



Important Instructions to examiners:

- 1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
- 2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
- 3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
- 4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
- 5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
- 6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
- 7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q.1.a) Attempt any THREE of the following:

MARKS 12

i) List the modeling techniques used in OMT. Explain the purpose of functional modeling.

(2 Marks for the listing the OMT technique & 2 Marks for the explanation of purpose of functional model)

Following technique used in OMT.

1. Analysis
2. System design
3. Object design
4. Implementation

Functional model gives functional aspect of a system.

- The functional model shows the- processes that are performed within an object and how the data changes as it moves between the model.
- The computations in the System are captured in functional models.
- Functional model shows the result of an action, but the model will not display the details on how the action is performed.
- Functional model are denoted by Data Flow Diagrams (DFDs).



ii) List any four diagrams suggested by Jacobson. State the importance of use case diagram.

(2 Marks for Listing 4 diagrams (1/2 mark each & 2 Marks for importance of use case diagram)

- Use case diagram.
- Interaction diagram.
 - Collaboration Diagram.
 - Sequence diagram.
- State chart and
- Activity diagram.

Importance of use case.

- Use cases express the entire system step-by-step detailing of the business process, bringing traceability between different phases of development.
- We use case diagram to illustrate the static use case view of a system.
- Use case diagrams are especially important in organizing and modeling the behaviors of a system.

iii) Explain the following terms:

1) Class 2) Object 3) Aggregation 4) Multiple inheritance.

(1 Mark for correct definition)


1) Class:- A class is a group of objects with similar properties (Attributes), Common behavior (operation) Common relationships to other objects and Common semantics.

e.g. a person, fruits, vegetables, animal, process are all classes.

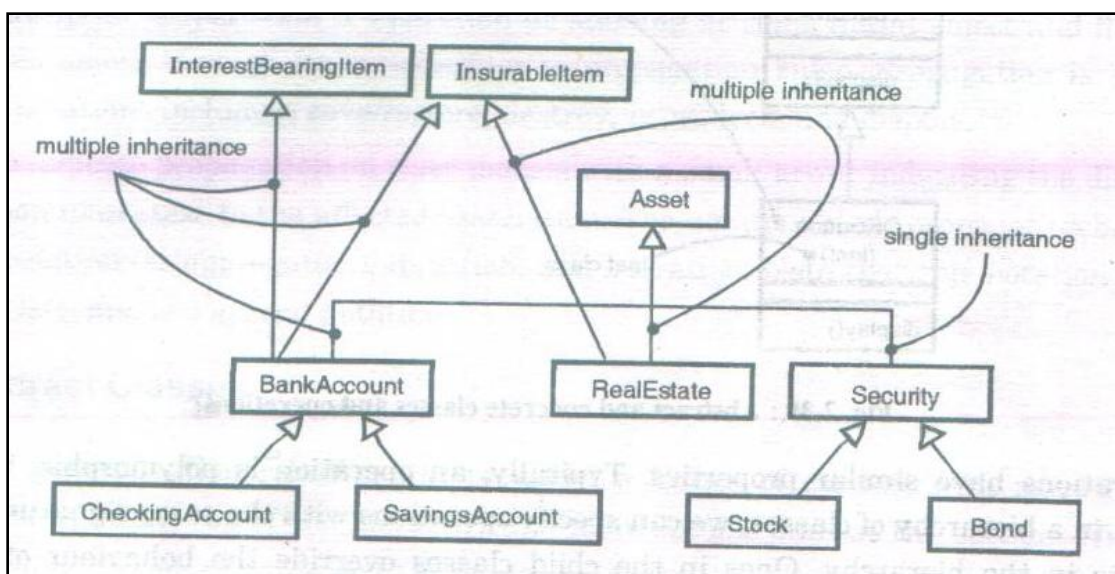
2) Object:- An object is a concept, abstraction or thing with crispy boundaries and meaning for the problem. Object is an instance of a class.

Object serves two purposes:-

1. They prompt understanding of the real world.
2. Provides a practical basis for computer implementation. E.g. mangoes, apple, abc etc.
- 3) Aggregation:- A plain association between two classes represents a structural relationship between peers, meaning that both classes are conceptually at the same level, no one more important than the other.

It is a special kind of association & is specified by adjoining a plain association with an open diamond  at the whole end.

- 4) Multiple inheritance:- it is defined as a class can inheritance features of two or more classes, this is known as multiple inheritance. Multiple inheritance permits a class to have more than one super class and to inherit features of all parents. This permits mixing of information from two or more sources. The advantage of M.I. is greater power specifying classes and an increased opportunity for reuse. It brings object modeling closer to the way people link.



iv) Write any four usage of UML.

(1Mark for each use of UML)

Unified Modeling language is a standard language for writing software blueprints.

It is very expressive language addressing all the views needed to develop & then deploy such system UML is language use for

- 1) Visualizing
- 2) Specifying
- 3) Constructing
- 4) Documenting



b) Attempt any ONE of the following:

MARKS 6

i) Write importance of Requirement Analysis. How CRC is helpful in analyzing system requirement. Explain.

(2 Marks for requirement analysis, 4 marks for CRC & Its Explanation)

With the help of clear requirement analysis you can create simple, easy to understand & use to prepare model, UML diagrams.

- An alternative to use case modeling is to write a requirements document that declares what the system will have to do.
- The goal of the requirement analysis is to fully specify the problem and application domain without introducing a bias to any particular implementation. There is no absolute line between the various design phases, nor is there any such thing as a perfect analysis.
- The requirement analysis gives proper defined data for model to be an effective means of communication with application experts who may not be computer expert.
- CRC (Class-Responsibility-Collaborator) card modeling is a simple yet powerful object oriented analysis technique.
- Their main purpose is to teach programmers the Object-oriented paradigm.
- CRC modeling includes the user, analysts, and developers in a modeling and design process, bringing together the entire development team to form a development team to form a common understanding of an OO development project.
- It is one of many tools that should be used in the collaborative design of used in the collaborative design of a system.
- Kent Beck & Ward Cunningham first introduced CRC cards.
- A CRC Model is a collection of cards (usually standard index cards or large) that are divided into three sections as shown in Fig. below



Class	
Responsibility	Collaborator

CRC card Layout

Advantages of CRC in analyzing system requirement.

- 1) The class, you make it easier talk through the designs.
- 2) Use of CRC Cards help to explore an Interaction between classes, typically to show how a scenario is implemented. Once you've figured out how the interaction works, you can document it with interaction diagram.
- 3) Use responsibilities to help you summaries the key responsibilities of a class. They are useful in highlighting different classes.

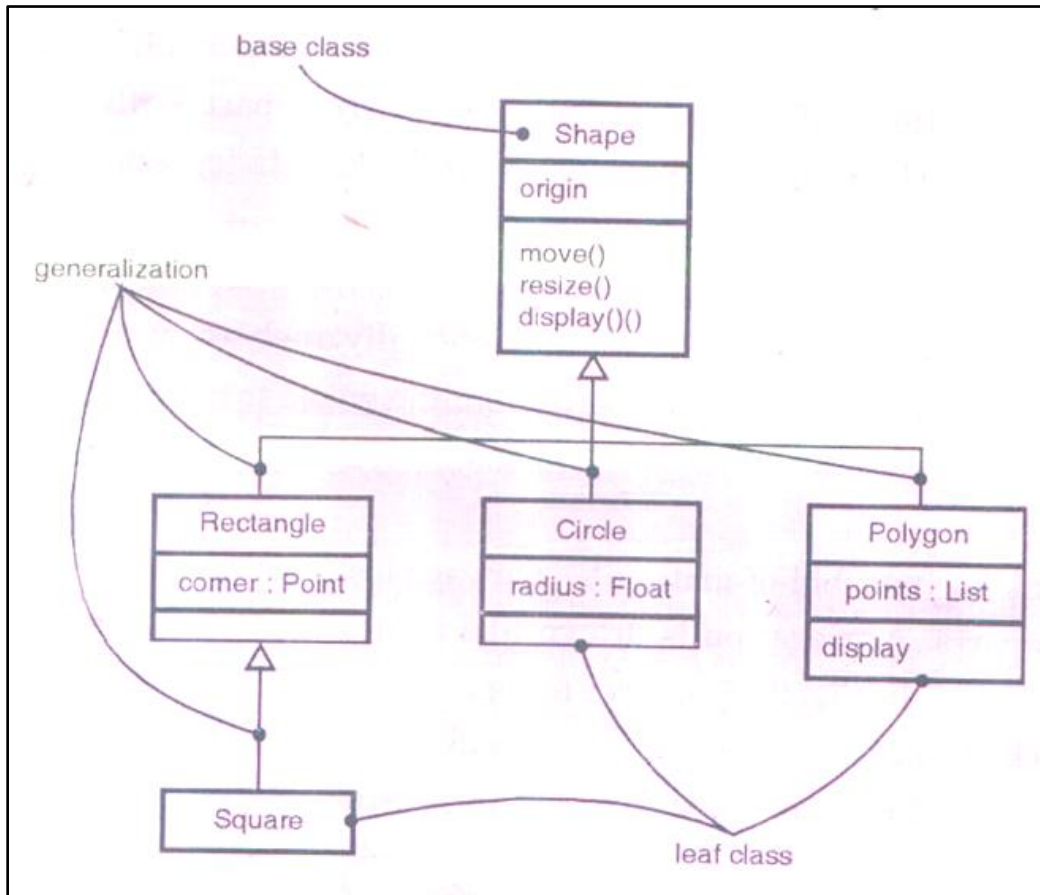
ii) Explain the generalization and association with suitable example and corresponding class diagram.

(Explanation of generalization & Association min. 4 each of 1 Mark, Example of each 2 Marks)

Generalization:

- A generalization is a relationship between a general thing (called the super class or parent) and more specific kind of that thing (called the subclass or child). Generalization is sometimes called an “is-a-kind-of” relationship: one thing (like the class Bay Windows) is-a-kind-of a more general thing (for example, the class Window).
- Generalization means that object of the child may be used anywhere the parent may appear, but not the reverse. In other words, generalization means that the child is substitutable for the parent. A child inherits the properties of its parents, especially their attributes and operations.

- Often but not always the child has attributes and operations in addition to those found in its parents.
- Generalization is rendered as a solid directed line with a large open arrowhead, pointing to the parent, as shown in example.

**Association:**

- An association is a structural relationship, specifying the object of one thing are connected to objects of another.
- For example, a Library class might have a one-to-many association to a Book class, indicating that each Book instance is owned by one Library instance.
- Furthermore, given a Book, we can find it owning Library, and given a Library, we can navigate to all its Books. Graphically , an association is rendered as solid line connecting the some or different classes.

- The associations are used when it necessary show structural relationship. There are four basic adornments that apply to an association.



