Total No. of Pages:01 Roll No.....

6th SEMESTER B.Tech.(CSE)

MID SEMESTER EXAMINATION (March-2020)

CO 326: Object Oriented Software Engineering

Time: 1:30 Hours Max. Marks: 30

Note: Answer all questions.
Assume suitable missing data, if any.

Total no. [redacted] Roll no. 2K20/CO/[redacted]

B.Tech.

MID TERM EXAMINATION March-2023

CO326 OBJECT ORIENTED SOFTWARE ENGINEERING

Time: 01:30 Hours Max. Marks: 30

Note : All questions carry equal marks.
Assume suitable missing data, if any.

Total No. of Pages

SIXTH SEMESTER — 124 — B.Tech.

MID SEMESTER EXAMINATION (March-2019)

CO326 OBJECT ORIENTED SOFTWARE ENGINEERING

Time: 1.5 Hours Max. Marks: 30

Note: Answer ALL questions.
Assume suitable missing data, if any.

Total No. of Pages 2 Roll No. .....

SIXTH SEMESTER B.Tech. (SE)

END SEMESTER EXAMINATION May-2019

CO326 OBJECT ORIENTED SOFTWARE ENGINEERING

Time: 3 Hours Max. Marks: 40

Note: Question No. 1 is compulsory.
Attempt any THREE from the remaining questions.
Assume suitable missing data, if any.

Total no. of Pages:01 Roll no. 2K10/IO/[redacted]

VI SEMESTER

B.Tech.

END TERM EXAMINATION May-2023

CO326 OBJECT-ORIENTED SOFTWARE ENGINEERING

Time: 03:00 Hours Max. Marks: 40

Note : Answer any *four* questions.
Assume suitable missing data, if any.

Page No.
Date

Q 1. Give the characteristics of an object-oriented system. What are the main advantages of object- oriented software development? 2020M (5)

Characteristics of object oriented system:

1. Abstraction: Simplifies complex reality by modeling relevant classes with attributes and operations.
2. Encapsulation: Hides internal state of objects, allowing access only through defined operations.
3. Inheritance: Allows creation of new classes based on existing ones inheriting attributes and behaviors.
4. Polymorphism: Enables objects of diff. classes to be treated as objects of a common superclass.
5. Modularity: Divides system into smaller, self contained units (objects) for independent development and maintenance.
6. Message Passing: Facilitates communication b/w objects by sending/receiving msg.

Advantages

→ Allows creation of reusable components saving time and effort in dev.
→ Allows for addition of new features and scalability.
→ Easier maintenance and modifications due to localized changes in object/classes.
→ Models/mirrors real world scenarios for easier representation and manipulation
→ Promotes collaboration among dev. teams by providing common framework and language.

Q 2. Consider the employee schema (emp-ID, emp-name, phone, email street and city). Give the class representation along with the attribute types. 2020M (4)

Page No. | Date

Q1. Answer the following:

a) Define the object "Employee" with possible attributes and operations. 2019M

```
class Employee {
public:
    int emp-ID;
    long phone;
    String name, email;
    String street, city;

    void displayDetails() {
        cout << "ID:" << emp-ID;
        cout << "Name:" << name;
        cout << "Phone:" << Phone;
    }

    void updatePhone(long & newPhone) {
        phone = newPhone;
        cout << "Updated";
    }

    void updateEmail(String & newEmail) {
        email = newEmail;
        cout << "Update";
    }
};
```

Employee
emp-ID : int name : String phone : long email : String city : String
displayDetails() updatePhone() updateEmail()

d) Define Actor. 2019M

An actor refers to an entity that plays a distinct role in interacting with or using the system. These roles often correspond to different user types, each with specific responsibilities and permissions.

Eg) Actors used in "library management system" are administrator, data entry operator, student, library staff and faculty.

Page No.
Date

Q1. Answer all the following questions:

i. What is the significance of rounds in the spiral model? 2019E

Each round represents a cycle of development activities including planning, risk analysis, development, evaluation - The mode repeats the four phases for the following rounds :

Round 0 : Feasibility Study
Round 1 : Cost of Operation
Round 2 : Top-level requirement analysis
Round 3 : Software Design
Round 4 : Design, Implementation and Testing.

ii. Compare iterative model and incremental model in view of following parameters: user involvement, requirement change, maintenance, knowledge of developers, within budget. 2019E

[5+5=10]

Q3. Differentiate between with suitable example:
a) Traditional software development life cycle (SDLC) with object oriented SDLC. 2019M

Page No.
Date

Q.1 [a] Difference between the conventional approach and the object-oriented approach of software development. 2023M [5] [CO1]

	TRADITIONAL APPROACH	OBJECT ORIENTED APPROACH
1.	The system is viewed as a collection of processes.	The system is viewed as a collection of objects.
2.	Data flow diagram, ER diagram, data dictionary and structured charts are used to describe the system	UML models including use case diagram, class diagram, sequence diagrams, components diagram etc are used to describe system.
3.	Reusable source code may not be produced.	The aim is to produce reusable source code.
4.	DFD depict the processes and attributes	Classes are used to describe attributes and functions that operate on these attributes.
5.	It follows a top-down approach for modelling the system.	It follows a bottom-up approach for modelling the system.
6.	It is non-iterative	It is highly iterative

Eg.) b) Define object interaction. 2019M

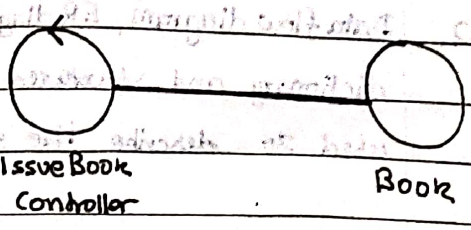
Object interaction refers to the way objects communicate and collaborate with each other to accomplish tasks or fulfill responsibilities within a software system. This interaction typically involves objects sending messages to each other, invoking methods, and sharing data as part of the overall system functionality. It is essential for ensuring modularity, encapsulation, and flexibility within OODs.

Page No.
Date

Q 3. Explain in detail the various types of relationships between classes with the help of suitable example. 2020M (3)

1. ASSOCIATION

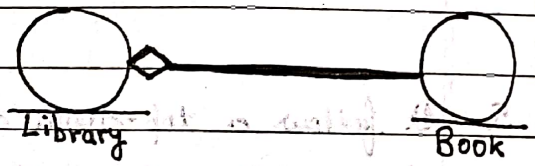
Association is a structural connection between classes. It signifies a relationship between classes. It signifies a relationship where objects of one class are linked to objects of another class



Eg:) In LMS, there could be an association / objects of IsseBook Controller are linked with Book.

