Tulsiramji Gaikwad-Patil College of Engineering and Technology
Wardha Road, Nagpur-441 108
NAAC Accredited

Department of Information Technology

Session 2018-2019 (Even Semester)

Fourth Semester **Subject: Object Oriented Methodology**

Unit – I

Syllabus

Introduction object-oriented development,Object Oriented Methodology, three Models, object oriented terms, object modeling Technique, object and classes links and associations, generalization and inheritance, grouping constructs a sample object module. Advanced object modeling; aggregation abstract classes,_multiple, inheritance, metadata, candidate keys.

Q1) What is Modeling? What does model serve? (7m)
Ans:

Modeling

1. A model is an abstraction of something for the purpose of understanding it before building it. Because, real systems that we want to study are generally very complex.
2. In order to understand the real system, we have to simplify the system. So a model is an abstraction that hides the non-essential characteristics of a system and highlights those characteristics, which are pertinent to understand it.
3. Efraim Turban describes a model as a simplified representation of reality.
4. A model provides a means for conceptualization and communication of ideas in a precise and unambiguous form.
5. The characteristics of simplification and representation are difficult to achieve in the real world, since they frequently contradict each other. Thus modeling enables us to cope with the complexity of a system.
6. Most modeling techniques used for analysis and design involve graphic languages.
7. Modeling is used frequently, during many of the phases of the software life cycle such as analysis, design and implementation. Modeling like any other object-oriented development, is an iterative process. As the model progresses from analysis to implementation, more detail is added to it.

Why do we model?

Before constructing anything, a designer first build a model. The main reasons for constructing models include:

- To test a physical entity before actually building it.
- To set the stage for communication between customers and developers.
- For visualization i.e. for finding alternative representations.
- For reduction of complexity in order to understand it.

Q2) What is meant by Object Orientation? Discuss various stages of OMT. (5M)

Ans:

Object Oriented Methodology is a methodology for object oriented development and a graphical notation for representing objects oriented concepts. We can call this methodology as OMT. The methodology has the following stages:

- System Analysis
- System Design
- Object Design
- Implementation

1. Analysis –
   - An analysis model is a concise, precise abstraction of what the desired system must do, not how it will be done.
   - It should not contain any implementation details. The objects in the model should be application domain concepts and not the computer implementation concepts.
2. System design –
   - The designer makes high level decisions about the overall architecture.
   - In system design, the target system is organized as various subsystems based on both the analysis structure and the proposed architecture.
3. Object design –
   - The designer builds a design model based on the analysis model but containing implementation details.
   - The focus of object design is the data structures and algorithms needed to implement each cycle.
4. Implementation –
   - The object classes and relationships developed during object design are finally translated into a particular programming language, database, or hardware implementation.
   - During implementation, it is important to follow good software engineering practice so that the system can remain the traceability, flexibility and extensibility.

Q3) Explain the following

1. **Object**
2. **Class**
3. **Links**
4. **Generalization**
5. **Metadata**
6. **Candidate Key (6M)**

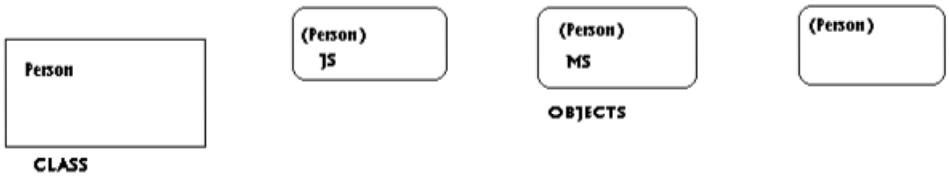
Ans:

1. **Object**

1. We can define an object as a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand.
2. All objects have identity and are distinguishable.eg:- Two apples can be described as 2 different objects even though they have the same descriptive properties.

2. **Classes**

1. An “object class” or “class” describes a group of objects with similar properties, common behavior, common relationships to other objects, and common semantics.
2. Eg: - Suppose you have a person class –you can term J S, M S, etc. to be objects of that class.