Q.2. Attempt any TWO of the following:

MARKS 16

a) Explain the following with suitable example and diagrams.

i) Link and multiplicity.

ii) Abstract classes.

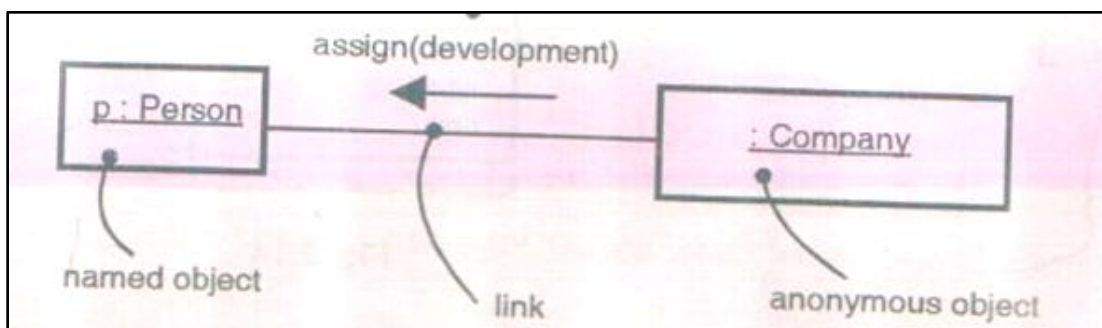
(i) (*Link and multiplicity -2 Marks for link & multiplicity explanation & 2 Marks for example & Diagram.*)

(ii) (*Abstract classes-3 Marks for abstract class, Defination & explanation , 1 marks for example & diagram*)

Link : Link and associations are the ways of connections among objects and classes

A link is physical connection between objects. mathematically, a link is defined as tuple. A link is an instance of an association.

A link shows relationship between two or more objects-





Multiplicity

Multiplicity specifies the number of one class that may relate to single instance of an associated class. Multiplicity constrains the number of related object.

An association represents a structural relationship among objects. In modeling situations it is important to state how many objects may be connected across an instance of an association. This “how many” is called the multiplicity of an association role.

A multiplicity can be stated at one end of an association, that it is specifying , for each object of class. At the opposite end, there must be that many objects at near end. A multiplicity can be shown of exactly one (1), zero or one (0..1), many (0..*).

Whenever a class is used, there may be any number of instances of that class. The number of instances a class may have is called its multiplicity.

Multiplicity is the specification of the range of all allowable cardinalities an entity may assume.

In the modeling, the multiplicity of a class can be specified by writing a multiplicity expression in the upper right corner of the icon.

Multiplicity can specify more complex multiplicities by using a list, such as 0..1,3..4,6..*, which would means any “any number of object other than 2 or 5”.

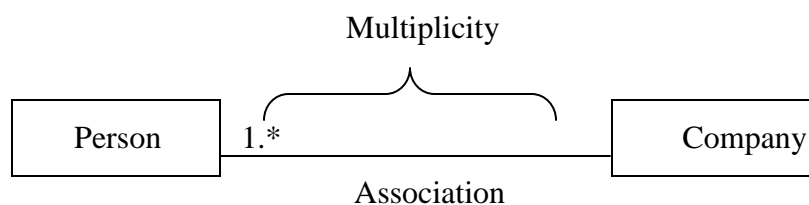
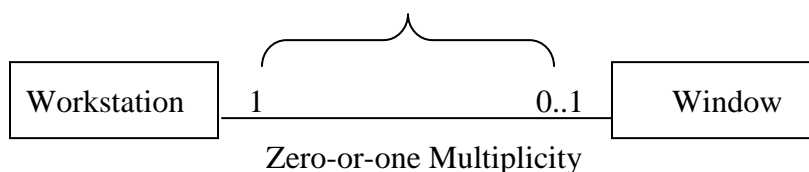


Fig. Multiplicity

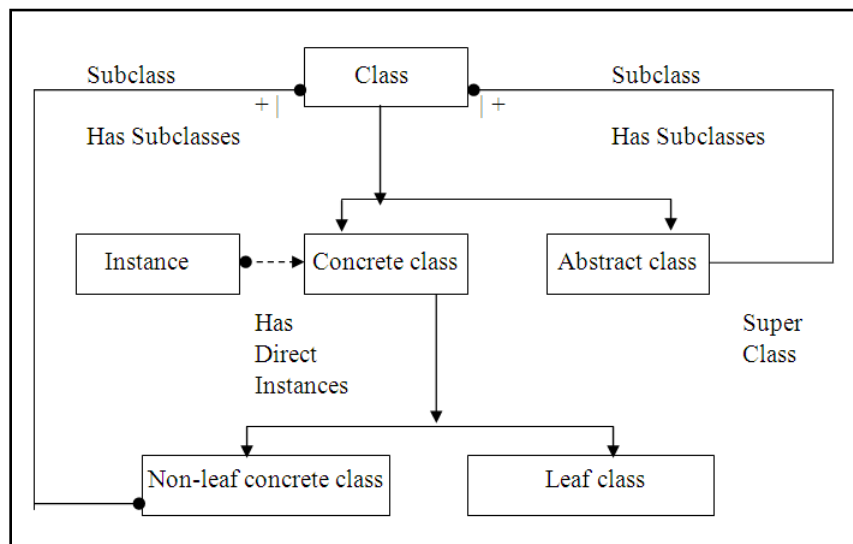


**ii) Abstract Classes:**

(Define 1Mark, every correct point will get 1 Mark, minimum 3 point (if diagram given 1 Mark))

Definition: a class that has no direct instances, but whose descendent classes have direct instances is known as abstract class.

- Abstract classes are incompletely implemented i.e the class has interfaces without implementations. Concrete classes have direct instances and they can be completely specified and completely implemented.
- Abstract classes are useful to create an abstract super class to encapsulate classes that participate in the same aggregation or association.
- Abstract classes are artificially introduced as mechanism for promoting code reuse.
- Abstract classes are frequently used to define methods to be inherited by subclass.
- An abstract class can define the protocol for an operation without supplying corresponding method, this is called as an abstract operation
- An abstract operation is specified by a comment in braces.



b) List and classify UML diagrams. Explain necessity of sequence diagrams.

(4 Marks listing & classification of UML diagram 4 Marks for brief explanation of necessity of sequence diagram.)

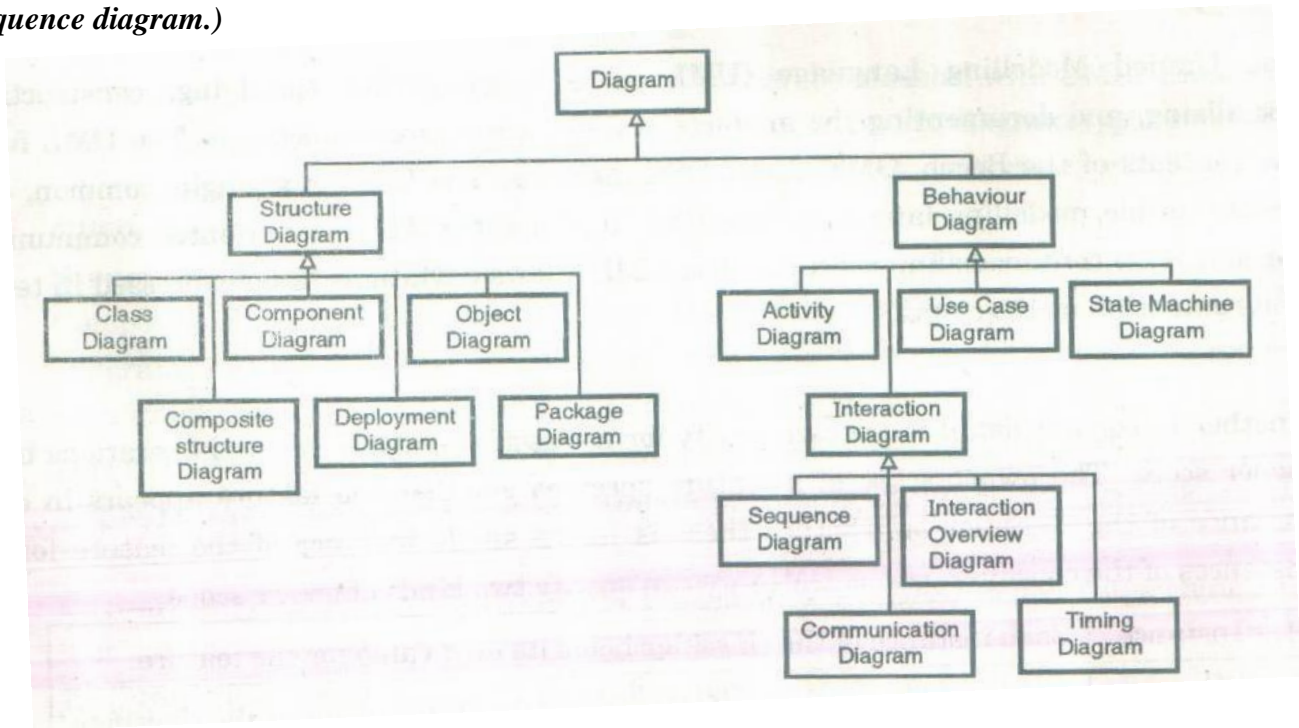


Diagram of projection into a system. The UML includes nine such diagrams.

1. **Class diagram** – A structural diagram that shows a set of classes, interfaces collaborations, and their relationships.
2. **Object diagram** – A structural diagram that shows a set of objects and their relationships.
3. **Use case diagram** – A behavioral diagram that shows a set of use cases and actors and their relationships.
4. **Sequence diagram** - A behavioral diagram that shows a set of use cases and actors and their relationships.
5. **Collaboration diagrams** - A behavioral diagram that shows an interaction, emphasizing the structural organization of the objects that send and receive messages.
6. **State chart diagram** – A behavioral diagram that show a state machine, emphasizing the even- ordered behavior of an object.
7. **Activity diagram** – A behavior diagram that shows a state machine, emphasizing the flow of activity to activity.



A sequence diagram is an interaction in diagram that emphasizes the time ordering of messages. A sequence diagram show a set of objects and the message sent and received by those object. The objects are typically named or anonymous instances of classes, but may also represent instances of other things, such as collaborations, components, nodes. You use sequence diagrams to illustrate the dynamic vies or a system.

Necessity of sequence diagram:

- A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.
- It is a construct of a Message Sequence Chart.
- A sequence diagram shows object interactions arranged in time sequence.
- A sequence diagram represents the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
- A sequence diagram represent the sequence of actions that occurs in system.
- A sequence diagram represents that dynamic behavior of a system.
- A sequence diagram are read left to right and descending.
- A sequence diagram is 2-Dimensional in nature. On the horizontal axis, it shows the life of the object that it represents, while on the vertical axis, it shows the sequence of the creation or invocation of these objects.



c) Draw and explain state chart diagram for ATM.

