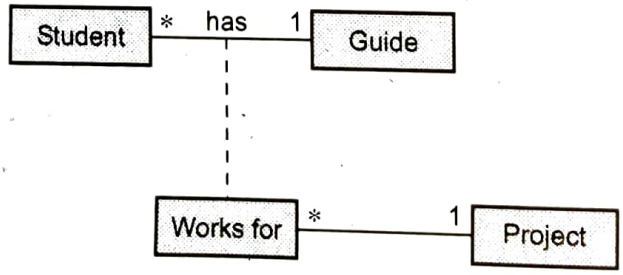


Fig. 2.5.15 (a)

2.6 Package Diagram

SPPU : April-18, 19, Marks 5

In UML, package is a general purpose mechanism for modeling the elements into groups. Using packages the elements can be organized properly. The package can be represented as shown in the fig below -

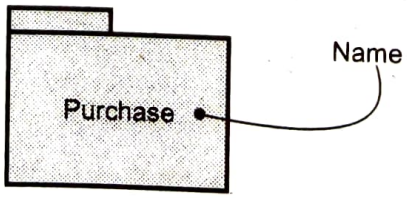
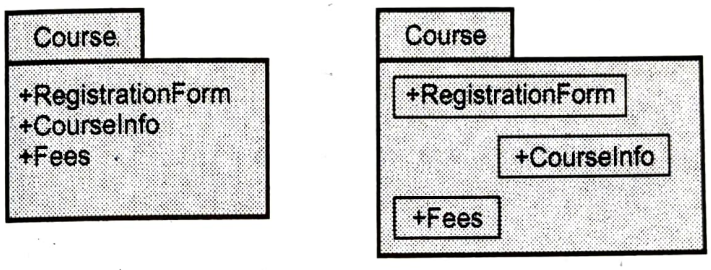


Fig. 2.6.1 Package

2.6.1 Owned Element

- A package may contain some other elements such as classes, interfaces, components, nodes, collaborations or even some other packages. Due to this ownership the composite relationship exists. That means when this package is deleted then the elements within it gets deleted.
- Every element is uniquely owned by exactly one package.
- The packages may have other packages. That mean it is possible to decompose the model hierarchically. For example a class named **course** can be represented as **Univeristy::Department::Course**
- Packages help to control the elements and arrange them in an organized manner.
- The contents of the packages can be displayed by text or graphically.
  For example -



Textual Nesting **Graphical Nesting**

Fig. 2.6.2 Owned elements

2.6.2 Visibility

- The visibility of the package can be controlled similar to the class.
- The element owned by a package is public to that element. Similarly, the protected elements can be seen by the children and the private elements cannot be seen outside the package. For example in Fig. 2.6.2 RegistrationForm is a public element of the package **Course**
- The public elements are specified by prefixing +.
- The protected elements are specified by prefixing #
- The private elements are specified by prefixing -
- Package visibility can be specified by prefixing ~

2.6.3 Designing Package Diagram

There are two approaches for designing the package diagram - 1. Modeling Group of Elements 2. Modeling Architectural Views

1. Modeling Groups of Elements

The purpose of packages is to model the elements into groups. Packages are use organize design. Following are the steps used to model the group of elements -

- Identify the **clump of elements** that are conceptually or semantically co-related
- Make a **package** of such elements.
- For each package mark the elements that are accessible outside the packag **public** and remaining as **private** or **protected** depending upon their characteris
- Explicitly connect the packages to one another using the **import dependency.**
- If there exists the specialized and general packages then connect them u **generalization** relationship.

