Descriptive Questions on UML

1. What is pattern? How do patterns help the software?

Ans: A pattern is an abstract solution to a commonly occurring problem in a given context. Patterns are more abstract and general.

Benefits:

a. Pattern provides a mechanism for the reuse of generic solutions for object oriented and other approaches.

b. Pattern offers a vocabulary for discussing the problem domain.

2. What is the difference between synchronous and asynchronous?

Ans.

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

3. Define event, state and transition.

Event is an occurrence that is of significance to the information system.

State of an object is determined by values of some of its attributes and the presence or absence of certain links with other objects. It reflects a particular condition for the object and normally persists for a period of time until a transition to another state is triggered by an event.

Transition is the movement from one state or activity to another triggered by an event. A transition may start and end at the same state.

4. Difference between Cohesion and Coupling.

Ans.

- ✓ *Cohesion: Cohesion is the degree to which the responsibilities of a single component form a meaningful unit.*

- ✓ *Coupling: Coupling describes the relationship between software components.*
- ✓ *Goal- Reduce coupling increase cohesion.*

5. *Explain briefly about MVC?*

MVC means Model View Controller architecture where-

- ✓ *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 accepts user input in the form of events that trigger the execution of operation within the model.*

6. *Define Integrity constraint, Normalization.*

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.*
- ✓ *Dependency constraints ensure that attribute dependencies, values are maintained consistently where the value of one attribute is calculated form other attributes are maintained consistently.*
- ✓ *Domain integrity ensures that attributes only hold permissible values.*

- ➢ ***Normalization**: Normalization is a technique that groups attributes based upon functional dependencies according to several rules to produce normalized data structures that are largely redundancy.*

7. *What are the advantage and disadvantage of singleton pattern?*

Benefits:

a. *Pattern provides a mechanism for the reuse of generic solutions for object oriented and other approaches.*

b. *Pattern offers a vocabulary for discussing the problem domain.*

Danger:

a. *Some people believe that the use of patterns can limit creativity.*

The use of pattern is an uncontrolled manner may lead to over design.

8. List the name of the fact finding techniques

Ans. There are 5 main fact finding techniques that are used by analyst to investigate requirements-

a) Background reading
b) Interviewing
c) Observation
d) Document Sampling
e) Questionnaires

9. What is multiplicity? Write down advantages of components.

Multiplicity devotes the range of values of the members of objects that can be linked to a single object by a specific association. Represent enterprise (or business) rules.

It is a constraint because it limits the behavior of a system. If a client can have only one staff contact it should be impossible to link a second.

10. Quality criteria for good design?

Ans. Functional, efficient, economical, reliable, secure, flexible, general, buildable, manageable, maintainable, usable, reusable.

11. Advantages & disadvantages of traditional waterfall life cycle.

Ans. Advantages:

a) Teams with specialized skill can be assigned to tasks in particular phases.
b) Progress can be evaluated at the end of each phase.
c) Attendant risk can be controlled and managed.

Disadvantages:

a) Real project rarely follow a simple sequential life cycle include.
b) Interactions are almost inevitable.
c) The elapsed time between inception and delivery is frequently too long.
d) It is unresponsive to changes in the technology or requirements.

12. Phases of waterfall life cycle.

a) System engineering
b) Requirements analysis.
c) Design.

d) Construction
e) Testing
f) Installation
g) Maintenance

13. What do you mean by Prototyping? What are the steps to prepare prototype?

In software development a prototype is a system or a partially complete system that is build quickly to explore some aspect of a system requirements and that is not intended as the final working system.

Main system require to prepare prototype

1) *Perform an initial analysis.*
2) *Define prototype objectives.*
3) *Specify prototype.*
4) *Construct prototype.*
5) *Evaluate prototype and recommend change.*

14. What do you mean by Incremental Development?

Incremental development involves some initial analysis to scope the problem and identify the major requirements. The requirements are that reviewed and those that deliver most benefit to the client become the focus of the first increment of development and delivery. The installation of the first increment provides valuable feedback to the development team and informs the development of the second increment and so on.

15. What is the difference between model and diagram?

Ans. Model:

Like any map, models represent something also. Models are usually both abstract and visible. They are useful in several different ways, precisely because they differ from the things that they represent-

a) *A model is quicker and easier to build.*
b) *A model can be used in simulations to learn more about the thing it represents.*
c) *A model can evolve as we learn more about a task or problem.*
d) *We can choose which details to represent in a model and which to ignore. It is an abstraction.*
e) *A model can represent real or imaginary things form any domain.*

Diagram: Diagrams are used to build models of system in the systems in the same way as architects use drawings and diagrams to model buildings.