2. AGGREGATION

Aggregation models a "whole-part" relationship, indicating that a class is part of another class. It represents "has-a" relationship

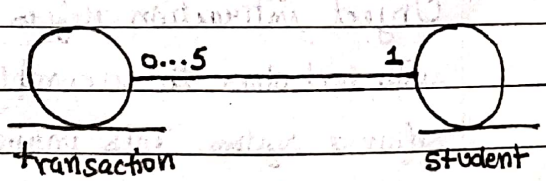


Eg:) In LMS, since 'Book' class is a part of 'Library', there is a aggregation relⁿ b/w them.

3. MULTIPLICITY.

Multiplicity specifies how many objects of one class are related to one object of another class. It represents the # instances related to one instance of another class

Eg:) Association b/w "Transaction" and "Student" indicate one tr is related to one student and one student may be related to upto 5 transactions.



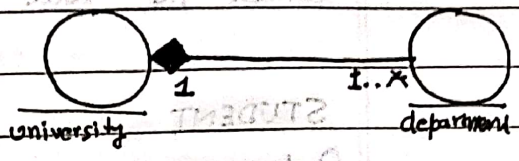
Representation:
Minimum_Value ---- Maximum Value

Eg:)
(0..*) → many
(1...*) → one or more
(1) → exactly one

Page No.
Date

4. COMPOSITION

Composition represents a strong form of the whole-part relationship, where the part class exclusively belongs to the whole class.



Eg.) In university system, the composition rel^n could exist b/w "University" and "Department" indicating the dept belong to a specific university.

5. DEPENDENCY

Dependency signifies that a class refers to another class. Changes in the referred class affect the class using it. Its a unidirectional rel^n.

Q.2 Explain Functional and Non-Functional Requirements in the context of requirement analysis with a good example. 2023M[5] [CO2, CO3]

FUNCTIONAL REQUIREMENTS

They describe "what" the system will do, i.e. the expectations from the system. Functional requirements are called product features. These describe the functionalities and features of application.

Eg.) In banking system, "Allow user to transfer funds b/w accounts" is a function requirement.

NON-FUNCTIONAL REQUIREMENTS

They describe the attributes of system quality, i.e. how well the system does what its has to do. Some of them are reliability, usability, maintainability, availability, portability, flexibility and testability.

Eg.) Requirement that "Ensure the system is available 99.9% of time".

Page No.
Date

Q 4. Consider a result management system of a university. Identify the objects, classes, attributes and methods. 2020M (3)

Let us define the classes with attributes and methods from which we will create objects.

STUDENT

Represents individual students enrolled in the university

ATTRIBUTES: Name, ID, Department, Semester, etc.

METHODS: Register for course, Drop course, View grades, Update personal information.

COURSE

Represents academic courses offered by the university

ATTRIBUTES: Course code, Name, Credit, Department, etc.

METHODS: Enroll Students, Assign instructor, Calculate Grade Distribution etc.

INSTRUCTOR

Represents faculty member who teaches courses.

ATTRIBUTES: Name, ID, Department, courses taught etc.

METHODS: Enter grades, View Student Rosters, View Course Schedule.

RESULT

Represents the result of a student's performance in a particular course.

ATTRIBUTES: Student ID, Course ID, Grade, Semister etc.

METHODS: Calculate GPA, Generate Transcripts, Calc. course average.

Page No.
Date

DEPARTMENT

Represents academic departments within the university.

ATTRIBUTES: Department code, Name, HOD, etc.

METHODS: Add new course, Assign instructor, view dept stats etc.

b) Explain the importance of requirements. How many types of requirements are possible? 2023E

A requirement is defined as "a condition or capability to which a system must conform."

Requirements provide a roadmap for building software, ensuring alignment with client expectations, reducing costly revisions and enabling project planning.

The types of requirements are:

→ FUNCTIONAL REQUIREMENTS: They define what the software does. These are essentially the features and expectations of the system, including what it won't do. They originate from stakeholders and directly relate to the system's functionality.

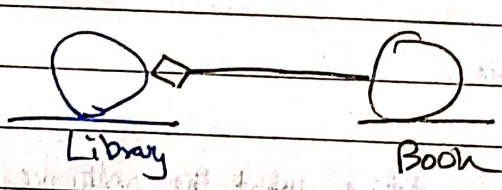
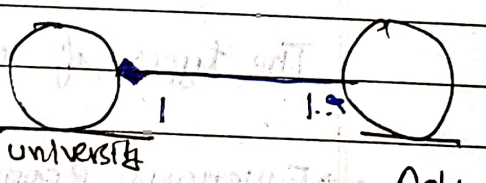
→ NON-FUNCTIONAL REQUIREMENTS: They describe how well the system performs. These are quality attributes that determine how effectively the software does its job. like reliability, usability, maintainability and security. These attributes impact user satisfaction and help measure software quality.

→ KNOWN REQ: Identified expectations of stakeholders that, if implemented correctly contribute to satisfaction. Think of them as explicit needs and desires communicated by the users.

→ UNKNOWN REQ: Unidentified needs that may not be immediately apparent due to lack of expertise or current technology limitations.

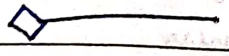
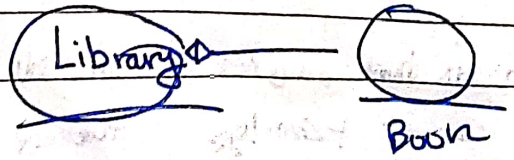
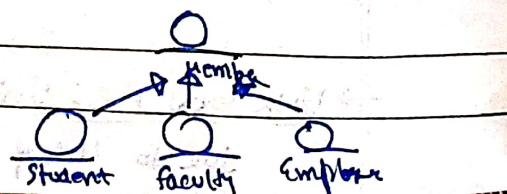
Page No.
Date

b)Differentiate between aggregation and composition relationships. 2023E

AGGREGATION	COMPOSITION
It provides whole-part kind of relationship b/w classes	It provides strong aggregation between classes.
An association b/w two objects describe a "has-a" rel^n	The most spcific type of aggregation that implies ownership
Destroying the owning object dont affect containing objects	_____ does affect _____
Represent by diamond symbol	_____ by filled diamond
Eg 	Eg) 

