# NOORUL ISLAM COLLEGE OF ENGINEERING, KUMARACOIL

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# SEVENTH SEMESTER

# CS 1402 – OBJECT ORIENTED ANALYSIS AND DESIGN

# TWO MARK AND SIXTEEN MARK Q & A

Prepared By,

R. Suji Pramila,

Lect/CSE.

# Part – A Questions and Answers

## UNIT-I

1. Write about the traditional development methodologies?
     Most traditional development methodologies are either algorithm centric or
   data centric.
       a.  Algorithm-centric methodology
             ◊  You think of an algorithm that can accomplish the task, and then
                build data structures for that algorithm to use.
       b.  Data- centric methodology
             ◊  You think how to structure the data, and then build the algorithm
                around that structure.

2.  Define objects.
     Object is a real- world entity, identifiably separate from its surroundings, has a
   well defined set of attributes and a well-defined set of procedures or methods.
   Properties (or attributes) describe the state (data) of an object. Methods
   (procedures) define its behavior. Object means a combination of data and logic
   that represents some real-world entity.
   (E.g.)  Car is an object
           Color, manufacturer, cost, owner etc are attributes.
           Drive it, lock it, tow it, carry passengers in it are all methods.

3.  Give a brief note on object behavior
     Object behavior is described in methods or procedures. Behavior denotes the
   collection of methods that abstractly describes what an object is capable of doing.
   Each procedure defines and describes a particular behavior of an object. Methods
   encapsulate the behavior of the object. Objects take responsibility for their own
   behavior.

4.  What do you mean by information hiding?
     Information hiding is the principle of concealing the internal data and
   procedures of an object and providing an interface to each object and providing an
   interface to each object in such a way as to reveal as little as possible about its
   inner workings.

           An object is often said to encapsulate the date and a program.

◊ Per-class protection.
  o Class methods can access any object of that class and not just the receiver
◊ Per-object protection.
  o Methods can access only the receiver.

5. Define class hierarchy
  ◊ Object oriented system organizes classes into a subclass-super class hierarchy.
  ◊ At the top of the class hierarchy are the most general classes and at the bottom are the more specific.
  ◊ A subclass inherits all of the properties and methods defined in its super class.
  ◊ Subclasses may refine or constrain the state and behavior inherited from its super class.
  ◊ A class may simultaneously be the subclass to some class and a super class to another class.
  ◊ Formal or abstract classes have no instances but define the common behaviors that con be inherited by more specific classes.

6. Write briefly about inheritance and explain the types of inheritance.
  ◊ Inheritance is the property of object-oriented systems that allows objects to be built from other objects.
  ◊ Inheritance is a relationship between classes where one class is the parent class of another derived class called base class or super class.
  ◊ Inheritance allows classes to share and reuse behaviors and attributes of all its super classes.
  ◊ Types of inheritance
    ◊ Dynamic inheritance.
      It allows objects to change and evolve over time. Since base classes provide properties and attributes for objects, changing base classes changes the properties and attributes of a class.
    ◊ Multiple inheritance.
      Some object-oriented systems permit a class to inherit its state (attributes) and behaviors form more than one super class.

7. What do you mean by polymorphism?
  Polymorphism means that the same operation may behave differently on different classes. Polymorphism allows us to write generic, reusable code more easily, because we can specify general instructions and delegate the implementation details to the objects involved.

8. Explain object relationship and associations.
  a. Association represents the relationships between objects and classes.
  b. Associations are bidirectional.

| **Pilot** | Can fly _____ Flown by | **Plane** |
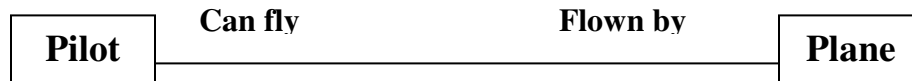
c. Cardinality specifies how many instances of one class may relate to a single instance of an associated class.

d. Cardinality constraints the number of related objects and often is described as being "one" or "many"

9. What do you mean by consumer-producer association?

It is also known as client-server association. It uses relationship. It is a one-way interaction in which one object requests the service of another object.

**Requesting for printing**

| Print server | ———————————————————— | Item |
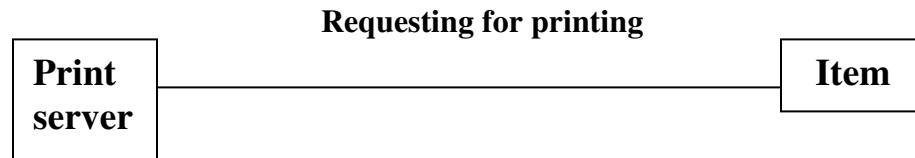
The object that makes the request is the consumer or client and the object receives the request and provides the service is the producer or server.
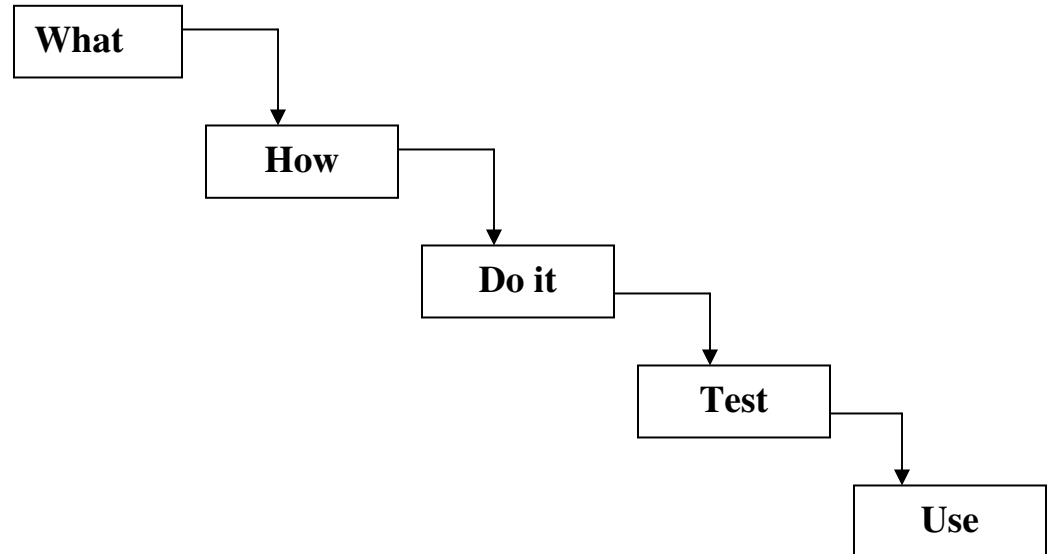
10. Write about static and dynamic binding?
   o Static binding
     ▪ The process of determining which functions to invoke at compile time is termed as static binding. Static binding optimizes the calls.
   o Dynamic binding
     ▪ The process of determining which functions to invoke at runtime are called as dynamic binding. They occur when polymorphic calls are issued. Dynamic binding allows some method invocation decisions to be deferred until the information is known.
     ▪

11. Define object persistence

A file or a database can provide support for objects having a longer lifeline, longer than the duration of the process for which they were created. This is called object persistence. An object can persist beyond application session boundaries, during which the object is stored in a file or a database. The object can be retrieved in an other application session and will have same state and relationship.

12. Define meta-classes.

If a class is an object, it must belong to a class which is called as meta-class or a class of classes. All the objects are instances of a class and all classes are instances of a meta-class. Meta-classes are used by the compiler.

13. What do you mean by software development process?

The software development process consists of
   ◊ Analysis
   ◊ Design
   ◊ Implementation

◊ Testing
◊ Refinement

The essence of software development process is to transform users' needs into a software solution that satisfies those needs. The main point is to build high-quality software.

14. Explain briefly the waterfall approach.
    a. It starts with deciding what is to be done
    b. Once the requirements have been determined, we next must decide how to accomplish them.
    c. Followed by a step in which we do it.
    d. We then must test the result to see if we have satisfied the users' requirements.
    e. Finally we use what we have done.

```
┌──────────┐
│  What    │──────┐
└──────────┘      │
             ┌────▼─────┐
             │   How    │──────┐
             └──────────┘      │
                          ┌────▼─────┐
                          │  Do it   │──────┐
                          └──────────┘      │
                                       ┌────▼─────┐
                                       │   Test   │──────┐
                                       └──────────┘      │
                                                    ┌────▼─────┐
                                                    │   Use    │
                                                    └──────────┘
```

15. Define collaboration.

The object-oriented programming community has adopted use-cases to a remarkable degree. Scenarios are a great way of examining who does what in the interactions among objects and what role they play; that is their interrelationships. This intersection among objects' roles to achieve a given goal is called collaboration.

