CS6502 OBJECT ORIENTED ANALYSIS AND DESIGN

UNIT-1

PART-A

1. **What is Analysis?**

   Analysis emphasizes an investigation of the problem and requirements, rather than a solution. Analysis is a broad term best qualified as in requirements analysis or object analysis.

2. **What is Design?**

   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 design can be implemented.

3. **Define Object-Oriented Analysis and Design**

   OOA-there is an emphasis on finding and describing the objects or concepts in the problem domain.

   OOD-there is an emphasis on defining software objects and how they collaborate to fulfill the requirements.

4. **Define Use Case.**

   Requirements analysis may include a description of related domain processes these can be written as use cases. Use cases are not an object oriented artifact-they are simply written stories. However, they are a popular tool in requirements analysis and are an important part of the unified process.

5. **Define a DomainModel**

   Domain model is a model in a set of diagrams that show domain concepts or objects.

6. **Define Interaction Diagrams**

   Object oriented design is concerned with defining software objects and their collaborations. A common notation to illustrate these collaboration is the interaction diagram. It shows the flow of messages between software objects and thus the invocation of methods.

7. **Define Design class Diagrams**

   It will show the attributes and methods of the classes.

8. **Define UML**

   The Unified Modeling Language 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.

9. **Define Software development process**

   It describes an approach to building, deploying and possibly maintaining software.

10. **What is the use of Unified Process ?**

    The UP has emerged as a popular software development process for building object-oriented systems.

11. **Define Iterative and Incremental Development.**

    The system grows incrementally over time iteration and thus this approach is also know as iterative and incremental development.

12. **Benefits of Iterative Development.**

    - Early visible progress
    - Managed complexity the team is not overwhelmed by analysis paralysis or very long and complex steps
    - The learning within an iteration can be methodically used to improve the development process itself, Iteration by iteration.

13. **Four major phses of UP**

    - Inception
    - Elaboration
    - Construction
    - Transition

14. **Define business Modeling**

    When developing a single application,this includes domain object modeling. When engaged in large scale business analysis or business process reengineering,this include dynamic modeling of the business process across the entire enterprise.

15. **Define Development Case**

The choice of UP artifacts for a project may be written up in a short document called the Development Case.

16. Define Heavy process

A heavy process is a pejorative term meant to suggest one with the following qualities

- Many artifacts created in a bureaucratic atmosphere
- Rigidity and control
- Elaborate long term detailed planning
- Predictive rather than adaptive

17. Define Extensions

Extensions are very important. They indicate all the other scenarios or branches both success and failure.

18. What are the key requirements artifacts?

The use case model

The supplementary Specification

The glossary

The vision

The Business Rules

19. Define Elaboration

Elaboration is the intial series of iterations during which the team does serious investigation,implements the core architecture clarifies most requriments,and tackles the high –risk issues.

20. Define quality attributes.

These include usability,reliability and so forth. There are two types observable at execution and not observable at execution.

PART-B

1. Explain in detail about object oriented emphasizes representation of objects.
2. Explain in detail about the next generation POS system.
3. Draw use case diagram for library system by including the user and librarian in the use case diagram and related use cases by using the relations.
4. Explain the use case diagram with example
5. Explain the UP phases and schedule oriented terms in detail.
6. Explain in details about the different types of requirements.
7. What are the reminder questions to find actor and goals in use case modeling.
8. Explain in details about the requirements in context and low-level feature lists.
9. Explain the next generation POS with example.
10. Explain in detail about the UP artifact and process context.