Q5. Differentiate between aggregation and generalization. State a problem of processing of order and generating invoices of sales in a medical store. Design a class diagram. Describe the role of each class, relationship between classes and operations defined. [5]
END 2019M

iv. Define aggregation and generalization with an example. 2019E

AGGREGATION	GENERALIZATION
It provides a whole-part kind of relationship between classes.	It signifies parent-child relationship among classes.
An association b/w 2 objects describe the "has-a" relationship.	A generalization denotes "is a" rel^n for combining similar class of object in single general class
A diamond symbol denotes 	a arrow symbol denote 
Eg) 	Eg) 

Page No.
Date

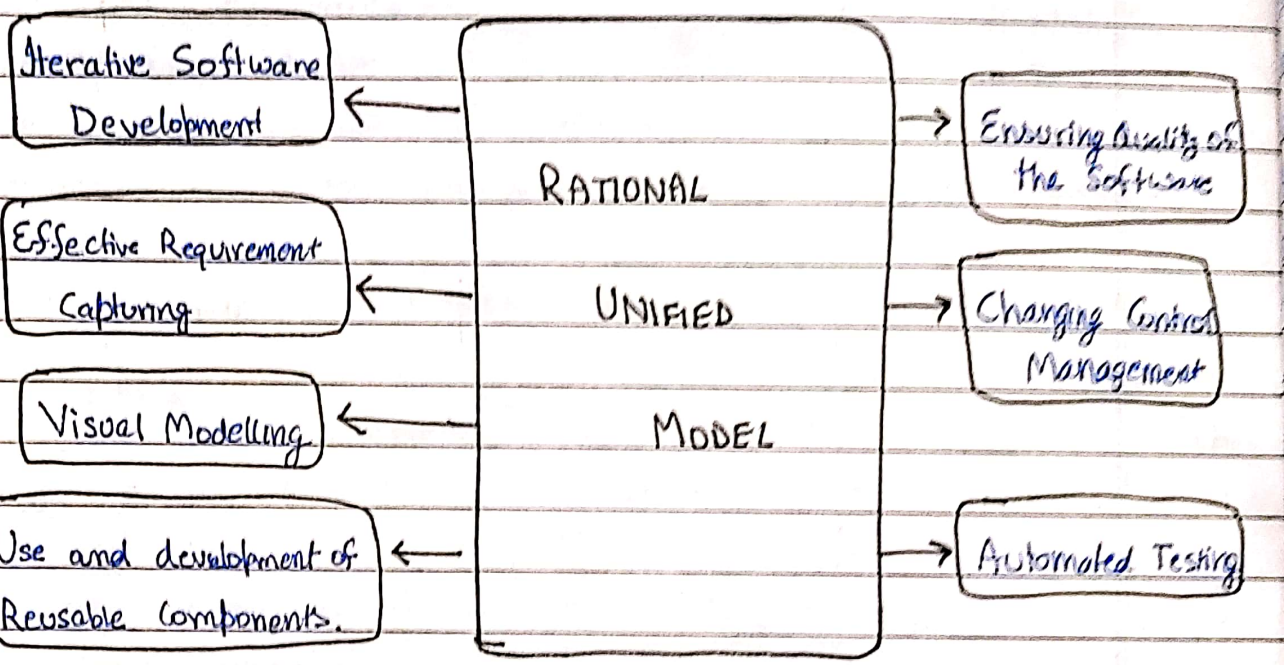
ii. Explain modifiability in requirement specification. 2019E

In SRS, modifiability is the ability to adapt changes in future without compromising the quality, performance or reliability of the s/w. Modifications should be perfectly indexed and cross-referenced.

Page No.
Date

Q 5. What is RUP? Explain the various phases of RUP along with their outcomes. 2020M (5)

Rational Unified Process (RUP) is a software development process framework maintained by Rational Software, offering adaptability to meet organization's need. It advocates for iterative development where software progresses through a series of short-length mini projects known as iterations. It serves as a foundation of the UML.



The phases of RUP are:-

1. INCEPTION: Defines project feasibility, scope and high-level requirements. Identifies actors, significant use cases and establishes initial risks.

2. ELABORATION: Detailed planning and architectural design. Addresses significant risks, creates prototypes, and develops detailed iteration plans.

3. CONSTRUCTION: Develops and tests the product based on the elaboration phase, optimizes resources, verifies quality and tests all functionalities.

Page No.
Date

4. TRANSMON: Hands over the usable product to the customer. Involves delivering, training users and maintaining the software. Includes Beta-testing and bug fixes.

OUTCOMES

INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
→ vision document	→ Prototypes	→ Software product	→ Product release
→ Business case	→ Updated risk list	→ Implementation model	→ Beta test reports
→ Risk assessment	→ Development case	→ Deployment plan	→ Release notes
→ Iteration plan	→ Software architecture description document.	→ Test suite	→ Training material
→ Software development plan		→ Test outline	→ End user support material.
→ Tools	→ Design and data model	→ Training materials	
→ Initial project glossary	→ Implementation model	→ Iteration plan	
→ Initial use case model	→ Use case model	→ Design model	
→ Project repository	→ Detailed iteration plan	→ Documentation manuals	
→ Prototypes.	→ Test plan		
	→ Test automation architecture		

e) Explain UML. 2019M

UML is a standard language for specifying, visualizing, constructing and documenting the artifacts of software systems. UML was created by OMG and in January 1997. UML is a graphical notation for modelling various aspects of software systems.

UML

STRUCTURAL DIAGRAMS
→ Class diagram
→ Object diagram
→ Package diagram
→ Component diagram
→ Deployment diagram

BEHAVIORAL DIAGRAMS
→ Activity diagram
→ State transition diagram
→ Use case diagram
→ Interaction diagram.

Page No.
Date

Q1. Describe Jacobson's methodology. What are the benefits of UML? (4)
2019M (SE)

The Jacobson's methodology known as Object-Oriented Software Engineering consists of five modules:

1. The requirement model: The aim of this model is to gather software requirements.
2. The analysis model: The goal of this model is to produce ideal, robust and modifiable structure of an object.
3. The design model: It refines the objects keeping the implementation environment in mind.
4. The implementation model: It implements the objects.
5. The test model: The goal of the test model is to validate and verify the functionality of the system.

Jacobson's methodology focuses on capturing system functionalities via use cases and actors, determining interfaces among use cases. Analysis involves developing an ideal model, identifying interface, database-related, and control objects grouped into subsystems. Design refines objects to the implementation environment, converting them into blocks, then modules. Modules are integrated and tested in the final model.

