CS8592 OBJECT ORIENTED ANALYSIS AND DESIGN **L T P C**
3 0 0 3

OBJECTIVES:
The student should be made to:

- To understand the fundamentals of object modeling
- To understand and differentiate Unified Process from other approaches.
- To design with static UML diagrams.
- To design with the UML dynamic and implementation diagrams.
- To improve the software design with design patterns.
- To test the software against its requirements specification

UNIT I UNIFIED PROCESS AND USE CASE DIAGRAMS **9**

Introduction to OOAD with OO Basics – Unified Process – UML diagrams – Use Case – Case study – the Next Gen POS system, Inception -Use case Modeling – Relating Use cases – include, extend and generalization – When to use Use-cases

UNIT II STATIC UML DIAGRAMS **9**

Class Diagram— Elaboration – Domain Model – Finding conceptual classes and description classes – Associations – Attributes – Domain model refinement – Finding conceptual class Hierarchies – Aggregation and Composition – Relationship between sequence diagrams and use cases – When to use Class Diagrams

UNIT III DYNAMIC AND IMPLEMENTATION UML DIAGRAMS **9**

Dynamic Diagrams – UML interaction diagrams – System sequence diagram – Collaboration diagram – When to use Communication Diagrams – State machine diagram and Modeling – When to use State Diagrams – Activity diagram – When to use activity diagrams Implementation Diagrams – UML package diagram – When to use package diagrams – Component and Deployment Diagrams – When to use Component and Deployment diagrams

UNIT IV DESIGN PATTERNS **9**

GRASP: Designing objects with responsibilities – Creator – Information expert – Low Coupling – High Cohesion – Controller Design Patterns – creational – factory method – structural – Bridge – Adapter – behavioral – Strategy – observer –Applying GoF design patterns – Mapping design to code

UNIT V TESTING **9**

Object Oriented Methodologies – Software Quality Assurance – Impact of object orientation on Testing – Develop Test Cases and Test Plans

TOTAL: 45 PERIODS

OUTCOMES:

At the end of the course, the students will be able to:

- Express software design with UML diagrams.
- Design software applications using OO Concepts.
- Identify various scenarios based on software requirements.
- Transform UML based software design into pattern based design using design patterns.
- Understand the various testing methodologies for OO software.

TEXT BOOK:

1. Craig Larman, "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development", Third Edition, Pearson Education, 2005.
2. Ali Bahrami-Object Oriented Systems Development-McGraw Hill International Edition-1999.

REFERENCES:

1. Erich Gamma,and Richard Helm,Ralph Johnson,John Vlissides,"Design patterns:Elements of Reusable Object-Oriented Software",Addision-Wessley,1995.
2. Martin Fowler, "UML Distilled: A Brief Guide to the Standard Object Modeling Language",Third edition, Addison Wesley, 2003.
3. Paul C. Jorgensen, "Software Testing:- A Craftsman's Approach", Third Edition, Auerbach Publications, Taylor and Francis Group, 2008.

CO-PO MATRIX

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	-	-	3	-	-	-	-	3	3
CO2	3	3	3	-	3	-	-	-	-	-	3	3
CO3	3	3	3	-	-	2	-	-	-	-	3	3
CO4	2	1	1	-	-	-	-	-	-	-	3	3
CO5	1	1	2	-	-	-	-	-	-	-	3	3
CO6	1	1	2	-	-	-	-	-	-	-	3	3
AVG	2.1	2	2.3	-	3	0.8	-	-	-	-	3	3

CO-PSO MATRIX

CO	PSO1	PSO2	PSO3
CO1	2	3	-
CO2	3	3	-
CO3	3	3	-
CO4	3	3	-
CO5	3	3	-
CO6	3	3	-
AVG	2.8	3	-

UNIT-I

UNIFIED PROCESS AND USE CASE DIAGRAMS

Introduction to OOAD with OO Basics – Unified Process – UML diagrams – Use Case –Case study – the Next Gen POS system, Inception -Use case Modeling – Relating Use cases – include, extend and generalization – When to use Use-cases

PART-A

1. What is an object?(NOV/DEC2009)(NOV/DEC2018) (R)

An object is a combination of data and logic; the representation of some real-world entity.

2. What is Object Oriented System development methodology? (R)

Object oriented system development methodology is a way to develop software by building self-contained modules or objects that can be easily replaced, modified and reused.

3. What is the main advantage of object-oriented development?((NOV/DEC2009) (R)

Why we need object oriented development? (NOV/DEC2012)

- High level of abstraction.
- Seamless transition among different phases of software development.
- Encouragement of good programming techniques.
- Promotion of reusability.

4. Distinguish between method and message in object. (NOV/DEC2008) (NOV/DEC 2015) (AN)

S.NO	Methods	Messages
1.	Methods are similar to functions, procedures or subroutines in more traditional programming languages.	Messages are essentially non-specific function calls.
2.	Method is the implementation.	Message is the instruction.
3.	In an object oriented system, a method is invoked by sending an object a message.	An object understands a message when it can match the message to a method that has the same name as the message.

5. What is Analysis and Design?(MAY/JUNE 2013) (R)

Analysis emphasizes an investigation of the problem and requirements, rather than a solution. For example, if a new computerized library information system is desired, how will it be used?

Design emphasizes a conceptual solution that fulfills the requirements, rather than its implementation. For example, a description of a database schema and software objects. Ultimately, designs can be implemented.

6. What is Object-Oriented Analysis and Design? (NOV/DEC2013) (MAY/JUNE2014) (APR/MAY 2017) NOV/DEC 2020 (R)

Object-oriented analysis, is emphasis on finding and describing the objects—or concepts—in the problem domain. For example, in the case of the library information system, some of the concepts include Book, Library, and Patron.

Object-oriented design, is emphasis on defining software objects and how they collaborate to fulfill the requirements. For example, in the library system, a Book software object may have a title attribute and a getChap-ter method.

7. What is UML? (MAY/JUNE2012) (MAY/JUNE 2013) (R)

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

8. What are the primary goals in the design of UML?(NOV/DEC 2016) (R)

1) Provide users a ready - to use expressive visual modeling language so they can develop and exchange meaningful models.
2) Provide extensibility and specialization mechanism to extend the core concepts.
3) Be independent of particular programming language and development process.
4) Provide a formal basis for understanding the modeling language.
5) Reverse engineering
6) Support higher - level development concepts.
7) Integrate best practices and methodologies.

9. Define Class Diagram. (U)

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

10. Define Activity Diagram. (U)

An activity diagram is a graphical representation of an executed set of procedural system activities and considered a state chart diagram variation. Activity diagrams describe parallel and conditional activities, use cases and system functions at a detailed level.

11. What is interaction diagram? Mention its types. (R)

Interaction diagrams are diagrams that describe how groups of objects collaborate to get the job done interaction diagrams capture the behavior of the single use case, showing the pattern of interaction among objects.

There are two kinds of interaction models

Sequence Diagram

Collaboration Diagram.

12. What is Sequence Diagram? (R)

A sequence diagram shows how objects are interacted in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects.

13. What is Collaboration Diagram? (R)

Collaboration diagrams convey the same information as sequence diagrams, but focus on object roles instead of the times that messages are sent.

In a collaboration diagram, the object-role rectangles are labeled with either class or object names (or both). Colons precede the class names (:).

14. Define Start chart Diagram. (U)

Start chart diagram shows a sequence of states that an object goes through during its life in response to events. A state is represented as a round box, which may contain one or more compartments. The compartments are all optional.

15. What is meant by implementation diagram? (R)

Implementation Diagrams show the implementation phase of systems development such as the source code structure and the run- time implementation structure.

There are two types of implementation diagrams:

1. Component Diagrams.
2. Deployment Diagrams.

16. Define Component Diagram? (U)

A Component diagrams shows the organization and dependencies among a set of components. A component diagrams are used to model the static implementation view of a

system. This involves modeling the physical things that reside on a mode, such as executable, libraries, tables, files and documents.

