Marks - 25

UNIT - IV

:- Design :-

The design activity begins when the requirement document for the s/w to be developed is available & the architecture has been designed.

During the design further refine the architecture. It focuses on module view.

That is during design we determine

⇒ what modules the system should have &

⇒ which modules have to be developed.

4.1 * Design concepts:

The design of a system is correct if a system built precisely according to the design ~~[illegible]~~ satisfies the requirement of that system.

To evaluate a design, we should use some evaluation criteria, so main criteria for evaluation is "Modularity".

⇒ A name "module" is a functional unit that can be logically seperable part of a program. ~~A system is said~~

⇒ Thus modularity is a means of problem partitioning in the context of s/w design. To produce modular designs some criteria must be followed.

PAGE NO.:

DATE: / /

The criterias are as follows

1) coupling
2) cohesion
3) open-closed principle.

4.1.1 coupling:- (between modules)

Defn:- coupling is a measure of relative interdependence among various module. It provides the strength of interconnections or a measure of interconnections among modules in a given program structure.

In general, modules must be loosely coupled with other modules in a s/w system in order to handle & modify a module independent of others. usualy highly coupled modules have strong interdependencies.

Factors affecting coupling between module are as follows.

1) The complexity of the interfaces across modules
2) The type of interconnection among modules
3) The type of communication between the modules.

V.V. Imp

In object oriented (OO) system, three different types of coupling exists between module.

1) Interaction coupling
2) component coupling
3) Inheritance coupling.

⇒ Interaction coupling :- It occurs due to methods of a class invoking methods of other classes It is similar to the function calling another function. The worst form of coupling is, if methods directly access interal parts of other methods. The coupling is lowest if the methods communicode directly through Parameters. Thus coupling is lower if only data is Passed, but higher if control information is Passed.

⇒ component coupling:- It refers to the interaction between two classes where a class has variables of other class.

For eg:- A class "c" has an instance variable (object) of type ~~anothe~~ another class "C1"

=> **Inheritance coupling:-**

It is due to inheritance relationships between classes. That is inheritance coupling exists if one class is a direct or indirect subclass (derived class) of the other.

**
**
Imp

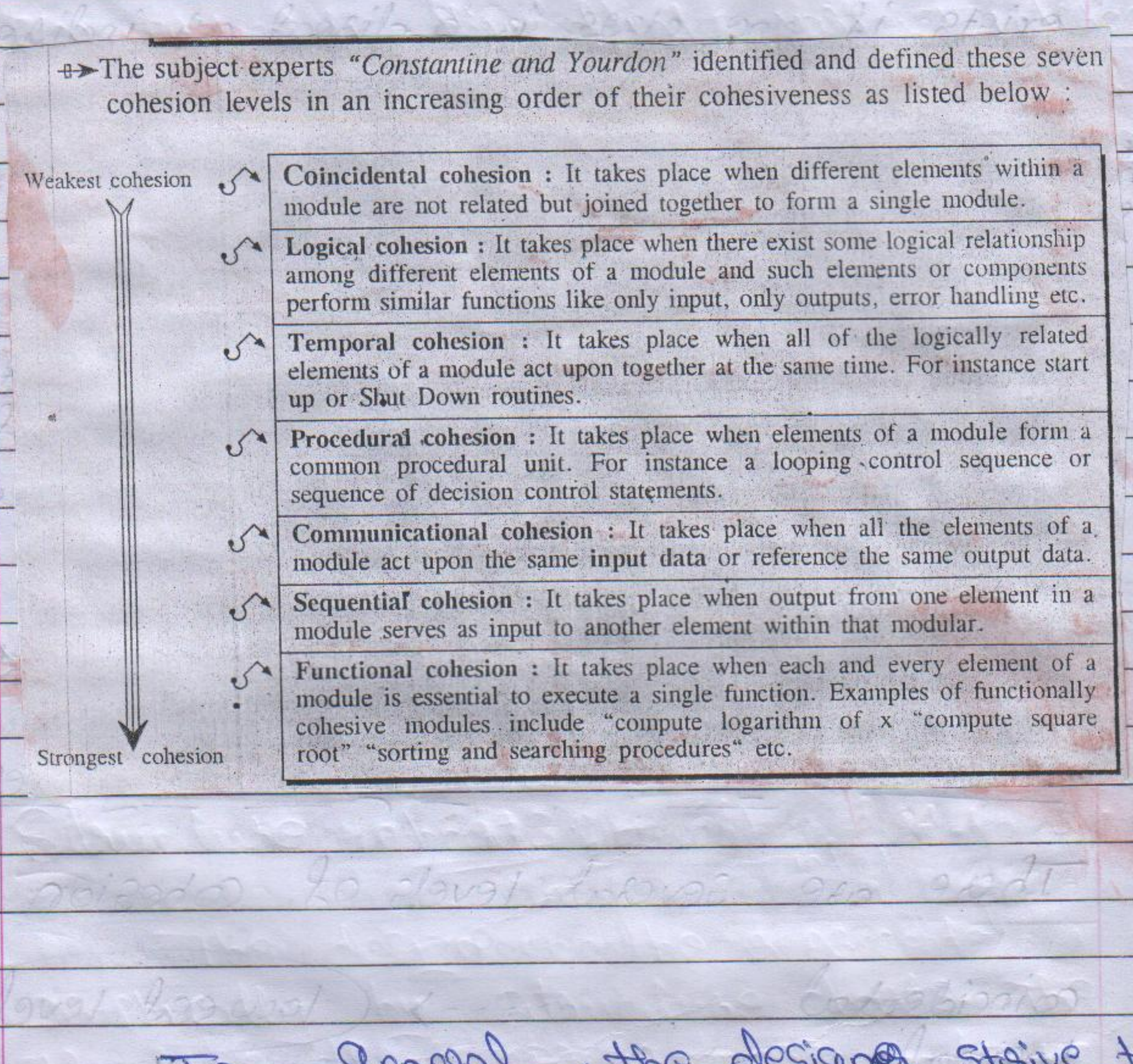
4.1.2 Cohesion (of a module)

Defn: Cohesion is a measure of the relative functional strength with which the different elements within a module are associated or related.

cohesion of a module specify how tightly the internal elements of a given module are bounded together or integrated together.

There are several levels of cohesion

1 coincidental -----> (lowest level)
2 Logical
3 Temporal
4 Procedural
5 communicational
6 Sequential
7 Functional ------> (Highest level)



→The subject experts *"Constantine and Yourdon"* identified and defined these seven cohesion levels in an increasing order of their cohesiveness as listed below :

Weakest cohesion
Strongest cohesion

Coincidental cohesion : It takes place when different elements within a module are not related but joined together to form a single module.