Jacobson's methodology emphasizes behavioral aspects over other

Page No.
Date

Q 7. What is object oriented design (OOD)? How is the OOD different from coding a software? 2020M (5)

Q.1 (a) Explain macro and micro process of Booch Methodology. 2023 E [3][CO1]

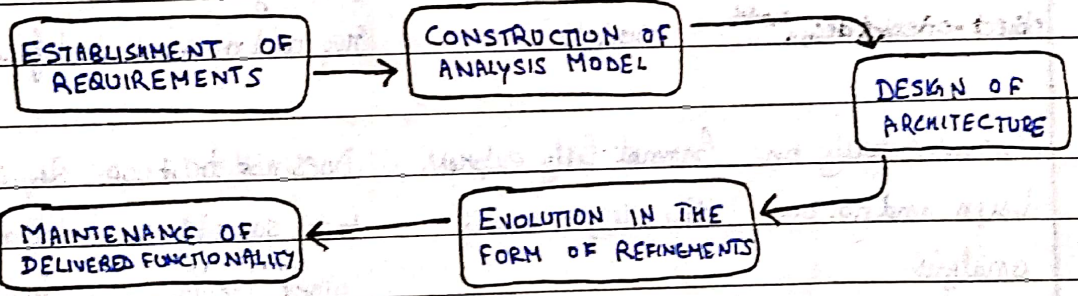
Object oriented design is a methodology proposed by Grady Booch in 1991. It aims to provide a foundation for implementing OOS. It focuses on identifying interfaces at various levels and emphasizes an iterative approach where designers utilize creativity in developing OOS. It follows an incremental and iterative life cycle, combining analysis, design and implementation.

OOD differs from coding in that it is a systematic approach to structuring and organizing software systems, whereas coding involves translating designs into executable code.

The Booch Methodology is broadly divided into two processes:

MACRO-PROCESS:

It is a high-level process that describes the activities to be followed by the development team throughout the life cycle.



It begins with establishing requirements through context diagrams and prototypes, followed by analysis to capture core system requirements and identify risks. The design phase focuses on constructing the system architecture, including identifying layers, mapping classes to subsystems, and planning releases. The evolutionary phase involves iterative implementation to add functionality, and the maintenance phase covers post-deployment activities.

P.T.O →

Page No.
Date

MICRO-PROCESS

It is a lower level process, which is recursive in nature. These steps include identifying classes and objects, determining their semantics, understanding their relationships, and specifying interfaces and implementations. This involves finding classes and objects at a specific level of abstraction, understanding their meaning in terms of attributes and operations, determining interactions among them.

Notably, Booch methodology does not prescribe formal notations for constructing visual diagrams.

Q1. Compare and contrast various object oriented methodologies. 2019 (4)

FEATURE	BOOCH	RUMBAUGH	JACOBSON	COAD/YOURDAN
APPROACH	object-centred	object-centred	User-centred	object-centred
PHASES COVERED	Analysis, design, implementation	"	All	Analysis, design
STRENGTH	Strong method for producing detailed object-oriented design models	Strong method for producing object models.	Strong method for producing user driven req. and OOA models	Strong method for OOA through five defined steps.
WEAKNESS	Focuses entirely on design and not on analysis	Cannot fully express the requirements	Does not treat OOP to the same lvl as other methods	Requires extensive documentation during analysis.
Relⁿ	Uses associations	Directed on	Sequential rel⁻	Link
Diagram	Class, State, object, timing, module, process	DFD, State transition, class/object	Use case dig	—

Page No.
Date

Q2. i. Mention the models in the analysis phase of the Rumbaugh Methodology and explain their roles for describing the system. 2018E

Analysis phase is composed of three submodels given below:

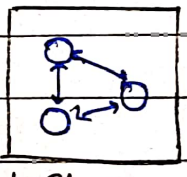
1. OBJECT MODEL: It captures the static aspect of the system. In this model, the requirements are stated in the problem statement. The relevant classes along with inheritance relationships are extracted from the problem statement.

2. DYNAMIC MODEL: It captures behavioural aspects of the object models and describes state of the objects.
This model identifies state and events in the classes identified in object models.

3. FUNCTIONAL MODEL: It represents the functional aspects of the system in terms of operations defined in the classes.
This model depicts the functionality of the system by creating data flow diagrams of the system in order to understand the processes (in the form of input and output information).

iii. Define the term cohesion in the context of the object-oriented design of systems. 2019E

Cohesion refers to the degree to which the elements of within a module / class belong together and are focussed on a single task or responsibility. High cohesion indicates that the elements within a module are closely related and work towards a common goal, while low cohesion suggests that the elements may be scattered and not closely related to each other.


Class

High cohesion is desirable as it leads to more maintainable, understandable and reusable code.

Page No.
Date

Q4. How can you judge the goodness of the object oriented design? Explain in brief about all the quality parameters. 2019M [5]

The set of design principles for improving the quality of software quality related to cohesion, coupling, inheritance and clarity of design include :-

i.) COHESION: The attributes and operations in a class should be highly cohesive. Each operation should be designed to fulfil one single purpose.

ii.) COUPLING: Interactions between classes should be kept minimum. The no. of messages exchanged b/w classes should be reduced. Inheritance based coupling should be increased.

iii.) DESIGN CLARITY: Ensure clear, concise, unambiguous, and consistent vocabulary, class names, attributes, and operations to convey intended meaning and purpose, with clear class responsibilities.

iv.) CLASS HIERARCHY DEPTH: The concept of generalization - specialization should be used wherever its necessary. Unnecessary use of inheritance should be avoided. Inheritance heirarchy should represent the solution to a problem

v.) SIMPLE CLASSES / OBJECTS: Excessive attributes should be avoided in a class. Class definition should be simple, clear, concise and understandable.