17. Define Deployment Diagram. (U)

Deployment Diagram shows the configuration of run-time processing elements and the software components, processes, and objects that live in them. Deployment diagrams are used to model the static deployment view of a system. A deployment diagram is a graph of modes connected by communication association.

18. What is the UP? (R)

A software development process describes an approach to building, deploying, and possibly maintaining software. The Unified Process has emerged as a popular iterative software development process for building object-oriented systems.

19. What is Iterations? (R)

A key practice in both the UP and most other modern methods is iterative development. In this lifecycle approach, development is organized into a series of short, fixed-length (for example, three-week) mini-projects called iterations

20. What is Iterative and Evolutionary Development? (R)

The iterative lifecycle is based on the successive enlargement and refinement of a system through multiple iterations, with cyclic feedback and adaptation as core drivers to converge upon a suitable system. The system grows incrementally over time, iteration by iteration, and thus this approach is also known as iterative and incremental development. Because feedback and adaptation evolve the specifications and design, it is also known as iterative and evolutionary development.

21. What is Inception? (APRIL/MAY 2011) (R)

Inception is the initial short step to establish a common vision and basic scope for the project. It will include analysis of perhaps 10% of the use cases, analysis of the critical non-functional requirement, creation of a business case, and preparation of the development environment.

22. Define Use case modeling? (NOV/DEC2013) (U)

Use case modeling is a form of requirements engineering. How to create an SRS in what, we might call it as the "traditional" way. Use case modeling is a different and complementary way of eliciting and documenting requirements.

23. Define Use case generalization? (R)

Use case generalization is used when you have one or more use cases that are really specializations of a more general case

24. What are the Phases of Unified Process? (R)

The Unified Process has 4 phases:

Inception: Requirements capture and analysis

Elaboration: System and class-level design

Construction: Implementation and testing

Transition: Deployment

25. **What do you mean by use cases and actors? (NOV/DEC 2011) (U)**

An actor is something with behavior, such as a person (identified by role), computer system, or organization; for example, a cashier. **A use case** is a collection of related success and failure scenarios that describe an actor using a system to support a goal.

26. What is Software verification and validation? (NOV/DEC 2008) (R)

Verification: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

- Validation: Are we building the right product?
- Verification: Are we building the product right?

27. List the relationships used in use cases (MAY/JUNE2012) (R)

List the two ways to relate use cases and give suitable examplesst. (Nov/Dec 2020)

1. The include Relationship
2. The extend Relationship
3. The generalize Relationship

UML relationships are grouped into the following categories:

Category	**Function**
Activity edges	Represent the flow between activities
Associations	Indicate that instances of one model element are connected to instances of another model element
Dependencies	Indicate that a change to one model element can affect another model element

Generalizations	Indicate that one model element is a specialization of another model element
Realizations	Indicate that one model element provides a specification that another model element implements
Transitions	Represent changes in state

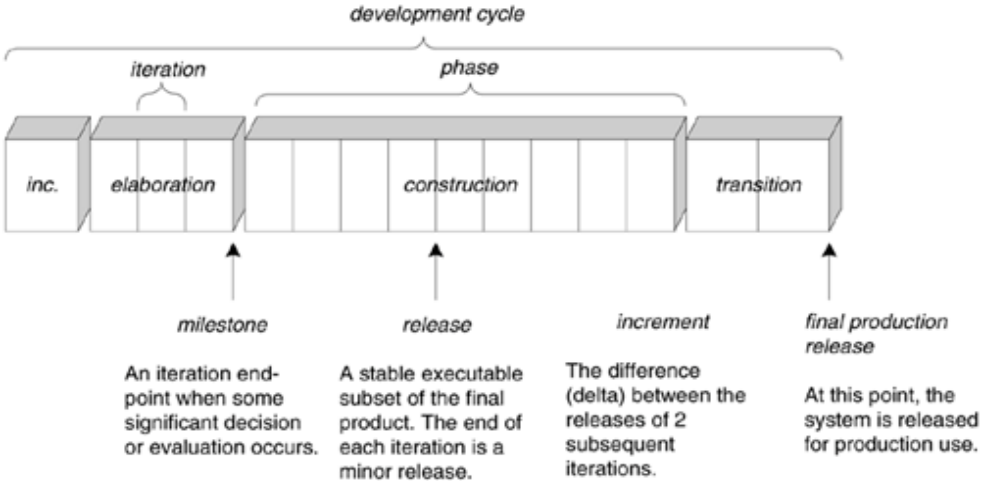
28. List out the step to find the use case? (NOV/DEC2012) (R)

Use cases are defined to satisfy the goals of the primary actors. Hence, the basic procedure is:

1.	Choose the system boundary. Is it just a software application, the hardware and application as a unit, that plus a person using it, or an entire organization?
2.	Identify the primary actors those that have goals fulfilled through using services of the system.
3.	Identify the goals for each primary actor.
4.	Define use cases that satisfy user goals; name them according to their goal. Usually, user-goal level use cases will be one-to-one with user goals, but there is at least one exception, as will be examined.

29. What is Unified Approach/ Process?(NOV/DEC-2018) (R)

A software development process describes an approach to building, deploying, and possibly maintaining software. The Unified Process has emerged as a popular iterative software development process for building object-oriented systems.



30. **Define the following (R)**

a. Forward Engineering b. Reverse Engineering

- **In reverse engineering**, a UML tool reads the source or binaries and generates (typically) UML package, class, and sequence diagrams. These "blueprints" can help the reader understand the big-picture elements, structure, and collaborations.
- **In forward engineering**, before programming, some detailed diagrams can provide guidance for code generation (e.g., in Java), either manually or automatically with a tool.

31. What is the need for modeling? (MAY/JUNE2014) (R)

Modeling is the process of planning a system of interacting objects for the purpose of solving a software problem. It is one approach to software design

32. How to Apply Activity Diagrams? (R)

A UML activity diagram offers rich notation to show a sequence of activities, including parallel activities. It MAY be applied to any perspective or purpose, but is popular for visualizing business workflows and processes, and use cases.

33. What are the three ways and perspectives to apply UML? (NOV/DEC 2015) (NOV/DEC 2016) (APR/MAY 2017)(R)

- UML as sketch- Informal and incomplete diagrams
- UML as blueprint
- UML as programming language

34. Define an object. Identify the probable attributes that will be modeled in a library database for the object BOOK. (R) (NOV/DEC 2017)

An object is a combination of data and logic; the representation of some real- world entity. The Attributes are ISBN, Title, Author and Publisher.

35. Outline the purpose of using use cases, to describe requirements. (NOV/DEC 2017)(U)

In software and systems engineering, a **use case** is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system to achieve a goal.

PART-B

1. What is UML? Explain the three perceptive to apply UML? **(R)**
2. Give a brief discussion on object oriented system in life cycle. **(NOV/DEC2008) (U)**