Logical cohesion : It takes place when there exist some logical relationship among different elements of a module and such elements or components perform similar functions like only input, only outputs, error handling etc.

Temporal cohesion : It takes place when all of the logically related elements of a module act upon together at the same time. For instance start up or Shut Down routines.

Procedural cohesion : It takes place when elements of a module form a common procedural unit. For instance a looping control sequence or sequence of decision control statements.

Communicational cohesion : It takes place when all the elements of a module act upon the same **input data** or reference the same output data.

Sequential cohesion : It takes place when output from one element in a module serves as input to another element within that modular.

Functional cohesion : It takes place when each and every element of a module is essential to execute a single function. Examples of functionally cohesive modules include "compute logarithm of x "compute square root" "sorting and searching procedures" etc.

In general, the designer strive to maintain high cohesion and at the same time allow for low coupling, so that s/w design can be easily modifiable in order to achieve greater function independence.

The OO system cohesion types a

1) method (2) class
③ Inheritance

PAGE NO.:
DATE: / /

4.1.3 Open- Closed Principle:

It was developed by an expert Mr. Bertrand Meyer. . It is a design concept widely accepted & practiced in an OO computing envidonment.

The basic Principle is "S/w entities should be open for extension but closed for modification"

According to this Principles changes and/or modification can be done only by adding new source code & should not by altering the existing old Program code.

This design Principle can be executed with oops design by employing Inheritance and Polymor Phism features.

The advantages of open-closed Principles are:

⇒ It permits addition of code for enhancements as requirements of end-user grows.

⇒ It will new allow the changes to the existing code & therefore risk can be minimised.

4.2. Function oriented Design:-

In function oriented design, the design of s/w system is divided into various interacting units. Each unit becomes a function.

This approach is more used in smaller system but not preferred for larger system.

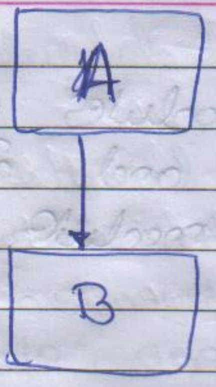
It is suitable for procedure oriented system design.

various models that can created are
1> structure charts
2> Data dictionaries
3> Data flow models.

4.2.1 Structure charts:- (flow of data & control info)

The design can be represented graphically by structure charts. The structure of a program is made up of the modules of that program together with ~~interaction~~ connections between the modules.

~~The~~ An arrow from module 'A' to module 'B' represents that module 'A' invokes module 'B'. 'B' is called subordinate of 'A', and 'A' is called the superordinate of 'B'.



calling module (superordinate)

called module (subordinate).

The modules uses data and flags. The charts drawn to comprise a variety module such as -

-> I/P
-> O/P
-> Transform
-> co-ordinate
-> composite.

The use of data and flag can be as shown below.

●↓ Flag going from superordinate to subordinate

○↓ Data going from superordinate to subordinate.

↑● Flag going from subordinate to superordinate

↑○ Data going from subordinate to superordinate.

=> I/P & O/P modules :-

what the modules gets from invoker is called I/P and what the receiver gets from the module is called o/p.

=> transform modules :-

The function Processes the I/P & Produces the o/p.

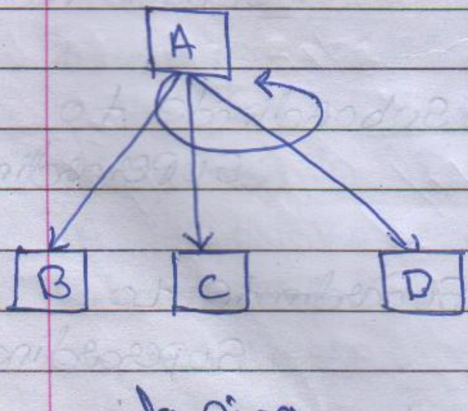
=> co-ordinate modules :-

These modules prime concern is managing the flow of data to and from different subordinates.

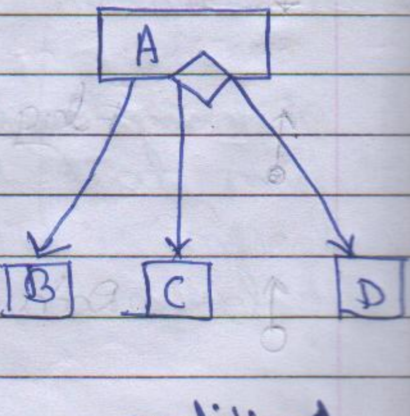
=> composite modules :-

A module can perform functions of more than one type of module.

The iteration & decisions on a structured chart is as shown below



looping



conditional checking.

4.2.2 Structured design methodology :-

Structured design methodology partitions the system ~~at~~ the very top level into various subsystems, one for managing each major I/P, one for managing each major O/P, and one for each major transformation.

The following are the steps that needed to be followed for designing the structure chart -

1) Draw the data flow diagram for given problem.
2) Identify the I/P & O/P data elements
3) convert DFD into first cut structure ~~chart with help~~ which is called first level factoring.
4) Refine the structure for I/P, O/P & transform branches.

Example :-

⇒ Explain the word counting problem through DFD & factoring.

Step 1:- Draw DFD for given problem.

4.3 Object oriented Design:-

The S/w Product can be designed using object/structured oriented method. There are many advantages and dis advantages with these system.

Advantages:-

1) The object oriented (OO) techniques focus on real world concepts due to which communication between the system engineer & customer becomes easy.

2) The complex system can be Partioned into small modules. This helps in managing the large complex system.

3) One component of the system can be reused in the same system or in another system. This process is called reusability.

Disadvantages:

1) These system are very slow.
2) These system require more money.

