OBJECT - ORIENTED METHODOLOGIES :→ Each methodology is based on modelling the business problem and implementing the application in an object-oriented fashion. The difference lie primarily in the documentation of information and modelling notation and language. An application can be implemented in many ways to to meet the same requirement and provide the same functionality. Two people using the same methodology may produce application design that look radically different This does not necessarily mean that one is right & one is wrong, just that they are different.
Methodology & modelling notation developed by Rumbaugh et al Booch, and Jacobson which are the origin of the Unified Modelling Language (UML)

RUMBAUGH ET AL.'S OBJECT MODELING TECHNIQUE:- OMT presented by Rumbaugh which describe a method for analysis, design and implementation of a system using object-oriented ~~system~~ Technique OMT is a fast, intuitive approach that expressed the class, Attribute method, Inheritance and anociation very easily. Dynamic behavior of the object described by dynamic model using state transitions and their description within a system. OMT consists of four phases

1. Analysis
2. system design
3. Object design
4. Implementation.

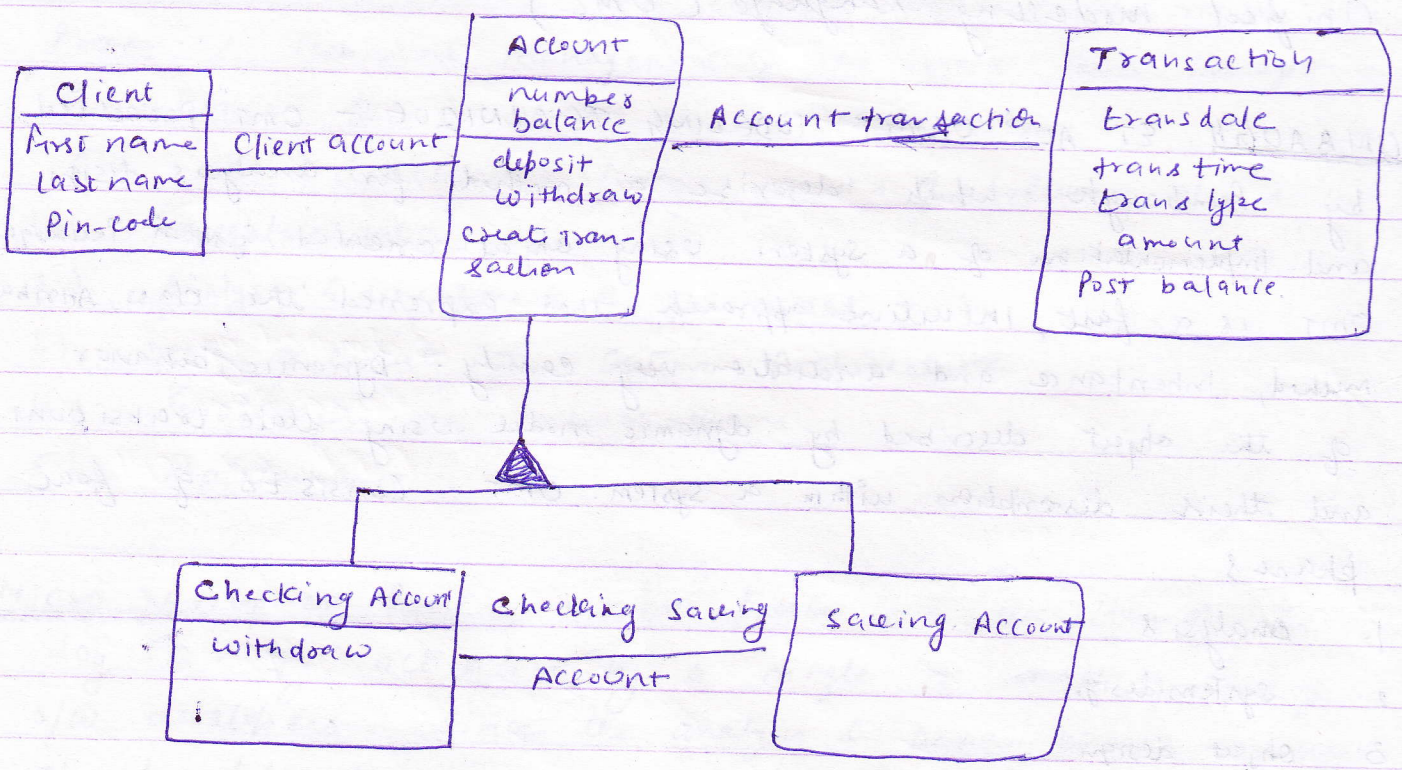
OMT Separates modelling into three different parts:

1) An object model, presented by object model and the data dictionary
2) Dynamic model prosented by state diagram, event flow diagram

3.) A functional model : Presented by data flow & constraints

OBJECT MODEL :- The object model describe the str. of object, their Identity, Relationship with other object, attribute, and operation The object diagra. model is Represented graphically with an object diagram & the diagram contains classes are Interconnected by association lines.