3. Briefly explain the different phases of Unified process**? (APRIL/MAY 2011) (NOV/DEC2013) (APR/MAY 2017)(U)**
4. Explain about the use case model**? (APRIL/MAY 2011) (R)**
5. What do you mean by Unified Process in OOAD? Explain the phases with suitable diagrams**. (NOV/DEC 2011) (NOV/DEC2012)(MAY/JUNE 2013) (NOV/DEC 2016) (U)**
6. By considering the Library Management system, perform the Object Oriented System Development and give the use case model for the same (use include, extend and generalization**). (NOV/DEC 2011) (AN)**
7. Explain the different phases of Unified Process. **(MAY/JUNE 2012) (R)**
8. Explain with an example, how use case modeling is used to describe functional requirements. Identify the actors, Scenario and use cases for the example. **(MAY/JUNE 2012) (MAY/JUNE 2013) (R)**
9. Explain the following **(NOV/DEC2009)**
   (i) Object Modeling Technique
   (ii) Compare Aggregation and Composition with a suitable example. **(R)**
10. Explain with an example, how use case modeling is used to describe functional requirements. Identify the actors, scenario and use cases for the example. **(NOV/DEC2013) (MAY/JUNE2014) (AN)**
11. List various UML diagrams and explain the purpose of each diagram. **(R) (MAY/JUNE2014)**
12. How to Apply State Machine Diagrams? Explain with example. **(AP) (NOV/DEC2013) (MAY/JUNE2014)**
13. Explain the state chart diagram with a suitable example. Also define its components and Use. **(U) (NOV/DEC 2011) (MAY/JUNE 2013)**
14. Explain about deployment diagram and Component diagram notations with example. **(R) (APRIL/MAY2011)(MAY/JUNE2012) (NOV/DEC2008) (MAY/JUNE2014)**
15. Explain about Unified Process Phase. **(R) (NOV/DEC 2015)**
16. Explain about Use-Case for a case study of your choice. **(AN) (NOV/DEC 2015)**
17. With a suitable example explain how to design a class. Give all possible representation in a class (such as: name, attribute, visibility, methods and responsibilities). **(AN)**
18. Explain interaction diagrams with a suitable example. **(R)**

19. What do you mean by Unified Process in OOAD? Explain the phases with suitable diagrams. **(U)**
20. By considering your own application, perform the Object Oriented System Development and give the use case model for the same (use include, extend and generalization). **(AN) (NOV/DEC 2016)**
21. Explain about UML diagrams in detail with neat example. **(APR/MAY 2017)(R)**
22. Present an outline of object oriented analysis and object oriented design. **(NOV/DEC 2017) (R)**
23. Why the unified process has emerged as a popular and effective software development process? **(NOV/DEC 2017)(AN)**
24. Model a state transition diagram for the following scenario:

    Here is what happens in a microwave oven:

    - The oven is initially in an idle state with door open, where the light is turned on.
    - When the door is closed it is now in idle but the light is turned off.
    - If a button is pressed, then it moves to initial cooking stage, where the timer is set and lights are on and heating starts.
    - At any movement the door may be opened, the cooking is interrupted, the timer is cleared and heating stops.
    - Also while cooking, another button can be pushed and extended cooking state starts, where the timer gets more minutes. At any moment door can be opened here also.
    - If the timer times out, then cooking is complete, heating stops, lights are off and it sounds a beep.
    - When the door is open, again the oven is in idle state with the door open.**(C) (NOV/DEC 2017)**
25. Model a use case diagram for the following scenario:

    Deepthi super market wants a subsystem to process supply orders via the web. The users will supply via a form their name, password, account number and a list of supplies along with an indication of the quantities desired. The subsystem will validate the input, enter the order into a database and generate a receipt with the order number, expected ship date and the total cost of the order. If the validation step fails, the subsystem will generate an error message describing the cause of the failure. **(C) (NOV/DEC 2017)**

26. "A component represents a modular, deployable and replaceable part of a system that encapsulates implementation and exposes a set of interfaces". Elucidates with an example. **(U) (NOV/DEC 2017)**
27. **Explain briefly the elements of Use Case Diagram.(Nov/Dec-2018)**
28. Elaborate use case modeling process with suitable examples**.(R) (Nov/Dec-2020)**
29. With suitable example explain the use case include relationship and extend relationship**.(U) (Nov/Dec-2020)**

UNIT II

UNIT II STATIC UML DIAGRAMS

Class Diagram— Elaboration – Domain Model – Finding conceptual classes and description classes – Associations – Attributes – Domain model refinement – Finding conceptual class Hierarchies – Aggregation and Composition – Relationship between sequence diagrams and use cases – When to use Class Diagrams

PART A

1. What is GRASP?(APRIL/MAY 2013) (R)

General Responsibility Assignment Software Patterns (or Principles), abbreviated GRASP, consists of guidelines for assigning responsibility to classes and objects in object-oriented design.

2. What is Responsibility-Driven Design? (R)

A popular way of thinking about the design of software objects and also larger scale Components 3 are in terms of responsibilities, roles, and collaborations. This is part of a larger approach called responsibility-driven design or RDD.

3. What is Responsibilities? (R)

The UML defines a responsibility as "a contract or obligation of a classifier". Responsibilities are related to the obligations or behavior of an object in terms of its role.

4. Define Pattern and design pattern? What is the use of Patterns? (MAY/JUNE2012) (MAY/JUNE 2013) (NOV/DEC2013) (MAY/JUNE2014) (NOV/DEC 2016) (R)

A pattern is a named problem/solution pair that can be applied in new context, with advice on how to apply it in Novel situations and discussion of its trade-offs

In software development, a pattern (or *design pattern*) is a written document that describes a general solution to a design problem that recurs repeatedly in many projects. III Year/VI SEM/2021-2022 (Even)

Software designers adapt the pattern solution to their specific project. Patterns use a formal approach to describing a design problem, its proposed solution, and any other factors that might affect the problem or the solution.

5. What are the GRASP patterns? (R)

General Responsibility Assignment Software Patterns (or **Principles**), abbreviated **GRASP**, consist of guidelines for assigning responsibility to classes and objects in object-oriented design. The different patterns and principles used in GRASP are:

- Controller
- Creator
- Indirection
- Information Expert
- High Cohesion
- Low Coupling
- Polymorphism
- Protected Variations and
- Pure Fabrication.

All these patterns answer some software problem, and these problems are common to almost every software development project.

6. Define Creator? (U)

Creation of objects is one of the most common activities in an object-oriented system. Which class is responsible for creating objects is a fundamental property of the relationship between objects of particular classes. Consequently, it is useful to have a general principle for the assignment of creation responsibilities.

7. What is Controller? (R)

The Controller pattern assigns the responsibility of dealing with system events to a non-UI class that represents the overall system or a use case scenario. A Controller object is a non-user interface object responsible for receiving or handling a system event. A Controller defines the method for the system operation.

8. Define Low Coupling? (U) (APRIL/MAY 2011) (MAY/JUNE2014)

An element with low (or weak) coupling is not dependent on too many other elements; "too many" is context-dependent, but will be examined.

9. Define High Cohesion? (U) (NOV/DEC 2012)

A class has moderate responsibilities in one functional area and collaborates with other classes to fulfill tasks.

Assume a class exists called RDB Interface which is only partially responsible for interacting with relational databases. It interacts with a dozen other classes related to RDB access in order to retrieve and save objects.

10. What is Information Expert? (R)

Information Expert is a principle used to determine where to delegate responsibilities. These responsibilities include methods, computed fields and so on. Using the principle of Information Expert a general approach to assigning responsibilities is to look at a given responsibility, determine the information needed to fulfill it, and then determine where that information is stored. Information Expert will lead to placing the responsibility on the class with the most information required to fulfill it.

11. What is singleton pattern? (R)

The singleton pattern is a design pattern; it is desirable to support global visibility or a single access point to a single instance of a class rather than some other form of visibility.

12. What is adapter pattern? (R)

The adapter pattern is a design pattern that translates one interface for a class into a compatible interface. An adapter allows classes to work together that normally could not because of incompatible interfaces, by providing its interface to clients while using the original interface. The adapter is also responsible for transforming data into appropriate forms.

13. What is Facade Pattern? (R)

The Facade pattern, which can be used to provide the interface from one layer to the next. A facade can make a software library easier to use, understand and test, since the facade has convenient methods for common tasks; make code that uses the library more readable, reduce dependencies, more flexibility in developing the system.

14. What is Observer pattern? (R)

The observer pattern (a subset of the publish/subscribe pattern) is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributed event handling systems.