4.3.1 Object oriented concepts.

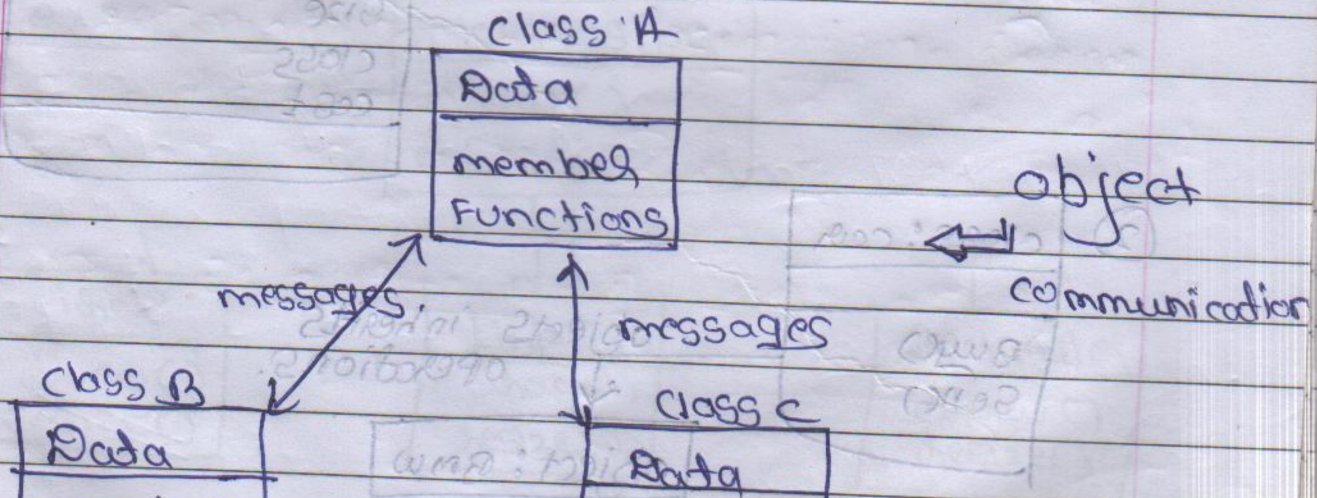
The OO technique evalves around objects and relationships among the objects.

OO concepts include.

1) classes and objects
2) Encapsulation and information hiding
3) Inheritance.
4) Polymarphism.

⇒ Classes and objects:-

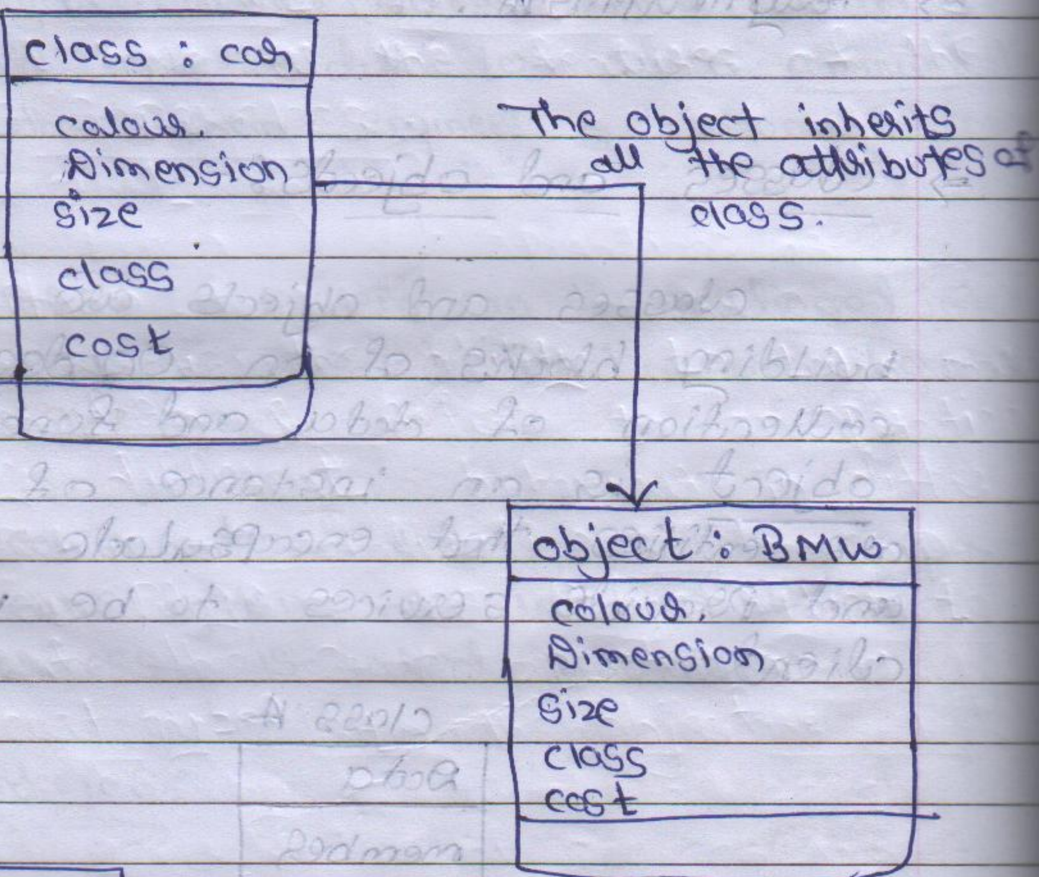
Classes and objects are the basic building blocks of an OO design. Class is collection of data and functions. Object is an instance of class. Objects are entities that encapsulate some state and provide services to be used by a client.



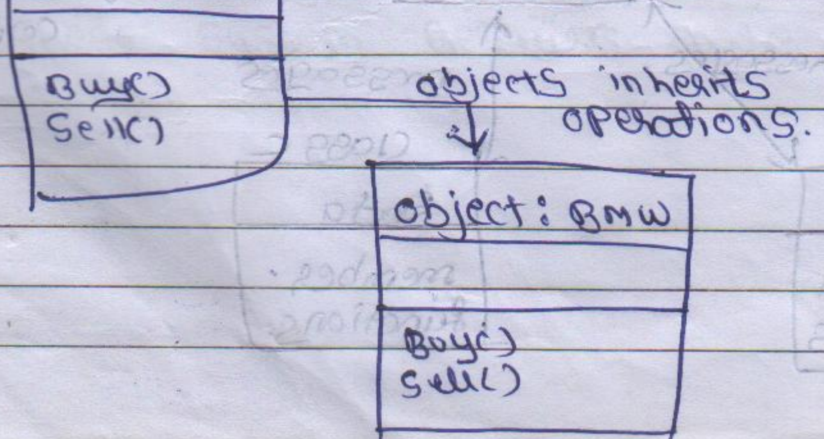
→ Relationships among objects :-

In OO system, an object interacts with other by sending a message to the object to perform some service it provides. This association is called link - a link that exists from object to another if the object uses services of other object.