UNIT 2

PART -A

1. **What is aggregation in UML?**

   Aggregation is a special type of 'has-a' relationship between classes where one of the two participating classes is part of other one. I.e. it's a 'whole-part' relationship. The class acting as 'whole' always has multiplicity of one.In UML representation aggregations are represented by an association that shows a rhomb on the side of the whole.
   For example A car has Wheel.
2. **What is composition?**

   Composition is a special type aggregation where the 'has-a' relationship is more strong. I.e. the part entity cannot exist without the whole entity. For example an university has departments which cannot exist on their own with the containing 'university' entity
3. **Define the characteristics of aggregation:**

   It does not imply ownership.It is an asymmetric relationship. It is a transitive relationship. It implies stronger coupling behavior (copy, delete, etc.).
4. **Differentiate Aggregation and containment?**

   Aggregation is the relationship between the whole and a part. We can add/subtract some properties in the part (slave) side. It won't affect the whole part.

Best example is Car, which contains the wheels and some extra parts. Even though the parts are not there we can call it as car.
But, in the case of containment the whole part is affected when the part within that got affected. The human body is an apt example for this relationship. When the whole body dies the parts (heart etc) are died

5. **Define Domain Model**

A domain model is a representation of real-world conceptual classes, not of software components. It is not a set of diagrams describing software classes or software objects with responsibilities

6. **What is meant by Activity diagram?**

An activity specifies the coordination of executions of subordinate behaviors, using a control and data flow model.

7. **What is activity parameter?**

Activity parameter nodes are object nodes at the beginning and end of flows that provide a means to accept inputs to an activity and provide outputs from the activity, through the activity parameter

8. **Define Swimlane.**

Swimlane is used for partitioning the children in an activity diagram

9. **Define Class**

A class is a description of objects that share the same attributes, operations, methods, relationships and semantics

A class is a template for creating objects.

A class models the data and behavior an object will have.

Classes can be domain (or analysis) classes, design classes, or implementation classes Definition of the class must include attributes, operations and constraints

10. **What is meant by object?**

An object is a specific instance of a class.

11. **What is meant by Attributes**

A data variable with object scope.

Examples: book attributes: title, author, publisher, ISBN

The value of an object's attributes define its state.

Attributes should not be accessible to entities outside the object.

12. Define Behaviors

Method: a function with object scope.

Methods can operate on that object's attributes.

Defines the objects behaviors - how it does what it does.

Methods define the objects responsibilities.

13. Define Conceptual Classes

• Symbol — words or images representing the concept

• Intention — definition of the concept

• Extension — the set of examples to which the concept applies

14. **How to Create a Domain Model — Three Steps**

1. Find the domain concepts
2. Draw them in a UML class diagram
3. Add associations and attribute.

15. **UML designs consist of**

Class and object diagram

Use cases

Interaction diagram

16. **UML class and object diagrams may have:**

Objects (of certain class), with

attributes

operations

Links between Objects,

Aggregations between Objects,

17. **What is meant by Operations?**

Operations are functions that may be applied to objects.Each operation hastarget object as implicit argument. Behaviour of object depends on its class (remember each object "knows" its class). Operations take parameters of certain type, and return result of certain type.

18. **Associations vs. Attributes**

Associations exist between objects;; attributes have simple type (not objects)..

Therefore

• Being mother off a person is represented through

• Students having an age is represented through

19**. Define composition**

Composition is a type of aggregation which links the

Lifetimes of the aggregate and its parts.

21. Define generalization

Generalization = relationship between a class and its more refined versions

PART B

1. Explain basic notation that have used in Activity diagram?
2. Explain the Domain Model and define all the classes involved in the model?
3. UML activity diagram for the Enroll in University use case.
4. Generate a Use Cases diagram for Example ATM System and explain?
5. Explain about Association and different types about associations
6. How to find conceptual classes?
7. Explain in detail about domain model refinement?
8. Discuss the concept of aggregation and composition?
9. Explain the concept of description classes?
10. Explain in detail about finding conceptual class hierarchy?