(5 Marks for ATM state chart diagram 3 Marks for brief explanation of ATM diagram requirements)

Steps for Building a state chart diagram for ATM.

1. Prepare a state diagram for each object class with non trivial dynamic behavior showing the events the object receives and sends.
2. Every scenario or event trace corresponds to a path through the state diagram. State chart diagram for ATM
3. Each branch in Control flow is represented by a state with more than one exit transition. In

ATM Example:

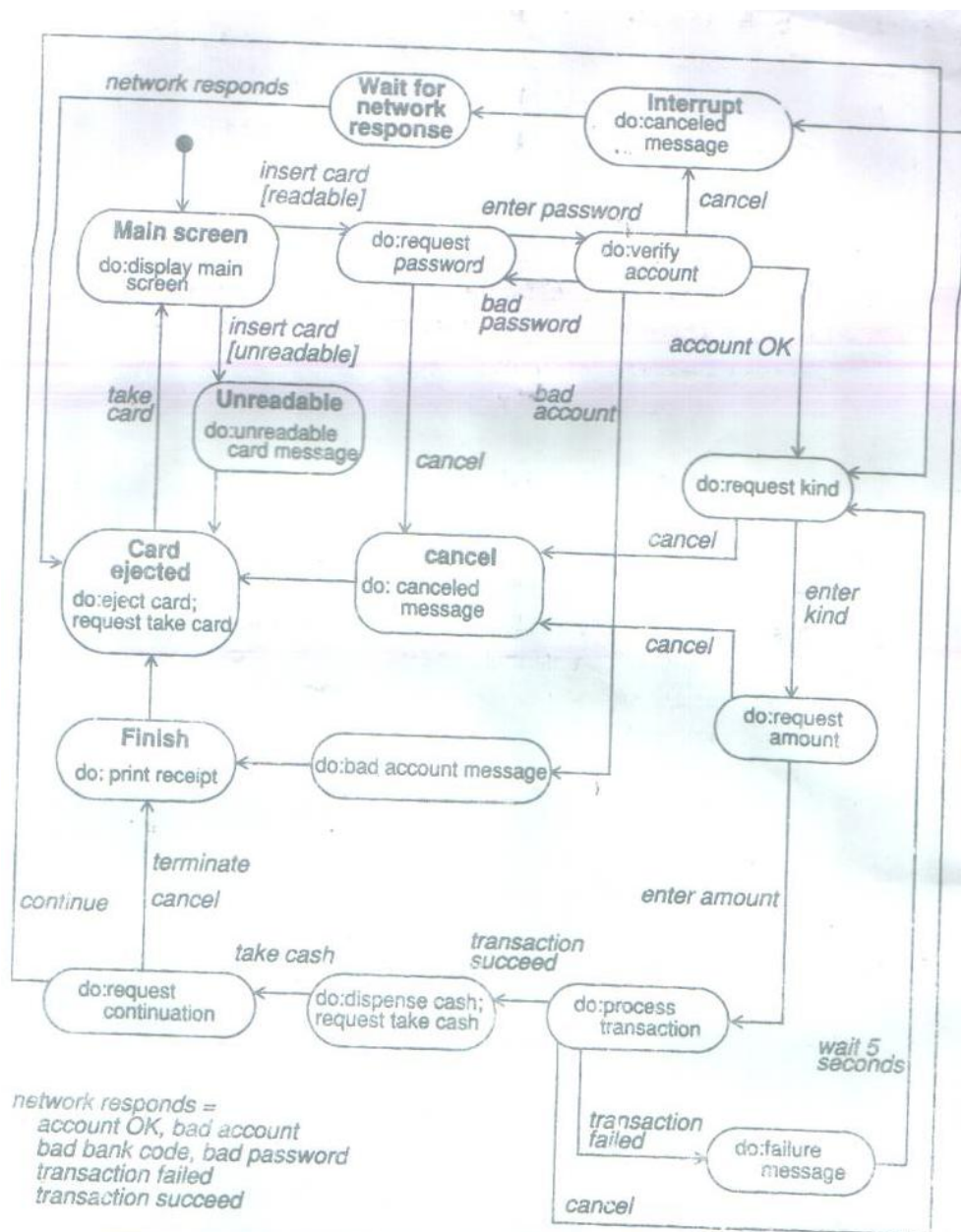
Object ATM, cashier station, consortium, and bank are actors that exchange events. Objects cash card, Transaction and Account are passive objects that are acted on and do not exchange events.

The customer and cashier are actors but their interactions with the entry stations are already shown. The customer and cashier objects are external to the system and need not be implemented within it anyway. Fig.1 shows the state chart diagram for the ATM.

Subject Code: 12260

Model Answer

**Subject Name : Object Oriented
Modelling & Design**



Q.3. Attempt any **FOUR** of the following:

MARKS 16

a) What is meta data. Give its examples.

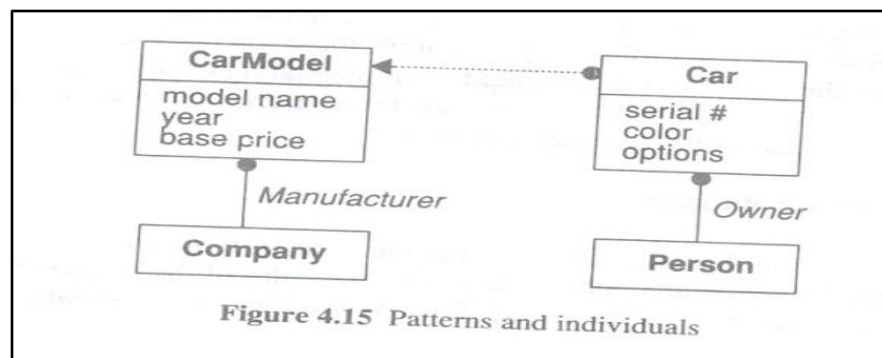
(Define meta data-1Mark, Instantiation -1Mark, example-2Marks)

- Define meta data: - Metadata is data that describes other data .for example, the definition of an class is metadata. Models are inherently metadata, since they describe the things being modelled. a relational database management system use metadata. A person can define database tables for storing information.
- Explain patterns and metadata:-(Instantiation)

Class describes a set of object instances of a given form. **Instantiation** relates a class to its instances.any pattern describes examples of the pattern;the relationship between pattern and example can be regarded as an extension of instantiation.

Real world things may be metadata.there are real world things that describe other real world things.A part description in a catalog describes manufactured parts.a blueprint describes a house.

Example:-



Explanation of example:-Each car model has its own attributes and associations.each car model object also describes a set of physical cars owned by persons.each car receives the common attributes from car model but also own list of particular attributes such as serial number,color and a list of options.

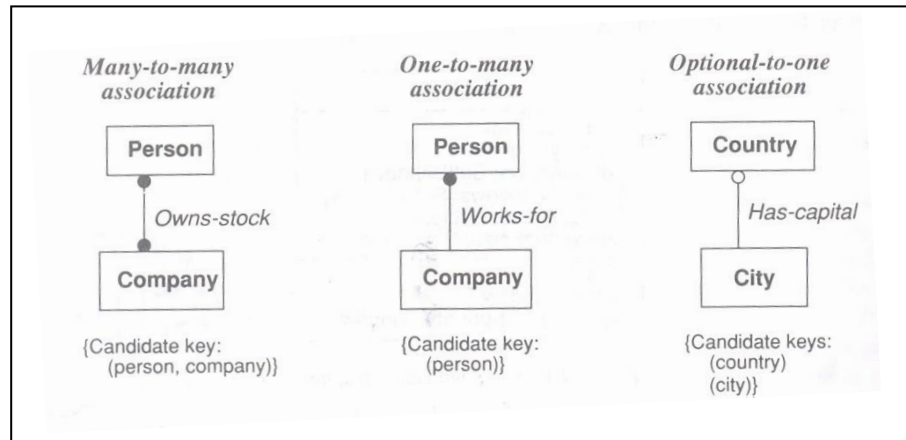
Car Model object as a pattern, a piece of metadata, that describes car objects.

b) Explain candidate key with suitable example.

(Define candidate key-2Marks, Example-2Marks=4Marks)

- **Candidate Key** – It is a minimal set of attributes that uniquely identifies an object or link. It means you cannot discard an attribute from the candidate key and still distinguish all objects and links. A class or association may have one or more candidate keys, each of which may have different combinations and numbers of attributes. The object id is always a candidate key for a class. One or more combinations of related objects are candidate keys for associations.

Notation: - A candidate key is delimited with braces in an object model.

Example :-

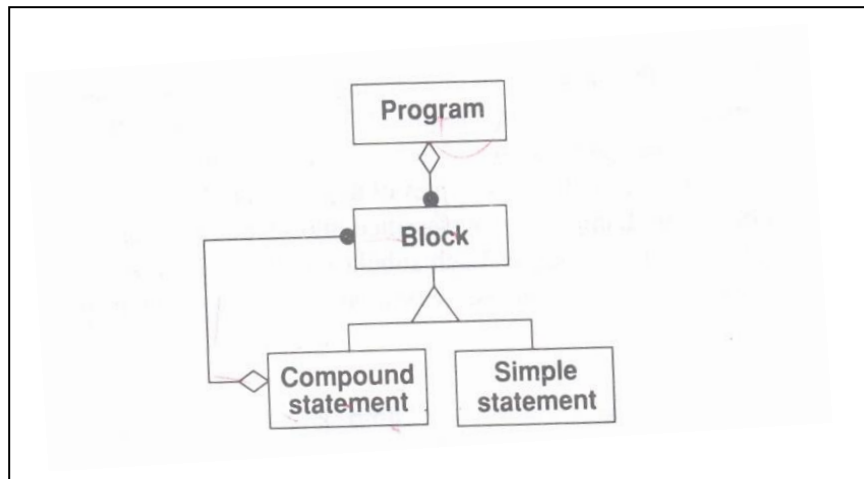
- A many- to- many association requires both related objects to uniquely identify each link.
- A one-to-many association has a single candidate key: the object on the many side.
- A one-to-one association has two candidate keys: either of the objects.

c) Explain recursive aggregation with suitable example.

(Define recursive aggregation-2Marks, Example -2Marks=4Marks)

- Aggregation definition-It is a 'has-a' relationship i.e. an object of the whole has objects of the part.
- A recursive aggregate contains, directly or indirectly, an instance of the same kind of aggregate; the number of potential levels is unlimited.

Example:-



- A computer program is an aggregate of blocks, with optionally recursive compound statements. The recursion terminates with the simple statement. Blocks can be nested to arbitrary depth.

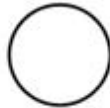
d) Define interface with suitable example and explain its importance.