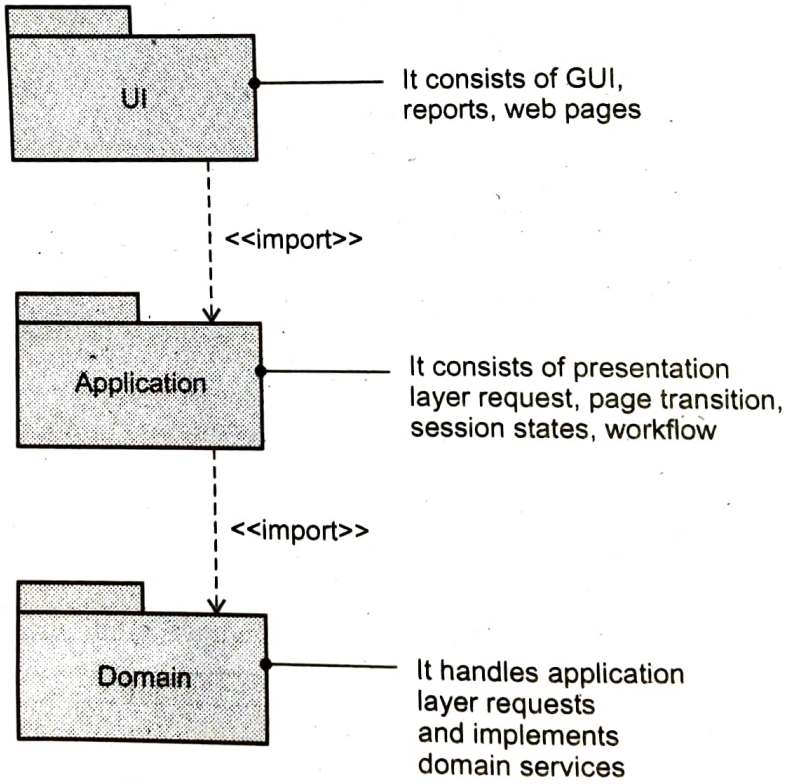


Fig. 2.6.3 Modeling groups of elements

2. Modeling Architectural Views

Following steps are followed to model the architectural views -

- Identify the architectural views that are significant to the given problem. Va architectural views are **Design View, Implementation View, Deployment V Interaction View and Use Case View.**
- Each view can be visualized, specified, constructed and documented by pl the corresponding elements into its belonging packages. See Fig. 2.6.4.
- The elements can be grouped further if necessary.

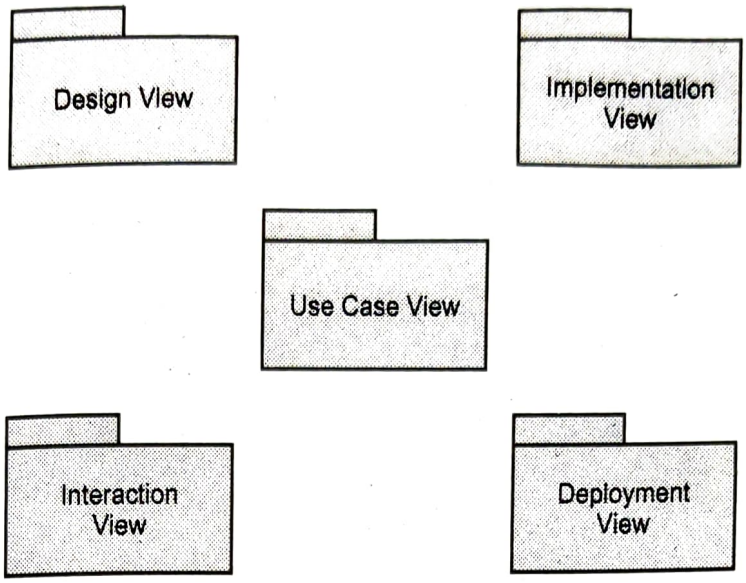


Fig. 2.6.4 Modeling architectural view

Review Questions

1. *Different elements used in package diagram.* **SPPU : April-18, In Sem, Marks 5**
2. *State package diagram with UML notations.* **SPPU : April-19, In Sem, Marks 5**

2.7 Component Diagram

SPPU : April-16, 17, 18, Dec.-19, March-20, Marks 6

- The components are used to represent the physical aspects of the system. The component represents physical and replaceable part of the system.
- The components can be executables, libraries, tables, files and documents.
- The component represents the physical packaging of the logical elements such as classes, interfaces and collaborations.
- The component is a physical part of the system that exists at the level of implementation platform.
- Graphically the component can be denoted by a rectangle.

2.7.1 Components and Classes

The components and classes both are the classifiers. But there lies the difference between the two -

Sr.No.	Components	Classes
1.	Components are the physical aspects of the system.	Classes are the logical aspects of the system.

2.	The components typically reside within the nodes.	The classes do not reside within the node.
3.	Components represent the physical packaging of bits on the implementation platform.	Using classes are the static design created for the implementation purpose.
4.	Components may implement classes and methods. But the components do not have the attributes and operations on their own.	Classes have attributes and operations.