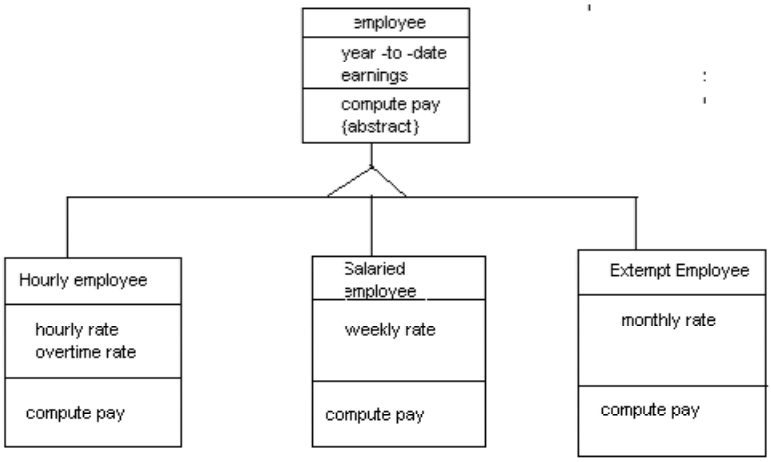


3. Link

1. A link is a physical or conceptual connection between object instances .We can say that a link is an instance of an association.
2. Eg: - J S works for Simplex company.

4. Generalization

1. Generalization is the relationship between a class and one or more refined versions of it. The class being refined is called the "superclass' and each refined version are called a "subclass".
2. For example, Equipment is the superclass of Pump and tank. Attributes and operations that are common to a group of subclasses are attached to the superclass and shared by each subclass.



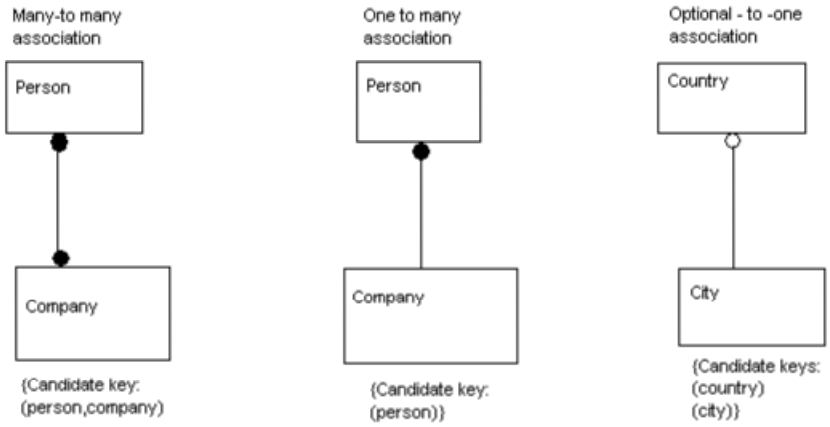
3. Generalization is sometimes called a „is-a" relationship. Each instance of a subclass is an instance of the superclass. The notation for generalization is a triangle connecting a superclass to subclass.

5. Metadata

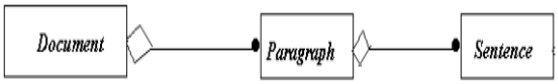
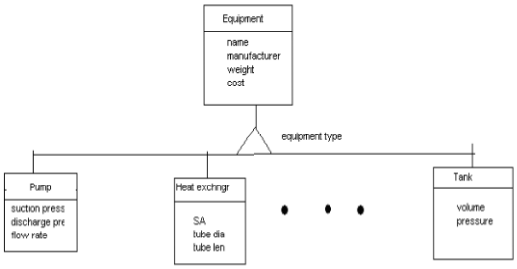
1. Metadata is data that describes other data.
2. For example, the definition of a class is metadata. Models are inherently metadata, since they describe the things being modeled.

6. **Candidate Keys**

1. It is a minimal set of attributes that uniquely identifies an object or link. It means you can"t discard an attribute from the candidate key and still distinguish all objects and links.
2. 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.
3. One or more combinations of related objects are candidate keys for associations.
4. A candidate key is delimited with braces in an object model.

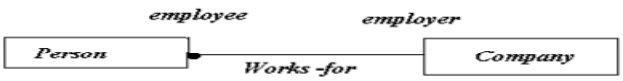


Q4) Differentiate between aggregation and generalization. (4m)
Ans:

Aggregation	Generalization
Aggregation is the "part-whole" or "a-part-of" relationship	**Generalization** is "is-a" relationship.
In this objects representing the components of something are associated with an object representing the entire assembly.	Generalization is the relationship between a class and one or more refined versions of it.
Aggregations are drawn like associations, except a small hollow diamond indicating the assembly end of the relationship	The notation for generalization is a triangle connecting a superclass to subclass.
	

