**Chapter-09(Object Interaction)**

1. ***What is interaction sequence diagram?***

Ans. Sequence diagram or interaction sequence diagram shows an interaction between objects arranged in a time sequence. Sequence diagram can be drawn at different levels of detail and also to meet different purpose ant several stages in the development life cycle.

* Interaction diagrams are two types-

1. Sequence diagram
2. Collaboration diagram
3. ***Difference between sequence diagram and collaboration diagram.***

|  |  |
| --- | --- |
| Sequence diagram | Collaboration diagram |
| 1. Sequence diagram shows an interaction between objects arranged in a time sequence. 2. Sequence diagrams have a time dimension. 3. It does no show the link between object. | 1. Collaboration diagram shows an interaction between object and the content of the interaction in terms of the links between the objects. 2. Do not have time dimension. 3. It shows the link between objects. |

1. ***What is an object lifeline and focus of control?***

Ans.

Object lifeline: An object lifeline represents the existence of an object during an interaction represented in a sequence diagram.

Focus of control: Focus of control indicates which operation is executing at a particular stage in an interaction represented in a sequence diagram.

1. ***Difference between Synchronous and Asynchronous message.***

Ans.

|  |  |
| --- | --- |
| Synchronous message | Asynchronous message |
| 1. Synchronous message or procedural call is shown with a full arrowhead. 2. It causes the invoking operation to suspended execution until the focus of control has been returned. | 1. Asynchronous message is shown with an open arrowhead. 2. It does not cause the invoking operation to Holt execution while it awaits a return. |

1. ***What is Callback:***

It may be necessary for an operation that has been invoked asynchronously to notify the object that invoked it when it has terminated. This is done by explicitly sending a message (known as callback) to the originating object.

**Chapter-10(Specifying Operations)**

1. ***What are the two main purpose of operation specification?***
2. They confirm the user’s view of the logical behavior of a model.
3. They also specify what the designers and programmers must product or meet the user’s requirements.
4. ***Difference between algorithmic and non-algorithmic technique to operation specification.***

|  |  |
| --- | --- |
| **Algorithmic technique** | **Non-Algorithmic technique** |
| 1. An algorithm defines the step-by-step behavior of an operation. 2. An algorithm also specifies the sequence in which the steps are performed. 3. Generally do not prefer in object-oriented development. 4. Describe the internal logic.   eg. Activity Diagram. | 1. A non-algorithmic approach defines only inputs and results. 2. If does not specifies the sequence. 3. Generally preferred in object-oriented because Non-algorithmic methods of operation specification emphasize encapsulation. 4. Do not describe.   eg. Decision table. |

1. ***What are the main component of OCL operation?***

The UML has also a formal language known as Object Constraint Language(OCL), which is intended mainly for specifying general constraints on a model.

Most OCL statement consists of the following three components-

1. A context within which the expression is valid (for example, a specified class).
2. A property within the context to which the expression applies(for example and attribute of the specified class).
3. An operation that is applied to the property (for example a mathematical expression that tests the value of the attribute)
4. ***Define Activity Diagram.***

Activity Diagram: It is a version of state chart diagram that focuses on a flow of activity driven by internal processing within an object rather than by events that are external to it. Activity diagrams can be used to specify the logic or procedurally complex operations.

**Chapter-11(Specifying Control)**

1. ***What do you mean by state chart?***

The state chart is a versatile technique and can be used within an Object-Oriented approach for other purpose than the modeling of object life cycles. A state chart must have at least one initial state.

1. ***Mention the important link between state-chart and iteration diagram.***

There is an important link between state chart and iteration diagrams. A model of state behavior in a state chart captures all the possible responses of a single object to all the use cases in which it is involved. By contrast a sequence or a collaboration diagram captures the responses of the entire object that are involved in a single use case.

1. ***What are the main approaches for state-chart?***

The steps involved in the life cycle to state modeling are as follows-

1. Identify major system events.
2. Identify each class that is likely to behave a state dependent response to these events.
3. For each of these classes produce a first-cut state chart by considering the typical life cycle of an instance of the class.
4. Examine the state chart and elaborate to encompass more detailed event behavior.
5. Enhance the state chart to include alternative scenario us.
6. Review the state chart to ensure that is consistent with the use cases. In particular check that the constraints that the state chart implies are appropriate.
7. Iterate through steps d, e & f until the state chart captures the necessary level of detail.
8. Ensure consistency with class diagram and interaction diagrams and other state charts.

**Chapter-13(System Design)**

1. ***What are the major elements of system design?***

Standard for code development and human computer interaction.