For example -

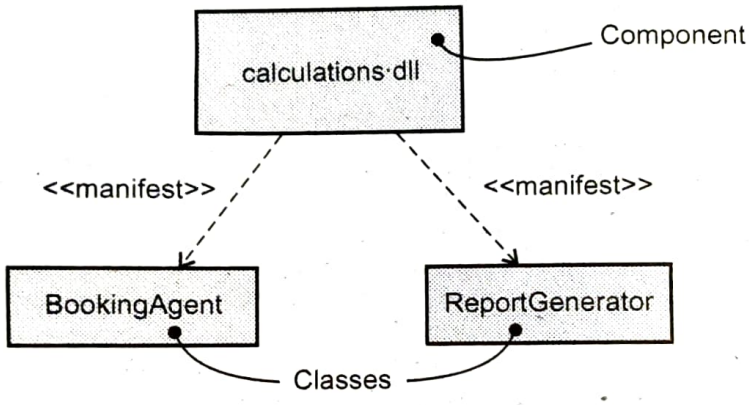


Fig. 2.7.1 Components and classes

2.7.2 Kinds of Components

There are three kinds of components -

1. **Deployment Component :** Using these components the executable systems can be formed. It includes DLLs i.e. Dynamic Link Libraries and EXE i.e. Executables. Using deployment models the classic object models such as .NET, CORBA which contains the database tables, web pages and executables - can be represented.
2. **Work Product Component :** These are the type of components that consists of source code files and data files from which the executable system can be developed.
3. **Execution Component :** These components are created as a consequence of executing system.

2.7.3 Stereotypes in Component Diagram

The tagged values can be used in components for specifying the properties of the components. The stereotypes are used to specify new type of components. Following are some standard stereotypes that can be used for components.

Sr. No.	Stereotype	Purpose
1	executable	It is used to specify that the component will be executed on the node.
2	library	It specifies a static or dynamic object library.
3	file	When a document contains the source code or data then that component is specified using this stereotype.
4	document	This stereotype denotes that the component represents a document.

2.7.4 Interfaces and Components

An interface is a collection of operations that are specified by the class or a component. There are various component based facilities such as COM+, CORBA or Enterprise JavaBeans that makes use of component and interface relationship.

In component based systems the whole system is decomposed into various components. One component accesses the services of other component via an interface. Due to this mechanism the location-independent (or platform-independent) operations are possible.

There are two types of interfaces used - **1) Provided Interface and 2) Required Interface**

Provided Interface - This interface is provided by one component as a service to other component.

Required Interface - This interface is used by one component when it requests to get services from other component.

One component may be associated with more than one required components. Similarly one component might be associated with provided as well as required interfaces.

A given interface may be provided by one component and is required by other component.

Following Fig. 2.7.2 represents the two methods of representing the components and interfaces. As shown in Fig. 2.7.2, in first method of representation the required interface is represented by a semicircle("Socket") and the provided interface is represented by a circle("lollipop" notation). The dependency relationship is shared by the interfaces.