Q5) Explain Rolename and qualifier. (4m)
Ans:
Rolename

1. A role is an end of an association. Role name is a name that uniquely identifies one end of an association.
2. Eg: - "Person works for a Company." Here the person plays the role of an employee and the company the role of the employer.

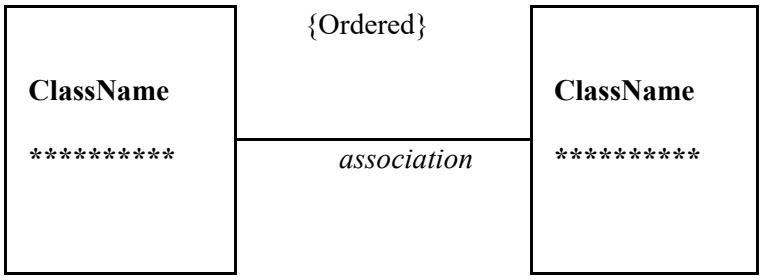


Qualification

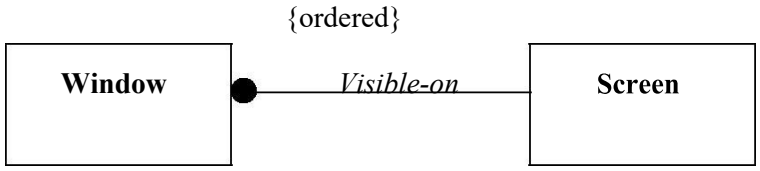
1. Qualifier is a special attribute that reduces the effective multiplicity of an association. It reduces the effective multiplicity of an association. We can qualify a one to many and many to many association.
2. For eg: - In the below figure a directory has many files .A file may only belong to a single directory. Within the context of a directory, the file name specifies a unique file .A file corresponds to a directory and a file name

Ordering

1. Usually the objects on the "many" side of an association have no explicit order, and can be regarded as a set.
2. Some times the objects on the many side of an association have order. Writing {ordered} next to the multiplicity dot as shown in Figure 2.7 indicates an ordered set of objects of an association.



3. Consider the example of association between Window class and Screen class. A screen can contain a number of windows. Windows are explicitly ordered.
4. Only topmost window is visible on the screen at any time. Figure 2.8 shows this example.



Q6) What are abstract class, concrete class? Explain the object model for concrete ad abstract class. (5m)
Ans:
Abstract Classes-

1. An abstract class is a class that has no direct instances but whose descendent classes have direct instances.
2. A concrete class is a class that is an instantiable; that is, it can have direct instances.
3. A concrete class may be leaf classes in the inheritance tree; only concrete classes may be leaf classes in the inheritance tree. Now to get a clear idea about look at the below figure.

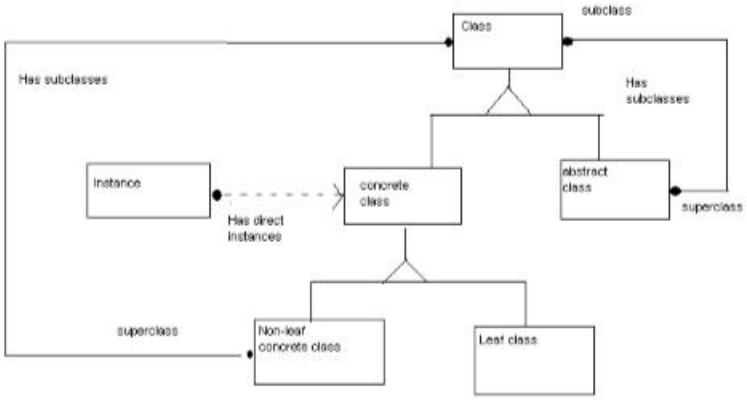


Fig.4.5 Object model defining abstract and concrete class

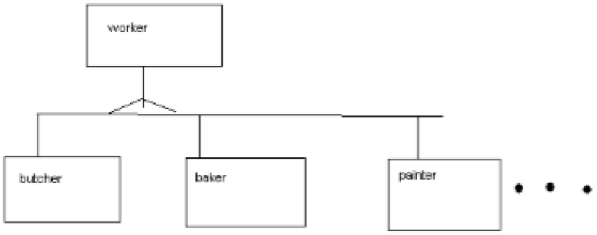
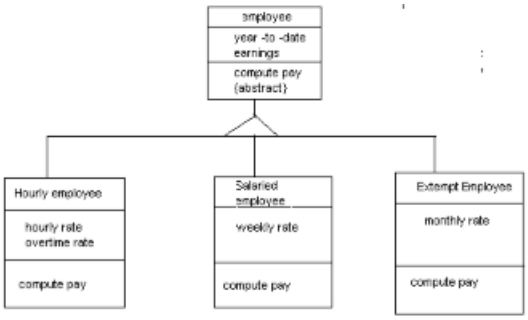