Adherence to these will lead to better design and hence improved s/w quality and produces a maintainable system.

v. Explain sequence diagram with an example. [2*5=10]

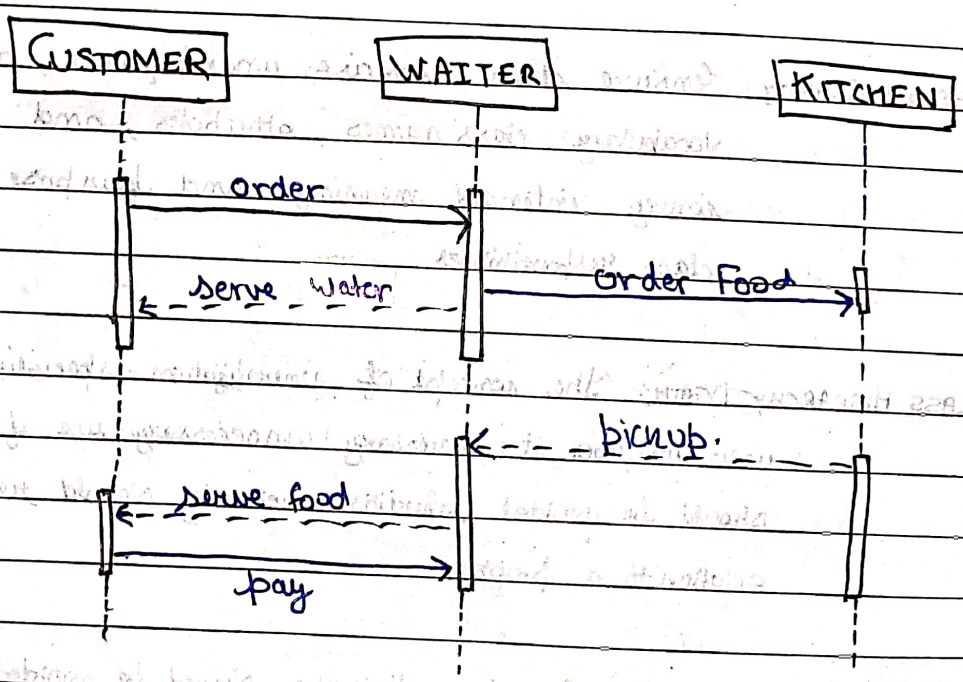
b) Sequence Diagram and Collaboration Diagram [5X2=10]
2019M

SEQUENCE DIAGRAM

A sequence diagram is an interaction diagram that depict the flow of control. The messages sent among objects are time ordered, i.e. the diagram tells us how and when objects of the system communicate with each other by passing messages.

Objects are depicted as boxes arranged horizontally, with vertical lines representing the lifelines (showing passage of time downwards) of objects. Messages are shown as arrow between the lifelines, indicating the dirn of communication. A thin rectangular box over lifeline shows activation representing the period in which an element is performing action.

Ex:



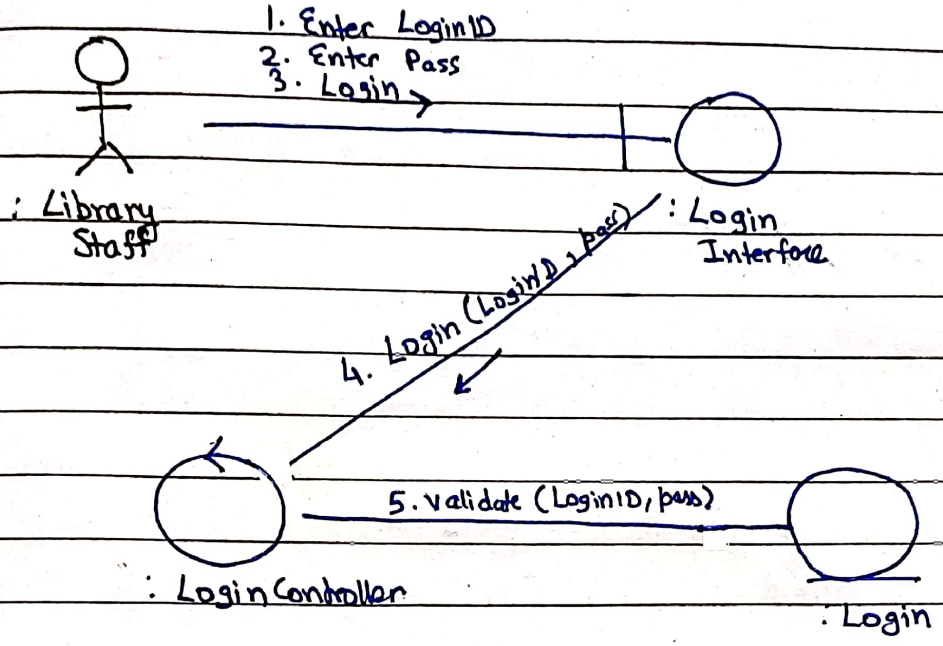
COLLABORATION DIAGRAM

A collaboration diagram illustrates the relationships and interaction between objects in a system. Unlike sequence diagram, here the sequence of events is not time ordered. Collaboration diagram depict the flow of events of a scenario in a use case.

Although the msg are shown here, the time for which object is alive and the time till when an object participates in an operation are missing in the diagram.

Page No.
Date

Ex) 'Login' use case.



Q.3 Consider the railway reservation system. Draw the sequence diagram for reservation and cancellation of trains. 2023E

Page No.
Date

Q.1 a) What is UML? Explain the objectives of modelling. 2023 E

Software modelling serves several crucial purposes:

1. Understanding problems: Modelling aids in comprehending the complexities of the problem domain.

2. Develop proper documents: Models serve as documentations for stakeholder aiding in communication and clarity.

3. Produce well-designed programs: Modelling facilitates the creation of robust system design for softwares.

4. Capture requirements: Models should accurately represent customer requirements and be verified accordingly.

5. Visualize and organize complex relⁿ: Modelling helps in visualizing and structuring intricate system relationships.

6. Produce maintainable systems: Well-designed models lead to high-quality, maintainable software system.

c) What is meant by model? 2019M [1X5=5]

A model refers to a representation or abstraction of a system, often created using classes and objects to describe its behavior, structure and interactions. Models help developers understand, design and implement software systems by capturing essential aspects of the system in a structured manner.