(Define interface 1Mark, Notation-1Mark, example-1Mark, importance-1 Mark =4)

- An interface is a collection of operations that specify a service of a class or component. An interface might represent the complete behavior of a class or component or only a part of that behavior. An interface defines a set of operation specifications (that is, their signatures) but never a set of operation implementations.
- Notation: - Graphically, an interface is rendered as a circle; in its expanded form, an interface may be rendered as a stereotyped class in order to expose its operations and other properties.

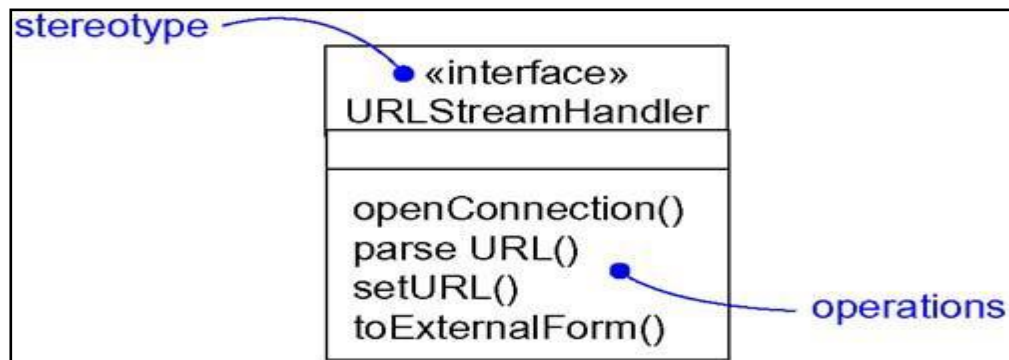


- An interface name must be unique within its enclosing package. Every interface must have a name that distinguishes it from other interfaces. A name is a textual string. That name alone is known as a simple name; a path name is the interface name prefixed by the name of the package in which that interface lives.

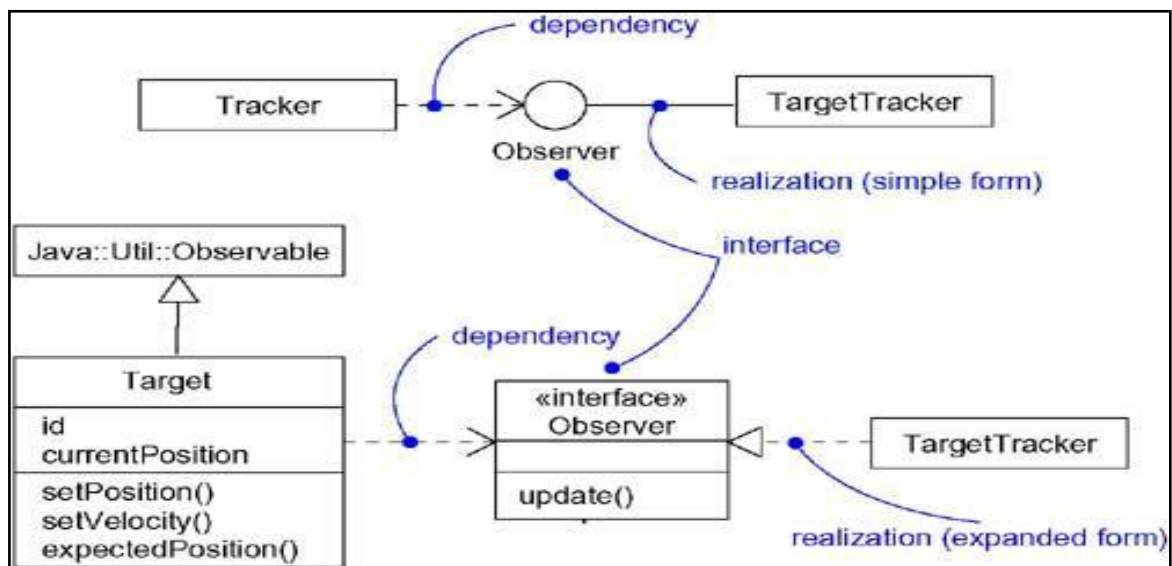


Interface Name

- Like a class, an interface may have any number of operations. These operations may be adorned with visibility properties, concurrency properties, stereotypes, tagged values, and constraints.
- When you visualize an interface in its normal form as a circle, by definition, you suppress the display of these operations. One can render an interface as a stereotyped class, listing its operations in the appropriate compartment. Operations may be drawn showing only their name, or they may be augmented to show their full signature and other properties, as shown below:



- Any example such as given below containing interface.



Importance:-

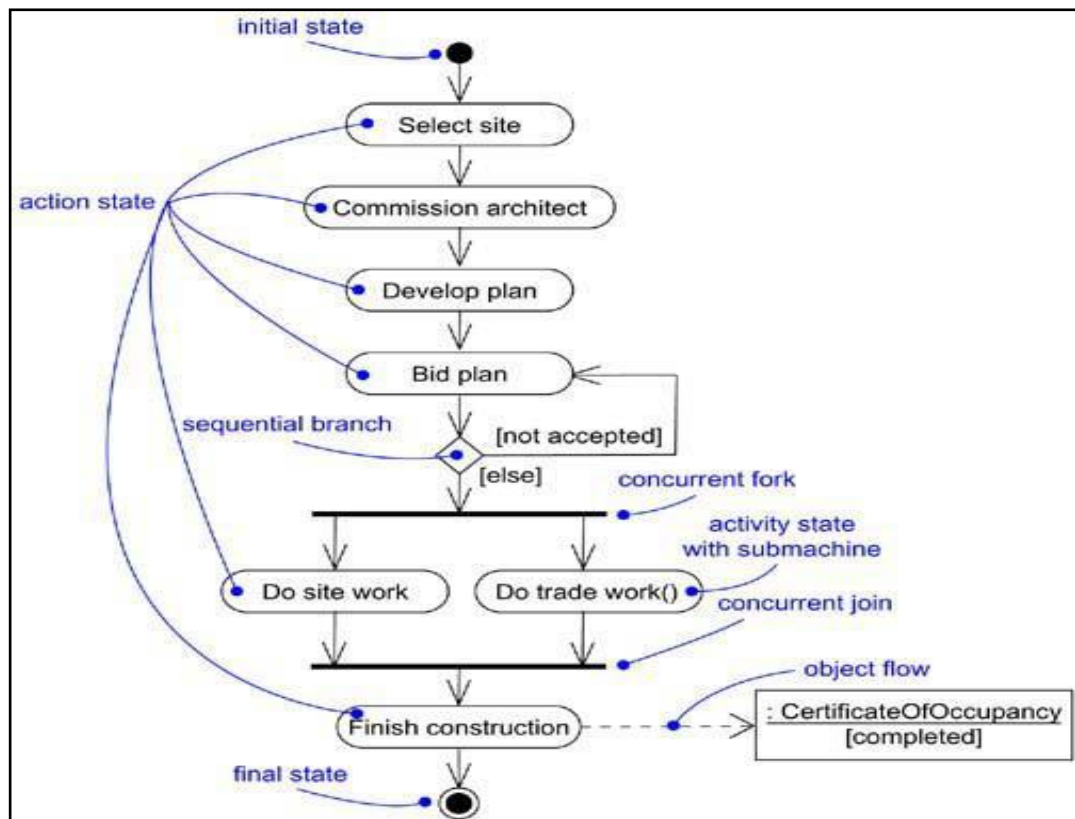
- An interface therefore describes the externally visible behavior of that element.
- An interface represent the complete behavior of a class or component or only a part of that behavior.
- An interface defines a set of operation specifications.

e) **How activity diagram helps you in modelling system.**

(Expiation of activity diagram-3 M & example 1 Mark =4 Marks)

- Define activity diagram:-**An activity diagram is essentially a flowchart, showing flow of control from activity to activity. You use activity diagrams to model the dynamic aspects of a system.
- With an activity diagram, you can also model the flow of an object as it moves from state to state at different points in the flow of control.
- Activity diagrams may stand alone to visualize, specify, construct, and document the dynamics of a society of objects, or they may be used to model the flow of control of an operation.

- An activity is an ongoing non atomic execution within a state machine. Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value. Graphically, an activity diagram is a collection of vertices and arcs.
- Activity diagrams commonly contain
 1. Activity states and action states
 2. Transitions
 3. Objects
- Example:-Any example can be included with activity diagram.





Q.4.a) Attempt any **THREE** of the following:

MARKS 12

i) Explain the importance of dynamic modelling. List diagrams used as dynamic modelling.

(Importance of dynamic modelling-2M,list-2M=4M)

Dynamic modelling:-

- The dynamic model shows the time-dependent behavior of the system and the objects in it.
- It is important for interactive systems, but insignificant for purely static data repository, such as database.
- The following steps are performed in constructing a dynamic model;-
 1. Prepare scenarios of typical interaction sequences.
 2. Identify events between objects
 3. Prepare an event trace for each scenario.
 4. Build a state diagram
 5. Match events between objects to verify consistency.

Diagrams used for dynamic modelling:-

1. State chart diagram
2. Sequence diagram
3. Collaboration diagram
4. Activity diagram

ii) How OOSE is helpful in terms of reusability. Discuss.

(Explanation of OOSE-2M & reusability explanation-2M= 4M)

(Since OOSE is vast topic examiner use his/her judgment and according reward marks)

About OOSE

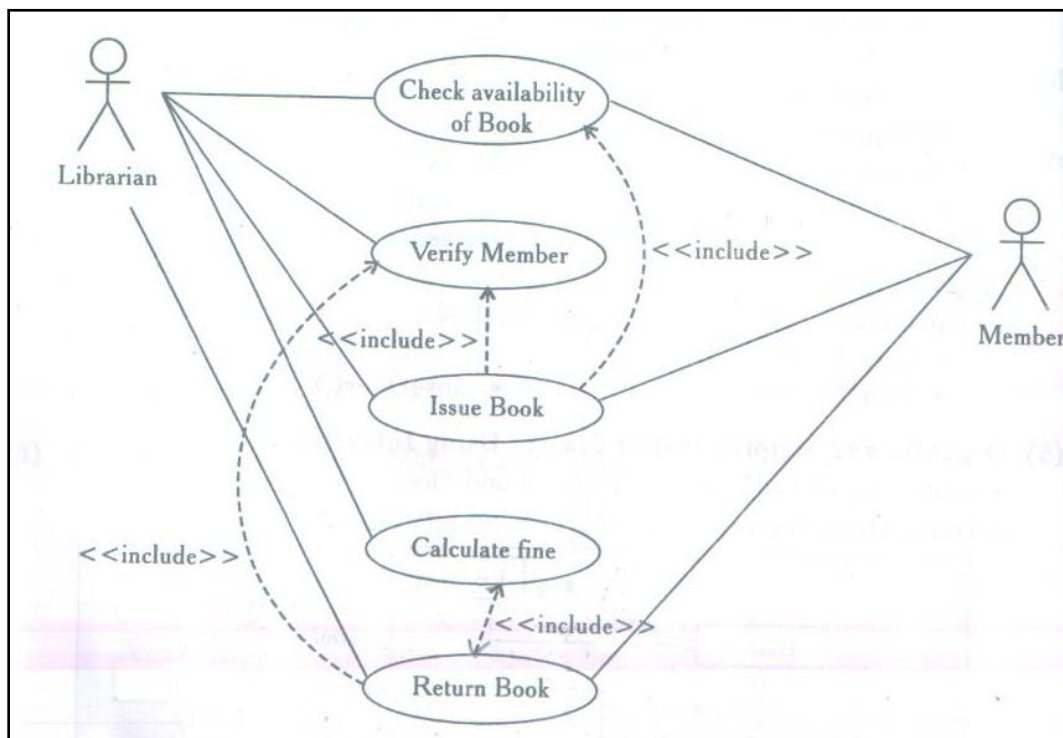
- **OOSE** – OOSE has a use case driven approach - In this approach, a use case model serves as a central model from which all other models are derived. A use case model describes the complete functionality of the system by identifying how everything that is outside the system interacts with the system.
- The **use case model** is the basis in the development phases such as **analysis**, construction (design and implementation) and testing.



