CMR INSTITUTE OF TECHNOLOGY

Internal Assessment Test 1 – September 2019

Solutions

Sub:	Software Engineering							**Code:**	18CS35
Date:	06/ 09/2019	Duration:	90 mins	Max Marks:	50	**Sem:**	III	**Branch:**	CSE / ISE

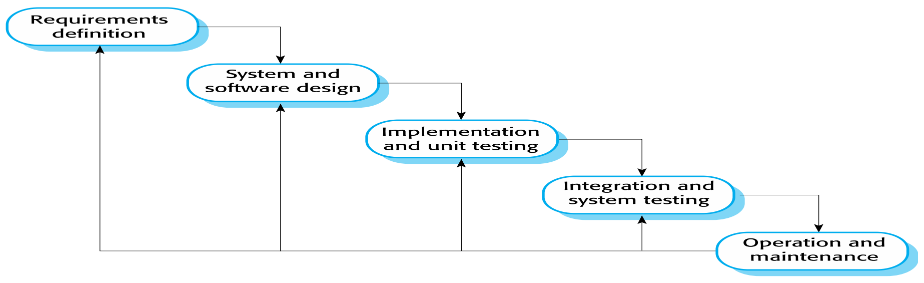
Note: Answer any five questions:

1 **10M**

a) What are the Attributes of Good software? (4M)

1. Maintainability - Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
2. Dependability and security - Software dependability includes a range of characteristics including reliability, security and safety.Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
3. Efficiency - Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
4. Acceptability - Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

b) Explain the Waterfall model with a neat diagram.(6M)


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

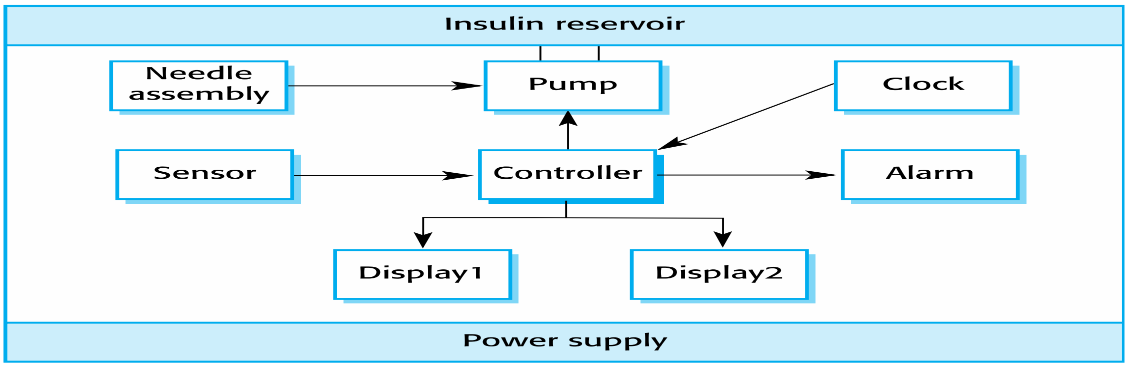
- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance

The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase. Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process. Few business systems have stable requirements.The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

2

10M

a) Explain an Insulin pump control system with a neat block diagram.(8M)



Comment on the essential high level requirements that this system must meet. Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected. Calculation based on the rate of change of blood sugar levels. Sends signals to a micro-pump to deliver the correct dose of insulin. Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage. The system shall be available to deliver insulin when required. The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.The system must therefore be designed and implemented to ensure that the system always meets these requirements.

b) List the fundamental activities of Software Engineering.(2M) – any 2

Some fundamental principles apply to all types of software system, irrespective of the development techniques or types used:

a. Systems should be developed using a managed and understood development process. Of course, different processes are used for different types of software.

b. Dependability and performance are important for all types of system.

