UNIT I

PART - A

1. What is an object?

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

2. What is the main advantage of object-oriented development?

• High level of abstraction

• Seamless transition among different phases of software development

• Encouragement of good programming techniques.

• Promotion of reusability.

3. What is Object Oriented System development methodology?

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

4. Distinguish between method and message in object.

Method Message

i) Methods are similar to functions, procedures or subroutines in more traditional programming languages. Message essentially are non-specific function calls.

ii) Method is the implementation. Message is the instruction.

iii) In an object oriented system, a method is invoked by sending an object a message. An object understands a message when it can match the message to a method that has the same name as the message.

5. What Is Analysis and Design?

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

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

6. What Is Object-Oriented Analysis and Design?

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

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

7. What is UML?

Unified modeling language is a set of notations and conventions and diagrams to describe and model an application.

8. What are the primary goals in the design of UML?

• Provide users a ready – to use expressive visual modeling language so they can develop and exchange meaningful models.

• Provide extensibility and specialization mechanism to extend the core concepts.

• Be independent of particular programming language and development process.

• 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.

9. Define Class Diagram.

The main static structure analysis diagram for the system, it represents the class structure of a system including the relationships between class and the inheritance structure.

10. Define Activity Diagram.

A variation or special case of a state machine in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations.

11. What is interaction diagram? Mention the types of interaction diagram.

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

There are two kinds of interaction models

• Sequence Diagram

• Collaboration Diagram.

12. What is Sequence Diagram?

Sequence diagram is an easy and intuitive way of describing the behaviors of a system by viewing the interaction between the system and its environment.

13. What is Collaboration Diagram?

Collaboration diagram represents a collaboration, which is a set of objects related in a particular context, and interaction, which is a set of messages exchanged among the objects with in collaboration to achieve a desired outcome.

14. Define Start chart Diagram.

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

15. What is meant by implementation diagram?

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

There are two types of implementation diagrams:

1. Component Diagrams

2. Development Diagrams.

16. Define Component Diagram?

A Component diagrams shows the organization and dependencies among a set of components. A component diagrams are used to model the static implementation view of a system. This involves modeling the physical things that reside on a mode, such as executable, libraries, tables, files and documents.

17. Define Deployment Diagram.

Deployment Diagram shows the configuration of run-time processing elements and the software components, processes, and objects that live in them.

Deployment diagrams are used to model the static deployment view of a system. A deployment diagram is a graph of modes connected by communication association.

18. What is the UP?

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

19. What is Iterations?

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

20. What is Iterative and Evolutionary Development?

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

21. What are the Phases of Unified Process?

The Unified Process has 4 phases:

– Inception: Requirements capture and analysis

– Elaboration: System and class-level design

– Construction: Implementation and testing

– Transition: Deployment

22. What is Inception?

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

23. Define Use case modeling?

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

24. Define Use case generalization?

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

Part –B(16 Marks)

1. Explain Types of UML Diagrams with example?

2. Explain Unified Phase and their types with an example?

3. Explain CASE STUDY: THE NEXTGEN POS SYSTEM?

UNIT II

PART - A

1. What is an Elaboration?

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

2. What is a domain model?

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

3. Define Conceptual Classes?

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

4. Define Description Class?

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

5. What are Three Strategies to Find Conceptual Classes?

1. Reuse or modify existing models.

2. Use a category list.

3. Identify noun phrases

6. What is an association?

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

7. What is an Attributes?

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

8. What About Attributes in Code?

The recommendation that attributes in the domain model be mainly data types does not imply that C# or Java attributes must only be of simple, primitive data types. The domain model is a conceptual perspective, not a software one. In the Design Model, attributes may be of any type.

9. What is a Derived Attributes?

The total attribute in the Sale can be calculated or derived from the information in the SalesLineItems. When we want to communicate that 1) this is a noteworthy attribute, but 2) it is derivable, we use the UML convention: a / symbol before the attribute name.

10. When to Define New Data Type Classes?

In the NextGen POS system an itemID attribute is needed; it is probably an attribute of an Item or ProductDescription. Casually, it seems like just a number or perhaps a string. For example, itemID : Integer or itemID : String.

11. Defining Conceptual Super classes and Subclasses?

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

12. What is Generalization?

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

13. What is Aggregation?

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

14. What is Composition?

Composition, also known as composite aggregation, is a strong kind of whole-part aggregation and is useful to show in some models. A composition relationship implies that 1) an instance of the part belongs to only one composite instance at a time, 2) the part must always belong to a composite and 3) the composite is responsible for the creation and deletion of its parts either by itself creating/deleting the parts, or by collaborating with other objects.