OMT Dynamic Model → Dynamic model depict the state, transition event & actions. It is a network of state & events. Each state receives one or more events, at which time it makes the transition to the next state.

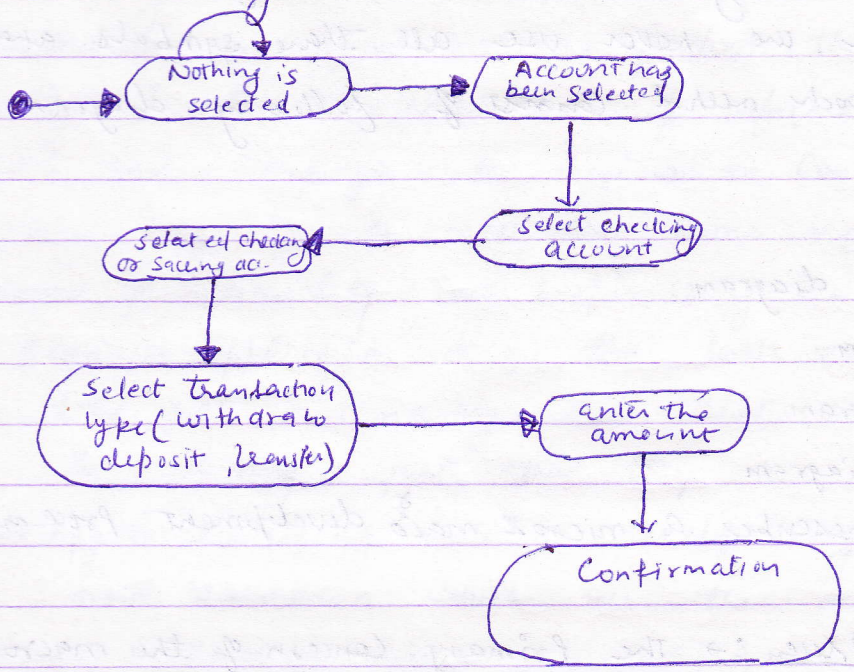


OMT Functional Model → OMT data flow diagram (DFD) shows the flow of data b/w different Processes in business. Data flow diagram use four Primary symbol

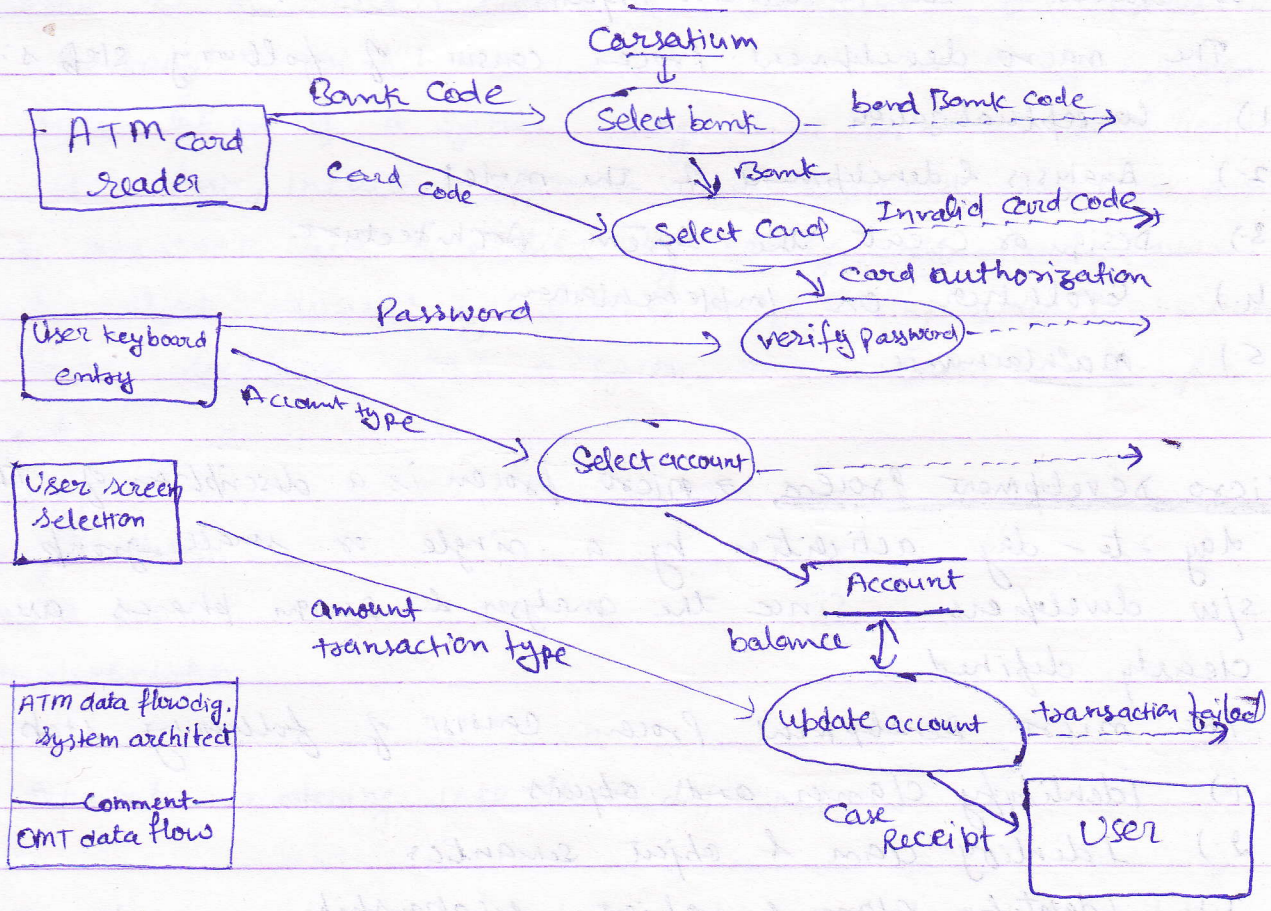
1.) Psaem, eg. verify Password.
2.) data flow eg. PIN CODE