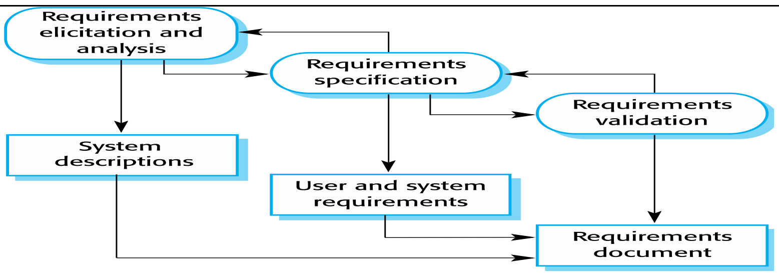
c. Understanding and managing the software specification and requirements (what the software should do) are important.

d. Where appropriate, you should reuse software that has already been developed rather than write new software.

3

a. With a neat diagram, explain the Requirement Engineering Process.(6M)

10M


Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

Requirements engineering process

- **Requirements elicitation and analysis**
  - What do the system stakeholders require or expect from the system?
  - Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
  - Elicitation (intelligence), collecting intelligence information from people as part of human intelligence
- **Requirements specification**
  - Defining the requirements in detail
  - The process of writing the user and system requirements in a requirements document.
  - User requirements have to be understandable by end-users and customers who do not have a technical background.
  - System requirements are more detailed requirements and may include more technical information.
  - The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.
  -
- **Requirements validation**
  - Checking the validity of the requirements
  - Concerned with demonstrating that the requirements define the system that the customer really wants.
  - Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

b)Differentiate between Functional and Non-Functional Requirements with an example for each.(4M)

Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

May state what the system should not do. Describe functionality or system services.

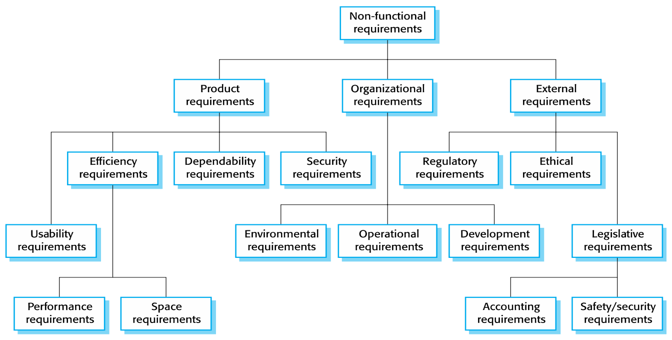
Depend on the type of software, expected users and the type of system where the software is used. Functional user requirements may be high-level statements of what the system should do.Functional system requirements should describe the system services in detail. Problems arise when functional requirements are not precisely stated.Ambiguous requirements may be interpreted in different ways by developers and users.Consider the term ‘search’ in requirement 1 -User intention – search for a patient name across all appointments in all clinics; Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

Mentcare system: functional requirements- A user shall be able to search the appointments lists for all clinics. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day. Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Non-functional requirements

Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.Often apply to the system as a whole rather than individual features or services.These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.

Process requirements may also be specified mandating a particular IDE, programming language or development method. Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless. Non-functional requirements may affect the overall architecture of a system rather than the individual components. For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components. A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.



Examples of non-functional requirements in the Mentcare system

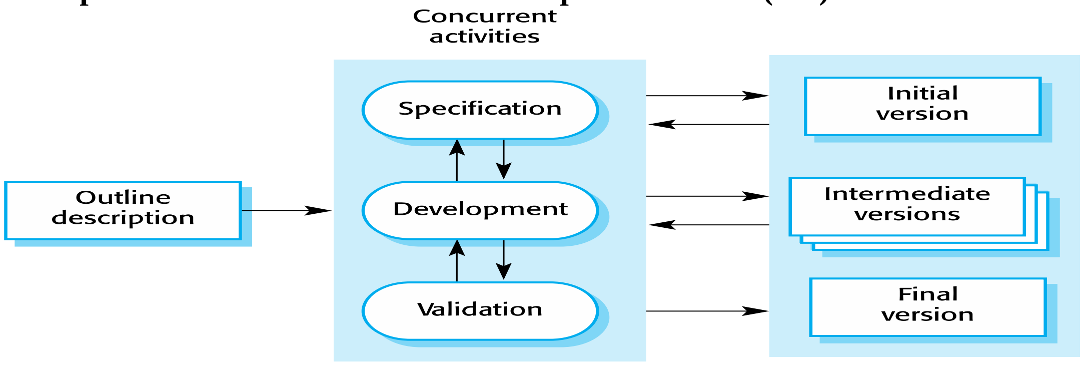
Product requirement

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

4 **a) Draw block diagram for illustrating incremental development model. State minimum two benefits and problems in the incremental development model. (7M)** 10M



Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

Benefits –

The cost of accommodating changing customer requirements is reduced.