15. Define Coupling. (U) (MAY/JUNE2012) (NOV/DEC2008) (NOV/DEC2013)

Coupling is a measure of how strongly one element is connected to, has knowledge of, or relies on other elements**.**

16. Distinguish between coupling and cohesion (NOV/DEC 2015) (NOV/DEC 2016) (APR/MAY 2017) (AN)

Cohesion	Coupling
Cohesion is the indication of the relationship within module.	Coupling is the indication of the relationships between modules.
Cohesion shows the module's relative functional strength.	Coupling shows the relative independence among the modules.
Cohesion is a degree (quality) to which a component / module focuses on the single thing.	Coupling is a degree to which a component / module is connected to the other modules.
While designing you should strive for high cohesion i.e. a cohesive component/ module focus on a single task (i.e., single-mindedness) with little interaction with other modules of the system.	While designing you should strive for low coupling i.e. dependency between modules should be less.
Cohesion is the kind of natural extension of data hiding for example, class having all members visible with a package having default visibility.	Making private fields, private methods and non public classes provides loose coupling.
Cohesion is Intra – Module Concept.	Coupling is Inter -Module Concept.

All good software design will go for **high cohesion** and **low coupling**.

17. When to use patterns? (U) (NOV/DEC 2015)

- *Design patterns* are solution for common design problems that *use* design principles to achieve these desirable goals of a good design.
- People only understand how to apply certain software design techniques to certain problems.
- Design patterns can speed up the development process by providing tested, proven development paradigms.

18. Define Cohesion and coupling. (R) (NOV/DEC 2017)

Coupling is a measure of how strongly one element is connected to, has knowledge of, or relies on other elements. An element with low (or weak) coupling is not dependent on too many other elements; "too many" is context-dependent, but will be examined. These elements include classes, subsystems, systems, and so on.

Cohesion (or more specifically, functional cohesion) is a measure of how strongly related and focused the responsibilities of an element are. An element with highly related responsibilities, and which does not do a tremendous amount of work, has high cohesion. These elements include classes, subsystems, and so on.

19. What is design pattern. (U) (NOV/DEC 2017)

Design patterns are solution for common design problems that *use* design principles to achieve these desirable goals of a good design.

20. What is the difference between a class and an object?(Nov/Dec-2018)(U)

The concepts of objects and classes are intrinsically linked with each other and form the foundation of object–oriented paradigm. An object is a real-world element in an object–oriented environment that may have a physical or a conceptual existence.

21. What is an Object Modeling Languages? (Nov/Dec-2018)(U)

An object model is a logical interface, software or system that is modeled through the use of object-oriented techniques. It enables the creation of an architectural software or system model prior to development or programming. An object model is part of the object-oriented programming (OOP) lifecycle.

22. List any two features of object based languages? (Nov/Dec-2018)(R)

- Good representation of reality through the modeling of classes.
- Code reuse through inheritance. Which makes easier to maintain it.
- Encapsulation of data, assuring its consistence.

23. What is multiplicity? R (Nov/Dec 2020)

The multiplicity is an indication of how many objects may participate in the given relationship or the allowable number of instances of the element. In a use case diagram, multiplicity indicates how many actors can take part in how many occurrences of a use case

24. What is meant by cohesion? (Nov/Dec 2020)

Cohesion is used to indicate the degree to which a class has a single, well-focused purpose.

PART B

1. What are the Design Principles? Explain in detail about object design in Inputs, Activities and Outputs? **(R)**

2. What are Patterns? Explain with a short example of Object Design with Grasp? **(R)**

3. How will you apply Grasp to object design? Discuss about responsibility of Creator, Information Expert, Low Coupling and controller with an example? **(R)** **(APRIL/MAY2011)**

4. How to keep objects focused, understandable, and manageable and as a side Effect, support Low Coupling? **(U)**

5. Apply Design pattern for Adapter pattern with example**? (AP) (APRIL/MAY 2011)**

6. Explain with an example the Factory method design pattern? **(R) (NOV/DEC 2017)**

7. Discuss in detail about Singleton GoF? **(R) (APRIL/MAY 2011)**

8. Explain Delegation Event Model GoF? **(R)**

9. What is GRASP? Explain the design patterns and the principles used in it. **(R) (MAY/JUNE2014)**

10. What is design pattern? Explain the GoF design patterns. **(R) (NOV/DEC 2011)**

11. Describe the concepts of creator. **(R) (MAY/JUNE2012)**

12. Explain about Low coupling, Controller and High Cohesion **(R) (MAY/JUNE2012&2013)**

13. Write short notes on Adapter, Singleton, Factory and observer patterns. **(R) (MAY/JUNE2012&13) (NOV/DEC2013) (MAY/JUNE2014)**

14. Explain GRASP: designing objects with responsibilities. **(R)** (**NOV/DEC2013)**

15. Differentiate Bridge and Adapter **(AN)** (**NOV/DEC2015)**

16. How will you design the behavioral pattern? **(U)** (**NOV/DEC2015)**

17. What is GRASP? Explain the design patterns and the principles used in it. **(R) (NOV/DEC2016)**

18. Explain the method of identifying the classes using the common class approach with an example. **(R) (NOV/DEC2016)**

19. Explain the following GRASP patterns: Creator, Information Expert, Low Coupling, High Coupling and Controller. **(R) (APR/MAY 2017)**

20. Explain in detail about the Factory pattern and mention the Limitations and applications of Factory pattern. **(R) (APR/MAY 2017)**

21. Explain with an example creator and information expert GRASP patterns. **(R) (NOV/DEC 2017)**

22. Draw and explain the class diagram for a banking applications**.(NOV/DEC-2018)(A)**

23. What is Domain model refinement? Explain with suitable examples**. (NOV/DEC-2018)(R)**

24. How will you find conceptual class Hierarchies? Give Example. **(NOV/DEC-2018)(R)**
Discuss the relationship between Sequence Diagram and Class Diagram. **(NOV/DEC-2018)(R)**

25. Differentiate Elaboration and inception. list any five and artifacts related to inception. **(NOV/DEC-2020)(R)**

26. With an illustration, explain the class hierarchies. also state the guidelines for defining a super class. **(NOV/DEC-2020)(R)**

UNIT-III

UNIT III DYNAMIC AND IMPLEMENTATION UML DIAGRAMS

Dynamic Diagrams – UML interaction diagrams – System sequence diagram – Collaboration diagram – When to use Communication Diagrams – State machine diagram and Modelling – When to use State Diagrams – Activity diagram – When to use activity diagrams Implementation Diagrams – UML package diagram – When to use package diagrams – Component and Deployment Diagrams – When to use Component and Deployment diagrams

1. What is an Elaboration? (R) or

List the objectives of Elaboration (MAY/JUNE2012) (MAY/JUNE2013) (MAY/JUNE2014)(Nov/Dec 2020)

It Build the core architecture, resolve the high-risk elements, define most requirements, and estimate the overall schedule and resources.

2. What is a domain model? (R) (APRIL/MAY 2011)(NOV/DEC2013)

A domain model is a visual representation of conceptual classes or real-world objects in a domain of interest. They have also been called conceptual models, domain object models, and analysis object models

3. Define Conceptual Classes? (U)

The domain model illustrates conceptual classes or vocabulary in the domain. Informally, a conceptual class is an idea, thing, or object. More formally, a conceptual class may be considered in terms of its symbol, intension, and extension.

4. Define Description Class? (U)

A description class contains information that describes something else. For example, a Product Description that records the price, picture, and text description of an Item.

5. What are Three Strategies to Find Conceptual Classes? (R)

1. Reuse or modify existing models.
2. Use a category list.
3. Identify noun phrases

6. What is an association? (R)

An association is a relationship between classes (more precisely, instances of those classes that indicates some meaningful and interesting connection.

7. What is an Attributes?(NOV/DEC-2018) (R)

An attribute is a logical data value of an object. It is useful to identify those attributes of conceptual classes that are needed to satisfy the information requirements of the current scenarios under development.

8. What is a Derived Attributes? (R)

If an attribute's value can be determined from the values of other attributes, then the attribute is derivable, and is said to be a derived attribute. Derived attributes are usually calculated from other attributes, such as multiplying an employee's monthly salary by twelve or deriving a person's full name from first name and last name attributes. The syntax of a derived attribute definition is a SQL statement.