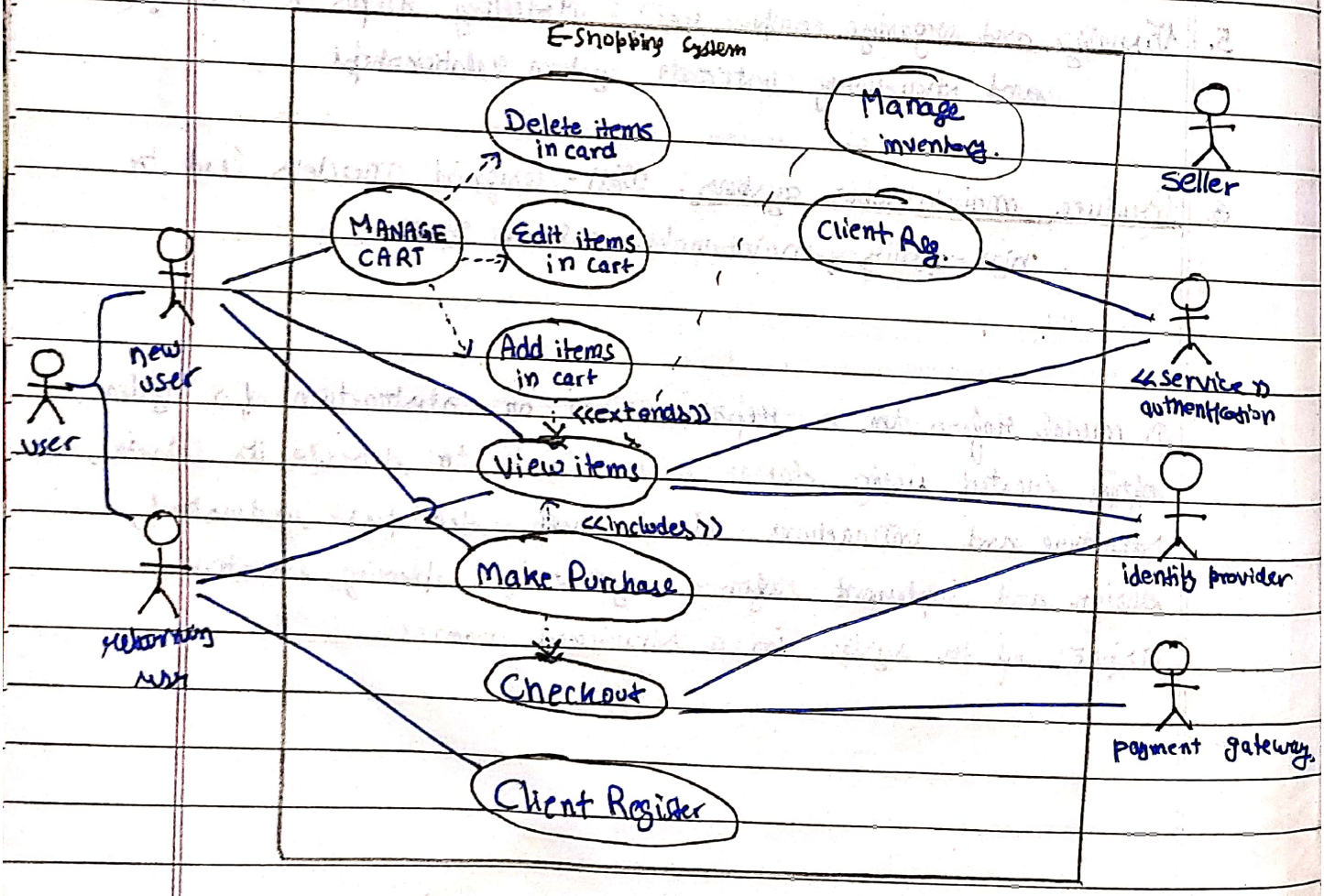
Page No.
Date

Q 6. Discuss the role of use cases in object oriented requirement analysis. Draw a use case diagram for e-shopping system. 2020M (5)

ROLE OF USE CASES

Use cases play a crucial role in object-oriented requirement analysis by providing structured outlines to describe system functionalities in natural language.

They focus on user interaction and goals, abstracting away implementation details. Use cases define system scope, actors and interactions, fostering a clear understanding of system behavior. Their discipline approach ensures systematic requirement capture, facilitating stakeholder comprehension and guiding devs in building s/w that meets expectations.



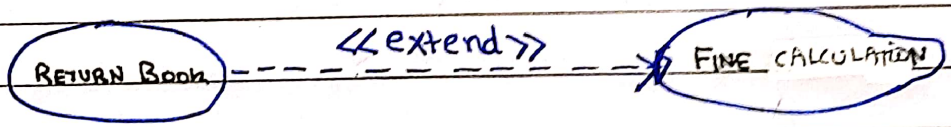
Page No.
Date

What is the purpose of extension and inclusion association between use cases.? Explain with the help of an example. [10] [CO2, CO3] 2023 E

EXTEND RELATIONSHIP

Extend relationship is used to model the occurance of some optional, alternative or special part of the use case. This relationship is used to extend the functionality of the original test cases.

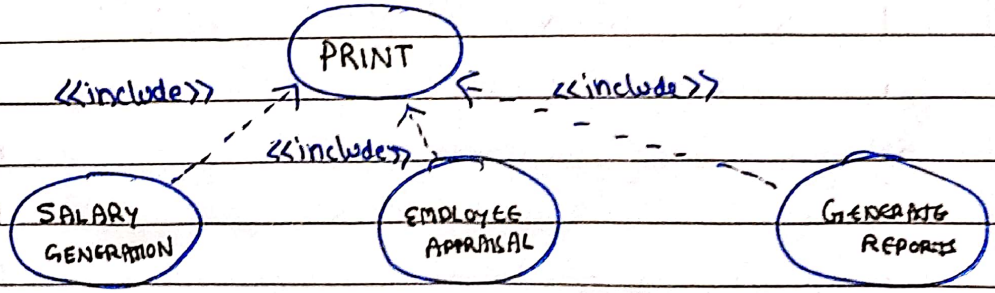
Ex) in LMS, the concept of "extend" occurs when a student returns a book after the due date, resulting in a fine calculation. This fine calculation is represented as a new use case that extends the "Return book" use case and is invoked whenever a book is returned late by a student.



INCLUDE RELATIONSHIP

The redundant and repeated functionalities amongst use cases can be modelled into include relationship. Thus, the redundancies can be grouped into a single use case. In order to use include relationship, there must be common text in ~~both~~ two or more use cases.

Ex) in Employee Mgmt. System, print use case is reqd by three use cases – salary generation, appraisal and generate report



Q.1 Consider a course scheduling system (CSS) for scheduling courses in Delhi Technological University. The purpose of CSS is to, 2022H

(i) enable entering data of courses, faculty members, the available facilities. -20

(ii) calculate and propose schedule for courses.

(iii) enable to manually update the proposed schedule, but keep track of the consistent schedule

- Entering programs and courses: An administrator should be able to enter new program and its running period. A data entry operator should be able to enter details of courses with their examiners. The examiner should be able to upload lecture, tutorials, projects, etc.
- Entering resources: An administrator should be able to enter information about lecture rooms and laboratories in which classes will be taken place.
- Scheduling: It provides schedule proposal, the days and time, and places where they can be scheduled. The scheduler allows some manual predefinition of the schedule. It shows if there are any conflicts.