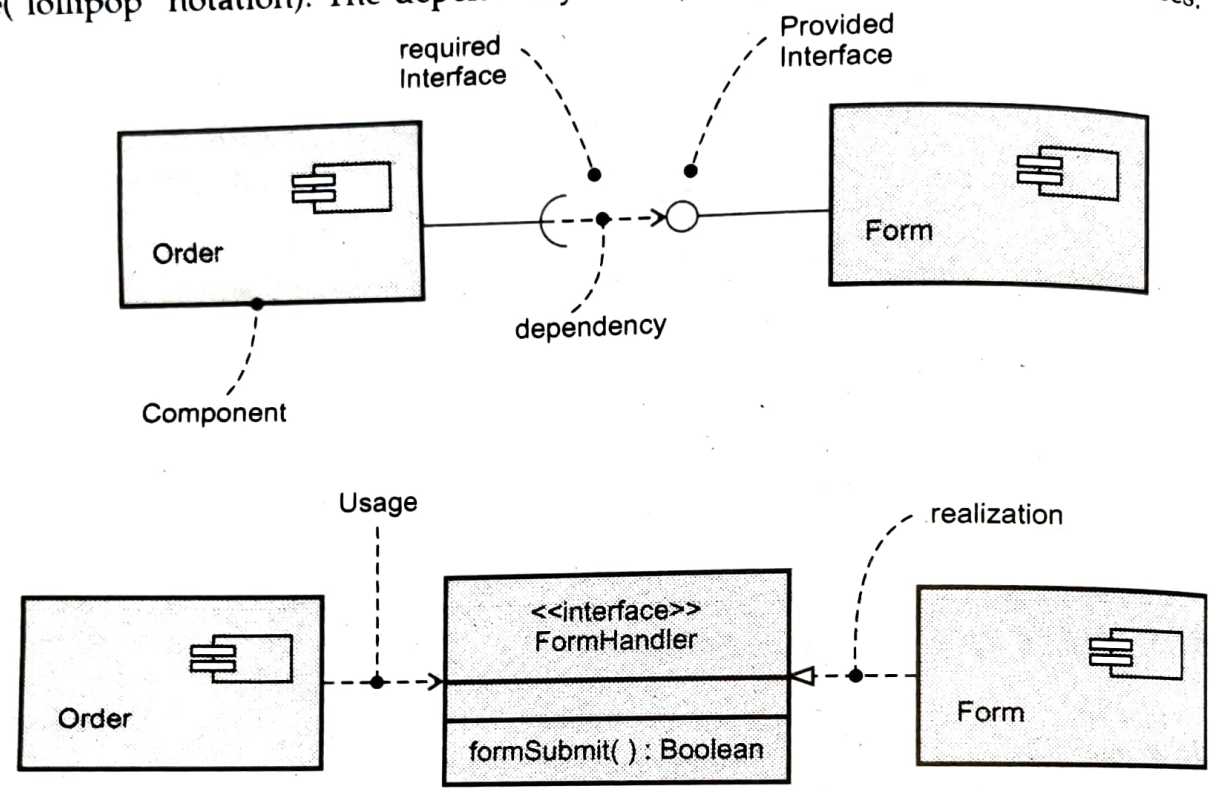


Fig. 2.7.2 Components and interfaces

In the second method of representation, the interface is declared using the stereotype <<interface>>. There is a dependency relationship to represent the required interface and the realization relationship is established between the interface and the component to show the provided interface.

2.7.5 Designing Component Diagram

- Component diagram and deployment diagram are the two types of models that focus on the physical aspect of the object oriented system.
- The component diagram represents the organization and dependencies among the set of components.
- The component diagram is used to model the static implementation view of the system.
- It contains the physical things(components) such as executables, libraries, files, tables and documents.
- The **component diagrams are basically the class diagrams** which mainly focus on system's **component.**

- For example

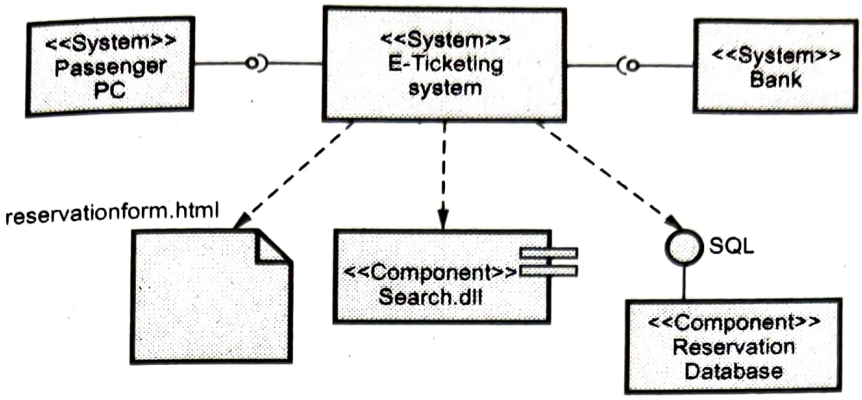


Fig. 2.7.3 Component diagram

Review Questions

1. *Explain port, provided interface and required interface with an example.*

SPPU : April-16, 17, In Sem, Marks 6

2. *Define interface. Explain provided interface and required interface with an example.*

SPPU : April-18, In Sem, Dec.-19, End Sem, March-20, End Sem, Marks 5

2.8 Composite Structure Diagram

SPPU : May-18, Marks 5

- Composite structure diagram in the Unified Modeling Language is a type of static structure diagram. It shows the **internal structure** of class and **collaborations** that this structure makes it possible.
- The structure contains various components such as -
- **Structured classifier :** It may contain an internal structure of connected elements each of which plays a role.
- **Role :** Role represents a participant within the internal structure of a structured Classifier.
- **Part :** Part is special kind of role (owned using composition).
- **Port :** A port represents interaction point through which an encapsulated classifier communicates with its environment.
- **Connector :** A Connector specifies links between two or more instances playing roles within a structured classifier.
- **Collaboration :** A collaboration is generally more abstract than a structured classifier. It is shown as a dotted oval containing roles that instances can play in the collaboration.