- The aim of analysis is to understand the system according to its functional requirements. The objects are found, organized and object interactions are described. The operations of objects and the internal view of objects is described as well during analysis.
- Construction encompasses **design** and **implementation** in source code. It is important that objects in the analysis phase can be found back during construction. This is called traceability. Besides traceability, components are important during construction. A component is an already defined piece of source code that can be used for implementing objects.
- In **testing** the system is verified, meaning that the correctness of the system is checked according to its specifications.
- **Reusability :-**
- Using OOSE technique referred by Ivar Jaccobson, we design the system in such a way that modules of the current system can be directly adopted in designing new system.
- OOSE combines object-oriented programming & conceptual modeling concepts.
- From object-oriented programming, OOSE mainly uses the concepts of encapsulation, **inheritance** and relationships between classes and instances.
- Conceptual modeling is used to create different models of the system or organization to be analyzed with respect to dynamic behavior.
- The **building blocks or objects can be reuse** for designing a model for any application.
- By organizing the analysis and design models around sequences of user interaction and actual usage scenarios, the method produces systems that are more robust and usable, adapting more easily to changing usage.
- For reusability, documentation of all aspects (use case, building blocks, objects, classes, etc) is required in each phase.

iii) Draw use case diagram of library system.

(Any use case diagram can be considered with proper notation for use case, actor, and dependencies etc. explaining library system. 4-Marks)



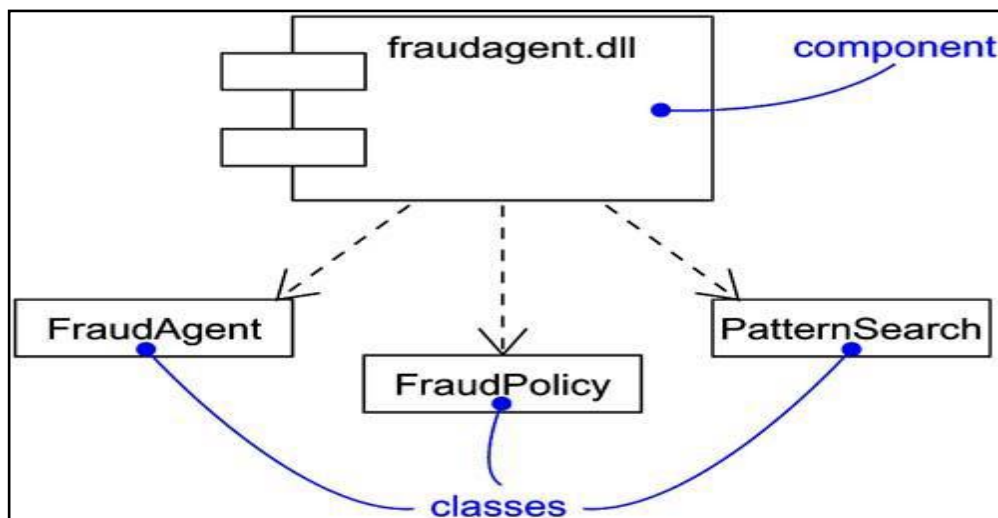
iv) Explain the importance of component diagram with example.

(Explanation 2Marks, example 2Marks=4Marks)

- A component diagram shows the organization and dependencies among a set of components. One can use component diagrams to model the static implementation view of a system.
- This involves modelling the physical things that reside on a node, such as executables, libraries, tables, files, and documents. Component diagrams are essentially class diagrams that focus on a system's components.
- Component diagrams are not only important for visualizing, specifying, and documenting component-based systems, but also for constructing executable systems through forward and reverse engineering.
- With the UML, one can use component diagrams to visualize the static aspect of these physical components and their relationships and to specify their details for construction
- A component diagram shows a set of components and their relationships.
- Graphically, a component diagram is a collection of vertices and arcs.



- Component diagrams commonly contain:-
 - 1.Components
 2. Interfaces
 3. Dependency, generalization, association, and realization relationships
- You use component diagrams to model the static implementation view of a system. This view primarily supports the configuration management of a system's parts, made up of components that can be assembled in various ways to produce a running system.
- When you model the static implementation view of a system, you'll typically use component diagrams in one of four ways.
 1. To model source code
 2. To model executable releases
 3. To model physical databases
 4. To model adaptable systems
- One example of component diagram:- Any diagram can be considered.



b) Attempt any ONE of the following:

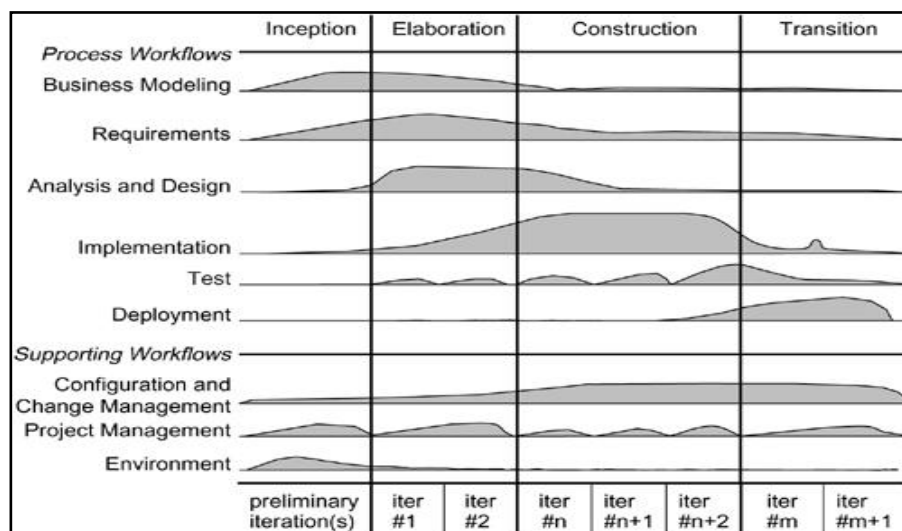
MARKS 6

i) Explain unified software development life cycle.

(2Marks-Diagram, 4Marks phases explanation=6Marks)

- The UML is largely process-independent, meaning that it is not tied to any particular software development life cycle. However, to get the most benefit the UML, you should consider a process that is
- Use case driven :- use cases are used as a primary artifact for establishing the desired behavior of the system, for verifying and validating the system's architecture, for testing, and for communicating among the stakeholders of the project.
- Architecture-centric:- a system's architecture is used as a primary artifact for conceptualizing, constructing, managing, and evolving the system under development.
- Iterative and incremental:- , an iterative and incremental process is *risk-driven*, meaning that each new release is focused on attacking and reducing the most significant risks to the success of the project.

There are four phases in the software development life cycle: inception, elaboration, construction, and transition. In the diagram, workflows are plotted against these phases, showing their varying degrees of focus over time.