3. data store : ACCOUNT is a data store in ATM

4. external entity : ATM Card Reader.



State Transition Diagram



BOOCH METHODOLOGY :- Booch defines a lot of symbols to document almost every design decision. If we work with this method, we never use all these symbols and diagrams. The Booch method consists of following diagram :

1.) Class diagram
2.) Object diagram
3.) State transition diagram
4.) Module diagram
5.) Process diagram
6.) Interaction diagram

Booch methodology Prescribes a micro & macro development process.

Macro Development Process :→ The primary concern of the macro process is technical management of the system. Such management is enforced less in actual object-oriented design.

The macro development process consists of following steps :

1.) Conceptualization
2.) Analysis & development of the model.
3.) Design or create the System Architecture.
4.) Evolution or Implementation.
5.) maintainance.

Micro Development Process :→ micro process is a description of the day-to-day activities by a single or small group of s/w developers. Since the analysis & Design phases are not clearly defined.

The micro Development Process consist of following step :

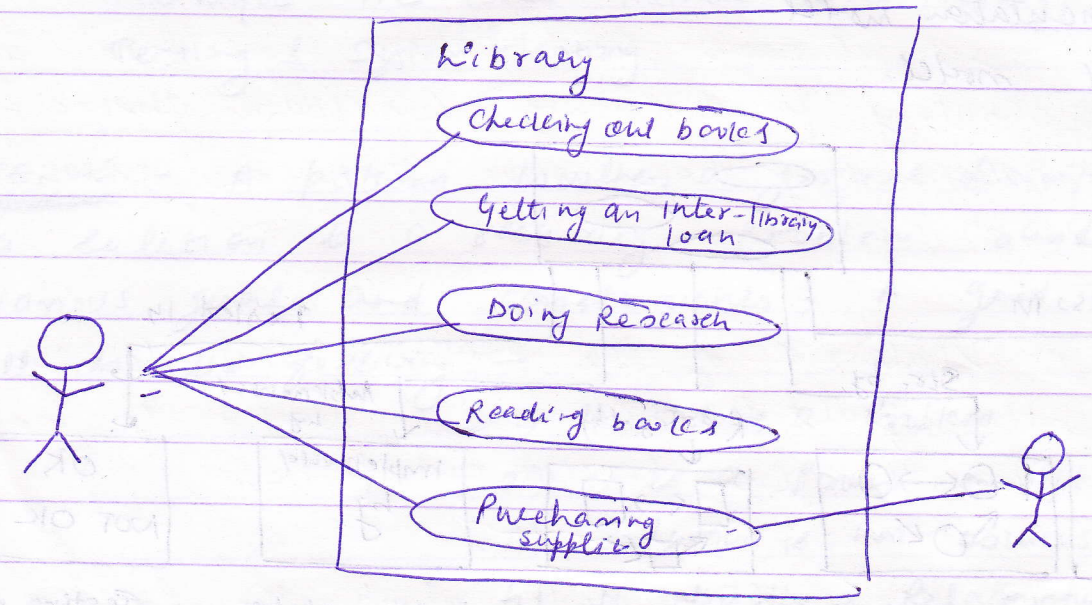
1.) identifyfy classes and objects
2.) Identify class & object semantics
3.) Identify class & object relationship
4.) identify class & object interfaces and implementation.

Jacobson ET AL. METHODOLOGIES :- Jacobson et al methodologies (OOSE & OOBE) Cover the entire life cycle and stress traceability b/w different Phases, both forward & backward. This allows the re-usability of code result is reduction of development time. There methodology use the use-case concept.