Ex:- (1)

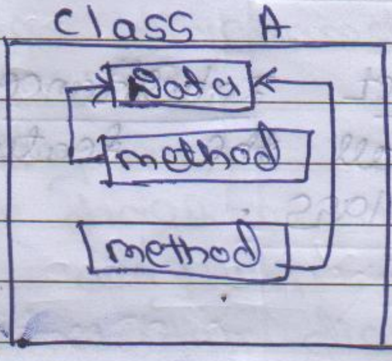


(2)



=> ~~P~~ Encapsulation :- means that the detailed implementation of a ~~computer~~ component is hidden from the rest of the system.

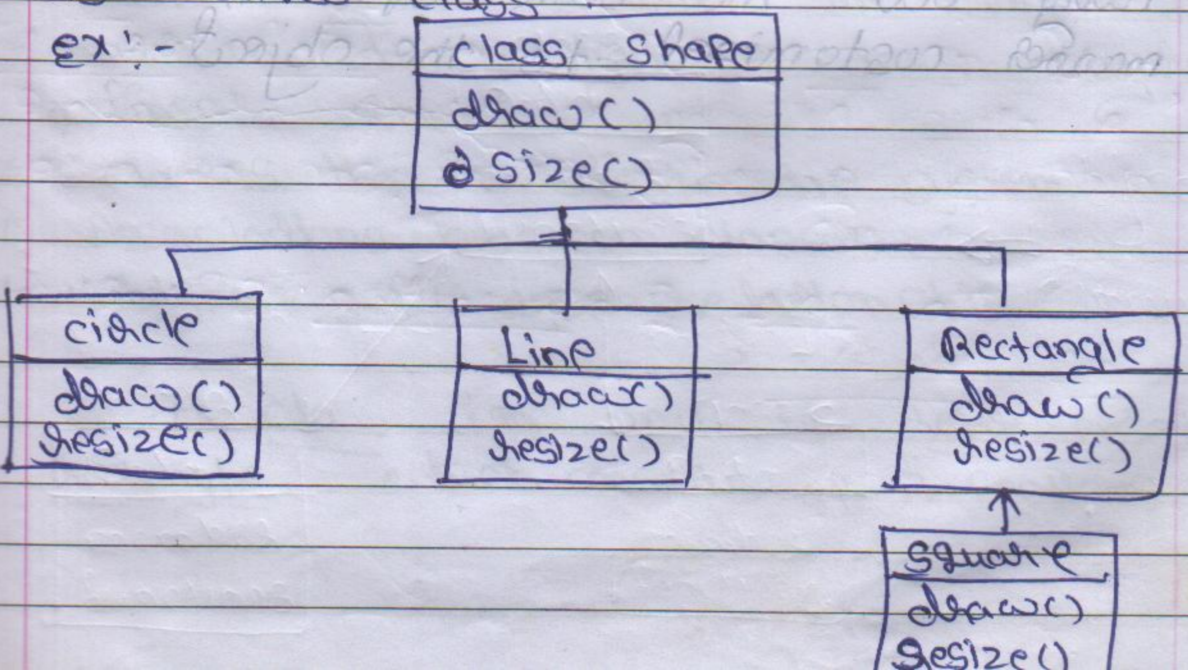
Def^n :- Encapsulation means binding of data and method together in a single entity is called class.



-> Inheritance :- It is a Property by which the new classes are created using the old classes.

The old classes are reffered as Base classes and the new classes are reffered as derived class.

ex:-



Inheritance is broadly classified into two types

1) Strict Inheritance
2) Non-strict Inheritance.

In strict Inheritance a subclass takes all the features from the base class / superclass and adds additional features.

In Non-strict Inheritance a subclass does' not have all the features of the base class / super class.

⇒ **Polymorphism :-**

Def^n → It is the ability to take more than one form & refers to an operation exhibiting different behaviour in different instances.

OO Program use PolymorPhism to carry out the same operation in a manner customized to the object.

Imp**

4.4 Unified Modeling Language :- (UML)

UML is a graphical notation for expressing OO design. It is called modelling language.

we will discuss modelling a s/w using UML with the help of following diagrams.

1) use case
*2) class diagram
*3) sequence diagram and collaboration.
4) Activity diagram.
*5) State transition diagram
6) component diagram.

Imp

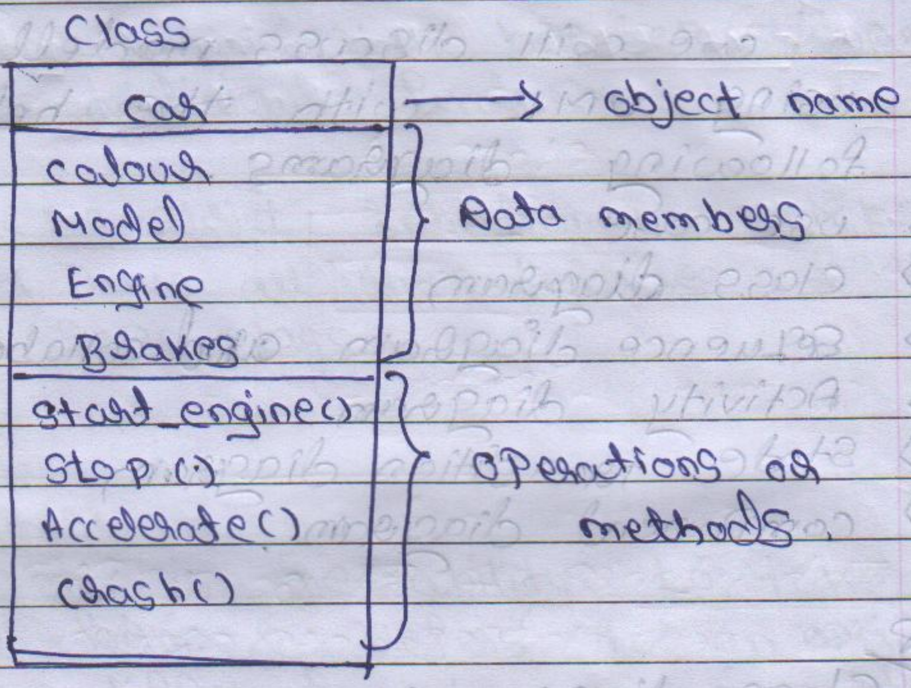
=> Class diagram :-

class diagram of UML is the central piece in a design or model. These diagram describes the classes that are there in the design. A class diagram defines

* classes that exist in the system
* Associations between classes.
* Subtype, super type relationship.

A class in UML is represented by rectangle which consists of 3 sections.

For eg:-objectcar consists of various attribute or data members such as model, Engine and Brakes. The object can communicate with other object by message passing.