Example :

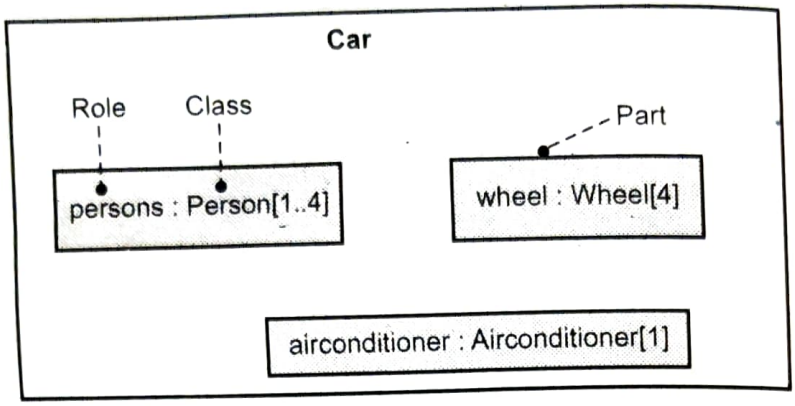


Fig. 2.8.1 Composite structure diagram

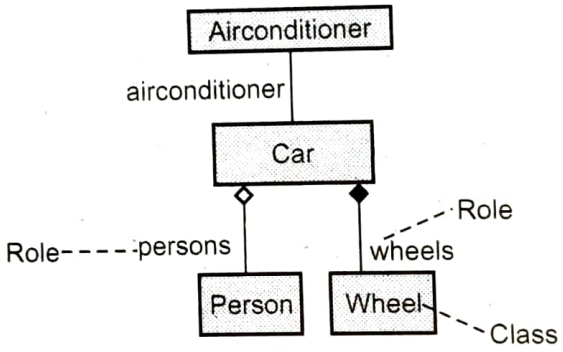


Fig. 2.8.2 Class diagram

- Note that the aggregation and composition relationship can be shown in Fig. 2.8.2 class diagram. The Car-Person is an aggregate relationship because **Person** can not be the physical part of the class **Car**, Similarly, **Wheel** is a physical part of a car in fact car can not be possible without the wheels, hence Car-Wheel is a composite relationship.

Review Question

1. *Explain the application of composite structure diagram.* **SPPU : May-18, End Sem, Marks 5**

2.9 Deployment Diagram

SPPU : April-18, March-20, Marks 5

- Deployment diagram is used to represent the physical aspect of object oriented system.
- The deployment diagram contains two things -
  1) Nodes
  2) Dependency or association relationship.

- Deployment diagram also contains the notes and constraints to add more semantic to the modeling.
- A node is a physical element that exists at run time and represents a computational resource. It generally has at least some memory and some processing capability.
- Graphically the node is represented as cube.
- Example -

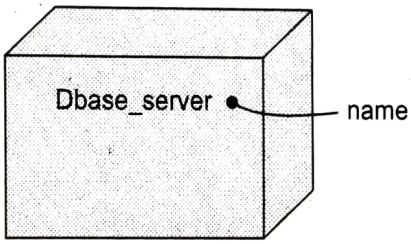


Fig. 2.9.1 Node

2.9.1 Nodes and Components

The nodes and components are similar to each other. Both the nodes and components have names, both participate in various relationships such as dependency, association and generalization. Both of them have instances. Both can be nested. Both the nodes and components can take part in interaction.

But there lies some difference between the two. Following are some differences between the nodes and components -

Sr. No.	Nodes	Components
1.	Nodes execute the components.	Components participate in execution of a system.
2.	The components are deployed within the nodes physically.	Components are nothing but the physical packaging of the logical elements such as classes, interfaces and collaborations.