Use-Case :- A Use-Case is an interaction b/w user and a system. The use-case model captures the goal of the user and Responsibilities of the system to the user. Use-case is described as -

- Nonformal text with no clear flow of events.
- Easy to read but with a clear flow of events to follow.
- Formal style using Pseudo code.

e.g



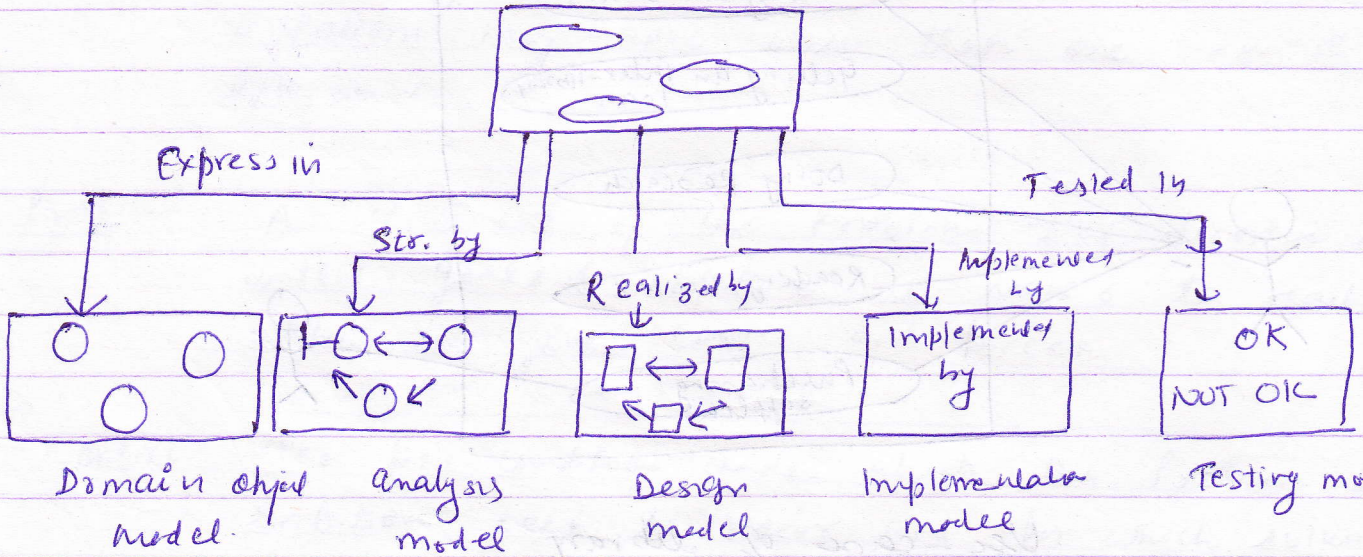
Use-case of library

OBJECT-ORIENTED S/W Engineering : OBJECTORY: ~~Object~~ OOSE also called the Objectory. It is a method of object-oriented development with the specific aim to fit the development of large, real time system

The development Process called Use-Case Driven development & use case scenario begin with a user of system initiating a seq^{n} of interrelated events. It is an approach to object-oriented analysis & design that centers on understanding the ways in which a system actually is used. By organizing the analysis & Design models around sequences of User interaction and actual Usage scenarios, the method produces system that are both more Usable & more Robust.

OBJECTORY IS BUILT around several different models:

- Use case model
- Domain object model
- Analysis Object model
- Implementation model
- Test model.



OBJECT - ORIENTED BUSINESS ENGINEERING : OOBE is object modelling at the enterprise level. Use case are act as central module for s/w engg. Process.

Analysis phase :- Analysis phase defines the system to be built in terms of Problem-domain object model, Requirement model, and the analysis model. The analysis process is iterative but the Requirement and analysis model should be stable before moving on to subsequent models.

Design & Implementation phases : The Implementation phase environment must be identified for the design model. This include factors such as Database System, distribution of process, constraint due to Programming language. The analysis object are translated into design object that fit the current Implementation environment.

Testing :- Finally, Jacobson describe several testing level & Technique. The level include Unit Testing, Integration Testing & System Testing.

PATTERNS :- A pattern involves a general description of a solution to a recurring problem bundle with various goals and constraints. A good Pattern will do the following :-

1) It Solves a Problem
2) It is a Proven concept
3) Solution is not obvious
4) It describe a Relationship
5) Pattern has a significant human Component.

Pattern type :- Generative & Nongenerative Patterns

Generative pattern :- Generative Patterns are patterns that not only describe a recurring problem, they can tell us how to generate something and can be observed in resulting System Architecture they helped shape.