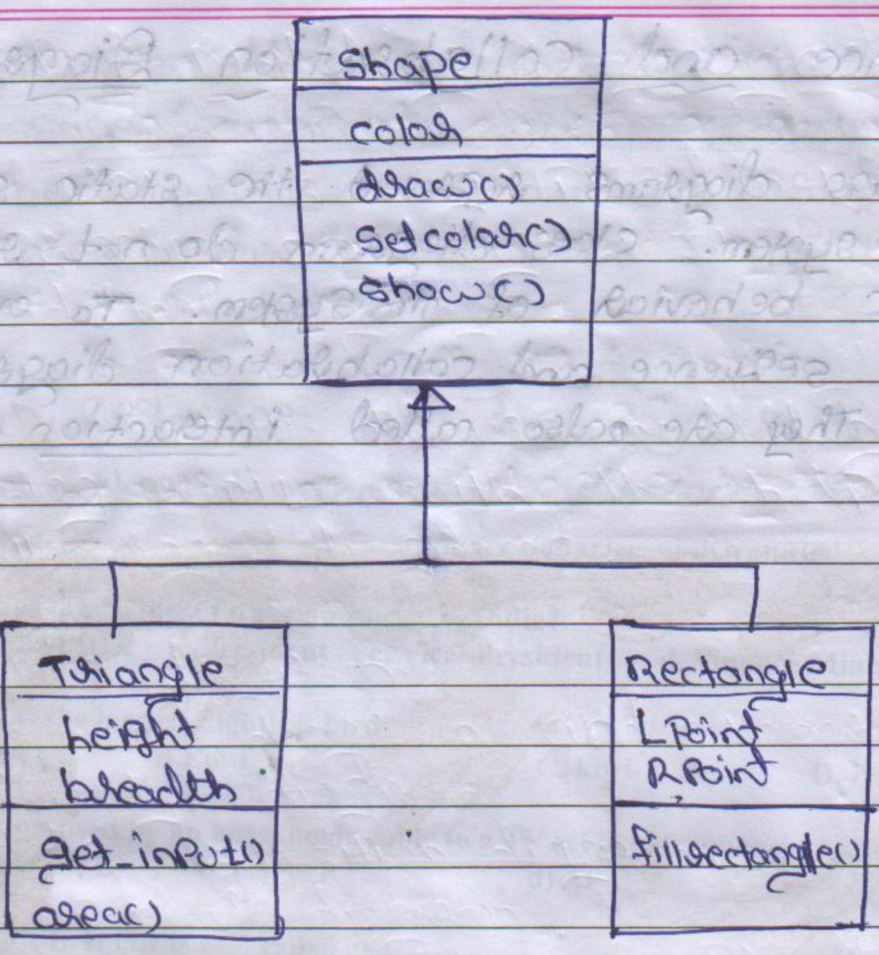
⇒ Generalization - Specialization relationship

It is specified by having arrows coming from the sub class to the super class with a empty triangle-shaped arrow head.

A class hierachy is formed where one class is generalization of one or more other classes.

In UML generalization is implemented as inheritance

16397



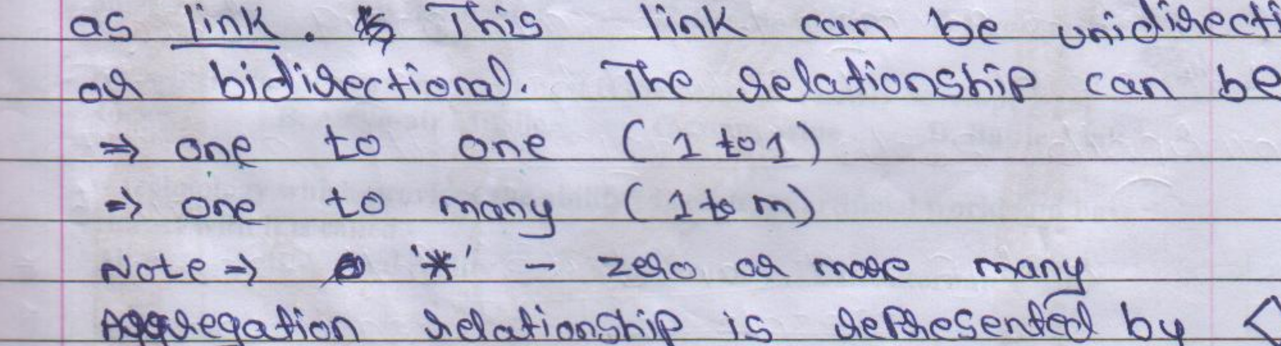
⇒ Association Relationship :-> allows the objects to communicate with each other. Association between classes can be refered to as link. ~~*~~ This link can be unidirectional or bidirectional. The relationship can be

⇒ one to one (1 to 1)

-> one to many (1 to m)

Note -> '*' - zero or more many

Aggregation relationship is represented by ◇



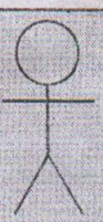
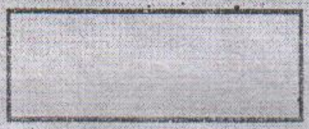
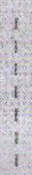
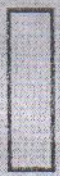
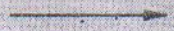
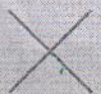
Imp

Sequence and collabration Diagrams :-

class diagrams represent the static structure of the system. class diagram do not reprent dynamic behavior of the system. To overcome this, sequence and collabration diagrams are used. They are also called interaction diagrams

Various notations used in sequence diagram are

fig 1 ⟹

	User - who is responsible for activating various events.
	An active object who takes part in various operations.
	It represents the life line i.e. a time span of the object.
	Active procedure on which object is active.
	To interact two objects this kind of association is shown. Along with this association the message is given
	Destruction of object.

A sequence diagram shows the series of messages exchanged between some objects. The various notations used in diagram are as shown in fig-"1".

=> In sequence all the objects that participate in interaction are shown at the top as boxes with object name.

