USN

Internal Assessment Test 1 – Feb. 2021

Sub:	**Software Engineering**				Sub Code:	**18CS35** Branch:	**CSE**
Date:	06 -02 - 21	Duration:	90 min's	Max Marks: 50	Sem / Sec:	III (Diploma Sections)	OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1.	What is Software Engineering? Explain Software engineering code of Ethics. List attributes of good software	[2+6+2]	CO1	L1
2(a)	Explain fundamental activities in Software Engineering.	[04]	CO1	L2
(b)	Explain the organization of Wilderness weather station system.	[06]	CO5	L3
3.	With neat diagram explain any two software development model. Enlist their pros and cons	[10]	CO1	L3
4.	Explain the organization of MHC-PMS (Mental Health Care Patient Management System) and also lists the goals and features of the system?	[10]	CO5	L3
5.	What is object oriented development (OOD)? Explain different stages in OOD.	[4+6]	CO5	L2
6.	Explain following terms Association, Links, Multiplicity, Sequence, Bag.	[2X5]	CO5	L2
7. (a)	Car can be manufacture by using following components like engine, gear box, brake, body and suspension. Its an one-to-one relationship with car for all the above components except brake and suspension. Based on the requirements one car may have many brakes and suspensions. Each brake is associated with each wheel and each wheel is associated with each tire. To have a smooth run on highways suspension is associated with wheel. Draw the class diagram by using multiplicity indicators.	[07]	CO5	L3
(b)	Draw association diagram for: A secretary can work for one or many managers. A manager can have many secretaries. Managers can have a group of secretaries. Some managers might have zero secretaries.	[03]	CO5	L3

Scheme of evaluation

Qno	Questions	Marks distribution
1	What is Software Engineering? Explain Software engineering code of Ethics. List attributes of good software	2M 6M 2M
2a	Explain fundamental activities in Software Engineering	4M
2b	Explain the organization of Wilderness weather station system. Diagram	4M 2M
3	Software development model. diagram Enlist their pros and cons	2M*2 1M*2 2M*2
4	The organization of MHC-PMS (Mental Health Care	8M

	Patient Management System) and also lists the goals and features of the system Diagram	 2M
5	What is object oriented development (OOD)? Explain different stages in OOD	4M 6M
6	Explain following terms Association, Links, Multiplicity, Sequence, Bag.	2*5M
7a	Car can be manufacture by using following components like engine, gear box, brake, body and suspension. Its an one-to-one relationship with car for all the above components except brake and suspension. Based on the requirements one car may have many brakes and suspensions. Each brake is associated with each wheel and each wheel is associated with each tire. To have a smooth run on highways suspension is associated with wheel. Draw the class diagram by using multiplicity indicators.	7M
7b	Draw association diagram for: A secretary can work for one or many managers. A manager can have many secretaries. Managers can have a group of secretaries. Some managers might have zero secretaries	3M

Solution:

1. A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective. A structured set of activities required to develop a software system. Many different software processes but all involve:
Specification – defining what the system should do;
Design and implementation – defining the organization of the system and implementing the system;
Validation – checking that it does what the customer wants;
Evolution – changing the system in response to changing customer needs.

Software engineering involves wider responsibilities than simply the application of technical skills. Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals. Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.
Issues of professional responsibility
Confidentiality -Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
Competence - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is out with their competence.

Intellectual property rights - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
Computer misuse - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

Eight Principles of Ethics –
PUBLIC - Software engineers shall act consistently with the public interest.
CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

List attributes:

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.

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.

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.

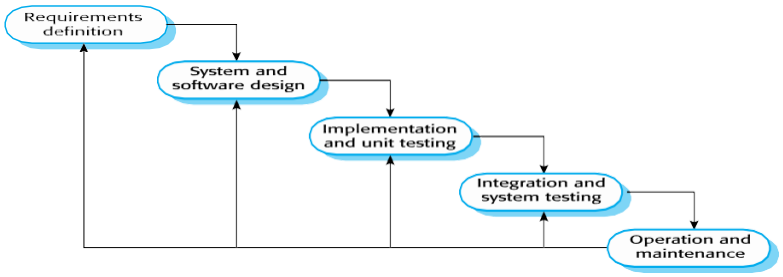
Acceptability

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.

2a) Explain fundamental activities in Software Engineering

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.