Non-generative pattern → are static & passive. They describe the recurring phenomena without necessarily saying how to reproduce them

Alexander explains that most are generative:
Each Pattern is a rule which describe what you have to do to generate the entity which it define. The pattern in the world merely exist. But the same pattern in our minds are dynamic. They have force. They are generative They tell us what to do, they tell us how we shall or may generate them

Pattern Template :→ Essential component for pattern templates are: Name, Problem, Forces, Solution, Examples, Resulting Context Rationale, Related pattern, Known uses,

Name :- A meaningful name. Good pattern name form a vocabulary for discussing conceptual abstraction. A Pattern may have more than one name in the literature.

Problem :- A statement of the problems that describe its intent: goals & objective it meant's to reach within the given context and forces.

Context: The pre-condition under which the Problem & Solution seen to recur and for which solution is desirable.

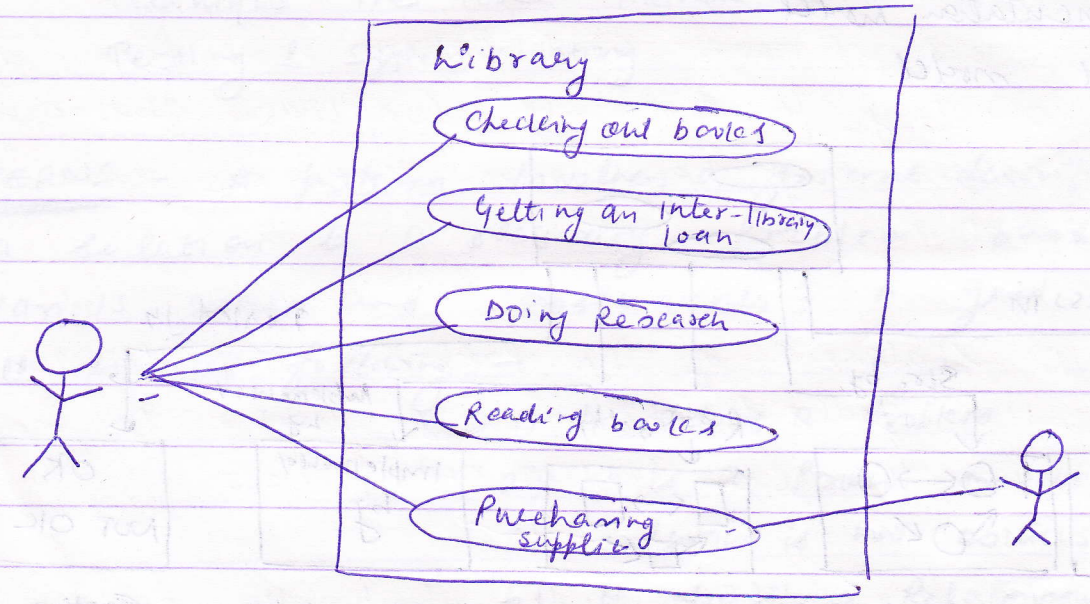
Forces : A description of the relevant forces and constraints and how they interact & conflict with one another and with the goals we wish to achive with some indication of their priorities.

Jacobson ET AL. METHODOLOGIES :- Jacobson et al methodologies (OOSE & OOBE) Cover the entire life cycle and stress traceability b/w different Phases, both forward & backward. This allows the re-usability of code result is reduction of development time. There methodology use the use-case concept.

Use-Case :- A Use-Case is an interaction b/w user and a system. The Use-Case model Captures the goal of the user and Responsibilities of the system to the user. Use-Case is described as -

- Nonformal text with no clear flow of events.
- Easy to read but with a clear flow of events to follow.
- Formal style using Pseudo code.