9. Defining Conceptual Super classes and Subclasses? (U)

It is valuable to identify conceptual super- and subclasses, it is useful to clearly and precisely understand generalization, super classes, and subclasses in terms of class definition and class sets.

10. What is Generalization? (R)(NOV/DEC-2018)

Generalization is the activity of identifying commonality among concepts and defining superclass (general concept) and subclass (specialized concept) relationships.

11. What is Aggregation? (R) (APRIL/MAY 2011)(MAY/JUNE2012&13) (NOV/DEC2013) (MAY/JUNE2014)

Aggregation is a vague kind of association in the UML that loosely suggests whole-part relationships (as do many ordinary associations). It has no meaningful distinct semantics in the UML versus a plain association, but the term is defined in the UML.

12. What is Composition? (R) (APRIL/MAY 2011) (MAY/JUNE2012)(MAY/JUNE2013) (NOV/DEC2013)

Composition, also known as composite aggregation, is a strong kind of whole-part aggregation and is useful to show in some models.

A composition relationship implies that

1) An instance of the part belongs to only one composite instance at a time

2) The part must always belong to a composite

3) The composite is responsible for the creation and deletion of its parts either by itself creating/deleting the parts, or by collaborating with other objects.

13. Define swim lane. (U (NOV/DEC2011) (MAY/JUNE2013)

A **swim lane** is a visual element used in process flow diagrams (or) flowcharts that visually distinguishes responsibilities for sub-processes of a business process. Swim lanes may be arranged either horizontally or vertically. In the accompanying example, the swim lanes are named customer, Sales, Contracts, Legal, and Fulfillment, and are arranged vertically.

14. State how a use case is represented? (R) (NOV/DEC2008)

Use cases are text documents, not diagrams, and use-case modeling is primarily an act of writing text, not drawing diagrams

15. What is Inception? (R) (APRIL/MAY 2011) (NOV/DEC 2017)

Inception is the initial short step to establish a common vision and basic scope for the project. It will include analysis of perhaps 10% of the use cases, analysis of the critical non-functional requirement, creation of a business case, and preparation of the development environment.

16. Why call a Domain Model a 'Visual Dictionary'? (U) (NOV/DEC 2016)

A domain model is a visual representation of conceptual classes or real-world objects in a domain of interest .Domain models are also called conceptual models, domain object models, and analysis object models.

- It visualizes and relates some words or conceptual classes in the domain.
- It depicts an abstraction of the conceptual classes
- The model displays a partial view, or abstraction, and ignores uninteresting (to the modelers) details.
- May be considered a visual dictionary of the noteworthy abstractions, domain vocabulary, and information content of the domain.

17. What are the tasks performed in elaboration? (R) (NOV/DEC 2015)

Elaboration is the initial series of iterations during which the team does serious investigation, implements (programs and tests) the core architecture, clarifies more requirements, and tackles the high risk issues.

18. How to create a Domain Model? (U) (NOV/DEC 2015) (NOV/DEC 2016)

The domain model can be created for the current iteration requirements under design with the following

i. Find the conceptual classes

ii. Draw them as classes in UML class diagram

iii. Add associations and attributes.

19. What is the relationship of a conceptual superclass to a subclass? (U)

(APR/MAY 2017)

A conceptual superclass definition is more general or encompassing than a subclass definition.

- All subclasses have the same attribute that can be factored out and expressed in the superclass.
- All subclasses have the same association that can be factored out and related to the superclass.

20. Define modular design. (R) (APR/MAY 2017)

Modular design, or "modularity in design", is a design approach that subdivides a system into smaller parts called modules or skids, that can be independently created and then used in different systems.

21. What is the purpose of extends and include relationship in usecase diagram. (U)

(APR/MAY 2017)

Use case include is a directed relationship between two use cases which is used to show that behavior of the **included** use case (the addition) is inserted into the behavior of the **including** (the base) use case.

The **include** relationship could be used:

- to simplify large use case by splitting it into several use cases,
- to extract **common parts** of the behaviors of two or more use cases.

27. When to create a subclass of a superclass? (R) (NOV/DEC 2017)

- All subclasses have the same attribute that can be factored out and expressed in the superclass.
- All subclasses have the same association that can be factored out and related to the superclass.

23. What is significance of UML?(Nov/Dec-2018)(R)

What is the purpose of UML diagram (Nov/Dec-2018)(R)

- Models help us to visualize a system as it is or as we want it to be.
- Models permit us to specify the structure or behavior of a system.
- Models gives us a template that guides us in constructing a system.
- Models document the decisions we have made.

PART B

1. Explain in Detail about the Elaboration Phase? **(R)**
2. What is a Domain Model? Explain the steps to create a Domain Model? **(R) APRIL/MAY 2011) (MAY/JUNE 2013)**
3. Explain about Description Classes and its uses with example? **(R)**
4. Explain briefly about Associations relation in classes? **(R)**
5. How will you refine the Domain model? **(U)**
6. What are Conceptual Classes explain with an example? **(R)**
7. Explain with an example of Aggregation and composition. **(C) (R) (NOV/DEC 2017)**
8. Explain about UML activity diagrams and modeling states? **(R) (APRIL/MAY 2011) (NOV/DEC2008**)
9. Explain the relationships that are possible among the classes in the UML representation with your own example. **(AN) (NOV/DEC 2009)** (**NOV/DEC 2011)**
10. Explain the following with an example:
    (i) Conceptual class diagram
    (ii) Activity Diagram (**R) (NOV/DEC 2011) (NOV/DEC2013)**
11. What are the various diagrams that are used in analysis and design steps of Booch methodology? Explain with your own example**. (R) (NOV/DEC 2009)(NOV/DEC 2008)**
12. Describe the strategies used to identify conceptual classes. Describe the steps to create a
13. Domain model used for representing conceptual classes **(R) (MAY/JUNE2012) (NOV/DEC2013) (MAY/JUNE2014)**
14. When to use activity diagrams. Describe the situations with suitable example. **(R) (MAY/JUNE2012)**
15. Write briefly about elaboration and discuss the differences between Elaboration and Inception With examples. **(R) (MAY/JUNE2014)**
16. What is Generalization? Explain with an example? **(R)**
17. Relate use case with include and extend, Generalization associations explain with proper illustrations. **(U) (NOV/DEC2012)**
18. Explain in detail the transformations in Object-oriented software development life cycle. Giving suitable examples. **(U)**
19. Write about elaboration and discuss the difference between elaboration and Inception with suitable diagram for university domain. **(AN)** (**NOV/DEC2015)**

20. Construct design for library Information System which comprises and following notations. (**NOV/DEC2015) (C)**
    - Aggregations
    - Compositions
    - Associations
21. Explain the method of identifying the classes using the common class approach with an example. (R)
22. For the Hospital management system draw the following UML diagrams.
    - Conceptual Class diagram (overall system)
    - Activity diagram (Billing). **(C)**
23. Write briefly about elaboration and discuss the difference between elaboration and inception with neat diagram. **(APR/MAY 2017)(R)**
24. **Explain the guidelines for finding conceptual classes with neat diagram. (R) (APR/MAY 2017)**
25. Explain about Aggregation and Composition with examples. **(U) (APR/MAY 2017)**
26. Explain with an example of concrete use case and an abstract use case. **(R)**

    (NOV/DEC 2017)
27. With an example, explain the need for Activity diagram. **(NOV/DEC-2018)(R)**
28. Justify the need for component and deployment diagrams with a suitable real time example. **(NOV/DEC-2020)(R)**
29. Differentiate state independent and state dependent objects. How to model them using state machine Diagrams?