15. What is UML Activity Diagrams?

A UML activity diagram shows sequential and parallel activities in a process. They are useful for modeling business processes, workflows, data flows, and complex algorithms.

16. How to Apply Activity Diagrams?

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

Part –B (16 Marks)

1. Explain Domain Models with an example?

2. Explain Conceptual Classes with an example and what are Three Strategies to Find Conceptual Classes?

3. Explain Descriptions with the Airline Domain example?

4. Explain Associations with Applying UML?

5. Explain Attribute with Applying UML?

6. What are Suitable Attribute Types? Explain Focus on Data Type Attributes in the Domain Model.

7. Explain Conceptual Superclasses and Subclasses with an example?

8. Explain Aggregation and Composition?

9. Explain UML Activity Diagrams and Modeling?

UNIT III
PART - A

1. What is a system sequence diagram?

A system sequence diagram (SSD) is a fast and easily created artifact that illustrates input and output events related to the systems under discussion. They are input to operation contracts and most importantly object design.

2. What are System Sequence Diagrams?

A system sequence diagram is a picture that shows, for one particular scenario of a use case, the events that external actors generate their order, and inter-system events. All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.

3. What is the Logical Architecture?

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

4. What is a Layer?

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

5. What is Software Architecture?

An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization these elements and their interfaces, their collaborations, and their composition.

6. What\'s the Connection Between SSDs, System Operations, and Layers?

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

7. What is controller?

A controller is the first object beyond the UI layer that is responsible for receiving or handling a system operation message.

8. What is UML Class Diagrams?

The UML includes class diagrams to illustrate classes, interfaces, and their associations. They are used for static object modeling.

9. Define Classifier?

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

10. What is UML Operations?

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

11. What is UML Method?

A UML method is the implementation of an operation; if constraints are defined, the method must satisfy them. A method may be illustrated several ways, including:

• in interaction diagrams, by the details and sequence of messages

• in class diagrams, with a UML note symbol stereotyped with «method»

12. What is UML Keyword?

A UML keyword is a textual adornment to categorize a model element. For example, the keyword to categorize that a classifier box is an interface is (shocking surprise!) «interface».

What are UML Properties and Property Strings?

In the UML, a property is \"a named value denoting a characteristic of an element. A property has semantic impact.\" Some properties are predefined in the UML, such as visibility a property of an operation. Others can be user-defined.

Properties of elements may be presented in many ways, but a textual approach is to use the UML property string {name1=value1, name2=value2} format, such as {abstract, visibility=public}. Some properties are shown without a value, such as {abstract}; this usually implies a boolean property, shorthand for {abstract=true}. Note that {abstract} is both an example of a constraint and a property string.

13. What is qualified association?

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

14. What is an association class?

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

15. What is a Sequence diagram?

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

16. What is a Communication diagram?

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

17. What are the Strengths and Weaknesses of Sequence vs. Communication Diagrams?

Part –B (16 Marks)

1. Explain System sequence diagrams with an Example?

2. Explain logical architecture and UML package diagrams?

3. What\'s the Connection Between SSDs, System Operations, and Layers?

4. Explain Logical architecture refinement?

5. Explain UML class diagrams?

6. Explain Inter-Layer and Inter-Package Interaction?

7. Explain UML Interaction Diagrams?

8. Explain Operations and Methods with an example?

UNIT IV

PART - A

1. What is GRASP?

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

2. What is Responsibility-Driven Design?

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

3. What is Responsibilities?

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

4. What are the two responsibilities?

The responsibilities are of the following two types: doing and knowing.

Doing responsibilities of an object include:

• doing something itself, such as creating an object or doing a calculation

• initiating action in other objects

• controlling and coordinating activities in other objects

Knowing responsibilities of an object include:

• knowing about private encapsulated data

• knowing about related objects

• knowing about things it can derive or calculate

5. Define Pattern?

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

6. What are the GRASP patterns?

They describe fundamental principles of object design and responsibility assignment. expressed as patterns.

7. How to Apply the GRASP Patterns?

The following sections present the first five GRASP patterns:

. Information Expert

. Creator

. High Cohesion

. Low Coupling

. Controller

8. Define Creator?

Creation of objects is one of the most common activities in an object-oriented system. Which class is responsible for creating objects is a fundamental property of the relationship between objects of particular classes.

9. What is Controller?

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

10. Define Low Coupling?

Low Coupling is an evaluative pattern, which dictates how to assign responsibilities to support:

• low dependency between classes;

• low impact in a class of changes in other classes;

• high reuse potential;

11. Define High Cohesion?

High Cohesion is an evaluative pattern that attempts to keep objects appropriately focused, manageable and understandable. High cohesion is generally used in support of Low Coupling. High cohesion means that the responsibilities of a given element are strongly related and highly focused. Breaking programs into classes and subsystems is an example of activities that increase the cohesive properties of a system.