e.g



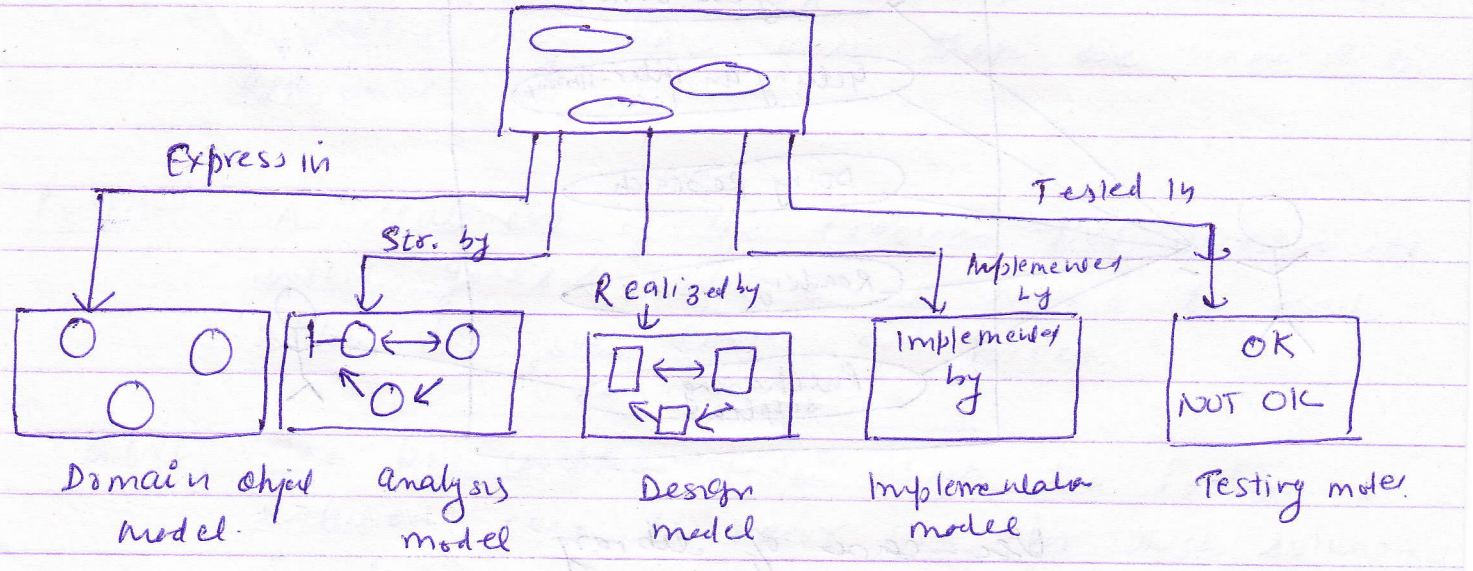
Use-case of library

OBJECT-ORIENTED S/W Engineering : OBJECTORY: ~~Object~~ OOSE also called the objectory. It is a method of object-oriented development with the specific aim to fit the development of large, real time system

The development Process called Use-Case Driven development & use case scenario begin with a user of system initiating a seq^{n} of interrelated events. It is an approach to object-oriented analysis & design that centers on understanding the ways in which a system actually is used. By organizing the analysis & Design models around sequences of user interaction and actual usage scenarios, the method produces system that are both more usable & more robust.

OBJECTORY IS BUILT around several different models:

- Use case model
- Domain object model
- Analysis Object model
- Implementation model
- Test model.



OBJECT - ORIENTED BUSINESS ENGINEERING : OOBE is object modelling at the enterprise level. Use cases are act as central module for s/w engg. Process.

Solution : solution should describe not only the static structure but also the dynamic behavior. static structure tells us the form & organization of pattern, but often the behavioral dynamics is what makes the pattern "come alive".

Example : example help the reader understand the pattern's use and applicability. Visual example and analogies oftenly very useful.

Resulting context : The state or configuration of the system after the pattern has been applied, including consequences (both good & bad) of applying the pattern, and other Problem & Pattern that may arise from new context. It describe post condition & side effect of the pattern.

Rationale : A justifying ~~explanation~~ explanation of steps or rules in the pattern and also of the pattern as a whole in term of how & why it resolves its forces in a particular way to be in alignment with desired goals Principles, and philosophies.

Related pattern : Related pattern often share common forces. They also frequently have an initial or resulting context that is compatible with resulting or initial context of another Pattern.

Known uses : Good pattern often begin with an abstract that provide a short summary or overview. This given a clear picture to the Readers.

Antipatterns :→ A Pattern represents a "best practice" where as an antipattern represents "worst Practice" or a "lesson learned".
Antipatterns come in two varieties

① those describing a bad solution to a problem that resulted in a bad situation.
② those describing how to get out of a bad situation and how to proceed from there to a good solution.

FRAMEWORKS - Frameworks are a way of delivering ~~information~~ application development pattern to support best practice sharing during application development - not just within one company, but across many companies - through an emerging framework market.

A framework is a way of presenting a generic solution to a problem that can be applied to all level in a development. However, design and s/w frameworks are the most popular. A def" of an object-oriented s/w framework is given by Gamma et al.

- A framework is a set of cooperating classes that make up a reusable design for a specific class of s/w. Framework captures the design decision that are common to application domain. Frameworks emphasise design reuse over code-reuse, though a framework will usually include concrete subclasses you can put to work immediately
- A framework is executable s/w, whereas design patterns represent knowledge and experience about s/w.

OBJECT-ORIENTED ANALYSIS: Analysis is the process of extracting the needs of a system and what the system must do to satisfy the user requirements. The goal of OOA is first understand the domain of problem and system responsibilities by understanding how the users use or will use the system.

OOA Process Consists of following steps:

1.) Identify the Actors
2.) Develop a simple business process model using UML Activity diagram
3.) Develop use case
4.) Develop interaction diagrams.
5.) Identify classes.

OBJECT-ORIENTED DESIGN: Booch provides the most comprehensive object-oriented design method. OOD the process consist of:

- Designing classes, their attributes, method, Association, structures and protocols, apply design axioms.
- Design the users layer
- Design and prototype user interface
- User satisfaction and usability tests based on usage
- Iterate and refine the design.