2b) Wilderness weather station

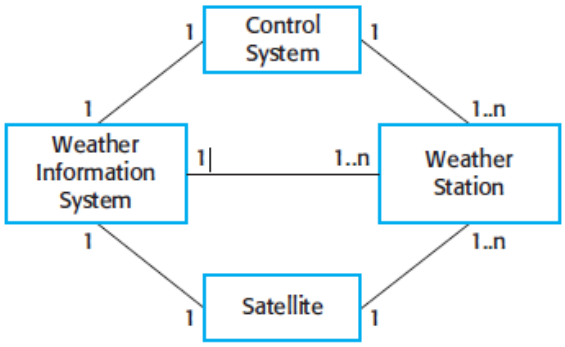
Setting the system boundaries helps you decide what features are implemented in the system being designed and what features are in other associated systems. In this case, you need to decide how functionality is distributed between the control system for all of the weather stations, and the embedded software in the weather station itself.

System context models and interaction models present complementary views of the relationships between a system and its environment:

1. A system context model is a structural model that demonstrates the other systems in the environment of the system being developed.
2. An interaction model is a dynamic model that shows how the system interacts with its environment as it is used.

The context model of a system may be represented using associations. Associations simply show that there are some relationships between the entities involved in the association. The nature of the relationships is now specified. You may therefore document the environment of the system using a simple block diagram, showing the entities

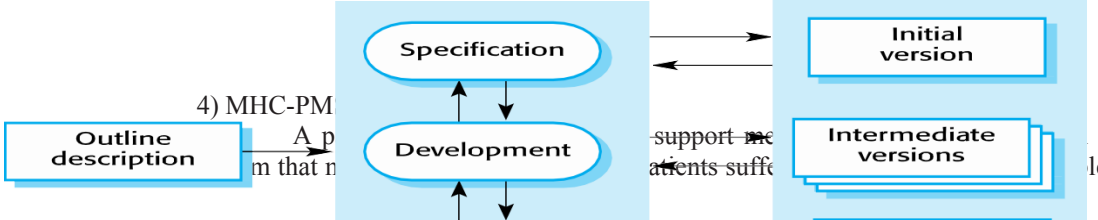
in the system and their associations. This is illustrated in Figure, which shows that the systems in the environment of each weather station are a weather information system, an onboard satellite system, and a control system. The cardinality information on the link shows that there is one control system but several weather stations, one satellite, and one general weather information system.
When you model the interactions of a system with its environment you should use an abstract approach that does not include too much detail. One way to do this is to use a use case model.



The systems in the environment of each weather station are a weather information system, an onboard satellite system, and a control system. The cardinality information on the link shows that there is one control system but several weather stations, one satellite, and one general weather information system.
When you model the interactions of a system with its environment you should use an abstract approach that does not include too much detail. One way to do this is to use a use case model.

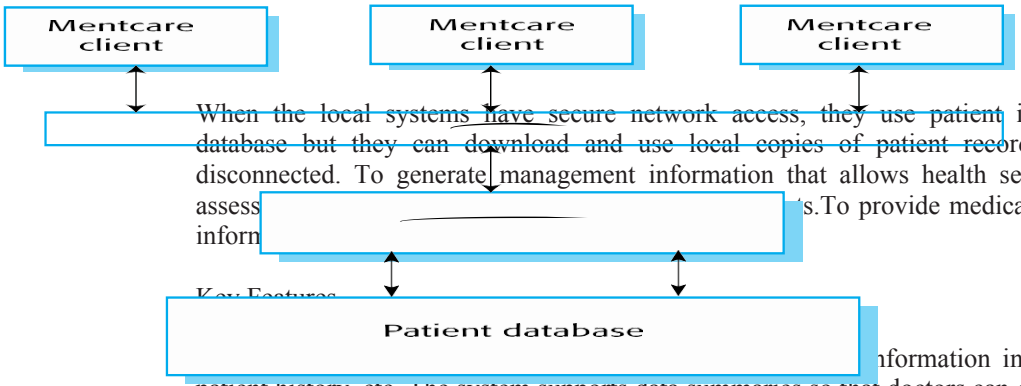
3)
Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.
Benefits –
The cost of accommodating 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.



4) MHC-PM
A p
m that n
support me
atients suffe
information
lems and the

treatments that they have received. Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems. To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres. Mentcare is an information system that is intended for use in clinics. It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.