– The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.

It is easier to get customer feedback on the development work that has been done.

– Customers can comment on demonstrations of the software and see how much has been implemented.

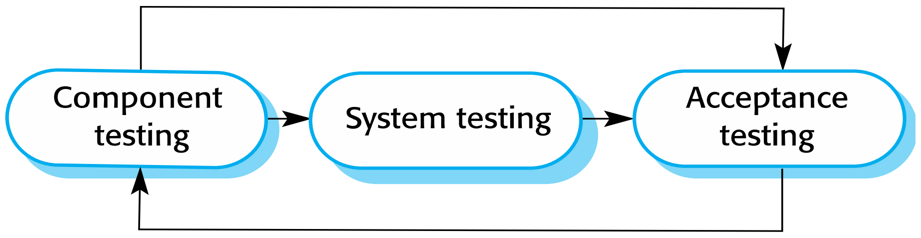
More rapid delivery and deployment of useful software to the customer is possible.

– Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Problems –

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

b) Explain the different stages in Software Testing. (3M)



- Component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Acceptance/Customer testing

– Testing with customer data to check that the system meets the customer’s needs.

5

a)Elaborate the major themes that are supported in object oriented technology. (6M) 10M

1. Abstraction
Abstraction lets you focus on essential aspects of an application while ignoring details i.e focusing on what an object is and does, before deciding how to implement it. It’s the most important skill required for OO development.

2. Encapsulation (information hiding)
It separates the external aspects of an object (that are accessible to other objects) from the internal implementation details (that are hidden from other objects). Encapsulation prevents portions of a program from becoming so interdependent that a small change has massive ripple effects.

3. Combining data and behavior
Caller of an operation need not consider how many implementations exist. In OO system the data structure hierarchy matches the operation inheritance

4. Sharing
OO techniques provide sharing at different levels. Inheritance of both data structure and behavior lets sub classes share common code. OO development not only lets you share information within an application, but also offers the prospect of reusing designs and code on future projects.

5. Emphasis on the essence of an object
OO development places a greater emphasis on data structure and a lesser emphasis on procedure structure than functional-decomposition methodologies.

6. Synergy
Identity, classification, polymorphism and inheritance characterize OO languages.Each of these concepts can be used in isolation, but together they complement each other synergistically.

b)Which are the models used to describe a system from different viewpoints? (4M)

A model is an abstraction, before building any system a prototype may be developed. The main purpose of model is for understanding of the system.We use three kinds of models to describe a system from different view points.

1.Class Model for the objects in the system & their relationships.

2. State model—for the life history of objects. Show how systems behave internally

3**. Interaction Model**—for the interaction among objects. Show the behaviour of systems in terms of how objects interact with each other

---

6 With the help of a sample class model explain the following; 10M
i. Association and Association end name
ii. Qualified association
iii. Association classes
iv. Multiplicity

I. Association and Association end name

- Associations are the means for establishing relationships among classes.An association is a description of a group of links with common structure and common semantics.E.g. a person WorksFor a company. If two classes in a model need to communicate with each other, there must be link between them, and that can be represented by an association (connector).
- Associations are inherently bi-directional. The association name is usually read in a particular direction but the binary association may be traversed in either direction. Association can be represented by a line between these classes **with an arrow indicating the navigation direction**. In case arrow is on the both sides, association has bidirectional association.