UNIT-IV

UNIT IV DESIGN PATTERNS

GRASP: Designing objects with responsibilities – Creator – Information expert – Low Coupling – High Cohesion – Controller Design Patterns – creational – factory method – structural – Bridge – Adapter – behavioural – Strategy – observer –Applying GoF design patterns – Mapping design to code

PART A

1. What is a system sequence diagram? (R) (APRIL/MAY 2011) (MAY/JUNE2012) (MAY/JUNE2014)

SSDs use sequence diagram notation to illustrate inter-system collaborations, treating each system as a black-box. It is useful to illustrate the new system events in SSDs in order to clarify:

• new system operations that the NextGen POS system will need to support

• calls to other systems, and the responses to expect from these calls

2. What is the Logical Architecture? (R)

The logical architecture is the large-scale organization of the software classes into packages (or namespaces), subsystems, and layers. It\'s called the logical architecture because there\'s no decision about how these elements are deployed across different operating system processes or across physical computers in a network.

3. What is a Layer? What is the use of view layer interface? (R) (NOV/DEC2009)

A layer is a very coarse-grained grouping of classes, packages, or subsystems that has cohesive responsibility for a major aspect of the system. Also, layers are organized such that \"higher\" layers (such as the UI layer) call upon services of \"lower\" layers, but not normally vice versa.

4. What's the Connection Between SSDs, System Operations, and Layers? (AN)

The SSDs illustrate these system operations, but hide the specific UI objects. Nevertheless, normally it will be objects in the UI layer of the system that capture these system operation requests, usually with a rich client GUI or Web page.

5. What is controller? (R)

The Controller pattern assigns the responsibility of dealing with system events to a non-UI class that represents the overall system or a use case scenario. A Controller object is a non-user interface object responsible for receiving or handling a system event. A Controller defines the method for the system operation.

6. Define Classifier? (U)

A UML classifier is "a model element that describes behavioral and structure features" Classifiers can also be specialized. They are a generalization of many of the elements of the UML, including classes, interfaces, use cases, and actors. In class diagrams, the two most common classifiers are regular classes and interfaces.

7. What is UML Operations? (R)

A UML operation is a declaration, with a name, parameters, return type, exceptions list, and possibly a set of constraints of pre-and post-conditions. But, it isn't an implementation rather, methods are implementations.

8. What is qualified association? (R)

A qualified association has a qualifier that is used to select an object (or objects) from a larger set of related objects, based upon the qualifier key.

9. What is an association class? (R)

An association class allows you treat an association itself as a class, and model it with attributes, operations, and other features. For example, if a Company employs many Persons, modeled with an Employs association, you can model the association itself as the Employment class, with attributes such as start Date.

10. What is a Sequence diagram? (R) (NOV/DEC2011)

Sequence diagrams illustrate interactions in a kind of fence format, in which each new object is added to the right.

11. What is a Communication diagram? (R)

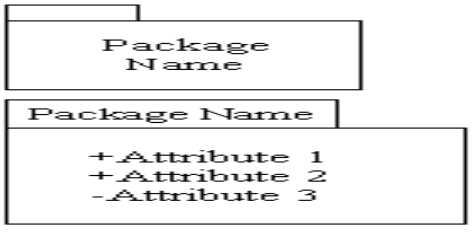
Communication diagrams illustrate object interactions in a graph or network format, in which objects can be placed anywhere on the diagram

12. What do you mean by sequence number in UML? Where and for what it is used? (U) (NOV/DEC2011) (NOV/DEC 2012-13)

The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. Each interaction message is assign by unique sequence number to identify the operation between the system object.

13. Define package and draw the UML notation for package. (U) (MAY/JUNE2012&13) (NOV/DEC2013)

Package diagrams organize the elements of a system into related groups to minimize dependencies among them.



14. What is interaction diagram? (R) (MAY/JUNE 2013) (NOV/DEC 2012)

This interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purposes of both the diagrams are similar. Sequence diagram emphasizes on time sequence of messages and collaboration diagram *emphasizes* on the structural organization of the objects that send and receive messages.

15. List the relationships used in Class Diagram. (R) (MAY/JUNE2014) (NOV / DEC 2015)

In a class diagram, there are following types of relationships amongst classes.

- **Association** - An Association is a connection between classes, showing how one class relates to another.
- **Aggregation -** Aggregation is a special case of association used to model a "Whole to its Parts" relationship. An Aggregate relationship can be thought of as a "Consists Of" relationship, where the Whole consists of the Parts.
- **Generalization** - The term generalization is used to specify the classification relationship between a general element and a more specific element. In fact, the term 'generalization' specifies a viewpoint focused on a classification hierarchy.
- **Dependencies**

16. What is meant by System Behavior? (R) (NOV / DEC 2015)

System behavior defines that "what a system will do in response to its external environment without referring to details on implementation" (e.g., use of technologies).

17. How to Naming System Events and Operations? (U) (NOV/DEC 2016)

System events should be expressed at the abstract level of intention rather than in terms of the physical input device. Choose event and operation names at an abstract level.

It also improves clarity to start the name of a system event with a verb (add..., enter..., end..., make...)

18. Define System Events and the System Boundary. (U)

To identify system events, it is necessary to be clear on the choice of system boundary. For the purposes of software development, the system boundary is usually chosen to be the software (and possibly hardware) system itself; in this context, a system event is an external event that directly stimulates the software.

19. What are the strength and weakness of sequence and collaboration diagram. (U)

(APR/MAY 2017) Type	Strengths	Weaknesses
Sequence	Clearly shows sequence or time ordering of messages large set of detailed notation options	Forced to extend to the right when adding new objects; consumes horizontal space
Communication	Space economical—flexibility to add new objects in two dimensions	More difficult to see sequence of messages Fewer notation options

20. Difference between logical architecture and layers. (APR/MAY 2017)(AN)

The logical architecture is the large - scale organization of the software classes into packages (or namespaces), subsystems, and layers. It's called the logical architecture because there's no decision about how these elements are deployed across different operating system processes or across physical computers in a network (these latter decisions are part of the deployment architecture).

- **A logical architecture** doesn't have to be organized in layers. But it's very common, and hence, introduced at this time.

A **layer** is a very coarse - grained grouping of classes, packages, or subsystems that has cohesive responsibility for a major aspect of the system. Also, layers are organized such that "higher" layers (such as the UI layer) call upon services of "lower" layers, but not normally vice versa. Typically layers in an OO system include:

- User Interface.
- Application Logic and Domain Objects
- Technical Services

21. Outline the key reason for modeling a package diagram. (U) (NOV/DEC 2017)

Package diagrams are used to structure high level system elements. Packages are used for organizing large system which contains diagrams, documents and other key deliverables.

Package Diagram can be used to simplify complex class diagrams, it can group classes into packages.

- A package is a collection of logically related UML elements.
- Packages are depicted as file folders and can be used on any of the UML diagrams.

22. Name two types of UML interaction Diagrams. (R) (NOV/DEC 2017)

Two types of interaction diagrams known as

1. Sequence diagram 2. Collaboration diagram.

23. Differentiate event, state and transition. (NOV/DEC 2020)

Event-occurrence that is relevant to an object or application.

State—the state of an object is determined by the value of some of its attributes and the presence or absences of links with other objects.

Transition—the movement from one state to another, triggered by an event

24. "Coupling should be low" – justify.

We can easily make changes to the internals of modules without worrying about their impact on other modules in the system. Low coupling also makes it easier to design, write, and test code since our modules are not interdependent on each other.