When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected. To generate management information that allows health service managers to assess s.To provide medical staff with timely inform

Key Features

nformation in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.

- Patient monitoring

– The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.

- Administrative reporting

The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc

Q5. What is object oriented development (OOD)? Explain different stages in OOD.

The process for OO development and graphical notation for representing OO concepts consists of building a model of an application and then adding details to it during design. The methodology has the following stages:

System conception : Software development begins with business analysis or users conceiving an application and formulating tentative requirements

Analysis : The analyst must work with the requestor to understand the problem, because problem statements are rarely complete or correct. The analysis model is a precise abstraction of what the desired system must do, not how it will be done. It should not contain implementation decisions.The analysis model has 2 parts:

- Domain model - a description of the real-world objects reflected within the system Eg: Domain objects for a stock broker
- Application – model - a description of the parts of the application system itself that are visible to the user.

Eg:- Application might include stock, bond, trade and commission.

Application objects might control the execution of trades and present the results.

System design: The development teams devise a high – level strategy – the system architecture for solving the application problem.

Class design : The class designer adds details to the analysis model in accordance with the system design strategy. The focus of class design is the data structures and algorithms needed to implement each class.

Implementation : Implementers translate the classes and relationships developed during class design into particular programming language, database or hardware. During implementation, it is important to follow good software engineering practice so that traceability to the design is apparent and so that the system remains flexible and extensible.

Q6. Explain following terms Association, Links, Multiplicity, Sequence, Bag.

1. **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.

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.

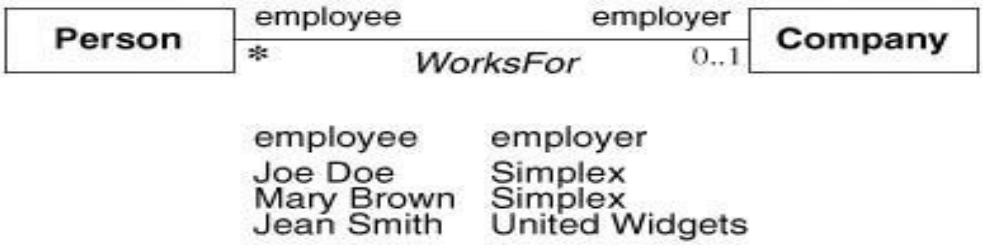


Figure 3.12 Association end names. Each end of an association can have a name.

II. Qualified association

- 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.

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

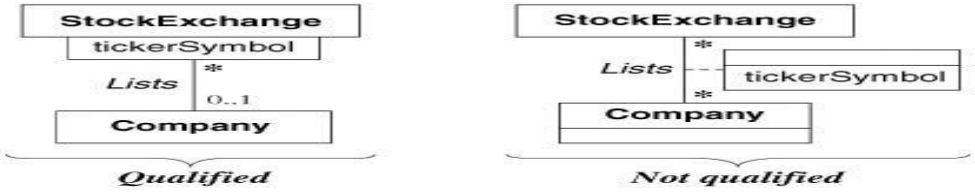
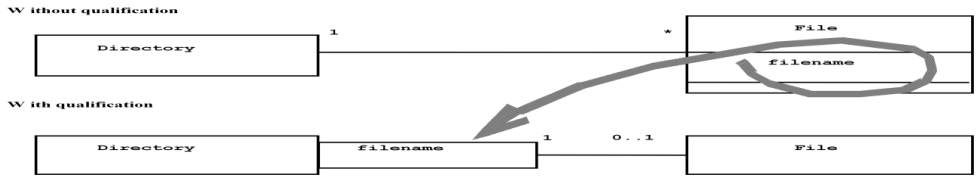


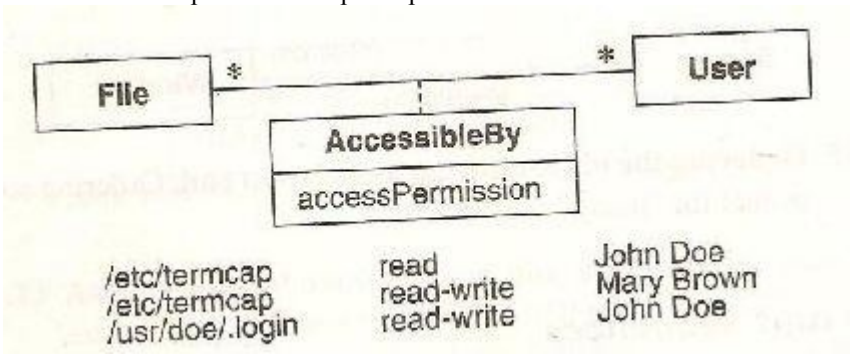
Figure 3.23 Qualified association. Qualification also facilitates traversal of class models.