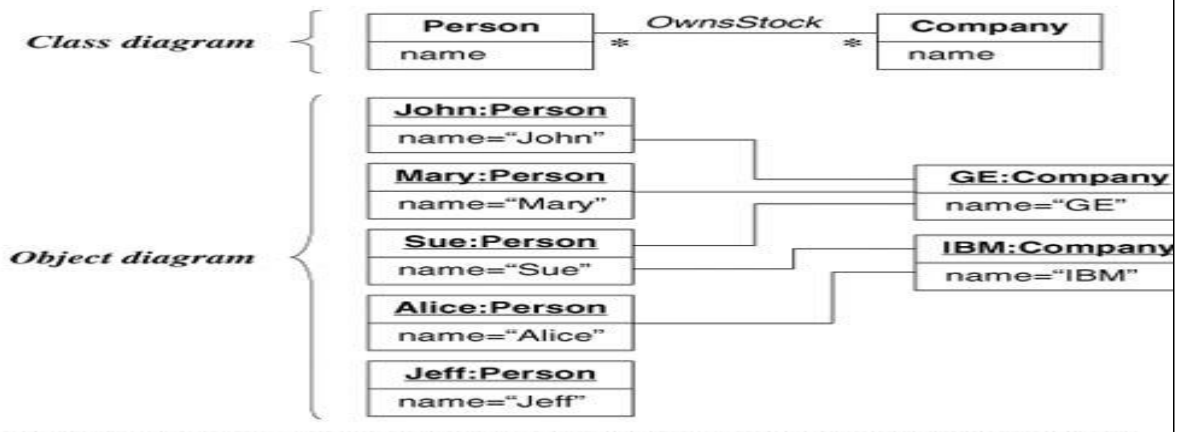
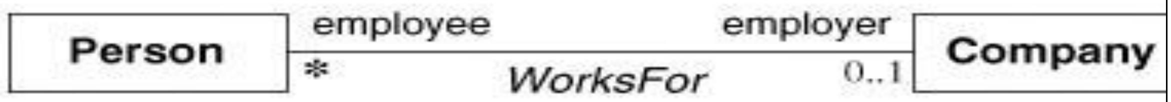


Figure 3.7 Many-to-many association. An association describes a set of potential links in the same way that a class describes a set of potential objects.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Association connects related classes and is also denoted by a line.Show association names in italics.

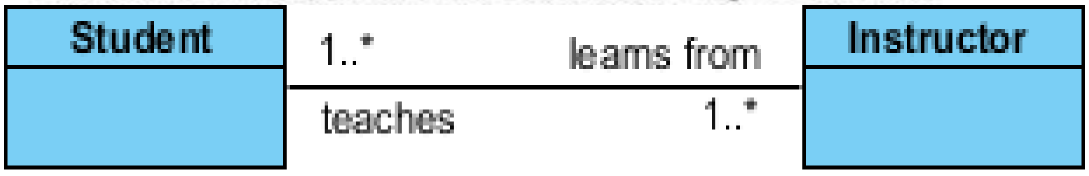
- **Association end name** Associations have ends. They are called 'Association Ends'. They may have names (which often appear in problem descriptions). Use of association end names is optional. But association end names are useful for traversing associations.



employee	employer
Joe Doe	Simplex
Mary Brown	Simplex
Jean Smith	United Widgets

Figure 3.12 Association end names. Each end of an association can have a name.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.



- **<u>II.Qualified association</u>**

- A qualified association is an association in which an attribute called Qualifier the objects for a 'many' association' end. A qualifier selects among the target objects, reducing the effective multiplicity from 'many' to 'one'.Both below models are acceptable but the qualified model adds information.