PART B

1. What is System sequence diagrams explain with any one scenario? **(R)**
2. Briefly explain relationship between System sequence diagrams and Use case diagrams with example? **(U)(APRIL/MAY 2011)(MAY/JUNE 2013) (NOV/DEC2013) (MAY/JUNE2014)**
3. What is the Logical Architecture? Explain about its layers with diagram. **(R) (MAY/JUNE2014)**
4. What are Package diagrams? How will you apply UML for Package Diagrams? **(AP) (MAY/JUNE2014)**
5. What are the guidelines for Design with Layers? **(R)**
6. Explain with suitable illustration logical layered architecture for the iteration of NextGen application. **(AP)**
7. Describe UML Class diagram notation with examples? **(U)**
8. Explain in detail about UML interaction diagrams? **(U) (APRIL/MAY 2011)**
9. Compare sequence verses collaboration diagram with example. **(AN) (MAY/JUNE2012)**
10. Describe the UML notation for class diagram with an example **(R) (MAY/JUNE2012) (MAY/JUNE2013)**
11. Describe the strategies used to identify conceptual classes. Describe the steps to create a domain model used for representing conceptual classes. **(R) (MAY/JUNE2012)**
12. With a suitable example explain how to design a class. Give all possible representation in a class (name, attribute, visibility, methods, responsibilities)**(C) (NOV/DEC 11&2012)**
13. What do you mean by interaction diagrams? Explain them with a suitable example**. (U) (NOV/DEC2011) (NOV/DEC2013)**
14. Explain the method of identifying the classes using the common class approach with an example. **(U) (NOV/DEC 2009)**
15. Consider the Hospital Management System application with the Following requirements• System should handle the in-patient, out-patient information through receptionist.
    a. Doctors are allowed to view the patient history and give their prescription.

There should be a information system to provide the required information. Give the use case, class and object diagrams. **(C) (NOV/DEC 2009)**

16. What is the purpose of an Access Layer class? Describe the process of creating the access Layer Classes and design an Access class for account class of the ATM Bank System. **(C) (NOV/DEC2008)**
17. Describe about Factory Pattern? **(R) (APRIL/MAY 2011)**
18. Discuss in detail about Singleton GoF? **(R) (APRIL/MAY 2011)**
19. How to add new SSDs and Contracts to the design diagram? List the relationships used in Class Diagram. **(AN) (NOV/DEC 2015)**
20. What is meant by System Behavior? **(R) (NOV / DEC 2015)**
21. What are the concepts involved in domain refinement? List the relationships used in Class Diagram **(R) (NOV/DEC 2015)**
22. What is design pattern? Explain the GoF design patterns. **(R) (NOV/DEC 2016)**
23. Explain the UML class, Sequence and Interaction diagram for Library Management System. **(U) (APR/MAY 2017)**
24. State Model-View Seperation principles and explain its motivations. **(R) (APR/MAY 2017)**
25. Consider the following use cases that play a role in a banking system:
    (i) Deposit
    (ii) Withdraw(Minimum balance has to be checked)

    Model Sequence diagram for the above two use cases. **(C) (NOV/DEC 2017)**
26. Explain with a diagram gang of four (GoF) patter summary and relationships. **(R) (NOV/DEC 2017)**
27. Model a class diagram for a "Banking System". State the functional requirements you are considering. **(C) (NOV/DEC 2017)**
28. Describe the features of Low coupling and High Coupling with suitable examples. **(NOV/DEC-2018)(R)**
29. What is GRASP? list and explain the nine object oriented design principles.
30. With an illustrated example diagram, brief on adapter pattern.

UNIT V

UNIT V TESTING

Object Oriented Methodologies – Software Quality Assurance – Impact of object orientation on Testing – Develop Test Cases and Test Plans

1. What is meant by an axiom? List the two design axioms of object oriented design. (R)

(NOV/DEC2008)

An axiom is a fundamental truth that always is observed to be valid and for which there is no counter example or exception.

Two design axioms:

Axiom 1: The independence axiom

Axiom 2: The information axiom.

2. What is Object Oriented Methodologies?(R)

A System of methods used ina particular area of study or activity.Number of methodology are based on modeling the business problem and implementing the application in an object oriented fashion.

3. List down the three major methodologies used in OOM.(U)

- Rambaugh method is well-suited for describing the object model or the static structure of the system.
- The Jacobson et al.method is good for producing user-driven analysis models.
- The Booch method produces detailed object-oriented design models.

4. What is SQA?(R)

SOFTWARE QUALITY ASSURANCE (SQA) is a set of activities for ensuring quality in software engineering processes that ultimately results, or at least gives confidence, in the quality of software products.

5. List down the Activities of SQA?(R)

SQA includes the following activities:

- Process definition
- Process training
- Process implementation
- Process audit

6. What are the processes doing in SQA?(R)

SQA includes the following processes:

- Project Management
- Project Estimation
- Configuration Management
- Requirements Management
- Software Design
- Software Development [Refer to SDLC]

- Software Testing [Refer to STLC]
- Software Deployment
- Software Maintenance

7. Write down the software quality management systems.(A)

Capability Maturity Model Integration (CMMI) and ISO 9000 are some quality management systems

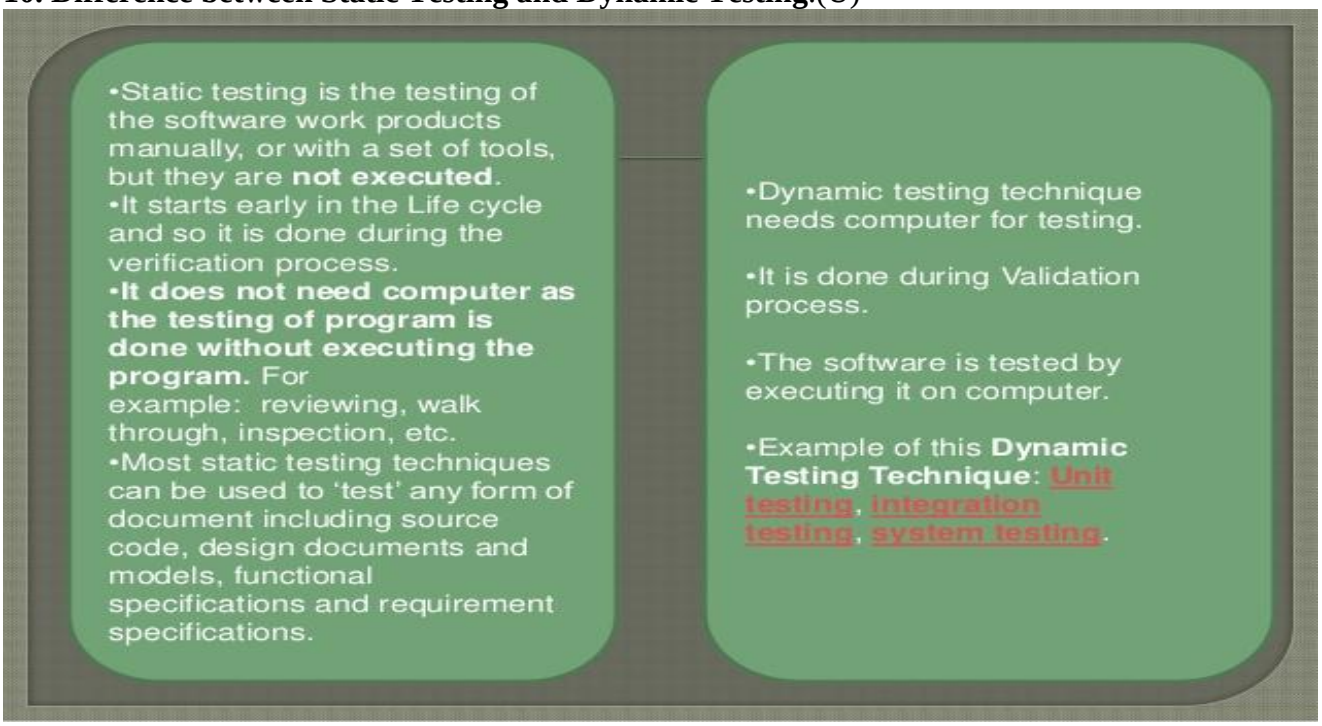
8. Write any three Components of SQA System.(A)

- Components of project life cycle activities assessment
- Components of infrastructure error prevention and improvement
- Components of software quality management
- Components of standardization, certification, and SQA system assessment
- Components of standardization, certification, and SQA system assessment
- Organizing for SQA – the human components

9. What is OOT?(U)

Object-Oriented Testing is a collection of testing techniques to verify and validate object-oriented software.