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.

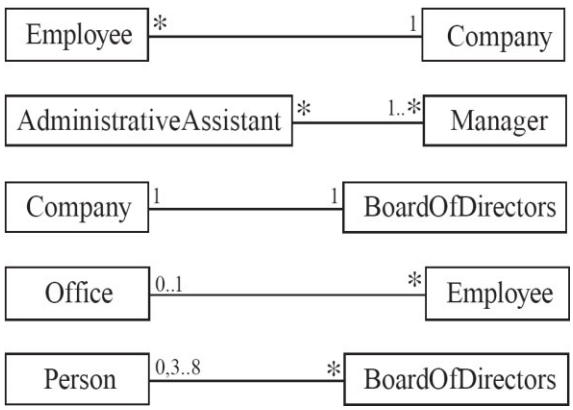


IV. Multiplicity

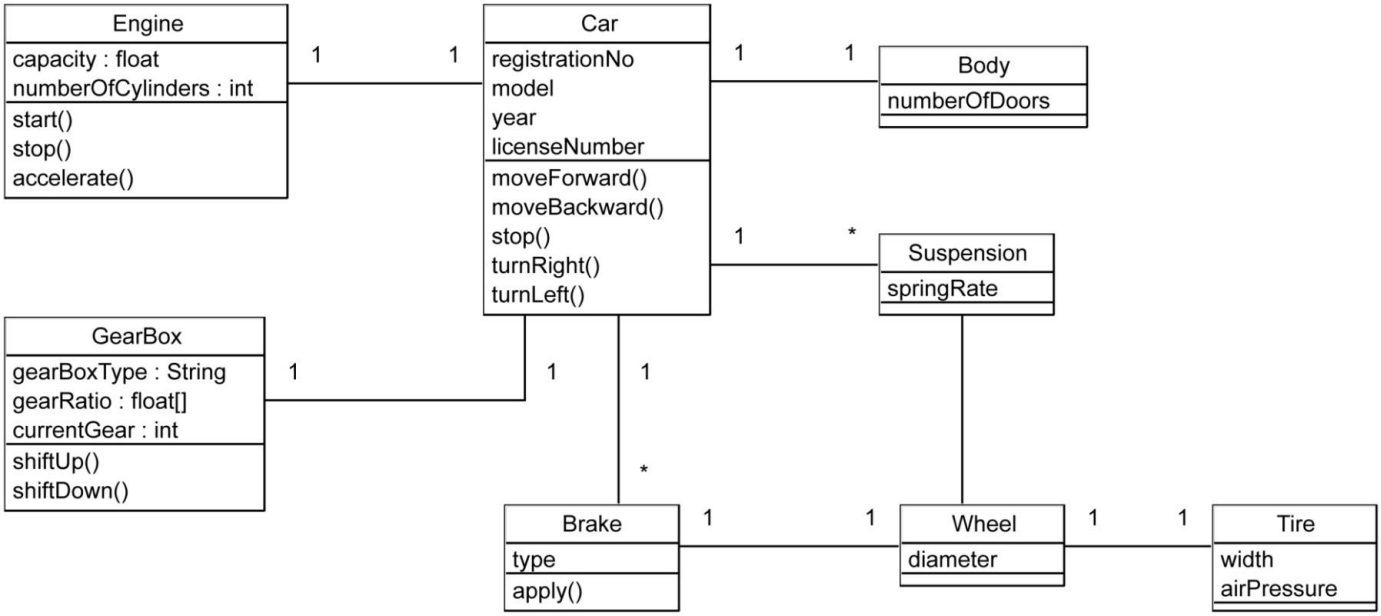
Multiplicity defines the number of objects associated with an instance of the association.

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)



7a)



7b)

Analyzing and validating associations

- A secretary can work for one or many managers
- A manager can have many secretaries
- Managers can have a group of secretaries
- Some managers might have zero secretaries.