ITERATIVE DEVELOPMENT AND CONTINUOUS SYSTEM → Real Repeat the entire process until, we are satisfy with the system. During this iterative process, our prototype will be incrementally transformed into the actual application. UA encourages the integration of testing plan from day 1 of the project. Usage scenarios can become test scenarios; therefore use cases will drive the usability testing. Usability testing is the process in which the functionality of S/W is measured.

The UA Proposed Repository: The idea promoted here to create a repository that allow the maximum re-use of previous experience and previously defined objects, patterns, frameworks and User Interfaces in an easily accessible manner with a completely available and easily utilized format. The advantage of Repositories from these projects might be useful. We can select any piece from a repository - from the definition of one data element, to a diagram, all its symbols, and all their dependent definitions, to entries - for reuse.

- The UA's underlying assumption is that if we design & develop application based on previous experience, creating additional application will require no more than assembling component from the library.

The repository should be accessible to many people. Furthermore, it should be relatively easy to ~~not~~ search the Repository for classes based on their attributes, methods or other characteristics.

The layered Approach to software development:-

layer approach consist of:

- The Business layer
- The User Interface layer
- The Access layer.

Business layer → It contains all the objects that represent the business (both data & behavior)

User Interface layer - It consist of objects with which the user interacts as well as objects needed to manage or control the interface.

The Access layer - The access layer contains objects that know how to communicate with the place where the data actually reside.

UML (Unified Modeling Language) - UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English, in a form known as object constraint language.
Primary goals in the design of UML were as follows

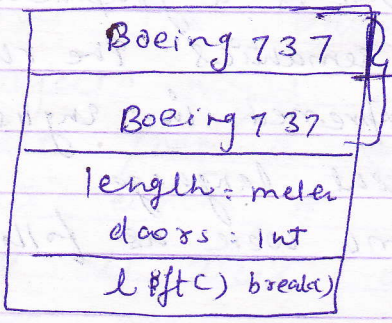
1.) Provide uses a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful model.
2.) Be Independent of particular Prog^{m} language and development process.
3.) Provide a formal basis for understanding the modeling language.
4.) Integrate best practices and methodologies.

UML DIAGRAM → UML Define nine Graphical diagram

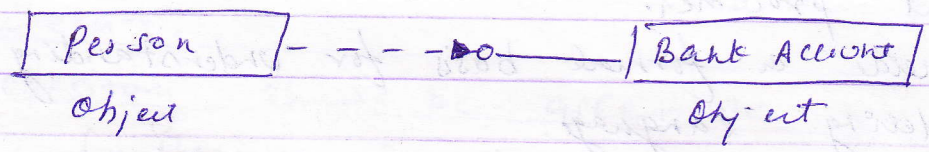
1.) Class Diagram (static)
2.) Use-case Diagram
3.) Behavior Diagram
4.) - Interaction Diagram
    - Sequence Diagram
    - Collaboration diagram
  - State chart Diagram
  - Activity Diagram
4.) Implementation Diagram
  - Component Diagram
  - Deployment Diagram

UML CLASS DIAGRAM – These Diagram show the static structure of model.

Class Notation – A class is drawn as a rectangle with three component separated by horizontal lines

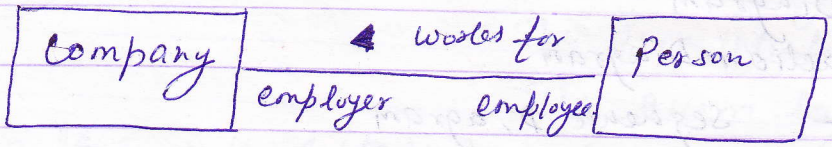


CLASS INTERFACE NOTATION – This notation is used to describe the externally visible behavior of a class for eg.

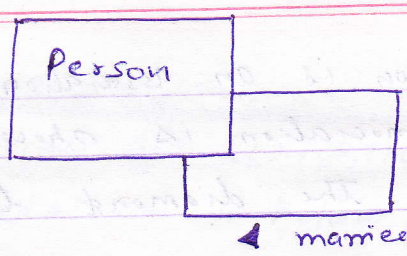


Person object may need to interact with the Bank account object to get the Balance.

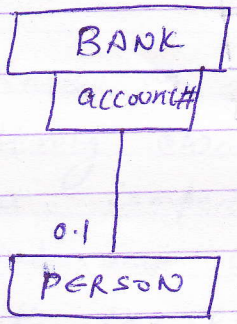
Binary Association Notation – A Binary Association is drawn as a solid path connecting two classes, OR both ends may be connected to the same class.



Association Role – Technical term for it is - Binary association. is drawn as a solid line connecting two class symbols. The end of an association, where it connects to a class, shows the association role.



Qualifier - A qualifier is an association attribute. for eg. a person object may be associated to a bank object An attribute of this association is the account#. the account# is the qualifier of this association.



MULTIPLICITY - specifies the Range of allowable associated clases. It is given for roles with-in association, Parts, within compositions, repetitions, and other purposes.