Example -

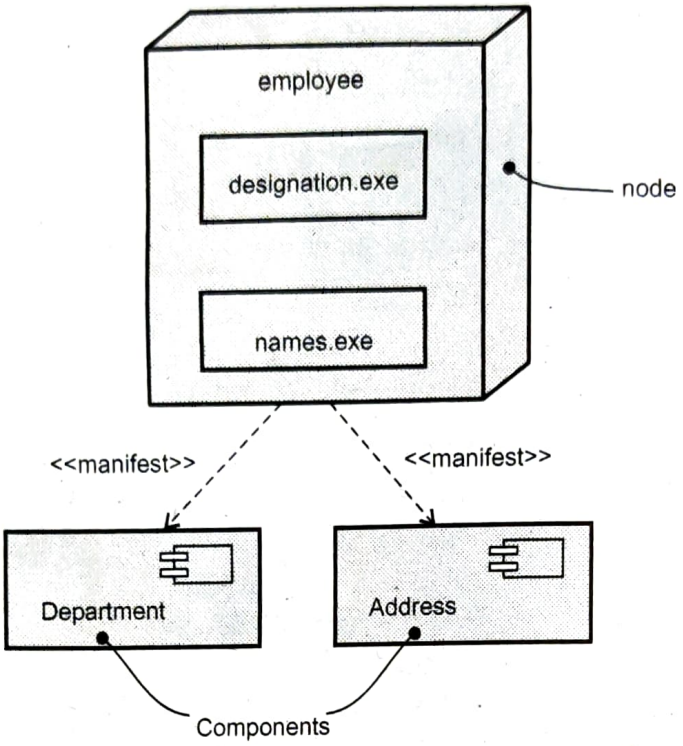


Fig. 2.9.2 Nodes and components

The set of objects that are allocated to a node as a group is called **distribution unit.**

2.9.2 Connections

- The **association** is the most common relationship used by the nodes among themselves. The association is actually a physical relationship such as Ethernet connection, shared bus or RS -232 cables.
- For distant processors the indirect connections such as satellite communication can also be used.
- **Example -**

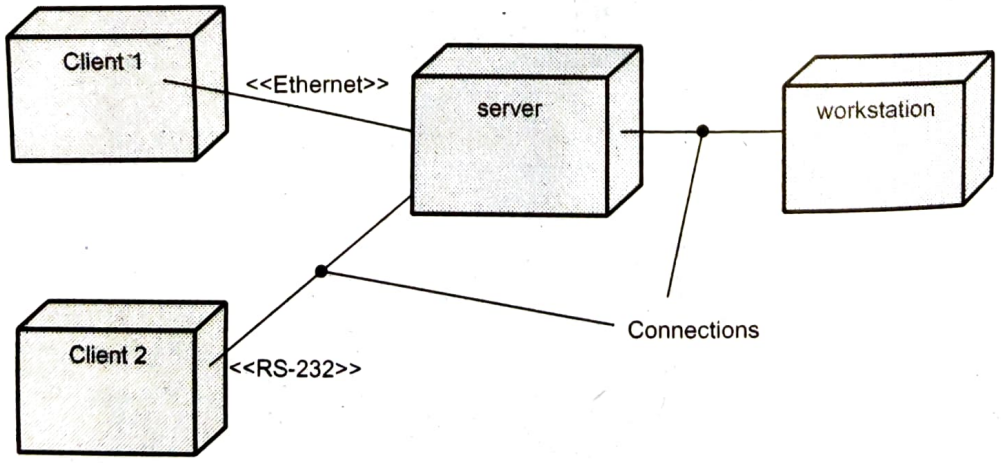


Fig. 2.9.3 Connections

2.9.3 Designing Deployment Diagram

- Component diagram and deployment diagram are the two types of models that focus on the physical aspect of the object oriented system.
- The deployment diagram shows the configuration of runtime processing modes and components that reside on them.
- The deployment diagram is used to model the static implementation view of the system. Mostly it models the hardware on which the system executes.
- The **deployment diagrams are basically the class diagrams** which mainly focus on system's **nodes.**
- Graphically the component diagram is a collection of nodes vertices and edges.
- For example -