16. Write the 80-20 rule.
    a. Documentation is another important activity, which does not end with object-oriented analysis but should be carried out throughout the system development.
    b. The 80-20 rule generally applies for documentation:
        ◊ 80 percent of the work can be done with 20 percent of the documentation.

17. Define Prototype. Give the types of prototype.

A prototype is a version of a software product developed in the early stages of the product's life cycle for specific, experimental purposes. A prototype enables you to fully understand how easy or difficult it will be to implement some of the features of the system. It also gives users a chance to comment on the usability and usefulness of the user interface design.

Prototypes have been categorized in various ways.
- ✓ Horizontal prototype
- ✓ Vertical prototype
- ✓ Analysis prototype
- ✓ Domain prototype

18. Write a brief note on RAD.
   a. Rapid application development (RAD) is a set of tools and techniques that can be used to build an application faster than typically possible with traditional methods.
   b. The developer sacrifices the quality of the product for the quicker delivery.
   c. RAD does not replace the system development life cycle but complements it, since it focuses more on process description and can be combined perfectly with the object-oriented approach.
   d. The main objective of RAD is to build a version of an application rapidly to see whether the system does what it is supposed to do.
   e. RAD involves a number of iterations. RAD encourages the incremental development approach.

19. Write about CBD?

Component based development is an industrialized approach to the software development process. Application development moves from custom development to assembly of prebuilt, pretested, reusable software components that operate with each other.

20. Why do we go for object oriented systems development?

The motivation factor behind object-oriented system development is the desire to make software development easier and more natural by raising the level of abstraction to the point where the level of abstraction to the point where applications can be implemented in the same terms n which they are described by users.

21. What is horizontal prototype?

It is a simulation of the interface ie, it has the entire user interface that will be in the full-featured system but contains no functionality.

22. Explain the domain prototype?

It is an aid for the incremental developmental of the ultimate software solution. It often is used as a tool for the staged delivery of subsystems to the users or other members of the developmental team.

23. Write about the Vertical prototype?

It's a subset of the system features with complete functionality. The principal advantage of this method is that the few implemented functionality can be tested in great depth.

24. Explain Analysis prototype?

It is an aid for exploring the problem domain. This class of prototype is used to inform the user and demonstrate the proof of a concept. It is not used as the basis of development. The final product will use the concepts exposed by the prototype, not its code.

# UNIT-II

1. Write about the four phases in OMT?

OMT consists of four phases. They are
- Analysis-The results are objects and dynamic & functional models.
- System design-The results are a structure of the basic architecture of the system along with high-level strategy decisions.
- Object Design-Produces a design document, consisting of detailed objects static, dynamic and functional models
- Implementation-This activity produces reusable, extendible, robust code.

2. What do you mean by object diagram?

The object model of OMT is represented graphically with an object diagram. The object diagram contains classes interconnected by association lines. Each class represents a set of individual objects. The association lines establish relationships among the classes. Each association line represents a set of links from the objects o f one class to the objects of another class

3. What are the primary symbols used in Data Flow Diagrams?

Data flow diagrams use four primary symbols:
- The **process** is any function being performed.(verify password in the ATM)
- The **data flow** shows the direction of data element movement.(PIN code)
- The **data store** is a location where data are stored.(Account in ATM)
- An **external entity** is a source or destination of a data element .(the ATM card reader)

| | |
|---|---|
| ⬭     → | **Process** |
| ▭ (two lines)     → | **Data Store** |
| →     → | **Data Flow** |
| ▭     → | **External entity** |
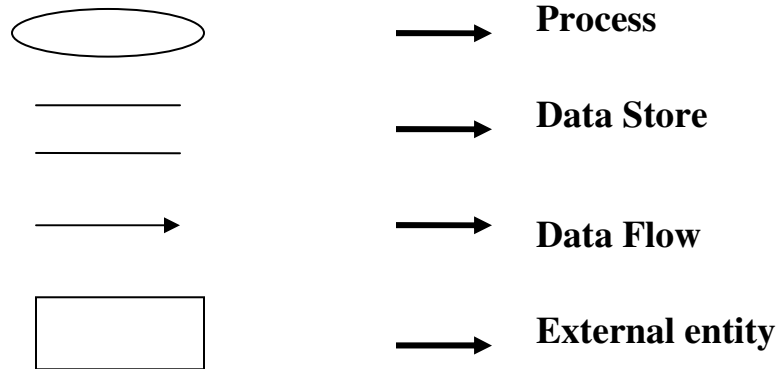
4. What are the diagrams used in Booch methodology?

The Booch methodology consists of the following diagrams:

   ◊ Class diagrams.
   ◊ Object diagrams.
   ◊ State transition diagrams.
   ◊ Module diagrams.
   ◊ Process diagrams.
   ◊ Interaction diagrams.

5. Give the steps involved in Macro development process in Booch methodology.

The macro development process consists of the following steps:

   o Conceptualization
      ▪ Establish the core requirements and develop a prototype.
   o Analysis and development of the model
      ▪ Use the class diagram to describe the roles and responsibilities of objects. Use the object diagram to describe the desired behavior of the system.
   o Design or create the system architecture.
      ▪ Use the class diagram to decide what classes exist and how they relate to each other, the object diagram to decide what mechanisms are used, the module diagram to map out where each class and object should be declared, and the process diagram to determine to which processor to allocate a process.
   o Evolution or implementation-
      ▪ Refine the system through much iteration.
   o Maintenance-Make localized changes to the system to add new requirements and eliminate bugs.

6. Give the steps involved in Micro development process in Booch methodology.

The micro process is a description of the day-to-day activities by a single or small group of software developers. It consists of the following steps.

      o Identify classes and objects.
      o Identify class and object semantics.
      o Identify class and object relationships.

o Identify class and object interface and implementation.

7. Write briefly about Use Cases.

Use cases are scenarios for understanding system requirements. A use case is an interaction between users and a system. The use-case model captures the goal of the user and the responsibility of the system to its users. The use-case model employs extends and uses relationships. The use cases are described as one of the following:

◊ Nonformal text with no clear flow of events
◊ Text with a clear flow of events
◊ Formal style using pseudo code.

8. Write short note on Objectory.

Object-oriented software engineering (OOSE), also called Objectory, is a method or object-oriented development with the specific aim to fit the development of large, real-time systems. Objectory, is a disciplined process for the industrialized development of software, based on a use-case driven design. Objectory is built around several different models:

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

9. Define patterns.

Design pattern identifies the key aspects of a common design structure that makes it useful for creating a reusable object-oriented design. Furthermore, it identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities. It describes when it applies, whether it can be applied in view of other design constraints and the consequences and trade-offs of its use.

A pattern is an instructive information that captures the essential structure and insight of a successful family of proven solutions to a recurring problem that arises within a certain context and system of forces.

10. Define proto-patterns.

Even if something appears to have all the requisite pattern components, it should not be considered a pattern until it has been verified to be a recurring phenomenon (preferably found in at least three existing system; this often is called rule of three). A proto-pattern is the "pattern in waiting" which is not yet known to recur. It is the pattern waiting to undergo some degree of peer scrutiny or review.

11. Define patterns template. Give some examples for components in pattern.

Every pattern must be expressed in the form of a rule which is called as a

template. It should establish a relationship between a context, a system of forces which arises in the context, and a configuration. Some of the essential components are:

- ◊ Name
- ◊ Problem
- ◊ Context
- ◊ Forces
- ◊ Solution
- ◊ Examples

12. Define anti-patterns.

An anti-pattern represents a worst practice while a pattern represents a best practice. Anti-patterns come in two varieties:

- Those describing a bad solution to a problem that resulted in a bad situation.
- Those describing how to get out of a bad situation

13. Define pattern mining. Give the steps involved in capturing pattern.

The process of looking for patterns to document is called pattern mining sometimes called reverse architecturing. The steps involved are:

- ◊ Focus on practicability.
    - o Patterns should describe proven solutions to problems rather than the latest scientific results.
- ◊ Aggressive disregard of originality.
    - o Pattern writers do not need to be the original inventor of the soln.
- ◊ Non anonymous review.
    - o Pattern submissions are shepherded rather than reviewed.
- ◊ Writers' workshops instead of presentations.
    - o Rather than being presented by the authors, the patterns are discussed in the writers' workshops.
- ◊ Careful editing.
    - o The pattern authors should have the opportunity to incorporate all the comments and insights during the shepherding.