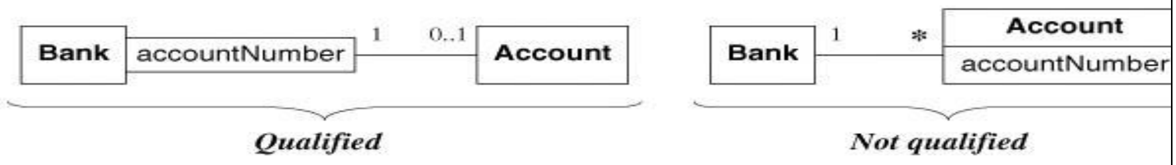


Figure 3.22 Qualified association. Qualification increases the precision of a model.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

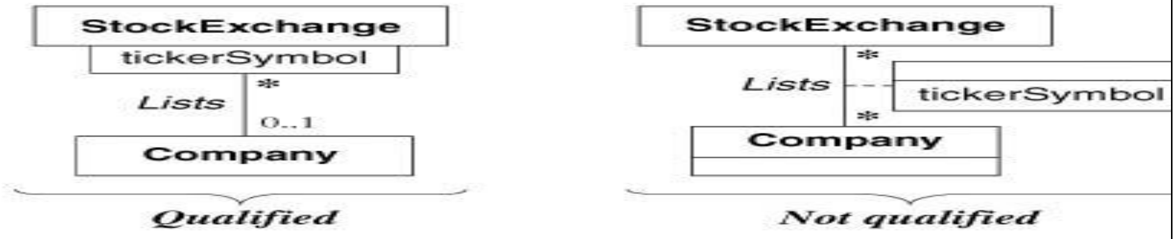
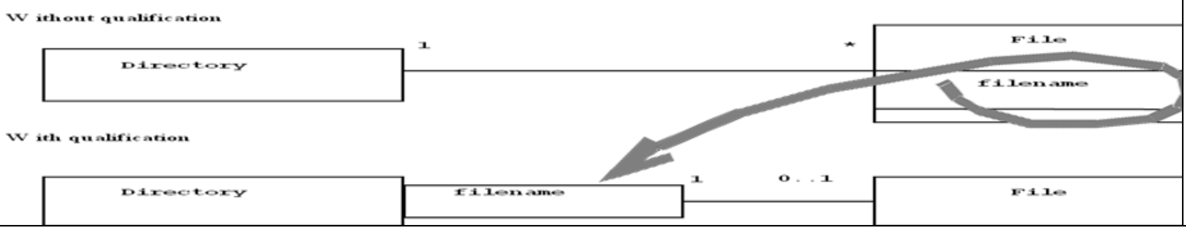


Figure 3.23 Qualified association. Qualification also facilitates traversal of class models.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

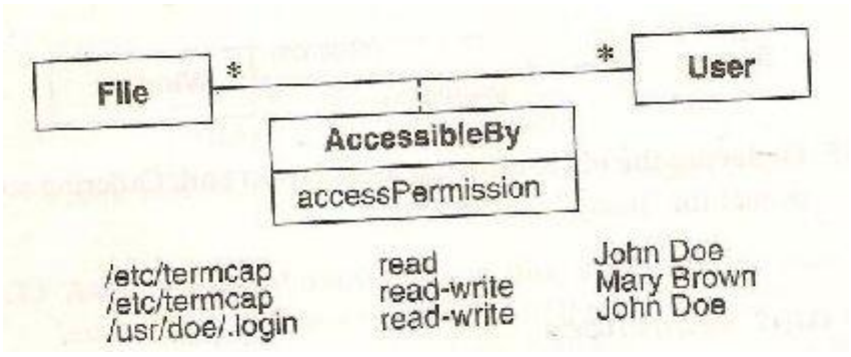
Adding a qualifier clarifies the class diagram and increases the conveyed information. In this case, the model including the qualification denotes that the name of a file is unique within a directory. Example of how a qualified association reduces multiplicity (UML class diagram).



-

III. Association classes

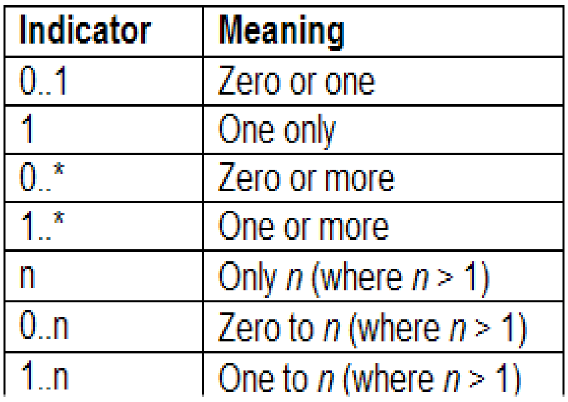
An **association class** is an association that is also a class.Like the links of an association, the instances of an association class derive identity from instances of the constituent classes. Like a class, an association class can have attributes and operations and participate in associations.