Example 6.7.4 : *Draw a sequence diagram for library management system.*

Solution :

fig = 2

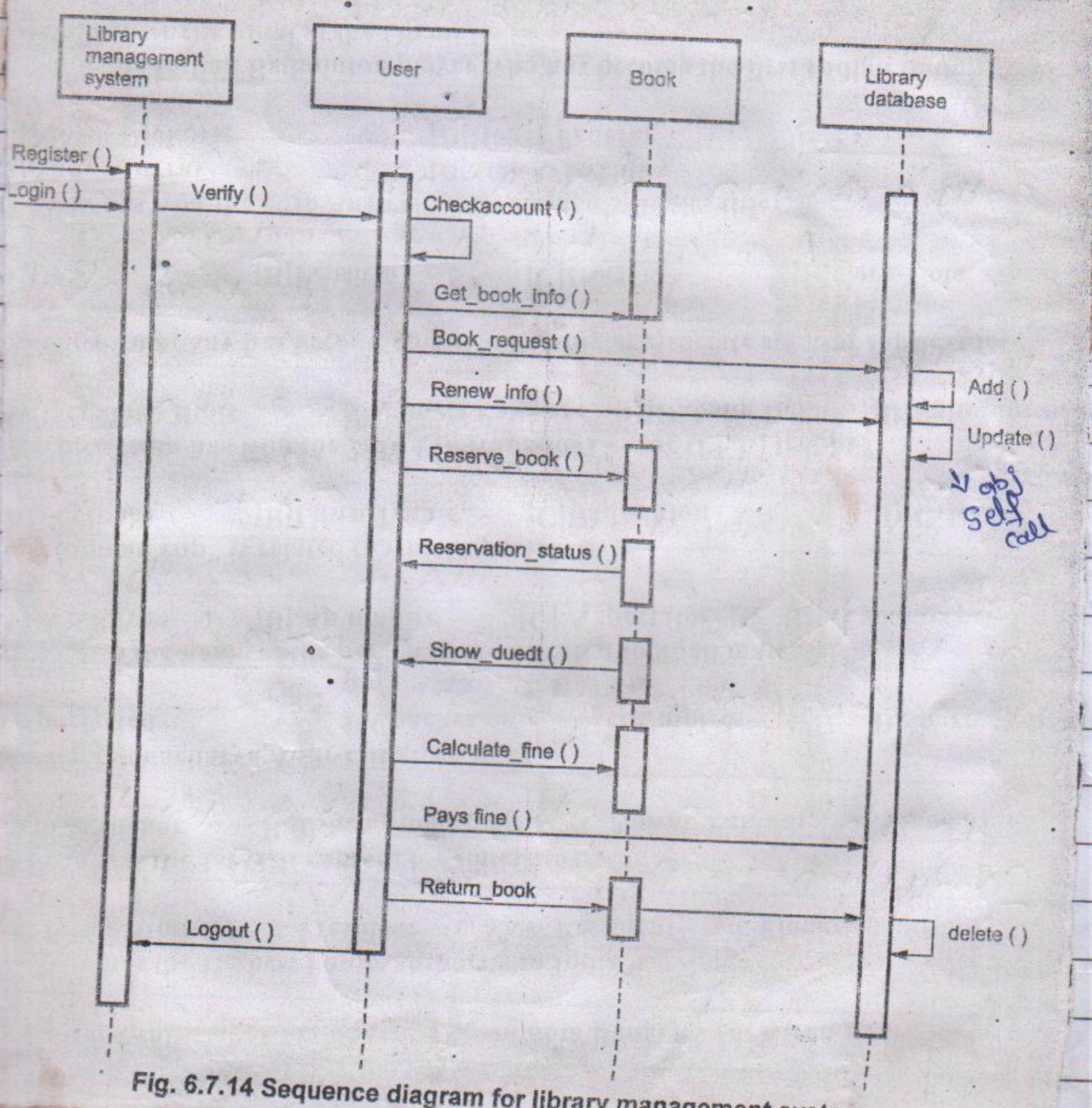


Fig. 6.7.14 Sequence diagram for library management system

=> For each object, a vertical bar representing its life line is drawn down wards.

=> A message from one object to another is represented as an arrow from the lifeline of one object to the lifeline of other objec

=> Each message is labeled with the message nam

=> An object can make a self call, which is shown as a message starting and ending in the same object life line.

collabaration diagrams :-

A collabaration diagram also shows how objects communicate. Instead of using a timeline based representation that is used by sequence diagrams, a collabaration diagram looks more like a state diagram.

Each object is represented in the diagram and the messages sent from one object to another as shown as numbered arrows from one object to the other.

collabaration diagram for printing a graduation report is ~~as~~ as in the fig.