14. Define frame work. Give the differences between design patterns and frameworks.

A frame work is a way of presenting a generic solution to a problem that can be applied to all levels in a development. Frameworks are a way of delivering application development patterns. A framework provides architectural guidance, captures the design decisions that are common to its application domain and thus emphasize design reuse over code reuse.

The major differences between design patterns and frameworks are as follows:

- ◊ A framework is executable software whereas design patterns represent knowledge and experience about the software.

◊   Design patterns are more abstract than frameworks.
◊   Design patterns are smaller architectural elements than frameworks.
◊   Design patterns are less specialized than frameworks.

15. Why do we go for unified approach?
    The main motivation for going to unified approach is to combine the best practices, processes, methodologies, and guidelines along with UML (Unified Modeling Language) notations and diagrams for better understanding object-oriented concepts and system development.

16. Write short note on UA proposed Repository.
    The repository allows the maximum reuse of previous experience and previously defined objects, patterns, frameworks, and user interfaces in an easily accessible manner with a completely available and easily utilized format. Everything from the user request to maintenance of the project should be kept in the repository. This will reduce the cost and development time. It should be relatively easy to search the repository.

17. Define model. Explain about the types of model.
    A model is an abstract representation of a system, constructed to understand the system prior to building or modifying it. It is a model of a simplified representation of reality. Models can represent static or dynamic situations.
    • Static model
      o It can be viewed as a snapshot of a system's parameters at rest or a specific point in time. They are needed to represent the structural or static aspect of a system. The UML class diagram is an example of static model.
    • Dynamic model
      o It can be viewed as a collection of procedures or behaviors that taken together reflect the behavior of a system over time. Dynamic modeling is the most useful during the design and implementation phases of the system development. The UML interaction diagrams and activity models are examples of dynamic models.

18. What are the advantages of Modeling?
    Good models are essential for communication among project teams. As the complexity of systems increases, so does the importance of good modeling techniques. Some of the advantages are as follows:
    ◊   Models make it easier to express complex ideas.
    ◊   The main reason for modeling is to reduction of complexity.
    ◊   Models enhance and reinforce learning and training.
    ◊   The cost of modeling analysis is much lower than the cost of similar perimentation conducted in real system.
    ◊   Manipulation of the model is much easier.

19. Define UML. Mention the primary goals in the design of the UML.
        The unified modeling language (UML) is a language for specifying,

constructing, visualizing and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The primary goals in the design of the UML were as follows:

◊ Provide users a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models
◊ Provide extensibility and specialization mechanisms to extend the core concepts.
◊ Be independent of particular programming languages and development processes.
◊ Provide a formal basis for understanding the modeling language.
◊ Encourage the growth of the OO tools market.
◊ Support higher-level development concepts.
◊ Integrate best practices and methodologies.

20. Give the nine UML graphical diagrams.
   a. Class diagram(static)
   b. Use-case diagram
   c. Behavior diagram(dynamic)
      i. Interaction diagram
         1. Sequence diagram
         2. Collaboration diagram
      ii. State chart diagram
      iii. Activity diagram
   d. Implementation diagram.
      i. Component diagram
      ii. Deployment diagram.

21. What is a Package?
   A package groups and manages the modeling elements, such as classes, their associations, and their structures. Packages themselves may be nested within other packages.

# UNIT-III

1. What is the purpose of analysis? Why do we need analysis?
   The main objective of the analysis is to capture a complete, unambiguous, and consistent picture of the requirements of the system and what the system must do to satisfy the users' requirements and needs. Object analysis is a process by which we can identify the classes that play a role in achieving the system goals and requirements. Hence, we are in need of analysis.

2. Why analysis is a difficult activity?
   Analysis is a creative activity that involves understanding the problem, its associated constraints, and methods of overcoming those constraints. The three most common sources of requirement difficulties are:

◊ Fuzzy descriptions:
Fuzzy and ambiguous description leads to requirement ambiguity.
◊ Incomplete requirements:
Certain requirements that are not included for variety of reasons might be necessary for successful system development.
◊ Unnecessary features:
Every additional feature could affect the performance, complexity, stability, and support costs of an application.

Analysis is a difficult activity because one has to understand the problem in some application domain and then has to define a solution that can be implemented with any software.

3. What do you mean by business object analysis?

Business object analysis is a process of understanding the system's requirements and establishing the goals of an application. The main intent of this activity is to understand users' requirements. The outcome of the business object analysis is to identify classes that make up the business layer and the relationships that play a role in achieving system goals.

4. Write a short note on use-case model?

A use-case model can be instrumental in project development, planning, and documentation of systems requirements. The use-case model describes the uses of the system and shows the courses of events that can be performed. It also defines what happens in the system when the use case is performed. It can also discover the classes and the relationships among subsystems of the system.

5. Define use-case.

Use cases are scenarios that describe how actors use the system. A use case is an interaction between users and a system. It captures the goal of the users and the responsibility of the system to its users. Jacobsons' definition of use case is, "A use case is a sequence of transactions in a system whose task is to yield results of measurable value to an individual actor of the system."

6. When 'extends' association is used?

The 'extends' association is used when you have one use case that is similar to another use case but does a bit more or is more specialized. It is like a subclass. Extends association is utilized to expand the common behavior to fit the special circumstances.

7. Define 'uses' association.

The uses association occurs when some of the use cases have subflows in common. Here to avoid describing a subflow more than once in several use cases, the common subflow can be extracted and made into a new use case of its own. The relationship among the other use cases and this new extracted use case is called a uses association.

When you want to share common sequences in several use cases, utilize the uses association by extracting common sequences into a new, shared use case. The uses association helps us to avoid redundancy by allowing a use case to be shared.

8. What is meant by railroad paradox? What do you infer from railroad paradox?

When rail roads were asked to establish new stops on the schedule they "studied the requirements", by sending someone to the station at eh designated time to see if anyone was waiting for a train. Of course, nobody was there because no stop was scheduled, so the railroad turned down the request because there was no demand.

The railroad paradox appears everywhere and goes like this:
◊ The product is not satisfying the users.
◊ Since the product is not satisfactory, potential users will not use it.
◊ Potential users ask for a better product.
◊ Because the potential users do not use the product, the request is denied.

9. Give the two-three rule?

The two-three rule is for identifying actors. The rule is stated as "start with naming at least two, preferable three, people who could serve as the actors in the system. Other actors can be identified in the subsequent iterations.

10. What is the 80-20 rule?

The 80-20 rule applies for documentation. 80 percent of the work can be done with 20 percent of the documentation. 20 percent is easily accessible and the remaining (80 percent) is available to those (few) who need to know.

11. Why is documentation an important part of analysis?

Documentation provides a valuable reference point and a form of communication to reveal issues and gaps in the analysis ad design. A document can serve as a communication vehicle among the project's team members. The main issue in documentation during the analysis phase is to determine what the system must do. Documentation can be an important factor in making a decision about committing resources.

12. List the approaches for identifying classes?

The four alternative approaches for identifying classes:
◊ The noun phrase approach.
◊ The common class patterns approach.
◊ The use-case driven, sequence/collaboration modeling approach.
◊ The classes, responsibilities and collaborators (CRC) approach.

13. What do you mean by relevant, fuzzy and irrelevant classes?

In the noun phrase approach the list of nouns is divided into three categories:
◊ Relevant classes:

Relevant classes have a purpose. They are clearly defined and so they are necessary.

◊ Fuzzy classes:
Fuzzy classes are those classes which we are not sure about.

◊ Irrelevant classes:
These classes have no purpose and are unnecessary. It is safe to scrap these irrelevant classes.

14. How would you select candidate classes for the list of relevant and fuzzy classes?

Candidate classes are selected from the relevant and fuzzy categories. The guidelines that are to be followed are:

◊ Redundant classes: Don't keep two classes that express the same information
◊ Adjectives classes: If the object behave differently when the adjective is applied then make a new class
◊ Attribute classes: Tentative objects that are used only as values should be defined as attributes and not as a class
◊ Irrelevant classes: Each class should have a purpose.