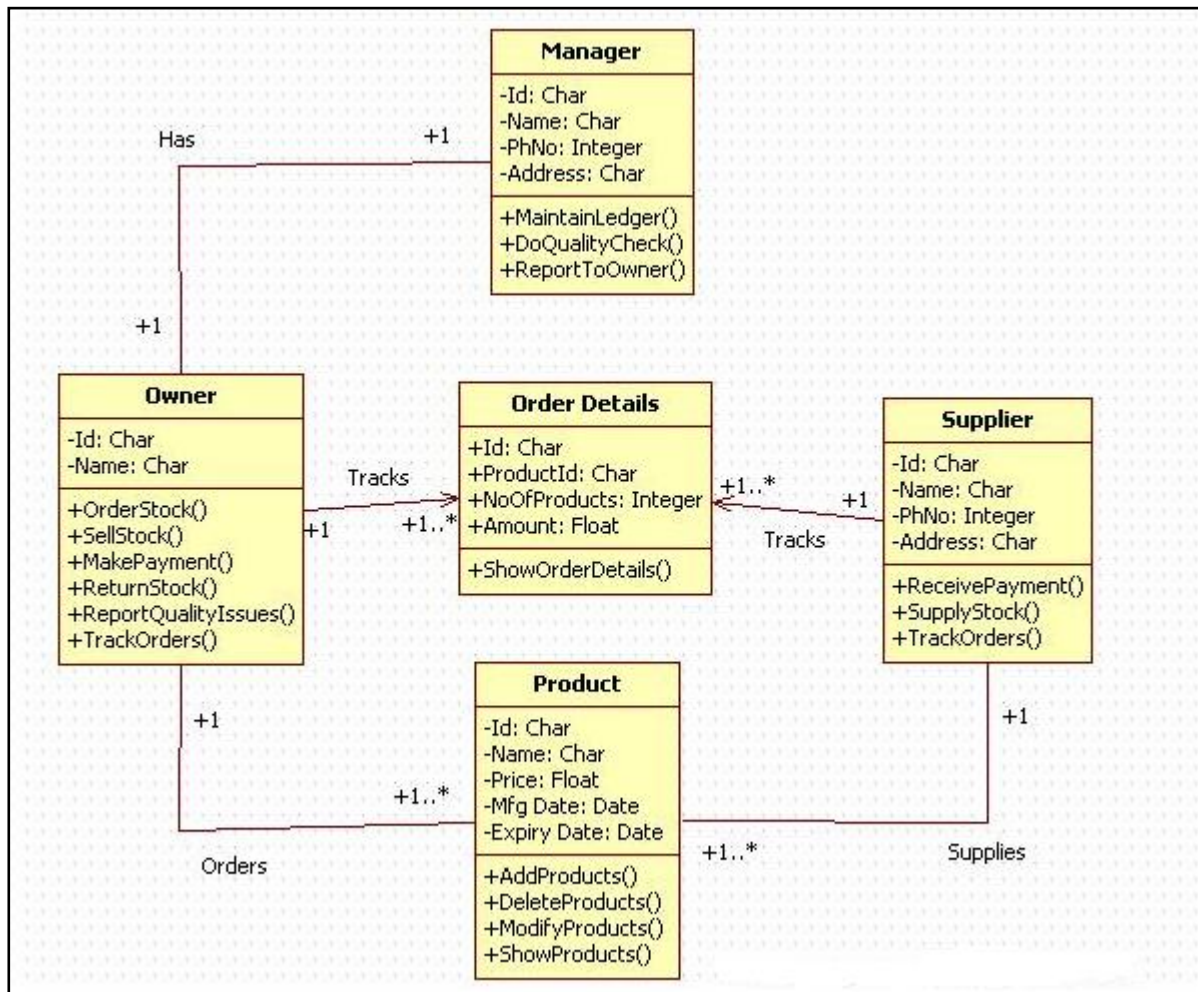


-
- **Inception** is the first phase of the process, when the seed idea for the development is brought up to the point of being• at least internally• sufficiently well-founded to warrant entering into the elaboration phase.
 - **Elaboration** is the second phase of the process, when the product vision and its architecture are defined. In this phase, the system's requirements are articulated, prioritized, and baselined. A system's requirements may range from general vision statements to precise evaluation criteria, each specifying particular functional or non-functional behaviour and each providing a basis for testing.
 - **Construction** is the third phase of the process, when the software is brought from an executable architectural baseline to being ready to be transitioned to the user community. Here also, the system's requirements and especially its evaluation criteria are constantly reexamined against the business needs of the project, and resources are allocated as appropriate to actively attack risks to the project.
 - **Transition** is the fourth phase of the process, when the software is turned into the hands of the user community. Rarely does the software development process end here, for even during this phase, the system is continuously improved, bugs are eradicated, and features that didn't make an earlier release are added.

ii) Draw class diagram for inventory system.

(Any suitable class diagram with class, association, multiplicity notations relevant to inventory system can be considered. (6Marks))

Example shown below.





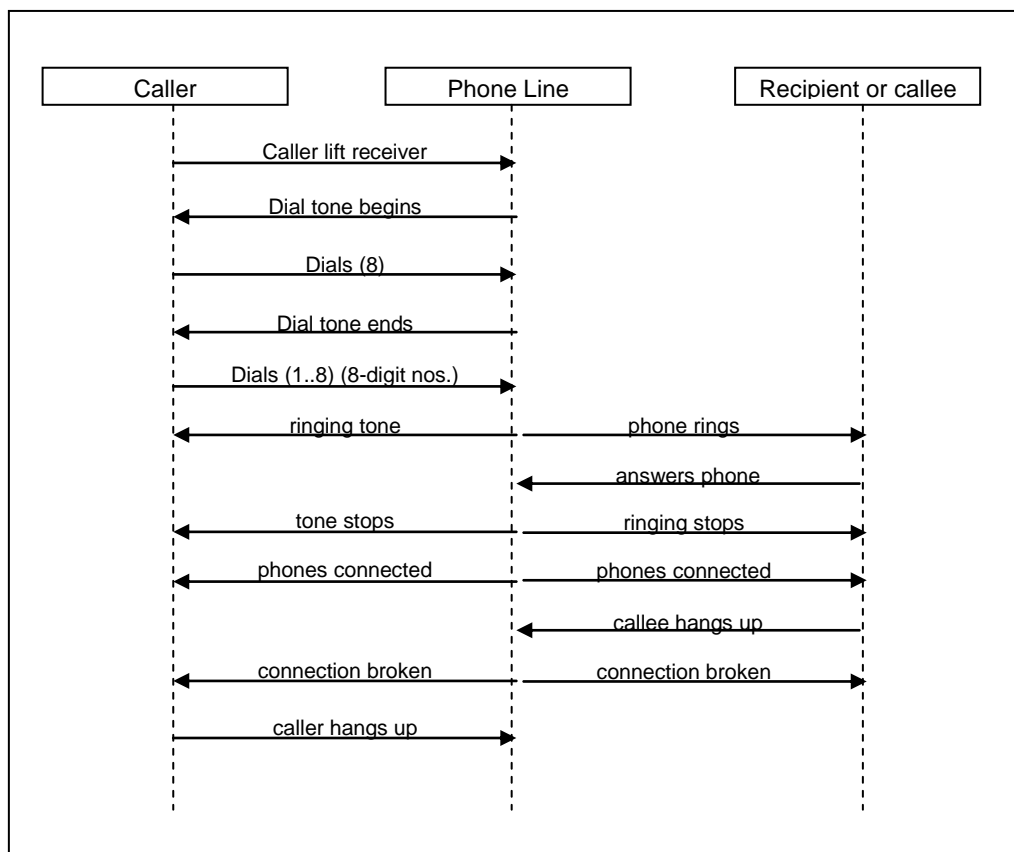
Q.5. Attempt any **TWO** of the following:

MARKS 16

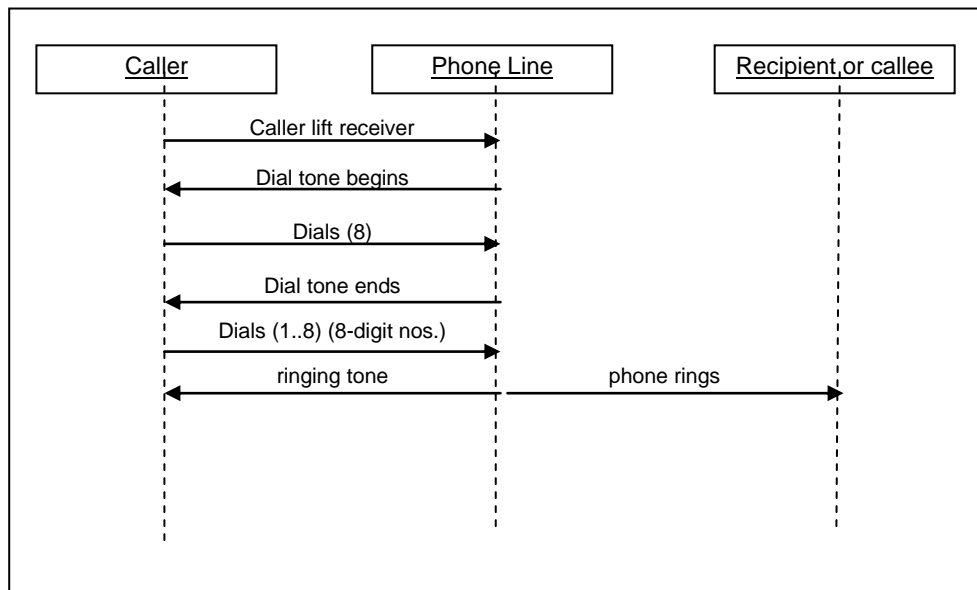
- a) Draw Sequence diagram for telephone (Landline) dialing and Explain consider all possible states.

(Following any Sequence diagram for telephone dialing - 5 Marks and Any 3 states - 3 Marks = 8 Marks)

Sequence Diagram for Telephone (Landline) dialing.



OR



Possible States of the telephone dialing system are :

1. Idle – When the system is on hook and no processing is going on.
2. Dial tone – When the caller lifts the receiver, dial tone state is activated.
3. Dialing – When the caller dials the first digit, this state is activated.
4. Busy tone – When the dialed number is busy
5. Ringing – When the connection is established, ring tone is given to the caller and callee.
6. Connected – When callee lifts up the receiver, the system is in connected state.
7. Disconnected – When the callee hangs up.
8. Connecting – after completing the dial operation the phone line tries to connected to number.

**b) Explain collaboration diagram with suitable example.**

(Explanation of Basics of Collaboration Diagram - 2 Marks, Explanation of Symbols/Notations/Elements - 2 Marks, Explanation of Features - 1 Marks, Any Example of Collaboration Diagram - 3 Marks = 8 Marks)

Collaboration diagram :

It is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. A collaboration diagram shows a set of objects, link among those objects and messages sent and received by those objects. The objects are typically named or anonymous instances of classes, but may also represent instances of other things, such as collaborations, components and nodes. You use collaboration diagrams to illustrate the dynamic view of a system. Sequence and collaboration diagrams are Isomorphic, meaning that you can convert from one to the other without loss of information. If you are using Rational Rose and you have drawn sequence diagram, its equivalent collaboration diagram can be drawn by just pressing F5 from keyboard. These diagrams are an extension of object diagrams.

Symbols /Notations/ Elements of Collaboration Diagram :

1. Object (Class Roles): The Objects describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Object : Class

2. Relation/Association (Association Roles): Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.

<<global>>

3. Messages: Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on. The a condition for a message is usually placed in



square brackets immediately following the sequence number. Use a * after the sequence number to indicate a loop.

Features of Collaboration diagram.

Collaboration diagrams have two main features which distinguish them from sequence diagram.

a. **Path** : There is a path for indication of how one object is linked to another. You can attach a path stereotype to the far end of a link (such as <<local>>, indicating that the designated object is local to the sender.) Typically, you will only need to render the path of the link explicitly for local parameter, global and self (but not association) paths.