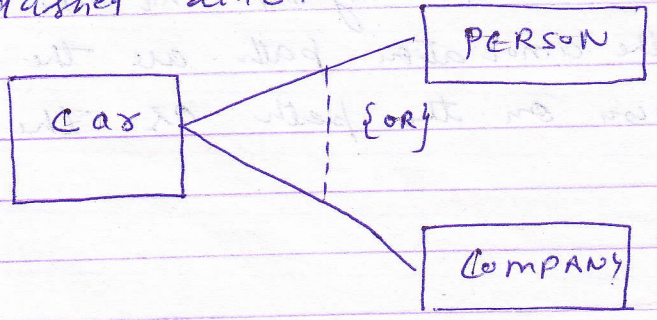
- shown by lower bound : upper bound

0 .. 1

0 .. * ← unlimited upper bound

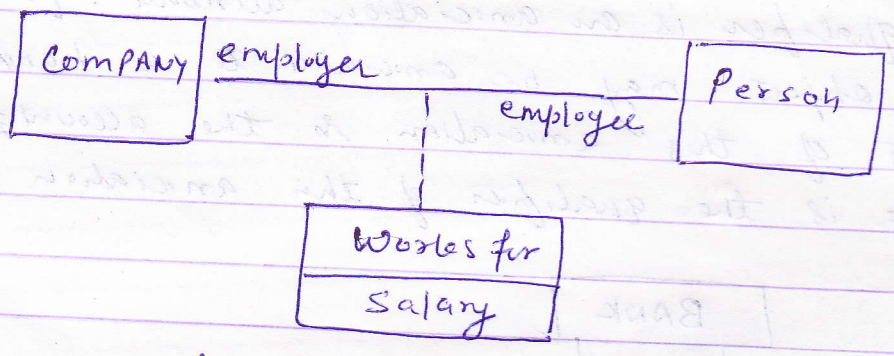
lower bound.

OR Association - This is shown as a dashed line connecting two or more associations, all of which must have a class in common, with the Constant string { } labeling the dashed line.

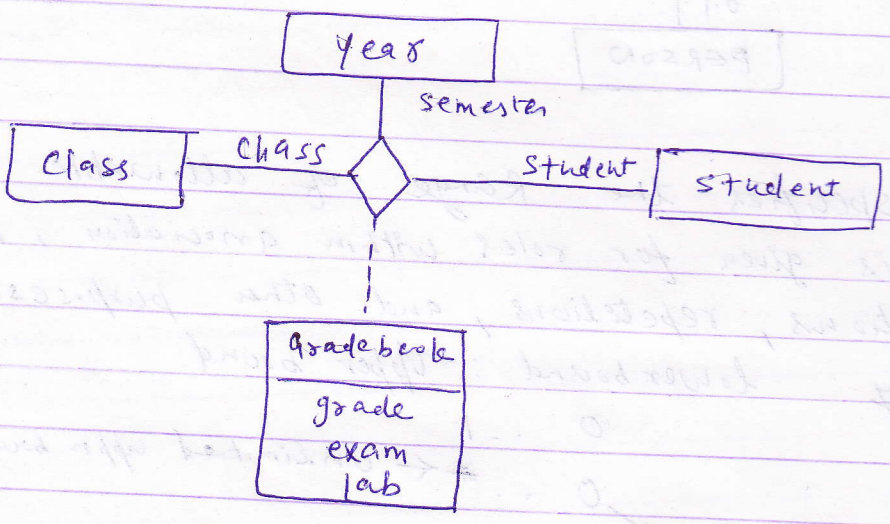


OR NOTATION.

N-ARY Association - An n-ary notation is an association among more than two classes. This association is shown by a large diamond with a path from the diamond to each participant class.



Association class

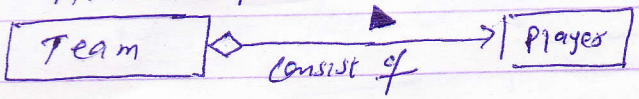


Ternary Association

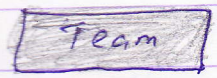
Association class - An association class is an association that also has class properties. It is shown by class symbol attached by a dashed line to an association path. The name in the class symbol and the name string attached to the association path are the same. The name can be shown on the path or the class symbol or both.

MICROPROGRAMMED CONTROL UNIT

AGGREGATION AND COMPOSITION (a-part-of) - Aggregation is a form of association. A hollow diamond is attached to the end of the path to indicate aggregation. However, the diamond may not be attached to both ends of a line, and it need not be presented at all



Composition - also known as the part of, is a form of aggregation with strong ownership to represent the component of a complex object. Composition also is referred to as a part-whole relationship. UML Notation for Composition is a solid diamond at the end of a path.



Generalization: is a relationship between a more general class and a more specific class. Generalization is displayed as a directed line with a closed, hollow arrowhead at the superclass end.