A statement of purpose for each candidate class must be formulated. If it cannot be done then eliminate that candidate class.

15. What is the common class patterns strategy? Give the list of patterns used.

The common class patterns approach is based on a knowledge base of the common classes that have been proposed researchers. The patterns used for finding the candidate class and object are:

◊ Concept class
◊ Events class
◊ Organization class
◊ People class
◊ Places class
◊ Tangible things and devices class

16. What is CRC?

Classes, responsibilities, and collaborators is a technique used for identifying classes' responsibilities, and collaborators and therefore their attributes and methods. Furthermore, CRC can help us identify classes. CRC is based on the idea that an object either can accomplish a certain responsibility itself or it may require the assistance of other objects.

CRC cards are 4"``X 6`` index cards in which all the information for an object is written is cheap, portable, readily available and familiar.

| Class Name | Collaborators |
|---|---|
| Responsibilities | … |
| … | |
| | |

17. What are the three steps in CRC process?

The classes, responsibilities and collaborators process consists of three steps:

◊ Identify classes' responsibilities (and identify classes).
◊ Assign responsibilities.
◊ Identify collaborators.
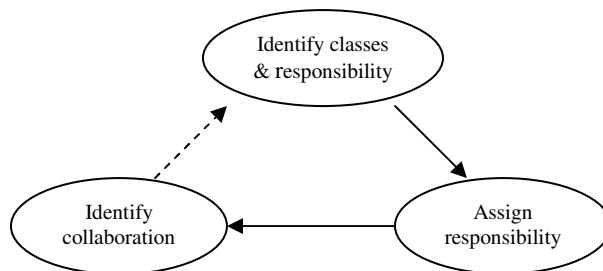


18. Give the guidelines for naming a class.

The guidelines for naming classes:

◊ The class name should be singular.
◊ One general rule for naming classes is that you should use names with which the users or clients are comfortable.
◊ The name of a class should reflect its intrinsic nature.
◊ Use readable name. Capitalize class names.

19. What is an association?

Association represents a physical or conceptual connection between two or more objects. Binary associations are shown as lines connecting two class symbols. Ternary and higher-order associations are shown as diamonds

connecting to a class symbol by lines, and the association name is written above or below the line.

| Class A | Association name | Class B |
|---------|------------------|---------|

Role of A        Role of B
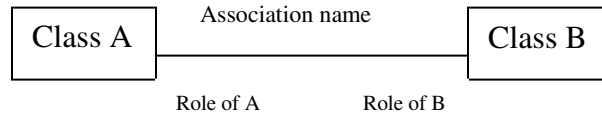
20. What is generalization hierarchy? Give the advantage.

Super class–subclass relationship also known as generalization hierarchy, allow objects to be built from other objects. Such relationships allow us to explicitly take advantage of the commonality of objects when constructing new classes. The super-sub class hierarchy is a relationship between classes, where one class is the parent (super or ancestor) class of another (derived) class. The real advantage of using this technique is that we can build on what we already have and more important, reuse what we already have.

21. What are some common associations?

The common association patterns which can be stored in the repository are based on some common associations:

◊ Location association:
Next to, part of, contained in.
◊ Communication association:
Talk to, order to.

22. How to eliminate unnecessary associations? How would you know it?

Some of the unnecessary associations are:

◊ Implementation association:
Defer implementation-specific associations to the design phase.
◊ Ternary associations:
Ternary or n-ary associations complicate the representation. So when possible restate ternary associations as binary associations.
◊ Directed actions (or derived) association:
Directed associations can be defined in terms of other associations. Since they are redundant, avoid these types of association.

The unnecessary associations are discovered by testing access paths to objects.

23. What do you mean by aggregation? What are the major properties of **a-part-of** relation?

A-part-of relationship, also called aggregation, represents the situation where a class consists of several component classes. A class that is composed of other classes does not behave like its parts; actually, it behaves very differently.

Two major properties of a-part-of relationship are:

◊ Transitivity:
If A is part of B and B is part of C, then A is part of C.

For example; a carburetor is part of an engine and an engine is part of a car; therefore, a carburetor is part of car.

◊ Antisymmetry:

If A is part of B, then B is not part of A. For example; an engine is part of part of a car, but a car is not part of an engine.

24. What guidelines would you see to identify a-part-of structures?

To identify a-part-of structures the following guidelines are provided:

◊ Assembly:

An assembly is constructed from its parts and an assembly-part situation physically exists.

◊ Container:

A physical whole encompasses but is not constructed from physical parts.

◊ Collection-member:

A conceptual whole encompasses parts that may be physical or conceptual.

25. Why do we need to identify the system's responsibilities?

We need to identify the system's responsibilities because responsibilities identify problems that are to be solved. A responsibility serves as a handle for discussing potential solutions. Once the system's responsibilities are understood we can start identifying the attributes of the system's classes.

26. How would you identify attributes?

◊ Attributes usually correspond to nouns followed by preposition phrases. Attributes also may correspond to adjectives or adverbs.

◊ Keep the class simple; state only enough attributes to define the object state.

◊ Attributes are less likely to be fully described in the problem statement.

◊ Omit derived attributes. They should be expressed as a method.

◊ Do not carry excess identification.

27. How would you identify methods?

◊ The sequence diagrams assist us in defining services that the objects must provide. These services are implemented as the methods for your objects.

◊ In a sequence diagram the events that occur between objects are drawn between the vertical object lines. An event is considered to be an action that transmits information; therefore these actions are the operations that the objects must perform.

◊ Methods also can be derived from the scenario testing.

28. Why do we need methods and messages in object-oriented system?

Objects not only describe abstract data but also must provide some services. Methods and messages are the workhorses of object-oriented systems. In an object-oriented environment, every piece of data, or object, is surrounded by a rich set of routines called methods. Methods are responsible for managing the value of attributes such as query, updating, reading and writing.

# UNIT-IV

1.  What is the need for axiomatic approach?

    The basic goal of the axiomatic approach is to formalize the design process and assist in establishing a scientific foundation for the object-oriented design process, so as to provide a fundamental basis for the creation of systems. Without scientific principles, the design field never will be systematized and so will be difficult to comprehend, codify, teach and practice.

2.  What are the main activities in design process?
    ◊ Designing classes (their attributes, methods, associations, structures, and protocols) and applying design axioms. If needed, this step is repeated.
    ◊ Designing the access layer.
    ◊ Designing the user interface (view layer classes).
    ◊ Testing user satisfaction and usability, based on the usage and use cases.
    ◊ Iterating and refining the design.

3.  Define axiom? What are the two design axioms applied to object-oriented design?

    An axiom is a fundamental truth that always is observed to be valid and for which there is no counterexample or exception. The axioms cannot be proven or derived but they cannot be invalidated by counterexamples or exceptions. There are two design axioms applied to object-oriented design. Axiom 1 deals with relationships between system components and Axiom 2 deals with the complexity of design.

    Axiom 1: The independence axiom. Maintain the independence of components.

    Axiom 2: The information axiom. Minimize the information content of the design.

4.  Define corollary? Give the corollaries derived from design axioms. (or) List the various design rules?

    A corollary is a proposition that follows from an axiom or another proposition that has been proven. A corollary is shown to be valid if its referent axioms and deductive steps are valid. The design rules or corollaries derived from design axioms are stated below.

    Corollary 1: Uncoupled design with less information content.
    Corollary 2: Single purpose.
    Corollary 3: Large number of simple classes.
    Corollary 4: Strong mapping.
    Corollary 5: Standardization.
    Corollary 6: Design with inheritance.

5. What do you mean by coupling?

Coupling is a measure of the strength of association established by a connection from one object or software component to another. Coupling is a binary relationship. For example A is coupled with B. Coupling is important when evaluating a design because it helps us focus on an important issue in design.

6. What do you mean by degree of coupling?

The degree of coupling is a function of
◊ How complicated the connection is.
◊ Whether the connection refers to the object itself or something inside it.
◊ What is being sent or received.

The degree or strength of coupling between two components is measured by the amount and complexity of information transmitted between them. Coupling increases with increasing complexity and decreases when the connection is to the component interface rather than to an internal component. Coupling is also lower for data connections than for control connections.