10. Difference between Static Testing and Dynamic Testing.(U)



11. What are the issues in object-Oriented Testing?(U)

1.) Basic unit of unit testing.

2.) Implication of Encapsulation

3.) Implication of Inheritance.

4.) Implication of Generality.

5.) Implications for testing processes

12. Write down the object-oriented testing techniques.(U)

Grey Box Testing:

- State **model based testing** − This encompasses state coverage, state transition coverage, and state transition path coverage.
- Use **case based testing** − Each scenario in each use case is tested.
- Class **diagram based testing** − Each class, derived class, associations, and aggregations are tested.
- Sequence **diagram based testing** − The methods in the messages in the sequence diagrams are tested

Subsystem Testing

The two main approaches of subsystem testing are −

Thread based testing − All classes that are needed to realize a single use case in a subsystem are integrated and tested.

Use based testing − The interfaces and services of the modules at each level of hierarchy are tested. Testing starts from the individual classes to the small modules comprising of classes, gradually to larger modules, and finally all the major subsystems.

13. What is Test case?(U)

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

14. What are the typical test cases parameters are used in testing?(U)

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters
- Actual Result
- Environment Information
- Comments

15. What is Test plan?(R)

Definition 1:

Test plan is developed to detect and identify potential problem before delivering the software to its users.

Definition 2:

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort.

16. What are the steps to create test plan?(R)

- Objectives of the test-what
- Development of a test case-how
- Test analysis-examining test output, and documentation the test result.

17. List down the test plan identifiers?(U)

S.No.	Parameter	Description
1.	Test plan identifier	Unique identifying reference.
2.	Introduction	A brief introduction about the project and to the document.
3.	Test items	A test item is a software item that is the application under test.
4.	Features to be tested	A feature that needs to tested on the testware.
5.	Features not to be tested	Identify the features and the reasons for not including as part of testing.
6.	Approach	Details about the overall approach to testing.
7.	Item pass/fail criteria	Documented whether a software item has passed or failed its test.
8.	Test deliverables	The deliverables that are delivered as part of the testing process,such as test plans, test specifications and test summary reports.
9.	Testing tasks	All tasks for planning and executing the testing.
10.	Environmental needs	Defining the environmental requirements such as hardware, software, OS, network configurations, tools required.
11.	Responsibilities	Lists the roles and responsibilities of the team members.
12.	Staffing and training needs	Captures the actual staffing requirements and any specific skills and training requirements.
13.	Schedule	States the important project delivery dates and key milestones.
14.	Risks and Mitigation	High-level project risks and assumptions and a mitigating plan for each identified risk.
15.	Approvals	Captures all approvers of the document, their titles and the sign off date.

18. Write down the Test Planning Activities?(R)

- To determine the scope and the risks that need to be tested and that are NOT to be tested.
- Documenting Test Strategy.
- Making sure that the testing activities have been included.
- Deciding Entry and Exit criteria.

- Evaluating the test estimate.
- Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.
- The Test artefacts delivered as part of test execution.
- Defining the management information, including the metrics required and defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test assets.

19. Define Test Oracle. (U)

Test Oracle is a document, or a piece of software that allows tester to determine whether a test has been passed or failed.

20. Define Test Bed. (U)

A test bed is an environment that contains all the hardware and software needed to test a software component or a software system.

21. What are the steps for mapping Designs to Code? (U) (NOV/DEC 2015)

Implementation in an object-oriented programming language requires writing source code for:

- Class and interface definitions
- Method definitions

22. List out the issues in OO Testing. (R)(NOV/DEC 2015)

- ❖ White-box testing methods can be applied to testing the code used to implement class operations, but not much else
- ❖ Black-box testing methods are appropriate for testing OO systems
- ❖ Object-oriented programming brings additional testing concerns
  - classes may contain operations that are inherited from super classes
  - subclasses may contain operations that were redefined rather than inherited
  - all classes derived from an previously tested base class need to be thoroughly tested

23. What is refactoring? (R)(NOV/DEC 2016)

Refactoring is "the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure". The concept of refactoring covers practically any revision or cleaning up of source code.

24. What is Regression testing? (R)(NOV/DEC 2016)

R**egression testing** is a type of software **testing** which verifies that software, which was previously developed and **tested**, still performs correctly after it was changed or interfaced

with other software. Changes may include software enhancements, patches, configuration changes, etc.

25. Mention steps involved in mapping design to code. (R) (APR/MAY 2017)

Creating class definitions from DCD

Creating Methods from Interaction diagram

26. What is unit testing? (R) (NOV/DEC 2017)

UNIT TESTING is a level of software testing where individual units / components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software.

27. Explain about OO Integration Testing. (R) (APR/MAY 2017)(Nov/Dec-2018)

When two or more modules are combined and tested, it is called *integration testing*. It occurs after unit testing and before validation testing.

31. **How is debugging different from testing? (Nov/Dec 2020)**

Testing is the process to find bugs and errors. Debugging is **the process to correct the bugs found during testing**. It is the process to identify the failure of implemented code. It is the process to give the absolution to code failure.

PART B

1. Explain in detail about object oriented Terminologies**.(R)**
2. Explain in detail about Object Oriented Models**.(R)**
3. Explain about Object-Oriented Metrics**.(R)**
4. Write detail in object-Oriented Testing Techniques**.(R)**
5. Write about Components of SQA System.(**R)**
6. What is test plan? List down the test plan identifiers**?(R)**
7. Explain detail about Test Planning Activities**.(R)**
8. What is test case? Explain about test case parameters with example**.(R)**
9. Consider the Hospital Management System application with the following requirements

   (i) System should handle the in-patient, out-patient information through receptionist.

   (ii) Doctors are allowed to view the patient history and give their prescription.

   (iii) There should be an information system to provide the required information.

   Give the state chart, component and deployment diagrams. **(C)**

   (NOV/DEC11, 12)

10. What is test case? Explain it with an Example **(R) (APRIL/MAY2008)**

11. What are the different testing strategies? Explain **(R) (APRIL/MAY 2008)**
12. Explain the various testing strategies. **(R) (NOV/DEC 2009)**
13. Give the use cases that can be used to generate the test cases for the Bank ATM Application**. (C) (NOV/DEC 2009)**
14. How will you measure the user satisfaction? Describe. **(U) (NOV/DEC2009)**
15. Perform the satisfaction test for any client/server application**.(R) (NOV/DEC 2009)**
16. Explain the object Oriented axioms and collaries. (R**) (NOV/DEC2008)**
17. Write notes on Implementation model (mapping design to code) **(R) (NOV/DEC2013)**
18. Elucidate the operation of mapping designs to code**. (NOV/DEC 2015)**
19. What is OO testing? Explain in detail about the concepts of OO testing in OOAD. **(NOV/DEC 2015)**
20. Consider the Hospital Management System application with the following requirements
    System should handle the inpatient. outpatient information through receptionist.
    Doctors are allowed to view the patient history and give their prescription.
    There should be a information system to provide the required information.
    Give the state chart, component and deployment diagrams. **(C)(NOV/DEC 2016)**
21. Explain the issues involved in OO? Testing. (APR/MAY 2017)(R)
22. Explain the Following
    (i) GUI Testing
    (ii) OO System Testing **(APR/MAY 2017)(R)**
20. How is a class testing different from conventional testing? Explain with an example. **(AN) (NOV/DEC 2017)**
21. Write a short note on system testing. **(R) (NOV/DEC 2017)**
22. What is integration testing? Explain the same with respect to object oriented systems. **(R) (NOV/DEC 2017)**
23. What is GUI based Testing? How does it help improving software design? Explain. **(NOV/DEC-2018)(R)**
24. How do you see the application of UML diagrams for Iterative Software Development? Explain. **(NOV/DEC-2018)(R)**
25. What are test cases? List the guidelines for developing quality assurance test cases. **(NOV/DEC-2020)(R)**
26. Suggest strategies to carry out unit testing and integration testing in an object oriented development environment. **(NOV/DEC-2020)(R)**