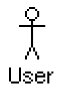
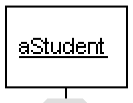
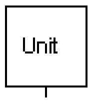
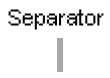
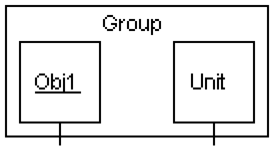
UNIT –III

PART A

1. What is mean by SSD?

   The UML system sequence diagram (SSD) illustrates events sequentially input from an external source to the system. The SSD will define the system events and operations. System sequence diagrams are a timeline drawing of an expanded use case. Events are related by time with the top events occurring first. System events are the important items. These are events that cause a system response.
2. Explain about Sequence Diagram Header Elements.

The header portion of the sequence diagram represents the components or objects of the system being modeled and are laid out horizontally at the top of the diagram. See an example sequence diagram here.

Actor	Represents an external person or entity that interacts with the system	
Object	Represents an object in the system or one of its components	
Unit	Represents a subsystem, component, unit, or other logical entity in the system (may or may not be implemented by objects)	
Separator	Represents an interface or boundary between subsystems, components or units (e.g., air interface, Internet, network)	
Group	Groups related header elements into subsystems or components	

3. What can be modeled using sequence diagrams?

   Sequence diagrams are particularly useful for modelling:

   - Complex interactions between components
   - Use case elaboration
   - Distributed & web-based systems
   - Complex logic
   - State machines

4. List the Benefits of using UML sequence diagram.

   - Help you discover architectural, interface and logic problems early
   - Sequence diagrams are valuable collaboration tools during design meetings because they allow you to discuss the design in concrete terms.
   - Sequence diagrams can be used to document the dynamic view of the system design at various levels of abstraction.

5. What is a UML sequence diagram?

   UML sequence diagrams are used to represent or model the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

6. What are the limitations of sequence diagram?

   When the set of communicating objects is large or highly variable, when there are many branch points (e.g. exceptions), when there are complex iterations, or synchronization issues such as resource contention. In such cases, sequence diagrams cannot completely describe the system's behavior, but they can specify typical use cases for the system, small details in its behavior, and simplified overviews of its behavior.

7. What is mean by UML package diagram?

   A package diagram in the Unified Modeling Language depicts the dependencies between the packages that make up a model.

8. List types of dependencies defined between packages.

   There are two special types of dependencies defined between packages:

   - package import
   - package merge

9. Explain package import.

   A package import is "a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace." By default, an unlabeled dependency between two packages is interpreted as a package import relationship.

10. Explain package import.

    A package merge is "a directed relationship between two packages that indicates that the contents of the two packages are to be combined. It is very similar to Generalization in the sense that the source element conceptually adds the characteristics of the target element to its own characteristics resulting in an element that combines the characteristics of both.

11. What is the use of package diagram?

Package diagrams can use packages containing use cases to illustrate the functionality of a software system. Package diagrams can use packages that represent the different layers of a software system to illustrate the layered architecture of a software system. The dependencies between these packages can be adorned with labels / stereotypes to indicate the communication mechanism between the layers.

12. What is class diagram?

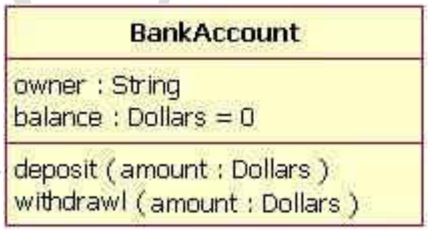
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

13. List the uses of class diagram.

The class diagram is the main building block in object oriented modeling. They are being used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed.

14. Give the UML notation of class diagram.

In the class diagram these classes are represented with boxes which contain three parts:



A class with three sections.

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

15. Explain about Visibility of a class.

To specify the visibility of a class member (i.e., any attribute or method) there are the following notations that must be placed before the member's name:

+ public	visible to all elements that can access the contents of the namespace that owns it.
# protected	visible to elements that have a generalization relationship to the namespace that owns it.
– private	only visible inside the namespace that owns it.
~ package	owned by a namespace that is not a package, and is visible to elements that are in the same package as its owning namespace

16. What is an instance member?

In the case of instance members, the scope is a specific instance. For attributes, it means that its value can vary between instances. For methods, it means that its invocation affects the instance state, in other words, affects the instance attributes.