7. What are the two types of coupling?

Object oriented design has two types of coupling. They are,
◊ Interaction coupling

Interaction coupling involves the amount and complexity of messages between components. It is desirable to have little interaction.
◊ Inheritance coupling

Inheritance is a form of coupling between super and subclasses. A subclass is coupled to its super class in terms of attributes and methods. Unlike interaction coupling, high inheritance coupling is desirable.

8. What do you mean by cohesion? Give the types of cohesion.

Cohesion can be defined as the interactions within a single object or software component. Cohesion reflects the "single-purpose ness" of an object. Cohesion helps in designing classes that have very specific goals and clearly defined purposes.
◊ Method cohesion

A method should carry only one function.
◊ Class cohesion

All the class's methods and attributes must be used by internal methods or derived classes' methods.
◊ Inheritance cohesion

Concerned with how classes are interrelated.

9. Differentiate coupling and cohesion?

Coupling deals with interactions between objects or software components while cohesion deals with the interactions within a single object or software component. Highly cohesive components can lower coupling because only a minimum of essential information need to b passed between components.

10. What do you mean by design patterns?

Design patterns are devices that allow systems to share knowledge about their design, by describing commonly recurring structures of communicating components that solve a general design problem within a particular context. A design pattern provides a scheme for refining the subsystems or components of a software system or the relationships among them. Design patterns are documented by writing essays in a fairly well-defined form.

11. Define OCL?

The rules and semantics of the UML are expressed in English, in a form known as object constraint language. Object constraint language (OCL) is a specification language that uses simple logic for specifying the properties of a system.

12. What do you mean by expressions? Give the syntax for some common expressions.

Expressions are stated as strings in object constraint language. The syntax for some common expressions is given here. The leftmost element must be an expression for an object or a set of objects. The expressions are meant to work on sets of values when applicable.
   ◊  Item. Selector: (e.g.) John. age
   ◊  Item. Selector [qualifier-value]: (e.g.) John. Phone[2]
   ◊  Set->select(Boolean-expression): (e.g.) company.employee-salary->30000

13. What are private, public and protected protocols?
   ◊  Private protocol of the class includes messages that normally should not be sent from other objects. The messages are accessible only to operations of that class. Only the class itself can use the method.
   ◊  The public protocol defines the stated behavior of the class so that it is accessible to all classes.
   ◊  If the methods or attributes have to be used by the class itself or its subclasses, a protected protocol can be used. In a protected protocol, subclasses can use the method in addition to the class itself.

14. What is encapsulation leakage?

Encapsulation leakage is lack of a well-designed protocol. The problem of encapsulation leakage occurs when details about a class's internal implementation are disclosed through the interface. As more internal details become visible, the flexibility to make changes in the future decreases.

15. What are the three basic types of attributes?
   The three basic types of attributes are
   ◊  Single-value attributes.
         The single-valued attribute has only one value or state.
   ◊  Multiplicity or multi value attributes.
         The multiplicity or multi valued attribute can have a collection of many values at any point in time.

◊   Reference to another object, or instance connection.

These attributes are required to provide the mapping needed by an object to fulfill its responsibilities, in other words, instance connection model association.

16. How do you present UML attribute?

The following is the attribute presentation suggested by UML.
*Visibility name: type-expression=initial-value*

*Visibility name* is +(public visibility), #(protected visibility), –(private visibility)
*Type-expression* is a language-dependent specification of the implementation type
*Initial-value* is a language-dependent expression for the initial value of a newly created object.

17. What are the different types of methods provided by a class?

A class can provide several types of methods:
◊   Constructor: Method that creates instances (objects) of the class.
◊   Destructor: The method that destroys instances.
◊   Conversion method: The method that converts a value from one unit of measure to another.
◊   Copy method: The method that copies the contents of one instance to another instance.
◊   Attribute set: The method that sets the values of one or more attributes.
◊   Attribute get: The method that returns the values of one or more attributes.
◊   I/O methods: The methods that provide or receive data to or from a device.
◊   Domain specific: The method specific to the application.

18. What are some characteristics of a bad design?

The five rules are,
◊   If it looks messy, then it's probably a bad design.
◊   If it is too complex, then it's probably a bad design.
◊   If it is too big, then it's probably a bad design.
◊   If people don't like it, then it's probably a bad design.
◊   If it doesn't work, then it's probably a bad design.

19. How do you present UML operation?

The presentation of operation in UML is as given below.
*Visibility name: (parameter-list): return-type-expression*

*Visibility name* is + (public visibility), # (protected visibility), – (private visibility)
*Parameter-list* is a list of parameters, separated by commas and each specified by

*Name: type-expression=default value,*

*Name* is the name of the parameter.
*Type-expression* is a language-dependent specification of the implementation type

*Default-value* is an optional value.

*Return-type-expression* is a language-dependent specification of the implementation of the value returned by the method

20. Define Package

A package groups and manages the modeling elements, such as classes, their associations, and their structures. Packages themselves may be nested within other packages. A package may contain both other packages and ordinary model elements. The entire system description can be thought of as a single high-level sub-system package with everything else init. All kinds of UML model elements and diagrams can be organized into packages.

21. What do you mean by persistence? Give some persistent data.

Persistence refers to the ability of some objects to outlive the programs that created them. The persistent data are those data that exist beyond the lifetime of the creating process. Some categories of persistent data are:
◊ Data that exist between the executions of a program.
◊ Data that exist between the versions of a program.
◊ Data that outlive a program.

22. Define transient data? Give some transient data?

Transient data are those data that cease to exist beyond the lifetime of the creating process. Some categories of transient data are:
◊ Transient results to the evaluation of expressions.
◊ Variables involved in procedure activation (parameters and variables with a localized scope.)
◊ Global variables and variables that are dynamically allocated.

23. What are the essential elements in providing a persistent store?

Essential elements in providing a persistent store are:
◊ Identification of persistent objects or reachability
◊ Properties of objects and their interconnections.
◊ Scale of the object store.
◊ Stability.

24. Define schema or meta-data?

Schema or meta-data contains a complete definition of the data formats, such as the data structures, types and constraints. The meta-data are usually encapsulated in the application programs themselves. In DBMS, the format of the meta-data is independent of any particular application data structure; therefore it will provide a generic storage management mechanism.

25. What is meant by database model? Give the different database models.

A database model is a collection of logical constructs used to represent the data structure and data relationships within the database. The different database models are,

◊ Hierarchical model
◊ Network model
◊ Relational model

26. Define DDL and DML.
Data definition language (DDL) is the language used to describe the structure of and relationships between objects stored in a database. This structure of information is termed as the database schema.
Data manipulation language (DML) is the language that allows users to access and manipulate data organization. The structured (SQL) is the standard DML for relational DBMS.

27. What is concurrency policy?
A concurrency control policy dictates what happens when conflicts arise between transactions that attempt access to the same object and how these conflicts are to be resolved. There are two policies,
◊ Conservative or pessimistic policy
Allows a user to lock all objects or records when they are accessed and to release the locks only after a transaction commits.
◊ Optimistic policy
Two conflicting transactions are compared in their entirety and then their serial ordering is determined.

28. What is shareability?
Data and objects in the database often need to be accessed and shared by different applications. With multiple applications having access to the object concurrently, it is likely that conflicts over object access will arise. The database must detect and mediate these conflicts and promote the greatest amount of sharing possible without sacrificing the integrity of data.

29. What do you mean by transaction?
The basic goal of the transactions is to provide each user with a consistent view of the database. A transaction is a unit of change in which many individual modifications are aggregated into a single modification that occurs in its entirety or not at all. Thus either all changes to objects within a given transaction are applied to the database (commit) or none of the changes (abort). This ability of transactions ensures atomicity.

30. Define client-server computing?
In client-server computing the calling module becomes the "client" and the called module becomes the "server". The connectivity allows applications to communicate transparently with other programs (process) regardless of their locations.
The client is a process (program) that sends a message to a server process requesting the server to perform a task (service). Client programs usually manage the

user interface portion of the application, validate data entered by the user and dispatch requests to server programs.

A server process (program) fulfills the client request by performing the task requested. Server programs generally perform database retrieval and updates, manage data integrity and dispatch responses to client requests.