Consider the CSS and construct the following:

(a) Use case diagram [5] [CO5]

(b) Use case description of any two use cases [5] [CO5]

(c) Identification of classes and their relationships [5] [CO5]

Page No.
Date

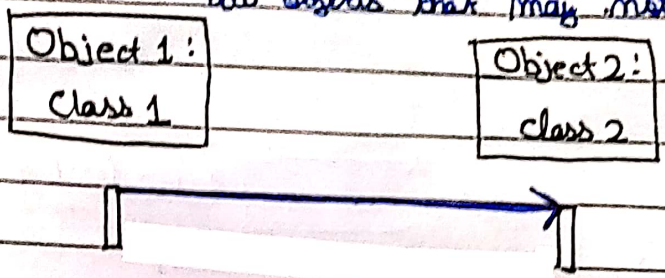
Page No.
Date

Page No.
Date

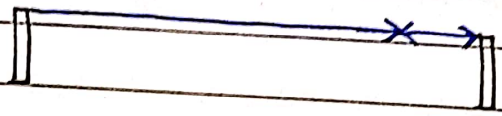
b) Describe the various types of messages in UML with their notation.
2023E [10] [CO5]

The various types of messages are as follows:

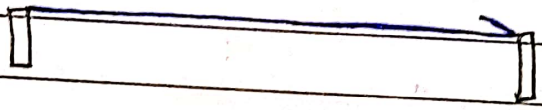
1. SIMPLE MESSAGE: A simple message is used to represent interaction b/w objects that may not be a procedure call.



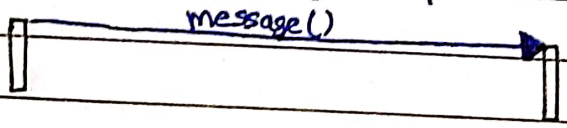
2. SYNCHRONOUS MESSAGES: When a synchronous type of message is sent, the sending object waits to recieve a response from receiving object.



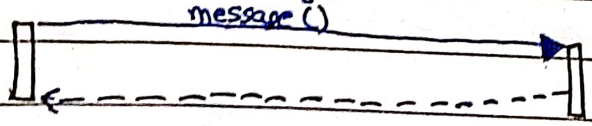
3. ASYNCHRONOUS MESSAGES: When an asynchronous type of msg is sent, the sending object does not wait for response from receiving object.



4. PROCEDURE CALL: The outer sequence resumes after the completion of the inner sequence of the procedure.



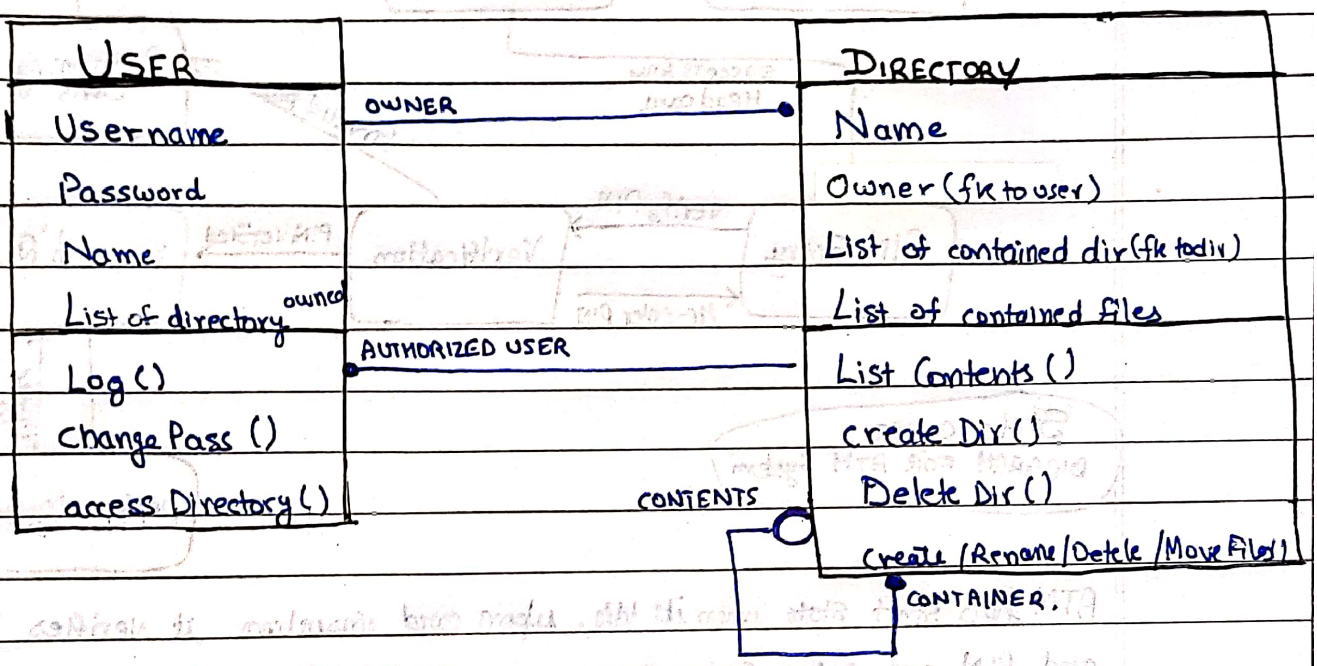
5. RETURN MESSAGE: If some value is given by a receiving object in response to some message obtained by the sending object, this is called a return message.