17. What is classifier class?

In the classifier member, the scope is the class. For attributes, it means that its value is equal for all instances. For methods, it means that its invocation does not affect the instance state. Classifier members are commonly recognized as "static" in many programming languages. To indicate that a member has the classifier scope, its name must be underlined.

18. What is mean by relationship found on class? Give its type.

A relationship is a general term covering the specific types of logical connections found on class and objects diagrams. UML shows the following relationships:

- Class level relation.
- Instance level relation.

19. What is an association?

An Association represents a family of links. Binary associations (with two ends) are normally represented as a line, with each end connected to a class box. Higher order associations can be drawn with more than two ends. In such cases, the ends are connected to a central diamond.

20. What is an Aggregation?

    Aggregation is a variant of the "has a" or association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As a type of association, an aggregation can be named and have the same adornments that an association can. However, an aggregation may not involve more than two classes.

PART B

1. Explain in detail about SSD in detail with an example.
2. Give the relation between sequence diagram and use cases.
3. Explain in detail about UML interaction diagrams.
4. Explain in detail about UML class diagram.
5. What is mean by logical refinement?
6. Give the relation between logical architecture and UML package diagram.
7. Explain in detail collaboration diagram.
8. Explain in detail about sequence diagram.
9. Explain in detail about the various types of relations in a class diagram.
10. Explain in detail about package diagram.

UNIT-4

PART-A

1. GRASP methodical approach to learning basic object design?

   GRASP patterns are a learning aid to help one understand essential object design,and apply design reasoning in amethodical,rational,explainable way.

2. Define GRASP patterns?

   GRASP patterns are really more accurately described as best practices

   - GRASP patterns outline best practices which can be employed in any object-oriented design
   - These best practices, if used properly, will lead to maintainable, reusable, understandable, and easy to develop software.

3. Define GRASP responsibilities?

   Responsibilities is “a contract or obligation of a classifier” (UML definition)

a. Responsibilities can include behaviour, data storage, object creation and more
b. They often fall into two categories:
   Doing
   Knowing

4. Define cohesion?
   cohesion – the degree to which the information and responsibilities of a class are related to each other
5. Define coupling?
   Coupling of classes is a measure of how strongly a class is connected to another class
6. What is creator?
   The Creator GRASP ensures that coupling due to object instantiation only occurs on closely related classes .An aggregate or container of a class is already coupled with that class
7. Define low coupling?
   Assign a responsibility so that coupling remains low
   - This is pretty vague, but it means that we try to keep low the number of classes to which a class is coupled.
   - Creator is a more specific case of Low Coupling, related to instantiation.
8. Define high cohesion?
   Assign a responsibility so that cohesion remains high
   - Again, this is vague, but it simply means that we should always try to maintain class cohesion.
   - This may prompt you to create a new class if other responsibilities exist that are similar/related to this one
9. What is creator and its responsibilities?
   Assign the responsibility for receiving and handling a system event message to a class that is either:
   - Representative of the entire subsystem (e.g. Façade Controller)
   - Representative of the entire use case scenario

10. What is facade controller?

    The GRASP mentioned that a Controller that represents an entire subsystem might be called a Façade Controller.

11. What is polymorphism?

    When related behaviours vary by type (class), assign the responsibility polymorphically to the specialization classes.This is basically the purpose of polymorphism, so it is natural for software developers to understand.This is not much of a pattern, and yet another best practice.

12. What about pure fabrication?

    To support high cohesion and low coupling, where no appropriate class is present invent one Even if the class does not represent a problem domain concept .This is a compromise that often has to be made to preserve cohesion and low coupling.