31. Differentiate distributed and cooperative processing?

The distributed processing means distribution of applications and business logic across multiple processing platforms. Distributed processing implies that processing will occur on more than one processor in order for a transaction to be completed.

Cooperative processing is computing that requires two or more distinct processors to complete a single transaction. It is a form of distributed computing in which two or more distinct processes are required to complete a single business transaction.

Cooperative processing can also be considered to be a distributed processing, if communication between processors is performed through a message-passing architecture.

32. What do you mean by distributed object computing?

Distributed object computing (DOC) utilizes reusable software components that can roam anywhere on networks, run on different platforms, communicate with legacy applications by means of object wrappers, and manage themselves and the resources thy control. It introduces a higher level of abstraction and promises the most flexible client-server system.

33. Write a short note on CORBA?

Common object request broker architecture (CORBA) is a standard proposed as a means to integrate distributed heterogeneous business applications and data. CORBA object request brokers (ORBs) implement a communication channel through which applications can access object interfaces and request data and services. The CORBA common object environment (COE) provides system level services such as life cycle management for objects accessed through CORBA, event notification between objects, and transaction and concurrency control.

34. What are the necessary characteristics that a system must satisfy to be considered as an object-oriented system?

The rules that make a system an object-oriented system are:

◊ The system must support complex objects.
◊ Object identity must be supported.
◊ Objects must be encapsulated.
◊ The system must support types or classes.
◊ The system must support inheritance.
◊ The system must avoid premature binding.

◊   The system must be computationally complete.
◊   The system must be extensible.

35. Differentiate object-oriented databases and traditional databases?
◊   The objects are an active component in an object-oriented database, in contrast to conventional database systems, where records play a passive role.
◊   The relational database systems do not explicitly provide inheritance of attributes and methods while object-oriented databases represent relationships explicitly.
◊   Object oriented databases also differ from the traditional relational databases in that they allow representation and storage of data in the form of objects.

36. Describe reverse and forward engineering?
◊   Creating an object model from an existing relational database layout (schema) often is referred to as reverse engineering.
◊   Creating a relational schema from an existing object model often is referred to as forward engineering.

37. Define object-relation mapping?
   In a relational database, the schema is made up of tables, consisting of rows and columns. In an object model, the counterpart to a table is a class which has a set of attributes (properties or data members) and methods (behaviors). The object-relation mapping is the mappings between a table and a class that is the mappings between columns and attributes, between a row and an object, and between a stored procedure and a method.

38. What are the different mapping capabilities to be defined?
   The mapping capabilities that must be defined are:

◊   Table-class mapping.
◊   Table-multiple classes mapping.
◊   Table-inherited classes mapping.
◊   Tables-inherited classes mapping.

39. Define referential integrity?
   Referential integrity means making sure that a dependent table's foreign key contains a value that refers to an existing valid tuple in another relation.

40. What do you mean by federated multidatabase systems?
   Federated multidatabase systems are the heterogeneous information systems which facilitated the integration of heterogeneous information sources, where they

can be structured, semi-structured and sometimes even unstructured. Some heterogeneous information systems are constructed on a global schema over several databases.

41. Define MDBS?

A multi database system (MDBS) is database systems that resides unobtrusively on top of existing relational and object databases, and file systems and presents a single database illusion to its users. An MDBS maintains a single global database schema. The local database systems actually maintain all user data. MDBS actually controls multiple gateways (or drivers). This way user can have the benefits of a database with a schema to access data stored in different databases and cross-database functionality.

42. Define neutralization (homogenization).

Neutralization also called homogenization is the process of consolidating the local schemata. The global schema is constructed by consolidating (integrating) the schemata of the local databases where the schematic differences among them are handled by neutralization.

43. What do you mean by ODBC?

Open database connectivity (ODBC) is an application programming interface that provides solutions to the multidatabase programming problem. ODBC provides a vendor-neutral mechanism for independently accessing multiple database hosts. The application interacts with the ODBC driver manager, which sends the application calls to the database. The driver manager loads and unloads drivers, performs status checks, and manages multiple connections between applications and data sources.

44. What are the activities involved in access layer design process?

The process of creating an access class for the business classes is as follows:

◊ For every business class identified, mirror the business class package.
◊ Define relationships.
◊ Simplify classes and relationships.
◊ Iterate and refine.

45. Write short note on creative process?

The creative process is a combination of the following:
◊ A curious and imaginative mind.
◊ A broad background and fundamental knowledge of existing tools and methods.
◊ An enthusiastic desire to do a complete an thorough job of discovering solutions once a problem has been defined.
◊ Being able to deal with uncertainty and ambiguity and to defer premature closure.

46. What are the steps in view layer macro process?
    The view layer macro process consists of two steps:

    ◊ For every class identified, determine if the class interacts with a human actor. If so, perform the following; otherwise, move to the next class.
        o Identify the view (interface) objects for the class.
        o Define the relationships among the view (interface) objects.
    ◊ Iterate and refine

47. Give the three UI design rules.
    ◊ UI design rule 1: Making the interface simple.
    ◊ UI design rule 2: Making the interface transparent and natural.
    ◊ UI design rule 3: Allowing users to be in control of the software.

48. What are the windows in user interface used for?
    Window commonly are used for the following purposes:
    ◊ Forms and data entry windows:
        Data entry windows provide access to data that users can retrieve, display and change in the application.
    ◊ Dialog boxes:
        Dialog boxes display status information or ask users to supply information or make a decision before continuing with a task.
    ◊ Application windows:
        An application window is a container of application objects or icons. It contains an entire application with which users can interact.

49. What are the three general steps in creating a user interface object?
    Creating a user interface generally consists of three steps.
    ◊ Create the user interface objects (such as buttons, data entry fields).
    ◊ Link or assign the appropriate behaviors or actions to these user interface objects and their events.
    ◊ Test, debug, then add more by going back to step 1.
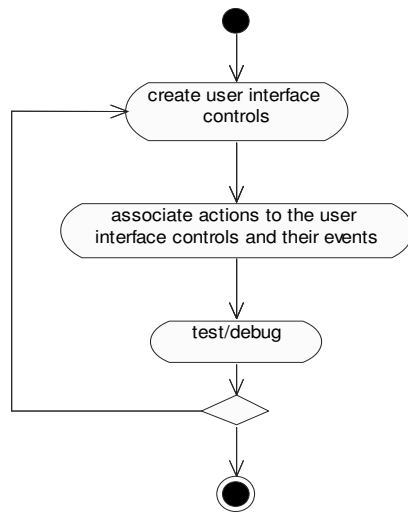
50. What is a Metaphor?

It is an analogy that relates two unrelated things by using one to denote the other.

# UNIT-V

1. What is the purpose of debugging?

Debugging is the process of finding out where something went wrong in the application, what we develop and correcting the code to eliminate the errors or bugs that cause unexpected results.

2. What are the types of errors that you could find in your program?
   ◊ Language (syntax) errors
   ◊ Run-time errors
   ◊ Logic errors

These are the various types of errors that would occur in the program that we develop.

3. Discuss Error-based testing?

This technique search a given class's method for particular clues of interests, then describe how these clues should be tested.

4. Discuss Scenario-based testing/usage-based testing?

It concentrates on what the user does, not what the product does. This means capturing use cases and the tasks users perform, then performing them and their variants as tests. They often are more complex and realistic than error-based tests. Scenario-based tests tend to exercise multiple subsystems in a single test, because that is what users do.

5. Name some testing strategies?

&#9674;  Black- Box Testing
&#9674;  White- Box Testing
     o  Path Testing
        &#9642;  Statement testing coverage
        &#9642;  Branch testing coverage
&#9674;  Top-Down Testing
&#9674;  Bottom-Up Testing

**6. What is the Impact of Object orientation on Testing?**
&#10149; Some types of errors could become less reasonable (not worth testing for)
&#10149; Some types of errors could become more reasonable (worth testing for)
&#10149; Some new types of errors might appear.

**7. Discuss Black-Box testing?**

In a Black box testing, the test item is treated as "black," since its logic is unknown; all that is known is what goes in and what comes out or input and output. It may be used for Scenario- based testing.

**8. Discuss White- Box testing?**

It assumes that the specific logic is important and must be tested to guarantee the system's proper functioning. It is used mainly in the error based testing.

**9. What do you mean by Top- down Testing?**

It assumes that the main logic or object interactions and systems messages of the application need more testing than an individual object's methods or supporting logic. It can detect the serious design flaws early in the implementation.