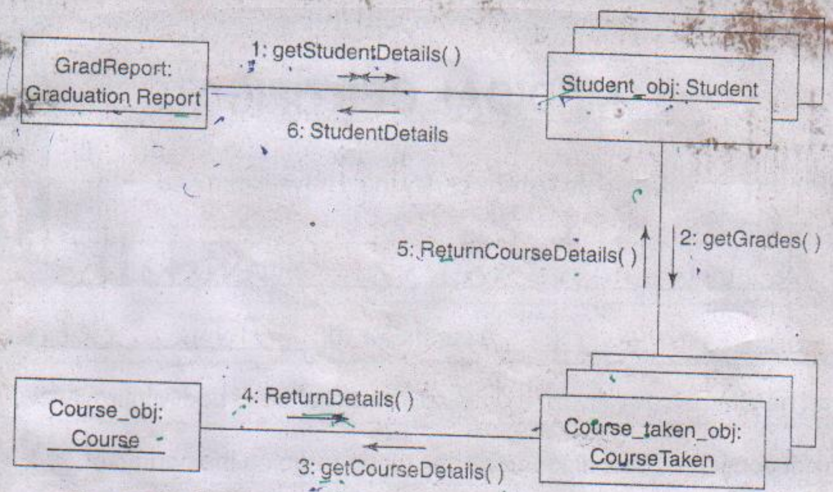
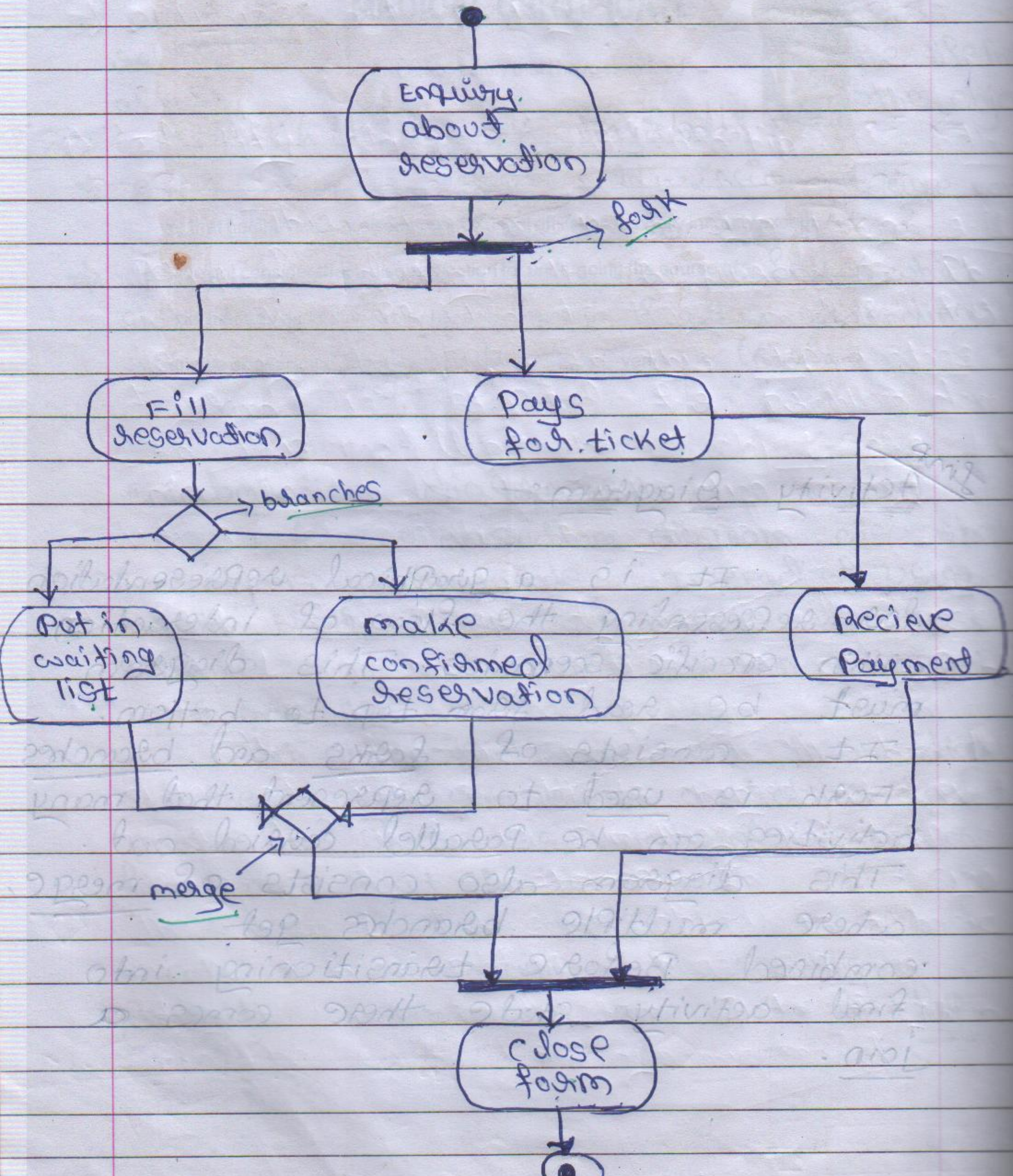


Figure 16 Collaboration diagram for printing a graduation report.

Imp Activity Diagram:-

It is a graphical representation for representing the flow of interaction within specific scenarios. This diagram must be read from top to bottom. It consists of forks and branches Fork is used to represent that many activities can be parallel carried out. This diagram also consists of merge, where multiple branches get combined. Before transitioning into final activity state there comes a join.

Activity diagram for railway reservatio
system is shown below.



Imp

State transition Diagram

State diagram are used to graphically represent the states of the system, ~~During~~ with transitions between the states taking place when some event ocur.

Notation	Description
1) ●	Initial state of state diagram.
2) ◉	Final state of state diagram.
3) ▭	State is represented as a rectangle.
4) ↓	arrows are used to indicate the transitions.

5 Design Methodology :-

An approach for creating an OO design consists of the following sequence of steps:

1) Identify classes and relationships between them.

2) Develop the dynamic model & use it to define operations on classes.

3) Develop functional model & use it to define operations on classes.

4) Identify internal classes and operations.

5) optimize and package.

1) Identify classes & relationships between them:

This step of the design is generally performed during the requirements phase when the problem is being modelled for producing the SRS. In this step a class diagram is prepared for problem domain. In this modelling, basic object are ~~inden~~ identified, the interaction among various classes is recognised.

2). Produce/Develop the dynamic model & use it ~~for~~ to define operations on classes:-

The dynamic model specifies how the state of various objects changes when events occur. An event is something that happens at some time instance. In OO design the request for some object is a basic event. various models such as sequence diagram, use case diagram, State diagram can be used for dynamic model.

PAGE NO.:
DATE: / /

3) Develop the functional model and use it to define operations on classes :-

A functional Model of a system specifies how the o/p values are computed in the system from the I/P values. In this modelling the focus is on functions. Hence during functional modelling the DFD's are produced.

4) Identify internal classes and operations:

using the static models and dynamic models the internal classes and corresponding operation are identified.

5) Optimization of the design :-

In this final step, the issue of efficiency is considered, keeping in mind that final structures should not deviate too much from the logical structure produced.

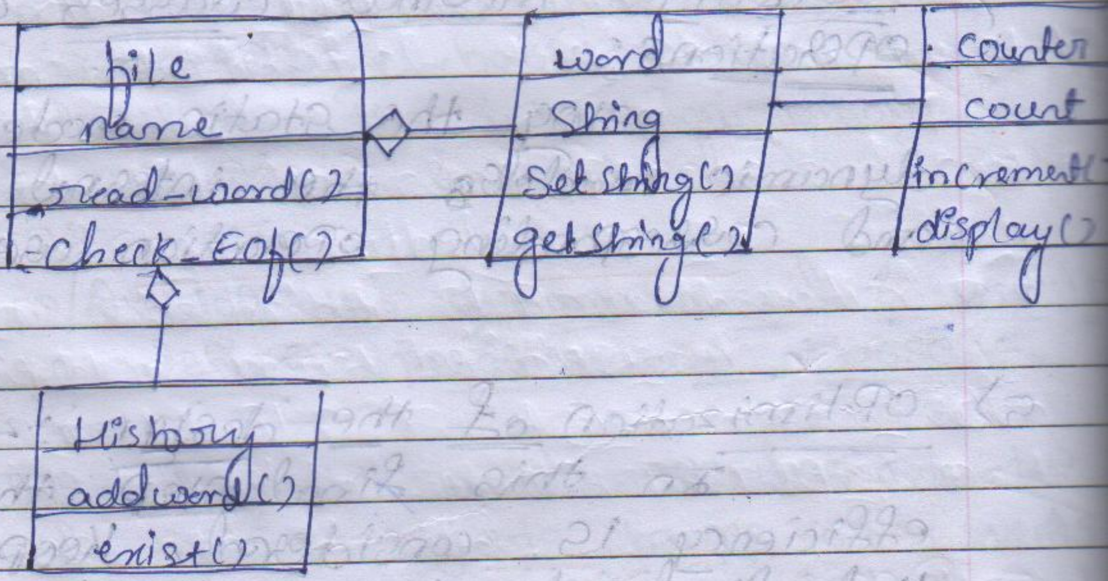
various optimizations are possible and a designer can use any method keeping in mind the modularity aspects also.