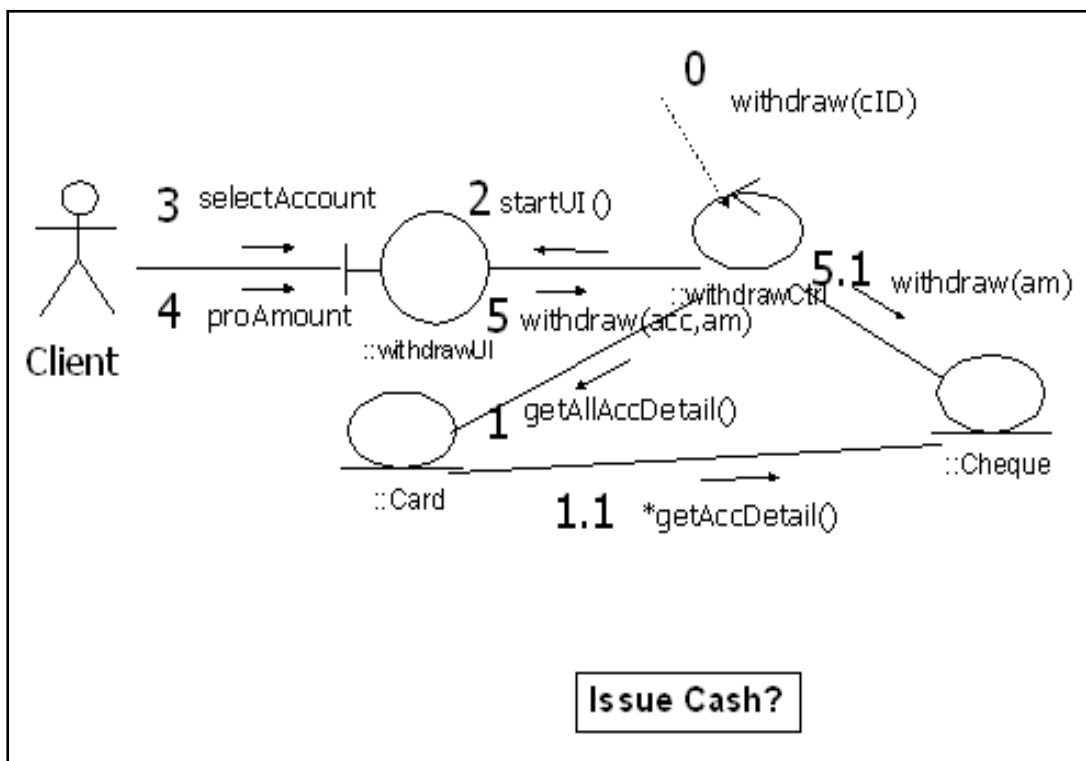
b. **Sequence number** : There is a sequence number for indication of the time order of message. You prefix the message with a number (starting with the messages numbered 1), increasing monotonically for each new message in the flow of control (2,3,4 and so on). To show nesting, you use Dewey decimal numbering (1 is the first message, 1.1 is the first message nested in message 1. 1.2 is the second message nested in message 1; and so on.) You can show nesting to an arbitrary depth. Note also that, along the same link, you can show many messages (possibly being sent from different directions), and each will have a unique sequence number.

Mostly, you can model straight, sequential flows of control. However, you can also model more complex flow, involving iteration and branching. Iterations represents a repeated sequence of messages.

It is important to note that you can show only simple branching in sequence diagram and more complex branching in collaboration diagrams.

Example of Collaboration Diagram:

A. ATM System:



c) What is object diagram? Explain with suitable example.

(Basic Explanation – 2 Marks, Object Diagram Symbols/Notation any 2 – 2 Marks, , Any Example - 4 Marks = 4 Marks)

Object Diagrams

- An object diagram shows a set of objects and their relationships at a point in time.
- An object diagram consists of the objects that collaborate, but without any of the messages passed among them.
- An object diagram is essentially an instance of a class diagram or the static part of an interaction diagram.
- Used to model the static design view or static process view of a system
- Object diagrams helps in modeling static data structures.
- Object diagrams commonly contain – Objects & Links
- Object diagrams are also closely linked to class diagrams. Just as an object is an instance of a class, an object diagram could be viewed as an instance of a class diagram.

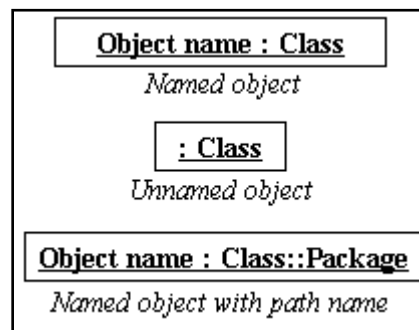


- Object diagrams describe the static structure of a system at a particular time and they are used to test the accuracy of class diagrams.

Object Diagram Symbols and Notations:

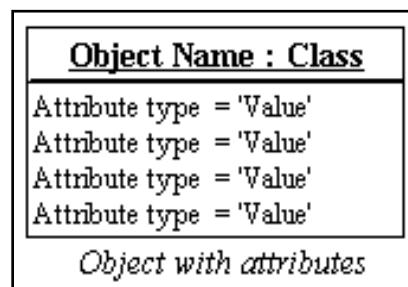
Object names

Each object is represented as a rectangle, which contains the name of the object and its class underlined and separated by a colon.



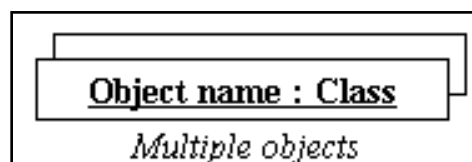
Object attributes

As with classes, you can list object attributes in a separate compartment. However, unlike classes, object attributes must have values assigned to them.



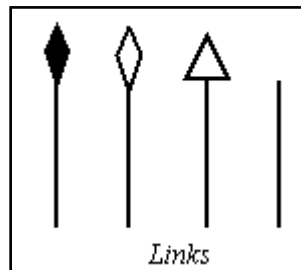
Multiplicity

You can illustrate multiple objects as one symbol if the attributes of the individual objects are not important.

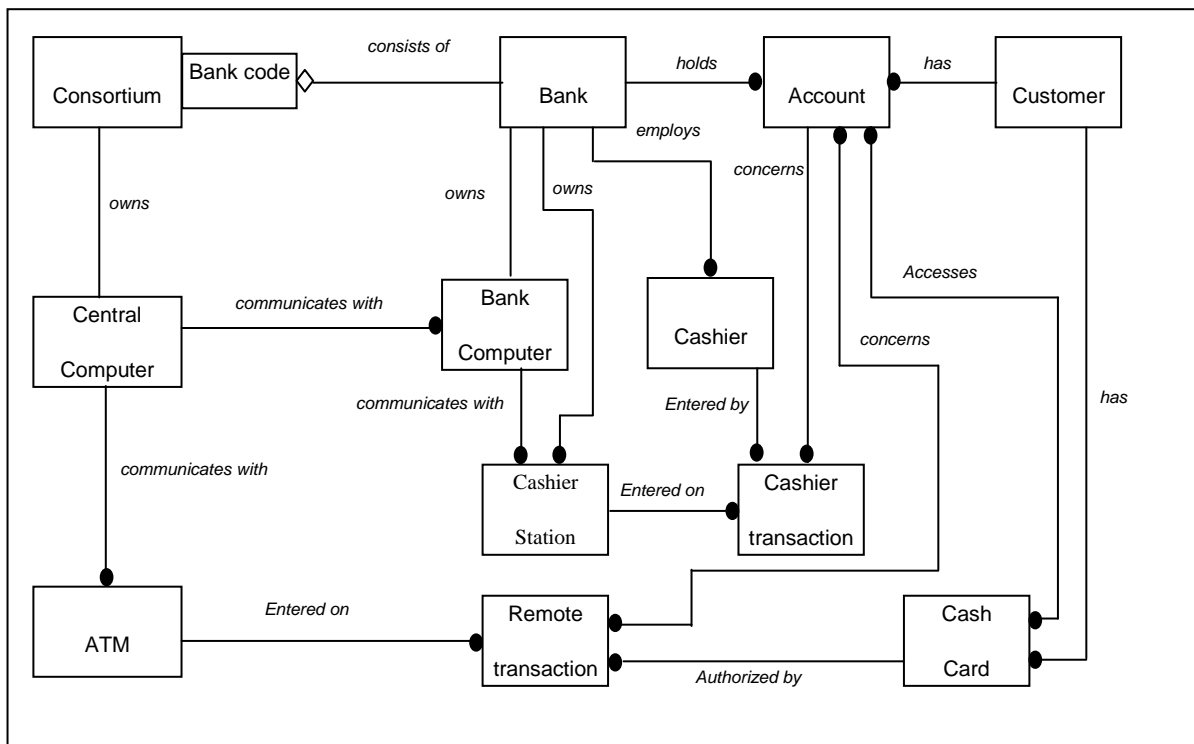


Links

Links are instances of associations. You can draw a link using the lines used in class diagrams.



Example: Object Diagram for ATM System





Q.6. Attempt any **FOUR** of the following:

MARKS 16

a) Write the scope of UML in analysing and designing a system.

(Any Explanation which aimed at Scope of UML in analysis and designing – 4 Marks)

Scope of the UML:

The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system.

1. The Unified Modeling Language combines the concepts of Booch, OMT, and OOSE. The result is a single, common, and widely usable modeling language for users of these and other methods.
2. The Unified Modeling Language pushes the envelope of what can be done with existing methods. As an example, the UML authors targeted the modeling of concurrent, distributed systems to assure the UML adequately addresses these domains.
3. The Unified Modeling Language focuses on a standard modeling language, not a standard process. Although the UML must be applied in the context of a process, it is our experience that different organizations and problem domains require different processes.
For example, the development process for shrink-wrapped software is an interesting one, but building shrink-wrapped software is vastly different from building hard-real-time avionics systems upon which lives depend.
4. The efforts concentrated first on a common metamodel (which unifies semantics) and second on a common notation (which provides a human rendering of these semantics). The UML authors promote a development process that is use-case driven, architecture centric, and iterative and incremental.

The Unified Modeling Language provides the following:

1. Semantics and notation to address a wide variety of contemporary modeling issues in a direct and economical fashion
2. Semantics to address certain expected future modeling issues, specifically related to component technology, distributed computing, frameworks, and executability.
3. Extensibility mechanisms so individual projects can extend the metamodel for their application at low cost. We don't want users to directly change the UML metamodel.



4. Extensibility mechanisms so that future modeling approaches could be grown on top of the UML.
5. Semantics to facilitate model interchange among a variety of tools.
6. Semantics to specify the interface to repositories for the sharing and storage of model artifacts.

b) Draw the appropriate symbols and write suitable example for following:

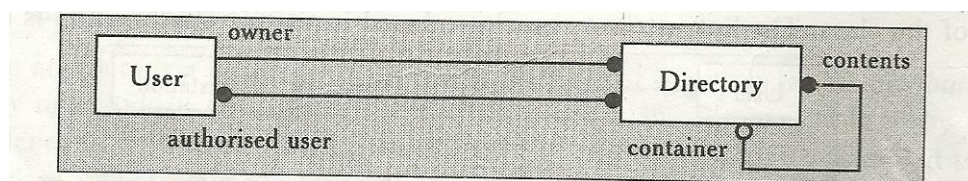
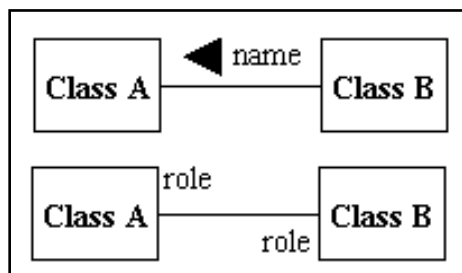
(Diagram of a Symbol -1/2 Marks each, Example – 1/2 Marks each 4 Marks)

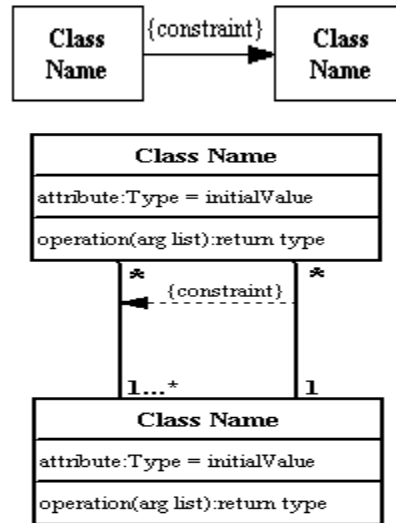
i) Role Name:

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

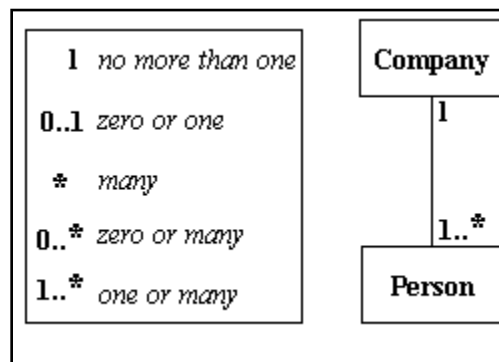
Note: It's uncommon to name both the association and the class roles.