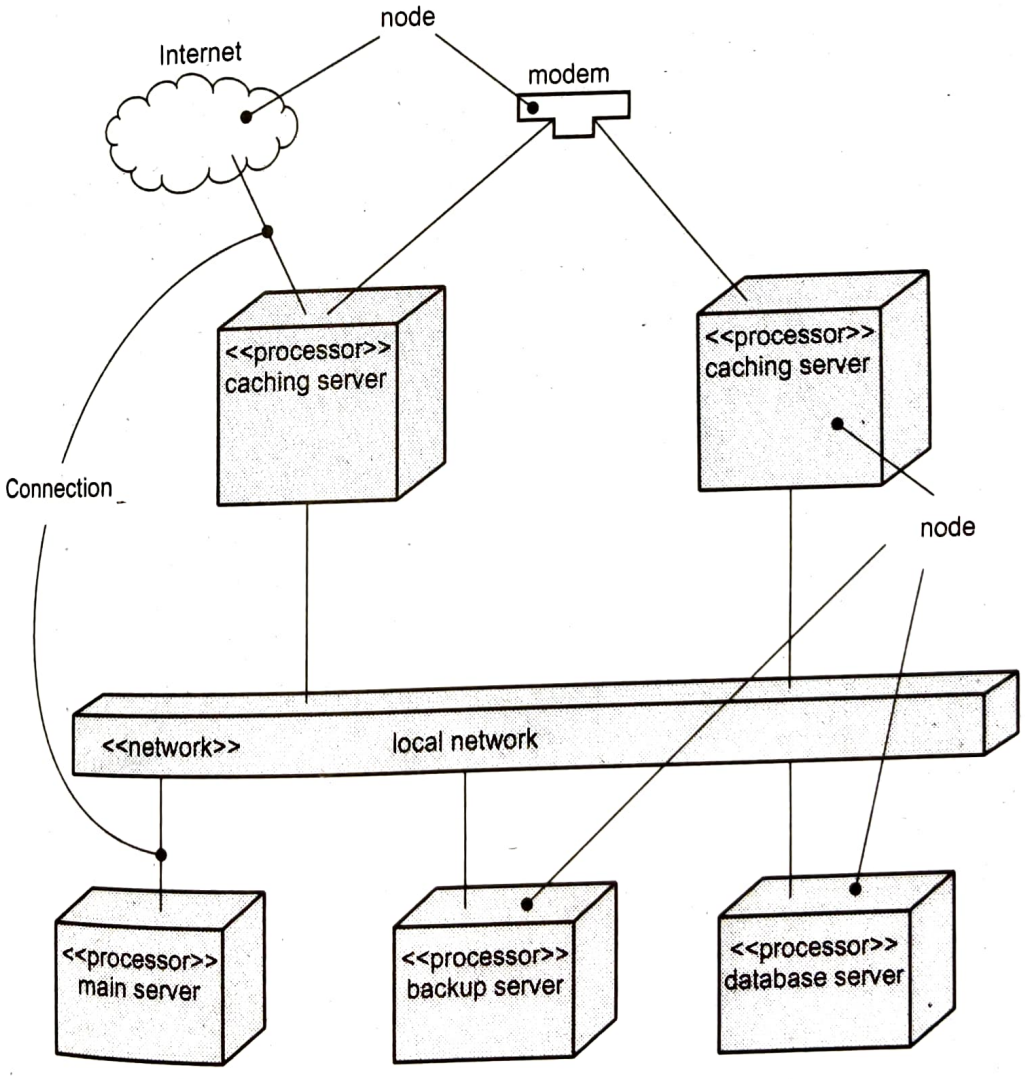


Fig. 2.9.4 Deployment diagram

2.9.4 Uses of Deployment Diagram

The deployment diagram is used for static deployment view of the system. Follow are the three ways in which the deployment diagram is used.

1. To model embedded systems

The embedded systems is a software intensive collection of hardware that interfa with physical environment. The software involved in embedded system basically con the hardware devices. The deployment diagram is used to model the devices a processors that form the embedded system.

2. To model the client/server systems

The client server is the most commonly used type of architecture. This architect separates the user interface from the persistent data. The topology of client/ser systems is modeled using the deployment diagram.

3. To model fully distributed systems

In distributed systems the components are globally distributed on multiple serv The deployment diagram is used to model the current topology of the system a distribution of the components across the system.

Example 2.9.1 *Design Deployment Diagram for online shopping cart*

Solution :

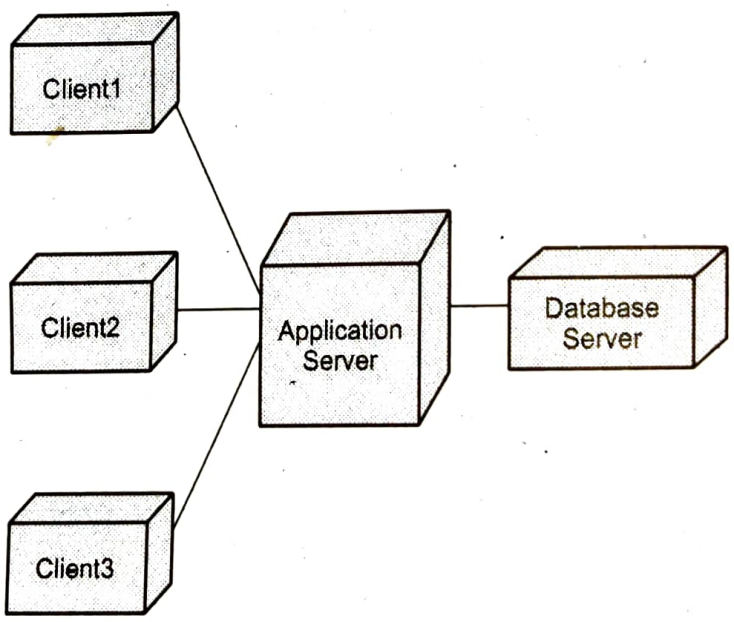


Fig. 2.9.5

Review Questions

1. *Explain different elements of deployment diagram. Draw a deployment diagram for J2EE web application with load balancing and clustering which shows specific server instances involved.*