13. Define indirection?

    To avoid direct coupling between objects, assign an intermediate object as a mediator

    1. Recall that coupling between two classes of different subsystems can introduce maintenance problems
    2. Another possibility is that two classes would be otherwise reusable (in other contexts) except that one has to know of the other

14. What is protected variations?

    Assign responsibility to create a stable interface around an unstable or predictably variable subsystem or component

    - If a component changes frequently, the users of the component will also have to be modified
    - This is especially time consuming if the component has many users

15. Define Larman?

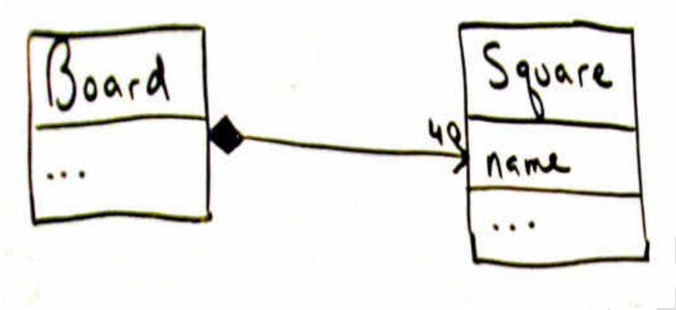
    Larman: "In OO design, a pattern is a named description of a problem and solution that can be applied in new contexts; ideally, a pattern advises us on how to apply the solution in varying circumstances and considers the forces and trade-offs."

16. What are the five GRASP patterns?

    - Creator

- Information Expert
- Low Coupling
- Controller
- High Cohesion

17. How does Create pattern develop this Design Class Diagram (DCD)?



Board has a composite aggregation relationship with Square I.e., Board contains a collection of Squares.

18.What about Contraindications or caveats?

- Creation may require significant complexity:
    - recycling instances for performance reasons
    - conditionally creating instances from a family of similar classes
- In these instances, other patterns are available…
    - We'll learn about Factory and other patterns later

19. Give example for Information Expert pattern or principle?

Name: Information Expert

Problem: How to assign responsibilities to objects?

Solution: Assign responsibility to the class that has the information needed to fulfill it?

- E.g., Board information needed to get a Square.

20. Benefits for Design patterns?

- Understandability: Classes are easier to understand in isolation.
- Maintainability: Classes aren't affected by changes in other components.
- Reusability: easier to grab hold of classes.

21. What should the class of the initial domain object be?

Initial domain object a class at or near the root of the containment or aggregation hierarchy of domain objects.

22. Design for Store—Create()?

- Store needs to be associated with Product Catalog.
- Store needs to be associated with Registers.
- Registers needs to be associated with Product Catalog.

PART-B

1.Explain in detail about use case realization with GRASP pattern?

2.Explain polymorphism, pure fabrication and protected variations?

3.Explain in detail about creating design class diagrams?

4.Explian the concept of implementation model and also explain the mapping designs to code?

5.Elloberate the iteration 2and its requirements?

6.Explain about use case realization with GoF design pattern?

7.Describe creating methods from interaction diagrams?

8.Explain in detail about order of implementation&test first programming?

9.Describe domain model versus design model classes?

10.Expail in detail of visibility between objects?

UNIT 5

PART -A

1. What is State Diagrams?

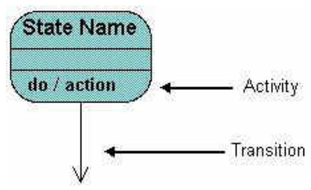
State diagrams are used to describe the behavior of a system. State diagrams describe all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and track the different states of its objects through the system.

2.When To Use: State Diagrams?

Use state diagrams to demonstrate the behavior of an object through many use cases of the system. Only use state diagrams for classes where it is necessary to understand the behavior of the object through the entire system. Not all classes will require a state diagram and state diagrams are not useful for describing the collaboration of all objects in a use case. State diagrams are other combined with other diagrams such as interaction diagrams and activity diagrams. [1]

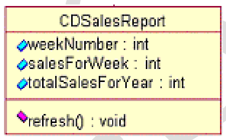
3.How to Draw: State Diagrams?