Symbol and Example :



**ii) Constraint:**Symbol and Example:**iii) Multiplicity (Cardinality)**

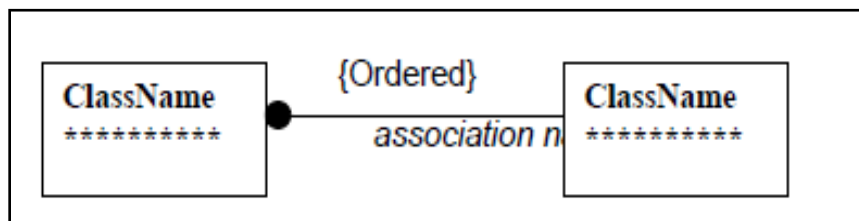
Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.

Symbol and Example:

iv) Ordering:

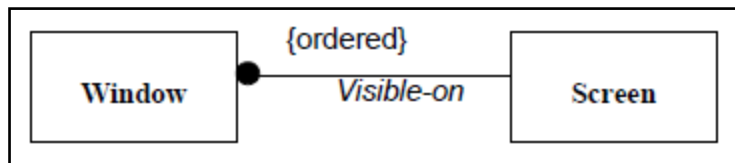
Usually the objects on the "many" side of an association have no explicit order, and can be regarded as a set. Sometimes the objects on the many side of an association have order. Writing {ordered} next to the multiplicity dot as shown in Figure indicates an ordered set of objects of an association.

Symbol and Example:



Figure

Consider the example of association between Window class and Screen class. A screen can contain a number of windows. Windows are explicitly ordered. Only topmost window is visible on the screen at any time. Figure shows this example.



Figure

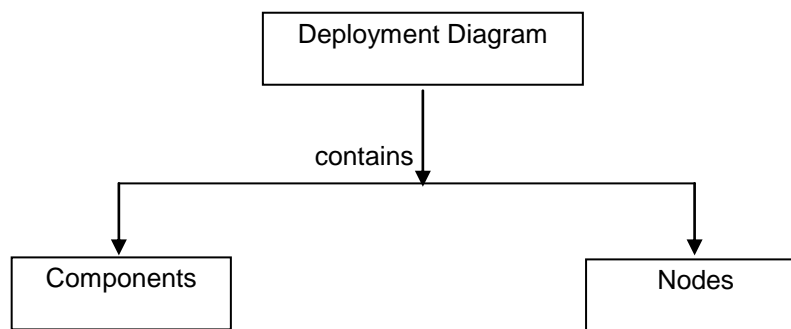


c) Draw the symbols used in deployment diagram and state use of it.

(Diagram of a Symbol -1 Marks each (2 Symbols)=, Use – 1 Marks each (2 Symbols)= 4 Marks)

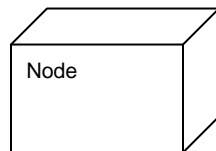
Basic Deployment Diagram Symbols and Notations

Deployment diagrams show the physical layout (locations) of the various hardware components (nodes) that compose a system, as well as the distribution of executable programs on this hardware. This diagram contains components and nodes, which represent processing or computational resources including computers, printers and so forth, (i.e. hardwares)

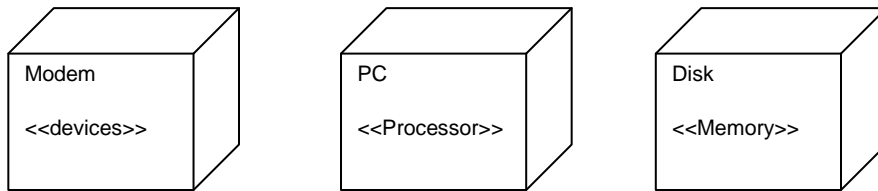


Use of Node.

Each hardware resource is represented by a cube, evoking the physical presence of the equipment within the system. Any system can be described by a small number of deployment diagrams, and a single diagram is often sufficient.

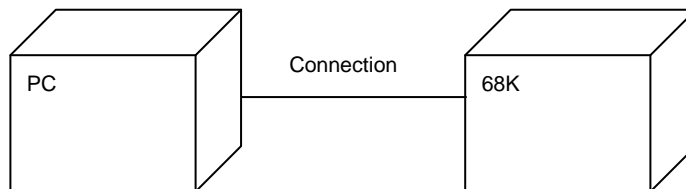


The nature of the equipment may be specified using a stereotype. The following example proposes three stereotypes to distinguish devices, processors, and memory. If necessary, the user has the possibility of defining other stereotypes.



Use of Association/Connection :

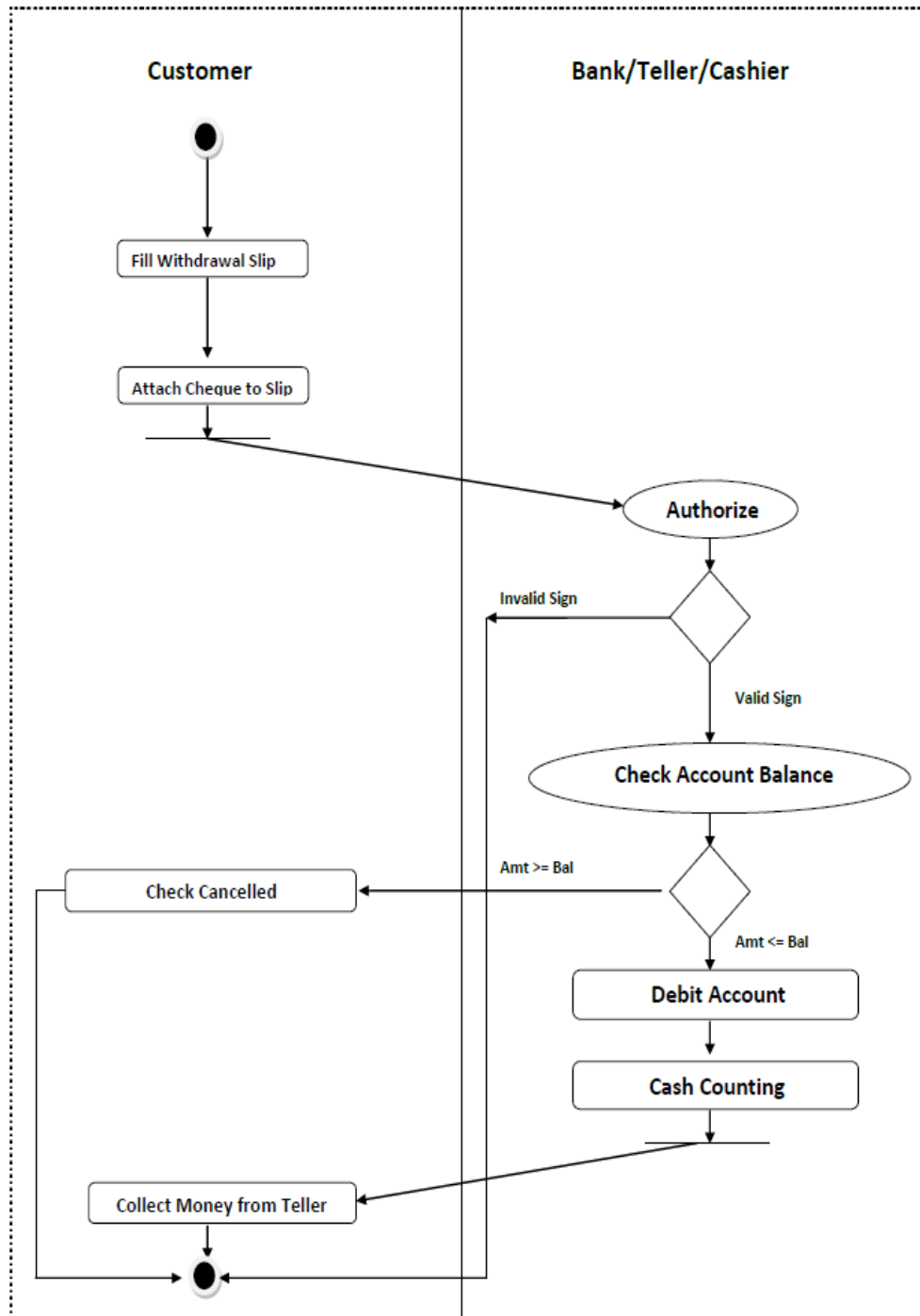
The various nodes that appear in the deployment diagram are connected to each other by lines representing a communication infrastructure, which is a priori bidirectional. The nature of that infrastructure may be specified using a stereotype.



Deployment diagrams may show node classes or node instances. As with other types of diagram, the graphical difference between classes and objects is implemented by underlining the object name.

d) Draw activity diagram for withdrawing money from bank through cheque.

(Relevant Diagram = 4 Marks)





e) **What is functional modeling? Give its significance.**

(Explanation - 3 Marks, Significance - 1 Mark = 4 Marks)

The functional model describes computations and specifies those aspects of the system concerned with transformations of values - functions, mappings, constraints, and functional dependencies.

The functional model captures what the system does, without regard to how or when it is done.

The functional model is represented graphically with multiple data flow diagrams, which show the flow of values from external inputs, through operations and internal data stores, to external outputs.

Following steps are performed in constructing a functional model:

1. Identify input and output values.
2. Build data flow diagrams showing functional dependencies.
3. Describe functions.
4. Identify constraints between objects.
5. Specify optimization criteria.

Significance :

- Function modeling is based on the concept of functions or processes, so they become the most important element in this approach
- The functional model describes computations within a system, i.e., what happens
- The functional model specifies the result of a computation without specifying how or when they are computed
- The functional model is represented graphically with multiple data flow diagrams