12. What is Information Expert?

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

13. What is singleton pattern?

The singleton pattern is a design pattern used to implement the mathematical concept of a singleton, by restricting the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system.

14. What is adapter pattern?

The adapter pattern is a design pattern that translates one interface for a class into a compatible interface. An adapter allows classes to work together that normally could not because of incompatible

interfaces, by providing its interface to clients while using the original interface. The adapter is also responsible for transforming data into appropriate forms.

15. What is Facade Pattern?

A facade is an object that provides a simplified interface to a larger body of code, such as a class library. A facade can:

• make a software library easier to use, understand and test, since the facade has convenient methods for common tasks;

• make code that uses the library more readable, for the same reason;

• reduce dependencies of outside code on the inner workings of a library, since most code uses the facade, thus allowing more flexibility in developing the system;

• Wrap a poorly-designed collection of APIs with a single well-designed API (as per task needs).

16. What is Observer pattern?

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

Part –B (16 Marks)

1. Explain GRASP: Designing objects with responsibilities

2. Explain GoF DESIGN PATTERNS

3. Explain Creator and Information Expert with an Example?

4. Explain Low Coupling and Controller with an Example?

5. Explain adapter and singleton with an example?

6. Explain factory and observer patterns.

UNIT V

PART - A

1. What is UML State machine Diagram?

A UML state machine diagram, illustrates the interesting events and states of an object, and the behavior of an object in reaction to an event.

2. Definitions: Events with example?

An event is a significant or noteworthy occurrence. For example:

• A telephone receiver is taken off the hook.

3. Definitions: States with an example?

A state is the condition of an object at a moment in time the time between events. For example:

• A telephone is in the state of being \"idle\" after the receiver is placed on the hook and until it is taken off the hook.

4. Definitions: States with an example?

A transition is a relationship between two states that indicates that when an event occurs, the object moves from the prior state to the subsequent state. For example:

• When the event \"off hook\" occurs, transition the telephone from the \"idle\" to \"active\" state.

5. What is State-Independent and State-Dependent Objects?

If an object always responds the same way to an event, then it is considered state-independent (or modeless) with respect to that event.

If for all events of interest, an object always reacts the same way, it is a state-independent object. By contrast, state-dependent objects react differently to events depending on their state or mode.

6. What is Deployment diagram?

A deployment diagram shows the assignment of concrete software artifacts (such as executable files) to computational nodes (something with processing services). It shows the deployment of software elements to the physical architecture and the communication (usually on a network) between physical elements.

7. What is Component Diagrams?

The Component Diagram helps to model the physical aspect of an Object-Oriented software system. It illustrates the architectures of the software components and the dependencies between them. Those software components including run-time components, executable components also the source code components.

8. What is Operation Contract?

An operation Contract describes the change in the state of the system when a system operation is invoked.

9. What is meant by an axiom? List the two design axioms of object oriented design.

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

Two design axioms:

Axiom 1: The independence axiom

Axiom 2: The information axiom.

10. Write the attribute presentation suggested by UML?

OCL can be used during the design phase to define the class attributes .The following is the attribute presentation suggested by UML.

Visibility name: type –expression-initial-value where visibility is

+ public visibility

protected visibility

- private visibility

Type – expression is language dependent specification .Initial – value is language dependent expression for the initial value of a newly created object.

11. What are the 3 relationships that can be shown in UML diagram? Define them.

1. Association how are objects associated? This information will guide us in designing classes.

2. Super-Sub Structure How are objects organized into super classes and sub classes? This information provides us the direction of inheritance.

3. Aggregation and a part of Structure what is the composition of complex classes? This information guides as in defining mechanisms that properly manage object within object.

12. What do you mean by layered architecture?

Layered architecture is an approach to software development that allows us to create objects that represents tangible elements of the business independent of how they are represented to the user through an interface or physically stored in a database.

13. Define Database Models And explain the categories.

A database model is a collection of logical constructs representing the data structure and data relationship within the database.

Database models is of two categories

1. Conceptual model

2. Implementation model

Conceptual Model: Focuses on logical nature of data. It deals with what is represented in the database.

Implementation Model: is concerned with how it is represented.

Part –B (16 Marks)

1. Explain UML state diagrams with an example?

2. Explain Operation contracts with an example?

3. Explain Mapping design to code with an Example?

4. Explain UML deployment diagrams with an example?

5. Explain UML component diagrams with an example?