Diagrammatical models are used extensively by system analysts and designers in order to-

a) Communicate ideas
b) Generate new ideas and possibilities
c) Test ideas and make predictions
d) Understand structures and relationships

A model provides a complete view of a system at a particular stage and form a particular perspective.

16. What is the purpose of Activity Diagram?

Activity diagram can be used to model different aspects of a system. At a high level, they can be used to model business activates in an existing or potential system.

Activity diagram can be used for the following purpose:

1) *To model a task*
2) *To describe a system function that is represented by a use case.*
3) *In operation specifications, to describe the logic of an operation.*
4) *In USDP to model the activities that make up the life cycle.*

17. What do you mean by Guard Condition?

Ans. Guard condition is a Boolean expression associated with a transition that is evaluated at the time the event fires. The transition only takes place if the condition is true. A guard condition is a function that may involve parameters of the triggering event and also attributes and links of the object that owns the state chart

18. What is use case? What is the purpose of use case?

Ans. Use case is descriptions of the functionality of the system from the user's perspective.
Use case diagrams are used to show the functionality that the system will provide and to show which users will communicate with the system in some way to use that functionality.
Purpose:
Use case is supported by behavior specifications there specify the behavior of each use case either using UML diagrams. Such as collaboration diagrams or sequence diagrams or in text from as use case descriptions.

Textual use case description provides a description of the interaction between the users of the system, termed actors and the high level functions within the system the use case.

19. What is Stereotypes? Describe include & exclude.

Ans. Stereotype:

A stereotype is a special use of a model element that is constrained to behave in a particular way Stereotypes can be show by using a keyword. Such as 'extend' or 'include' in matched quillements like <<extend>>. Stereotype can also be represented using special icon. The actor symbol in use case diagrams is a stereotyped icon- an actor is a stereotyped class and could also be shown as a class rectangle with the stereotype <<actor>> above the name of the actor.

<<extend>> is used when we wish to show that a use case provides additional functionality that way be required in another use case.

<<include>> applies when there is a sequence of behavior that is used frequently in a number of use cases and we want to avoid copying the same description of it into each use case in which it is used.

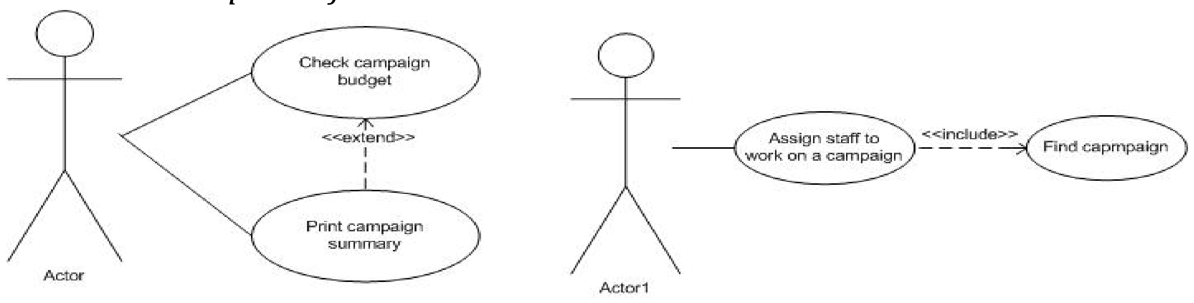

Fig: use case diagram showing <<extend>> and <<include>>

20. Define boundary class, entity class and control class?

Ans. Boundary class:

Boundary class is a stereotyped class that provides an interface to users or other system.

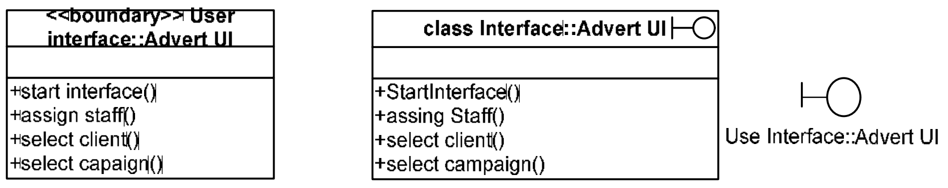

Fig: Alternative notations for boundary class stereotype.

Entity class:

Entity class is a stereotyped class that represents objects in the business domain model.

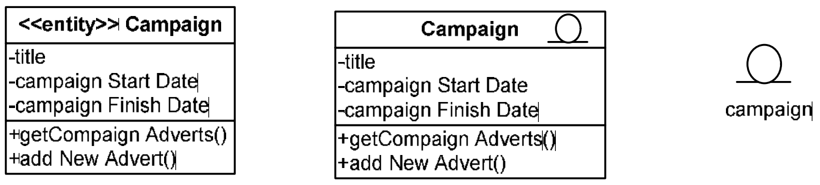

Fig: Alternative notation for an entity class.

Control class:

Control class is a stereotyped class that controls the interaction between boundary classes and entity classes.