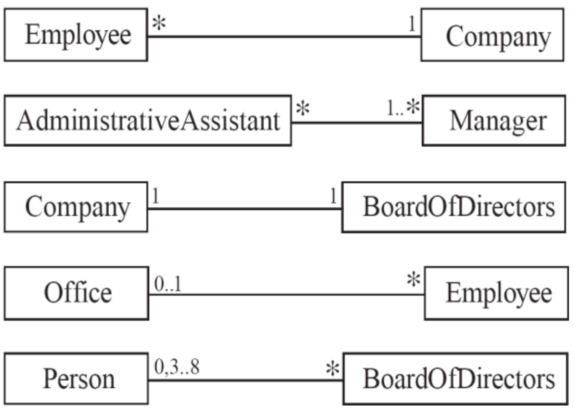
IV. Multiplicity

Multiplicity defines the number of objects associated with an instance of the association.

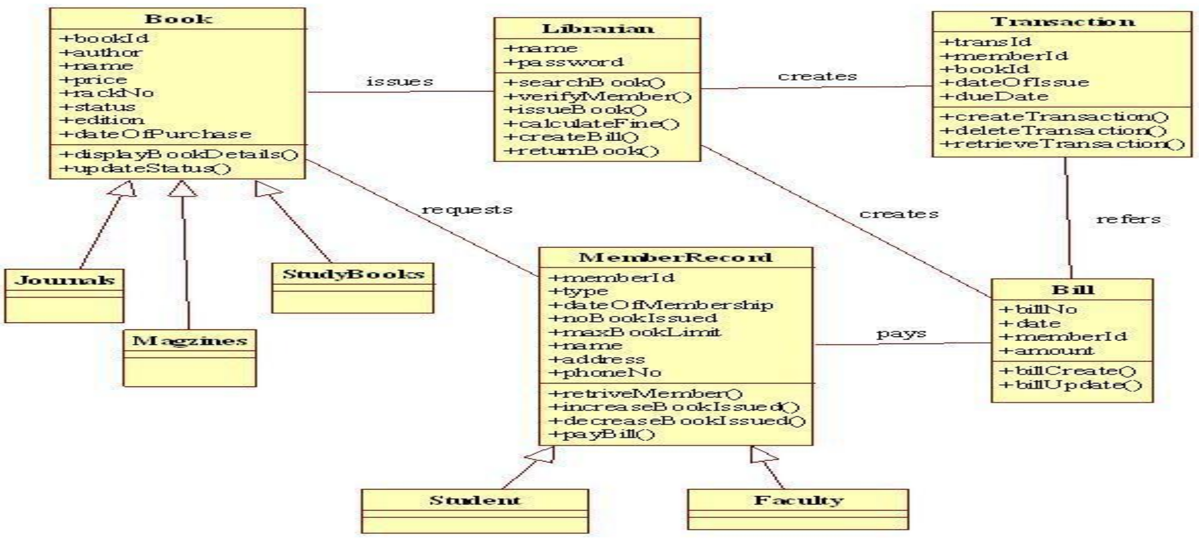
UML diagrams explicitly list multiplicity at the end of association lines.Intervals are used to express multiplicity:



Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
n	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)