SPPU : April-18, In Sem Marks 5

2. *Define component. Compare component and deployment diagram.*

SPPU : March-20, In Sem, Marks 5

2.10 Mutliple Choice Questions

Q.1 ___ is a blueprint for creating an object.

a Class
b Dependency
c Generalization
d Association

Q.2 Class is rendered as _____.

a square
b rectangle
c circle
d triangle

Q.3 CRC stands for _____.

a Class Relationship Collaboration
b Class Responsibility Component
c Class Responsibility Collaboration
d Collaboration Relation Component

Q.4 Modeling simple collaboration are in _____ diagram.

a class
b object
c use case
d activity

Q.5 A dependency relationship is rendered as a _____ .

a
b
c
d

Q.6 Inheritance in object-oriented system is used to _____.

a create new classes from existing classes
b add new operations to existing operations
c add new attributes to existing attributes
d add new states to existing states

Q.7 By polymorphism in object-oriented modelling we mean _____.

a the ability to manipulate objects of different distinct classes

b the ability to manipulate objects of different distinct classes knowing only their common properties

c use of polymorphic operations

d use of similar operations to do similar things

Q.8 Composition is a stronger form of _____ relations.

a aggregation

b encapsulation

c inheritance

d none of these

Q.9 An abstract class is _____.

a A class that has direct instances, but whose descendants may have direct instances.

b A class that has no direct instances, but whose descendants may have direct instances.

c A class that has direct instances, but whose descendants may not have direct instances.

d A class that has no direct instances, but whose descendants may not have direct instance.

Q.10 An aggregation _____.

a expresses a part-of relationship and is a stronger form of an association relationship.

b expresses a part-of relationship and is a weaker form of an association relationship.

c expresses an is-a relationship and is a stronger form of an association relationship.

d expresses an is-a relationship and is a weaker form of an association relationsh

Q.11 An object is selected for modelling a system provided _____.

a its attributes are invariant during operation of the system

b its attributes change during operation of the system

c it has numerous attributes

d it has no attributes relevant to the system

Q.12 _____ are part of the class operation specification format.

a Name
b Parameter list
c Return-type list
d All of the mentioned

Q.13 _____ are used to distribute files, libraries and tables across topology of the hardware.

a Deployment diagrams
b Use case diagram
c Sequence diagram
d Collaboration diagram

Q.14 Who consider diagrams as a type of class diagram, component diagram, object diagram and deployment diagram ?

a Structural
b Behavioral
c Non-behavioral
d Non structural

Q.15 _____ shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

a Use case diagram
b Deployment diagram
c Sequence diagram
d Class diagram

Q.16 _____ shows runtime processing for hardware.

a Use case diagram
b Deployment diagram
c Sequence diagram
d Class diagram

Q.17 Which among the following are not the valid notations for package and component diagram ?

a Notes
b Box
c Extension mechanisms
d Packages

Q.18 Components can be represented by which of the following ?

a Component symbols
b Stereotypes
c Rectangular boxes
d Component symbols and stereotypes

Q.19 An operation can be described as ?

a Object behavior
b Class behavior
c Functions
d Object & class behavior

Q.20 Which of the following diagram is used to model vocabulary of a system ?

a Object diagram
b Activity diagram
c Class diagram
d Use-case diagram

Q.21 _____ diagram is used to explain the data structures and the static snapshots of the things present in the class diagram.

a Use case
b Object
c Collaboration
d Sequence

Q.22 Select the view which is shown by object diagram_____.

a logical
b dynamic
c static
d none of these

Q.23 Which among the following are not the valid notations for package and component diagram ?

a Notes
b Box
c Extension mechanisms
d Packages

Q.24 A package diagram consists of the following ?

a Package symbols
b Groupings of use cases, classes, components
c Interface
d Package symbols, groupings of use cases, classes & components

Answer Keys for Multiple Choice Questions :

Q.1	d	Q.2	b	Q.3	c	Q.4	a
Q.5	b	Q.6	a	Q.7	b	Q.8	a
Q.9	b	Q.10	a	Q.11	b	Q.12	d
Q.13	a	Q.14	a	Q.15	b	Q.16	b
Q.17	b	Q.18	d	Q.19	d	Q.20	c
Q.21	b	Q.22	c	Q.23	b	Q.24	d

❑❑❑