State diagrams have very few elements. The basic elements are rounded boxes representing the state of the object and arrows indicting the transition to the next state. The activity section of the state symbol depicts what activities the object will be doing while it is in that state.



4.Define Class diagram with example

The class diagram shows how the different entities (people, things, and data) relate to each other; in other words, it shows the static structures of the system.



5.Define Sequence diagram

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case.

6. What are the two dimensions of Sequence Diagram

A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent.

7. Define Component diagram

A component diagram provides a physical view of the system. Its purpose is to show the dependencies that the software has on the other software components (e.g., software libraries) in the system.

8.What about pure fabrication?

To support high cohesion and low coupling, where no appropriate class is present invent one Even if the class does not represent a problem domain concept .This is a compromise that often has to be made to preserve cohesion and low coupling.

9. Define indirection?

To avoid direct coupling between objects, assign an intermediate object as a mediatorRecall that coupling between two classes of different subsystems can introduce maintenance problemsAnother possibility is that two classes would be otherwise reusable (in other contexts) except that one has to know of the other

10.What is protected variations?

Assign responsibility to create a stable interface around an unstable or predictably variable subsystem or component

- If a component changes frequently, the users of the component will also have to be modified
- This is especially time consuming if the component has many users

11.Give example for Information Expert pattern or principle?

Name: Information Expert

Problem: How to assign responsibilities to objects?

Solution: Assign responsibility to the class that has the information needed to fulfill it?

- E.g., Board information needed to get a Square.

12.Benefits for Design patterns?

- Understandability: Classes are easier to understand in isolation.
- Maintainability: Classes aren't affected by changes in other components.
- Reusability: easier to grab hold of classes.

13.What should the class of the initial domain object be?

Initial domain object a class at or near the root of the containment or aggregation hierarchy of domain objects.

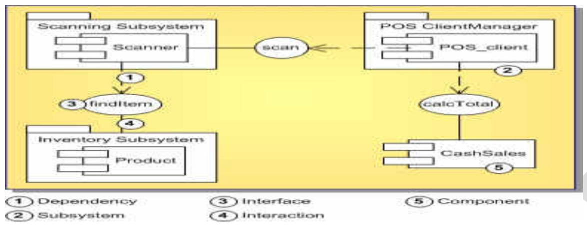
14.Define component

The software industry and literature use the term "component" to refer to many different things. It is often used in the broad sense to mean "a constituent part". It is also frequently used in a narrow sense to denote specific characteristics that enable replacement and assembly in larger systems.

15.Define component diagram.

A component is a container of logical elements and represents things that participate in the execution of a system. Components also use the services of other components through one of its interfaces. Components are typically used to visualize logical packages of source code (work product components), binary code (deployment components), or executable files (execution components).

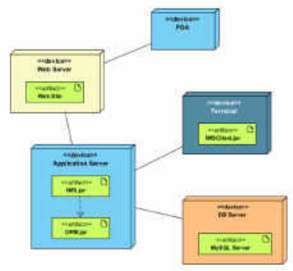
16.Give the sample diagram that shows the dependencies and interactions between software components for cash register



17.Define Deployment Diagram

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes.[1] To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

18.Draw the deployment diagram



19.Define operation.

An operation is a specification of a transformation or query that an object may be called to execute.

20.What is mean by inception phase?

Contracts are not motivated during inception. They are given in a detailed manner and it will be more.

PART B

1. Explain the various Use case Models with sample use case diagram?

2. Explain the component and deployment diagram by making use of library management as an example with respective diagrams?

3.Draw the state machine diagram for ATM system and explain how the modules are processed

4.Describe creating methods from interaction diagrams?

5.Explain in detail about order of implementation&test first programming?

6.Explian the concept of implementation model and also explain the mapping designs to code?

7.Draw the compile time component and explain the diagram.

8.Explain about the runtime components and explain it with diagram.

9.Explain about the Deployment diagram with example diagram.

10.Explain operation contracts expressed with the OCL.

11.Explain operation contracts expressed with the UP.