Example : With an example explain word coun-ting problem for OO design

Solution: problem description: The file contains Various words. This file is opened and read, word by word

The count for distinct words is maintained. The history for the words is maintained

Class diagram



Use case

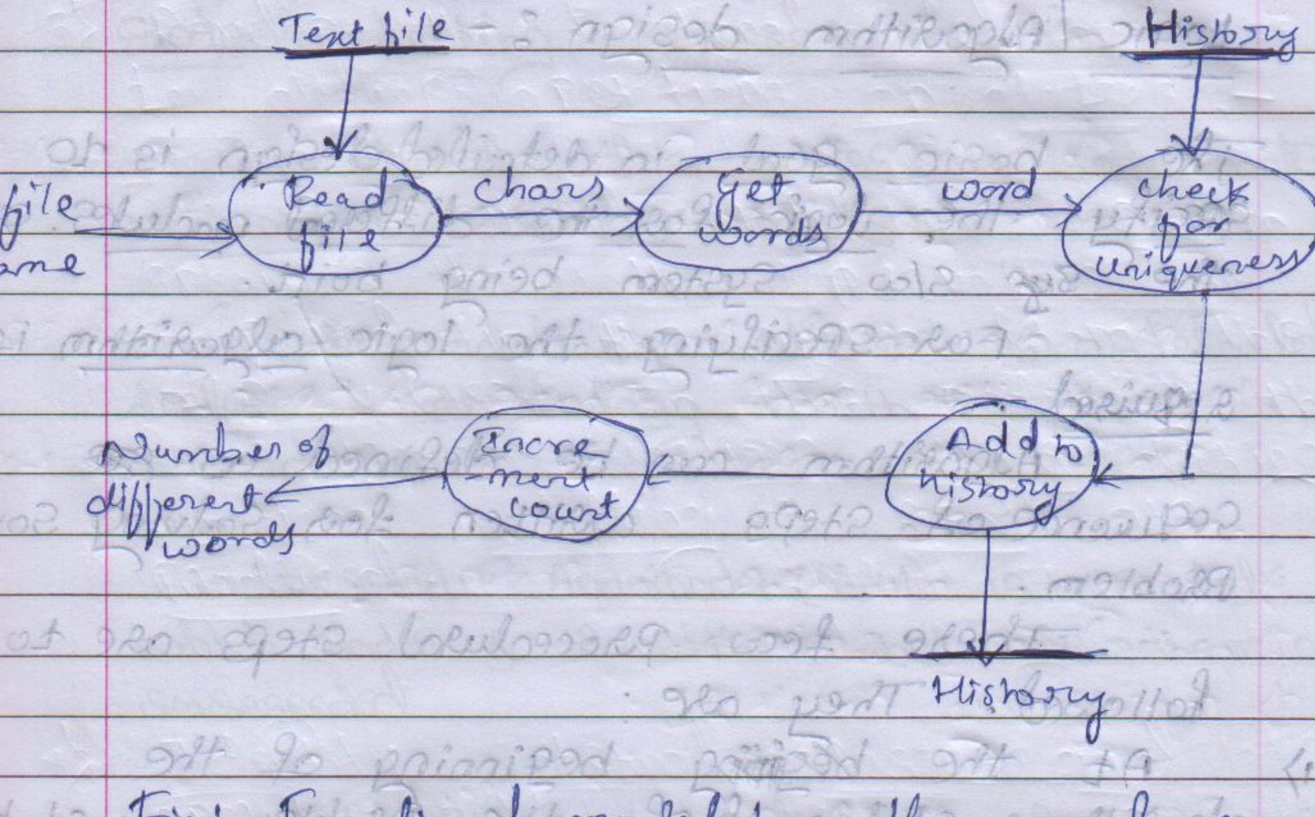


Fig1- Functional model for the word-counting problem.

4.6. Detailed Design:

It is a design methodology in which the system design module is focused in order to internal logic implementation. There are various methods for developing those are

1) PDL (Program design language) [Pseudo code]
2) Logic/ Alogirthm design.
3) State modeling classes

Logic / Algorithm design :-

The basic goal in detailed design is to specify the logic for the different modules. & the ~~sys~~ S/w System being built.

For specifying the logic / algorithm is required.

Algorithm can be defined as ~~se~~ sequence of steps written for solving Some problem.

There few procedural steps are to be followed. They are:

1) At the ~~begining~~ begining of the algorithm statement of the problem must be mentioned.

2) It should be clear. Precise and can be understood by anybody.

3) In the next step mathematical model for problem must be designed.

4) The next step is design of algorithm in which the data structure & program structure are decided.

** After complete design of algorithm ~~in~~ ~~which~~ its correctness should be verified.

Logic / Algorithm Design is a top-down method for developing detailed design.

State modelling classes:-

An object of a class has some state and many operations on its.

State modelling classes is a detailed design methodology with is used in OO design.

Finite State Automata (FSA) is a model in which states & transitions between them are represented.

A State transition diagram for an object consist of an initial state and transition Path exist for all other states of that object.

Finite State modelling of class-objects Provides better understanding on various operations as defined on the classes and on different states of its objects.

Thus FSA help to evaluate the logic for a given class operation as correct or not.

---o---

Q.7 verification:-

The O/P the System design Phase is the system design Specification's document..

This document have to be verified before starting the next Phase of SDLC.
There are various techniques for verification such as

1) Design walkthrough | Reviews.
2) critical design review.
3) consistency checkers.

=> Design walkthroigh :-

A team of reviews persons conduct system design review process to ensure good quality of design & whether the design meets necessary problem requirements / specifications.

Design review is intended to detect design errors or undesirable properties in the design.

During the walkthrough some of the faults are fixed but some are kept as it is for later correction.

-> ~~critical Design Review~~ :-

— o — o —