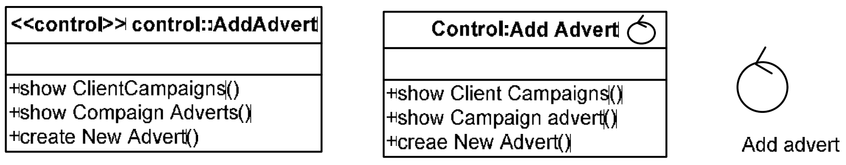

Fig: Alternative notation for a control class.

21. How does a collaboration diagram differ from class diagram?

Ans. A collaboration diagram shows only those classes that collaborate to provide the functionality of a particular use cases (or operation); the links that are shown are those that are required for that purpose.

A class diagram typically shows all the classes in a particular package and all the associations between them.

22. Distinguish between compositions from aggregation.

Ans.

Composition	*Aggregation*

1. Composition is a type of abstraction that encapsulates groups of classes that collectively have the capacity to be a reusable sub-assembly. Represent the whole and the other part of the whole. 2. Symbol ———◆ 3. A part can belong only one composition.	1. Aggregation represents a whole part association between two or more objects. 2. Symbol ———◇ 3. A part can belong more than one aggregation.

23. Difference between sequence diagram and collaboration diagram

Ans.

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. Don not have time dimension. 3. It shows the link between objects.

24. 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.

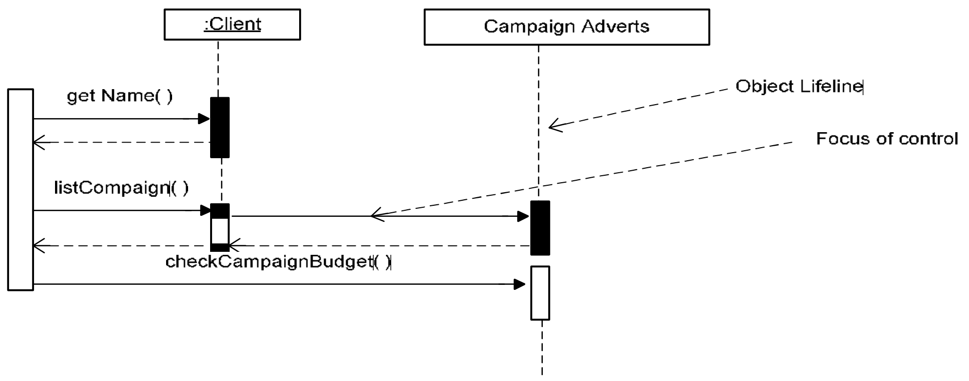

Fig: Sequence diagram showing object lifeline and Focus of control.

25. What is Layering and partitioning?

Ans. There are two general approaches to the division of a software system into subsystems. These are known as 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.

26.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.*

27. What is software architecture?

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

28. Difference between algorithmic and non-algorithmic technique to operation specification?

Ans.

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.*

29. Define Black box testing and white box testing?

SL	***Black Box***	***White Box***
1	*Focuses on the functionality of the system*	*Focuses on the structure (Program) of the system*
2	*Techniques used are :* · *Equivalence partitioning* ·*Boundary-value analysis* ·*Error guessing* ·*Race conditions* ·*Cause-effect graphing* ·*Syntax testing* ·*State transition testing* ·*Graph matrix*	*Techniques used are:* ·*Basis Path Testing* ·*Flow Graph Notation* ·*Control Structure Testing* 1. *Condition Testing* 2. *Data Flow testing* · *Loop Testing* 1. *Simple Loops* 2. *Nested Loops* 3. *Concatenated Loops* 4. *Unstructured Loops*
3	*Tester can be non technical*	*Tester should be technical*
4	*Helps to identify the vagueness and contradiction in functional specifications*	*Helps to identify the logical and coding issues.*

30. Define various levels of testing such as

- ***unit testing***

  Unit testing *is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use.*
- ***integration testing***

  Integration testing *is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.*
- ***sub-system testing***

  This phase involves testing collections of modules which have been integrated into sub-systems. Sub-systems may be independently designed. The most common problems which arise in large software systems are sub-system interface mismatches. The sub-system test process should therefore concentrate on the detection of interface errors by rigorously exercising the interfaces.

- ***system testing***

  Sub systems are integrated to make up the entire system. The testing process is concerned with finding errors that result from unanticipated interactions between sub-systems and system components. It is also concerned with validating that the system meets its functional and non-functional requirements.
- ***acceptance testing***

  This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system procurer rather than simulated test data. Acceptance testing may reveal errors and omissions in the system requirements definition because the real data exercises the system in different ways from the test data. It may also reveal requirements problems where the system's facilities do not really meet the user's needs or the system's performance is not acceptable.