1. ***What is software architecture?***

A software architecture is a description of the sub-systems and components of a software system and the relationship between them.

1. ***What is Layering and partitioning?***

Layering- The different sub-systems usually represent different levels of abstraction.

Partitioning- Usually means that each subsystem focuses on different aspect to the functionality of the system as a whole.

* Guidelines on the development of Layered architecture:

1. Define the criteria by which the application will be grouped into layers.
2. Determine the member of layers.
3. Name the layers and assign functionality to them.
4. Specify the services for each layer.
5. Refine the layering by iterating through steps i to l.
6. Specify interfaces for each layer.
7. Specify the structure of each layer.
8. Specify the communication between adjacent layer.
9. Reduce the coupling between adjacent layer.

* Open layered architectures are more difficult to maintain because each layer communicate with all lower layers hence increasing the degree of coupling in the architecture. A change to one layer amy ripple to many layers.
* A closed layer architecture may require more processing as messages have to be passed through interviewing layers.
* ***What is MVC?***
* Model provides the central functionality of the application and is aware of each of its dependent view and controllers components.
* View corresponds to a particular style and format of presentation of information to the user.
* Controller accept user input in the form of events that trigger the execution of operation within the model.

**Chapter-14(Object Design)**

* What levels of visibility may be assigned to an attribute or an operation?

Ans. Public, Private and Protected.

* Attribute is an element of the data structure that together with operation, defines a class.
* An attributes data type is declared in UML using the following system:

name ‘:’ type-expression ‘=’ initial-value ‘{’ properly storing

Data type

attribute name

balance : Money =0.00

* Operation: Operation in an aspect of the behavior that defines a class an element of the services that are provided by a class; a specification of an element of system functionality that will be implemented as a method of an object.

The syntax used for an operation is-

operation name’(’ parameter-list ’)’ ‘:’ return-type expression

* Object visibility: Visibility is an UML modeling element (eg, attributes or operations) may be designated with different levels of accessibility or visibility.

|  |  |  |
| --- | --- | --- |
| **Visibility symbol** | **Visibility** | **Meaning** |
| + | public | The feature is directly accessible by an instance of any class |
| - | private | The feature may only be used by an instance of the class that include it. |
| # | protected | The feature may be used them by instances if the class that includes it or of a subclass or descendant of that class. |
| ~ | package | The feature is directly accessible only by instance of a class in the same package. |

* Interfaces: An interface in UML is a group of externally visible (i.e. public) operations. An interface is equivalent to associations and only abstract operation.
* Further design guidelines:

1. Design clarity: design should be mode as easy as possible.
2. Don’t over design
3. Control Inheritance Hierarchies.
4. Keep message and operation simple
5. Design volatility.
6. Evaluate by scenario
7. Design by Delegation
8. Keep classes Separate

* Association: Association is a logical connection, usually between different classes.

1. One-to-one association
2. One-to-many
3. Many-to-many

* Integrity constraint:

A constraint that has to be enforced to ensure that the information system holds data that is manually consistent and is manipulated correctly.

* Referential integrity ensures that an object identifier in one object actually refers to an object that exists.
* Dependencies constraints ensure that attribute dependencies, values are maintained consistently where the value of one attribute is calculated from other attributes are maintained consistently.
* Domain integrity ensures that attributes only hold permissible values.
* Normalization: Normalization is a technique that group’s attributes based upon functional dependencies according to several rules to produce normalized data structures that are largely redundancy.

**Chapter-15(Design Patterns)**

1. **Difference between patterns and framework:**

|  |  |
| --- | --- |
| **Pattern** | **framework** |
| 1. A pattern is an abstract solution to a commonly occurring problem in a given context. 2. Patterns are more abstract and general. 3. Patterns are more primitive 4. A pattern cannot be directly implemented in particular software. | 1. Framework is a reusable mini-architecture that provides structure and behavior common to all application. 2. Frameworks are abstract and general. 3. Frameworks are primitive. 4. A framework can be directly implemented in a particular to software. |

1. **The three main categories of purpose that a pattern can have are-**
2. Creational Singleton Pattern
3. Structural Composite Pattern
4. Behavioral State Pattern
5. **Singleton Pattern:**

Can be used to ensure that only one instance of a class is created.

1. **Benefits and danger of using patterns:**

Benefits:

1. Pattern provides a mechanism for the reuse of generic solutions for object oriented and other approaches.
2. Pattern offers a vocabulary for discussing the problem domain.

Danger:

1. Some people believe that the use of patterns can limit creativity.
2. The use of pattern is an uncontrolled manner may lead to over design.