Page No. | Date

Q.4 Draw a class diagram for the case study given below. 2023M

A directory may contain many other directories and may optionally be contained in other directory. Each directory is used by only one user who is the owner and many users are authorized to use that directory. The identified classes must have their respective attributes and operations. Make necessary assumptions, if any. [5] [CO5]



Q2. What is the significance of state chart diagram while designing the system? Explain the structure of state chart diagram of ATM Machine and illustrate the working. 2019M [5]

OOS

A state chart diagram plays a crucial role in system design by offering a clear and concise way to visualize its dynamic behaviour. Its significant as it :

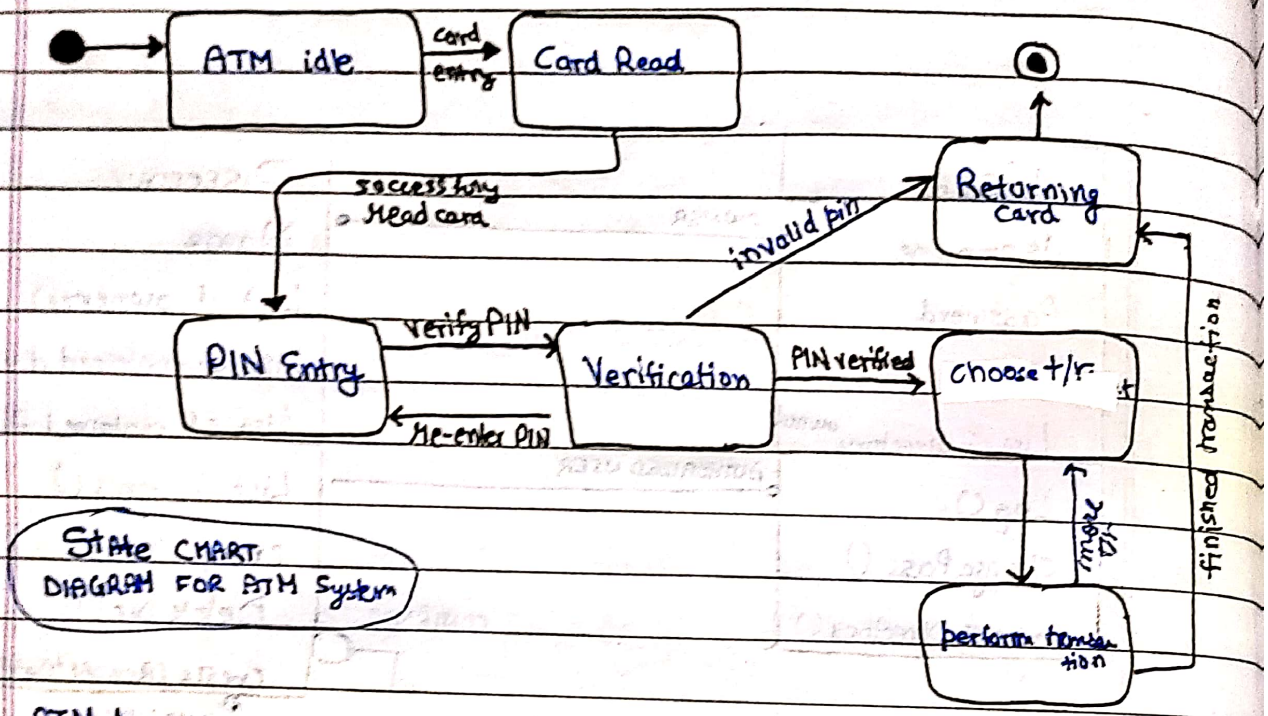
→ Visualizes system states and transitions for clear behaviour understanding

→ Facilitates communication and collaboration among stakeholders.

→ Helps identify and address potential issues early.

→ Serves as valuable documentation for future reference and maintenance

→ Simplifies complex systems for easier analysis and design.



STATE CHART DIAGRAM FOR ATM System

ATM has start state when its idle, upon card insertion it verifies the card and PIN and enters transaction zone. Here it takes you to choose t/r action like transfer/ withdraw. After each t/r it offers further option or card rejects if error occurs, like an invalid PIN or cancellations, it handles them accordingly.

Q3.

i. How UML diagrams constructive for the development of a typical software system? Explain state chart diagram with a suitable reason considering the example of employee information system.
2019E

out of syl?

Page No.
Date

Page No.
Date

Q.4 a) Consider a traffic light system. Draw its state chart diagram. Specify the rules according to which the system is controlled. 2023E

005

Page No.
Date

b) Differentiate between interaction diagrams and state chart diagrams.
2023E [10] [CO6] 005

Q.2 a) How is a class diagram different from an ER diagram 2023E

The main difference between a class diagram and ER diagram is that the Class Diagram represents the classes and associations between them in a software program while an entity-relationship diagram represents the entities and their relationship b/w them in a database.

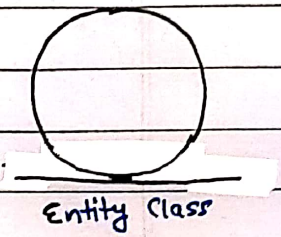
Thus, Class diagram helps to understand the static view of system while ERD helps to recognize entities and rel^n in a DB.

c) Explain various types of classes along with their notations.[10] [CO4]
2023E

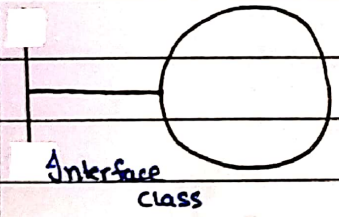
A class is a template which that groups attributes and operations together, and is used to create objects. In RUP classess are categorged into three types which allow the analyst to separate the functionality of system and simplify the identification process.

1. ENTITY CLASS

Entity classes include those classes that are going to persist in the system. These are the classes which have to be stored and maintained for a longer period (sometimes system's lifetime).



2. INTERFACE CLASS

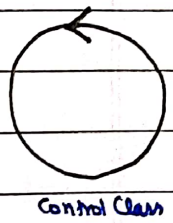


Interface classes handle the interactions in the system. They provide interface b/w the actor and the system. They are called boundary classes. These are used to model windows, buttons, communication protocols etc.

Their life is only as long as the use case exists.

3. CONTROL CLASSES

A control class is responsible for coordinating and managing entity and interface classes, handling the tasks and sequence of events within a system. It represents the dynamics of the system and ensures that a use case is completed by putting together necessary elements.



Page No.
Date

ii. Consider a program which takes a date as an input and checks whether it is a valid date or not. Its input is a triple of day, month and year with the values in the following ranges:
1<=month<=12
1<=day<=31
2000<=year<=2070
Generate boundary value analysis test cases. 2019E

Q4. [5+5=10]

i. Explain the expected benefits of using CASE tools for software system developers and software maintenance teams. 2019E

ii. Explain Halstead's software science metrics. How can we estimate program length? 2019E [5+5=10]

Q5.

i. Consider the database application project with the following characteristics: 2019E
a. The application has 45 key classes.
b. A graphical user interface is required.
Calculate the effort to develop such a project given 20 person days.

ii. Explain the Fountain model with the help of a diagram. What is the significance of arrows within the circle in this model? Also, list the advantages and disadvantages of this model. 2019E
[5+5=10]

Q.5 Write a Short note on:
(a) Software testing tools 2023E
(b) State-based testing 2023E
(c) Class testing 2023E [10] [CO7]

END