**10. Discuss about the Statement testing coverage and Branch testing coverage?**

The main idea of **statement testing** coverage is to test every statement in the object's method by executing it at least once.

The main idea behind **branch testing** coverage is to perform enough tests to ensure that every branch alternative has been executed at least once under some test.

**11. What is Path testing?**

Path testing is one form of white box testing. It makes that each path in a object's method is executed at least once during testing. Two types of Path testing are;
&#9674;  Statement testing coverage
&#9674;  Branch testing coverage

**12. What is Bottom - Up Testing?**

It starts with the details of the system and proceeds to higher levels by a progressive aggregation of details until they collectively fit the requirements for the system. This approach is more appropriate for testing the individual objects in a system.

**13. What is the objective of testing?**

◊ Testing is the process of executing a program with the intent of finding errors.
◊ A good test case is the one that has a high probability of detecting an as-yet undiscovered error.
◊ A successful test case is the one that detects an as-yet undiscovered error.

14. What is the necessary of a test plan?

A test plan is developed to detect and identify potential problems before delivering the software to its users. A test plan offers a road map for testing activities, whether usability, user satisfaction, or quality assurance tests. It should state the test objectives and how to meet them.

15. List the steps needed for a test plan?
◊ Objectives of the test
◊ Development of a test case
◊ Test analysis

16. Define regression testing?

All passed tests should be repeated with the revised program, called regression testing, which can discover errors introduced during the debugging process. When sufficient testing is believed to have been conducted, this fact should be reported and testing for this specific product is complete.

17. Define Beta testing and Alpha testing?

Beta testing, a popular, inexpensive and effective way to test software on a select group of the actual users of the system.

Alpha testing is done by in-house testers, such as programmers, software engineers, and internal users.

18. What is the purpose of configuration control system?

It provides a way of tracking the changes to the code. At a minimum, every time the code changes, a record should be kept that tracks which module has been changed, who changed it, and when it was altered, with a comment about why the change was made.

19. When is testing said to be successful?

Testing becomes successful when the steps below are followed;
◊ Understand and communicate the business case for improved testing.
◊ Develop an internal infrastructure to support continuous testing.
◊ Look for leaders who will commit to and own the process.
◊ Measure and document your findings in a defect recording system.
◊ Publicize improvements as they are made and let people know what they are doing better.

20. Define Usability?

ISO defines Usability as the effectiveness, efficiency, and satisfaction with which a specified set of users can achieve a specified set of tasks in particular environments. It requires,

◊ Defining tasks.
◊ Defining users
◊ A means for measuring effectiveness, efficiency, and satisfaction.

21. What are the issues in software quality?

◊ Validation – user satisfaction
◊ Verification – Quality assurance

22. What is Usability testing?

It measures the ease of use as well as the degree of comfort and satisfaction users have with the software.

23. What are the guidelines for developing usability testing?

◊ The usability testing should include all of software's components.
◊ Usability testing need not be very expensive, such as including trained specialists working in a soundproof lab with sophisticated recording equipment.
◊ All tests need not involve many subjects.
◊ Consider the user's experience as part of your software usability.
◊ Apply usability testing early and often.

24. Explain user satisfaction testing?

It is the process of quantifying the usability test with some measurable attributes of the test, such as functionality, cost, or ease of use.

25. Explain COTS and USTS?

**Commercial off –the –shelf (COTS)** software tools are already written and a few are available for analyzing and conducting user satisfaction tests.

**User satisfaction test spreadsheet (USTS)** automates many bookkeeping tasks and can assist in analyzing the user satisfaction test results.

26. Write about the user satisfaction cycle?

◊ Create a user satisfaction test for your own project
◊ Conduct test regularly and frequently
◊ Read the comments very carefully, especially if they express a strong feeling.
◊ Use the information from user satisfaction test, usability test, reactions to prototypes, interviews recorded, and other comments to improve the product.

Important benefit of user satisfaction testing is you can continue using it even after the product is delivered.

# I6 Marks Questions and Answers

1. Describe the various Object oriented concepts?
- Object – combination of data and logic that represents a real world entity.
- In an OOS everything is an object.
- Instance of a class.
- Class – used to distinguish objects from others.
- Object properties – represent the state of an Object.
- Behavior – the collection of methods that abstractly the object is capable of doing.
- Messages – Non-specific function calls.
- Encapsulation or Information Hiding – principle of concealing the internal data and procedures of an object.
- Class Hierarchy – organizing classes into sub class – super class hierarchy.
- Inheritance – property that allows the objects to be built from other objects.
  - Dynamic Inheritance
  - Multiple Inheritance
- Polymorphism – the same operation may behave differently on different classes.
- Association – relationship between objects and classes.
- Aggregation - collection of objects or an attribute which can be an object itself.

2. Describe the Software Development process.
- Set of activities that transforms user's needs into a software solution that satisfies the needs.
- A process can be divided into a number of sub-processes.
- Three types of transformation
  - Transformation 1 – Analysis
  - Transformation 2 – Design
  - Transformation 3 – Implementation
- Waterfall Approach
  - What
  - How
  - Do it
  - Test
  - Use
- Advantages and disadvantages of Waterfall Approach

3. How can we build a high quality Software?
- High quality products must meet user's needs and satisfaction.
- Should have minimal or no defects.
- Ultimate goal – user satisfaction.
- Four quality measures:
  - Correspondence
  - Validation
  - Correctness
  - Verification
- Verification – Am I building the product right?
- Validation – Am I building the right product?

4. Describe the Object oriented Systems Development?
- Object Oriented Analysis – Use case driven
  - Collaboration
  - Role
  - Use – case Modeling
- Object oriented design
  - Goal – design the classes
  - Design and refine classes, attributes, methods, structures and associations.
  - Guidelines of OOD.
- Prototyping
  - Horizontal
  - Vertical
  - Analysis
  - Domain
- Component based development
  - Component Wrapper
  - Software Components
  - Rapid Application Development
- Incremental Testing
  - Testing should be performed in every phase to find errors.
- Reusability

5. Describe Rumbaugh's Object Modeling Technique?
- Describes the dynamic behavior of objects in a system.
- Four phases.
  - Analysis – results are objects, dynamic and functional models.
  - System design – gives a structure of the basic architecture.
  - Object design – produces a design document.
  - Implementation – produces reusable code.
- Three different parts
  - Object Model – presented by object model and the data dictionary.
  - Dynamic model - presented by the state diagrams and event flow diagrams.
  - Functional Model – presented by data flow and constraints.

6. Give detailed notes about the Booch Methodology?
- Helps to design your system using the object paradigm.
- Is criticized for his large set of symbols.
- Consists of the following diagrams:
  - Class diagrams.
  - Object diagrams.
  - State transition diagrams.
  - Module diagrams.
  - Process diagrams.
  - Interaction diagrams.
- Two processes:
  - Macro development process
  - Micro development process.
- Macro development process.

- Primary concern – technical management of the system.
- Steps involved:
  - Conceptualization.
  - Analysis and development of the model.
  - Design or create the system architecture.
  - Evolution or implementation.
  - Maintenance.
- Micro development process.
  - Describes the day-to-day activities.
  - Steps involved:
    - Identify classes and objects.
    - Identify classes and object semantics.
    - Identify classes and object relationships.
    - Identify classes and object interfaces and implementation.

7. Give a detailed account of Jacobson methodology?
- Covers the entire life cycle and stress traceability between the various phases.
- Use cases
  - Scenarios for understanding system requirements.
  - Non formal text with no clear flow of events.
  - Text easy to read.
  - Formal style using pseudo code.
  - Can be viewed as concrete or abstract (not initiated by actors).
- Object oriented software Engineering: Objectory
  - Use case model.
  - Domain Object Model.
  - Analysis Object Model.
  - Implementation model.
  - Test model.
- Object oriented business Engineering
  - Analysis phase.
  - Design and Implementation phase.
  - Testing phase.

8. Describe patterns and the various pattern templates?
- Pattern – identifies a common structure that makes it useful for design.
- Involves a general description of a solution to a recurring problem.
- Properties of a good pattern:
  - It solves a problem.
  - It is a proven concept.
  - Solution is not obvious.
  - Describes a relationship.
  - Has significant human component.
- Generative patterns – tells us how to generate something.
- Non generative patterns – static and passive.
- Patterns templates.
  - Name.
  - Problem.
  - Context.
  - Forces.