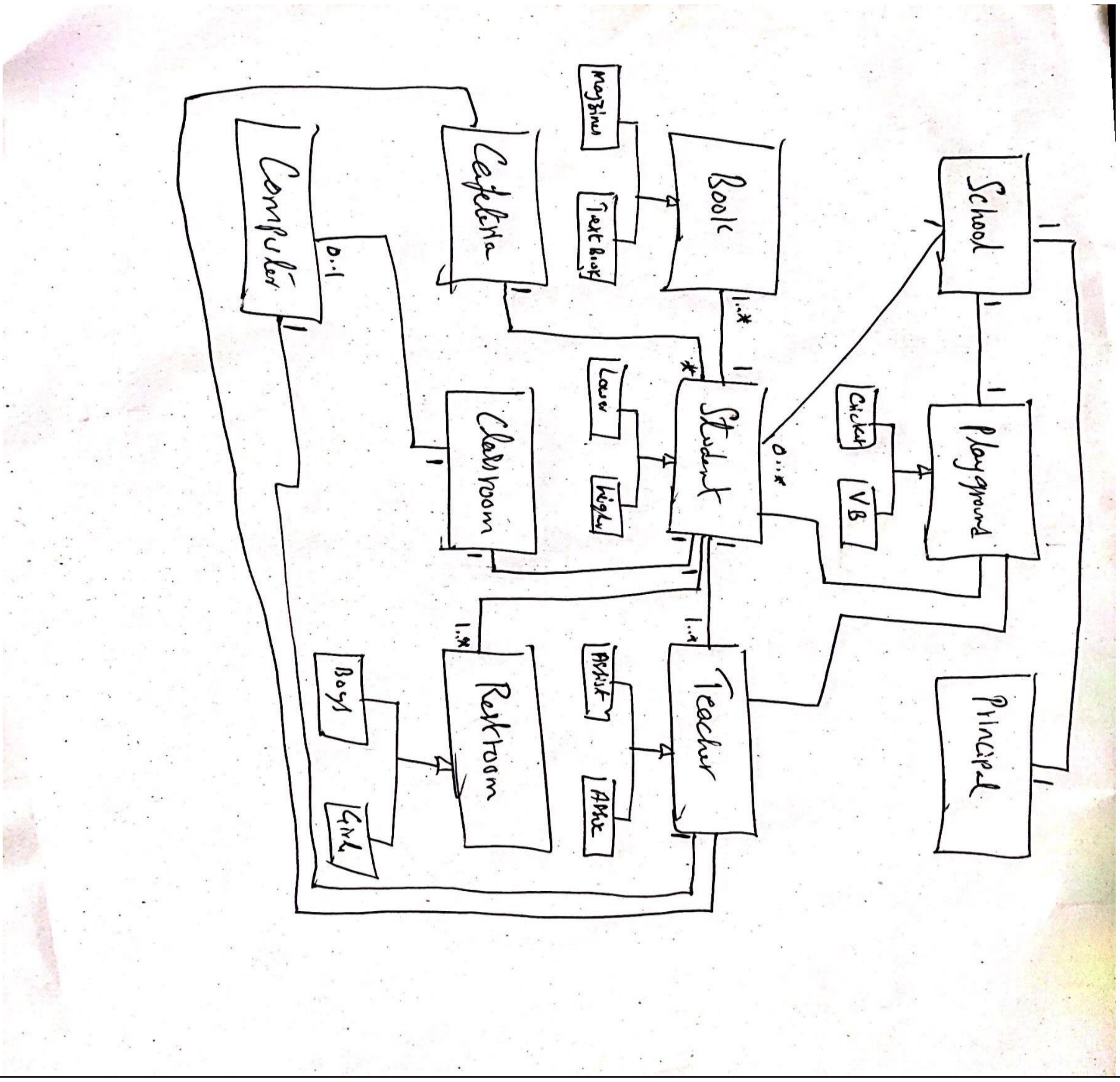
7. a) Design a class diagram for library management system.(3)



b)Design a class diagram for the following group of classes;

- **School, playground, principal, book, student, teacher, cafeteria, class room, rest room, and computer.**

In the class diagram, add minimum three relationships (associations, generalization). Use association names where ever needed and show multiplicity. (7)



Magzine
Text Book
Book
School
Computer
Cafeteria
0..1
1..*
*
Lower
Higher
Classroom
Student
0..*
Cricket
VB
Playground
1..*
1..*
Boys
Girl
Restroom
Assist
Admin
Teacher
Principal
1