- Solution.
- Examples.
- Resulting context.
- Rationale.
- Related patterns.
- Known uses.

9. Explain in detail about the Unified approach?
- Establishes a unifying and unitary framework by utilizing UML.
- The processes are:
  - Use case driven development.
  - Object oriented analysis.
  - Object oriented design.
  - Incremental development and prototyping
  - Continuous testing.
- Methods and technologies employed include:
  - UML
  - Layered approach.
  - Repository.
  - CBD.
- Object oriented Analysis.
  - Identify actors.
  - Develop a simple process model.
  - Develop the use case.
  - Develop interaction diagrams.
  - Identify classes.
- Object oriented design
  - Design classes, attributes, methods etc.
  - Design the access layer.
  - Design the prototype user interface.
  - User satisfaction and usability tests.
  - Iterate and refine the design.
- Continuous testing
- UML – modeling language
- Repository
  - Allows maximum reuse of previous experience.
  - Should be accessible by many people.
- Layered approach
  - The business layer.
    - Displaying results.
    - Data access details
  - The user interface or view layer.
    - Responding to user interaction.
    - Displaying business objects.
  - The access layer.
    - Translate request.
    - Translate results.

10. Describe the UML Class diagram?
- Also referred to as Object modeling.

- Collection of static modeling elements.
- Class notation – rectangle with three compartments.
- Object diagram – instance of a class diagram.
- Class interface notation – small circle with the interface name connected to the class.
- Binary association – solid path connecting two classes.
- Association role – end of an association.
- Qualifier – small rectangle attached to the end of an association.
- Multiplicity – specifies the range of allowable associated classes.
- OR association – dashed line connecting two or more associations.
- Association class – association that has class properties.
- N-ary association – association among more than two classes.
- Aggregation – hollow diamond attached to the end of the path.
- Generalization – relationship between a general class and a more specific class.

11. Demonstrate the guidelines for finding use cases and developing effective documentation?
- Guidelines for use cases.
  - For each user, find the tasks and functions.
  - Name the use cases.
  - Describe the use cases briefly.
  - Isolate users from actors.
  - Isolate actors from other actors.
  - Isolate use cases.
- Guidelines for effective documentation.
  - Common cover.
  - 80 – 20 rule.
  - Familiar vocabulary.
  - Make the document as short as possible.
  - Organize the document.

12. Give detailed notes about the Noun phrase approach?
- Look for the noun phrases through the use cases.
- Three categories:
  - Relevant classes.
  - Fuzzy classes.
  - Irrelevant classes.
- Identifying tentative classes.
  - Look for noun phrases and nouns in the use cases.
  - Some classes are implicit or taken from general knowledge.
  - All classes must make sense in the application domain.
  - Carefully choose and define class names.
- Selecting classes from the relevant and fuzzy classes.
  - Redundant classes.
  - Adjective classes.
  - Attribute classes.
  - Irrelevant classes.

13. Describe the CRC approach and naming classes?
- CRC approach.
  - Technique used for identifying classes responsibilities.

- CRC cards are 4" x 6" index cards.
- Cheap, portable and readily available.
- Stress the importance of creating objects.
- CRC process.
    - Identify classes' responsibilities.
    - Assign responsibilities.
    - Identify collaborators.
- Naming classes.
    - It should be singular.
    - Use comfortable names.
    - Should reflect its intrinsic nature.
    - Use readable names.

14. Give a detailed note about Associations?
- Represents a physical or conceptual connection between two or more objects.
- Identifying associations.
    - Is the class capable of fulfilling the required task by itself?
    - If not what does it need.
    - From what other class can it acquire what it needs?
- Guidelines for identifying associations.
    - A dependency between two or more classes.
    - A reference from one class to another.
- Common association patterns.
    - Location association: next to, part of, contained in.
    - Communication association: talk to, order to.
- Eliminate unnecessary associations.
    - Implementation association.
    - Ternary associations.
    - Directed actions or derived associations.

15. Give a detailed note on Super-sub class relationship and a-part-of relationship?
- Super Sub class relationship.
    - Also known as generalization hierarchy.
    - Allows objects to be built from other objects.
- Guidelines for identifying Super-sub relationship.
    - Top – down.
    - Bottom – up.
    - Reusability.
    - Multiple inheritances.
- A-part of relationship.
    - Also known as aggregation.
    - Represents a situation where a class contains several component classes.
- Properties.
    - Transitivity: A->B, B->C => A->C.
    - Antisymmetry: A is a part of B, and then B is not a Part of A.
- Patterns.
    - Assembly.
    - Container.
    - Collection-member.

16. Describe in detail about Object oriented database management systems?
- OODBMS – combination of OOPs and database technology.
- Rules that make an object oriented system are:
  - System must support complex objects.
  - Object identity must be supported
  - Object must be encapsulated.
  - Must support types or classes.
  - Must support inheritance.
  - Must avoid premature binding.
  - Must be computationally complete.
  - Must be extensible.

- Rules that make it a DBMS are:
  - Must be persistent.
  - Should be able to manage very large databases.
  - Must accept concurrent users.
  - Must be able to recover from H/w and S/w problems.
  - Data query must be simple.

17. State the differences between OODBMS and traditional database. Describe object – relational systems?
- OODBMS
  - Active objects.
  - Explicit relationships.
  - Has object identity and is persistent.
- Traditional database.
  - Passive records.
  - Implicit representations.
  - Does not have object identity and is not persistent.
- Object – relational systems
  - Reverse engineering.
  - Forward engineering.
  - Object relation mapping.
    - Table class mapping.
    - Table multiple class mapping.
    - Table inherited classes mapping.
    - Tables inherited classes mapping.
    - Keys for instance navigation.

18. Explain the steps involved in designing the access layer classes?
- Main idea – create a set of classes that know how to communicate with the places where data reside.
- Provides a link between the business or view objects and the data storage.
- Access layer performs two tasks.
  - Translate the request
  - Translate the results.
- Benefits.
  - Provides easy migration of developing technologies.
  - Able to address the modest needs of the two tier architecture.
- The process.

- Access class – interacts with a non-human actor.
- The steps involved are:
  - ° Mirror the business class package.
  - ° Define relationships.
  - ° Simplify classes and relationships.
  - ° Iterate and refine.
- The steps involved in designing with persistent attributes are:
  - ° Determine if the class has persistent data.
  - ° Mirror the business class package.
  - ° Define relationships.
  - ° Simplify classes and relationships.
  - ° Iterate and refine.

19. Explain the steps involved in designing the view layer classes?
- Responsible for two major aspects – i/p, o/p.
- Four major activities.
  - Macro level UI design process.
  - Micro level design process.
  - Testing usability and user satisfaction.
  - Refine and iterate.

20. Describe the purpose of view layer interface?
- Forms and data entry windows
  - Guidelines.
- Dialog boxes
  - Guidelines.
- Application or main windows
  - Guidelines.

21.Describe quality assurance test and testing strategies?
- Debugging
- Types of errors
  - Language or syntax.
  - Run time.
  - Logic.
- Types of QA
  - Error based
  - Scenario based
- Strategies.
  - Black box
  - White box
  - Top down
  - Bottom up.

22. Describe test cases and the impacts of object orientation on testing?
- Construct some test i/p cases and mention the expected output.
- Compare the output with the expected outcome
- Objectives
  - Testing

- ▪ Good test case
  - ▪ Successful test case.
- • Guidelines
- • Impact of object orientation
  - ▪ Impact of inheritance on testing
  - ▪ Reusability of tests.

23. Illustrate test plan and continuous testing?
- • Test plan – developed to detect and identify potential problems before delivering the S/w.
- • Steps involved.
  - ▪ Objectives.
  - ▪ Development.
  - ▪ Test analysis.
- • Regulation testing – discover errors during debugging.
- • Alpha testing.
- • Beta testing.
- • Guidelines.
- • Continuous testing.

24. Describe Usability Testing?
- • Measures the ease of use as well as degree of comfort and satisfaction.
- • Guidelines.
- • Recording the usability test.

25. Describe user satisfaction?
- • Is the process of quantifying the usability test with some measurable attributes.
- • Objectives.
- • Guidelines.