Fig 4.6 Concrete classes

4. All the classes are concrete classes**.**



Q7) Discuss various kinds of models of object modelling techniques. (6M)
Ans:

The OMT methodology uses three kinds of models

- Object Model-describes the static structure of the objects in a system and their relationships. This model mainly contains object diagrams.
- Dynamic Model-describes the aspects of a system that change over time. This model mainly contains state diagrams.
- Functional Model-describes the data value transformations within a system. This model contains the data flow diagrams.

Q8) Explain Aggregation in detail. (6M)
Ans:

1. Aggregation is another relationship between classes. It is a tightly coupled form of association with some

extra semantics.

2. It is the "part-whole" or "a-part-of" relationship in which objects representing the component of something are associated with an object representing the entire assembly.
3. Aggregations are drawn like associations, except a small hollow diamond indicating the assembly end of the relationship as shown in Figure 2.11. The class opposite to the diamond side is part of the class on the diamond side.

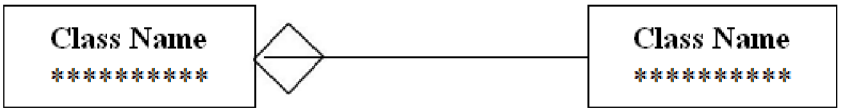


Figure 2.11

4. For example, a team is aggregation of players. This can be modeled as shown in Figure 2.12.

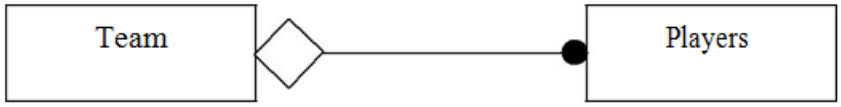
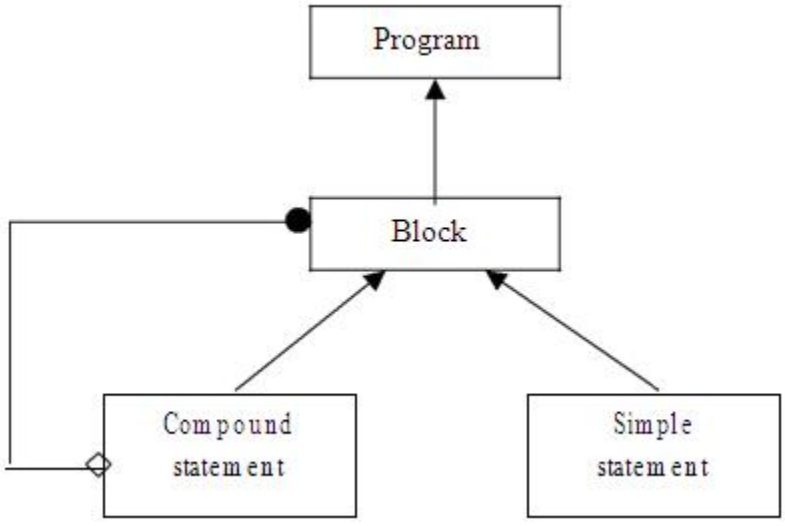
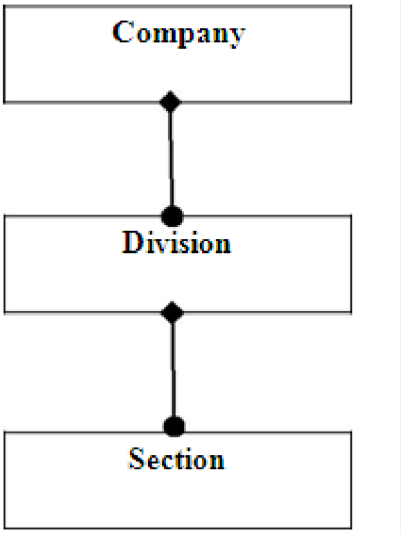


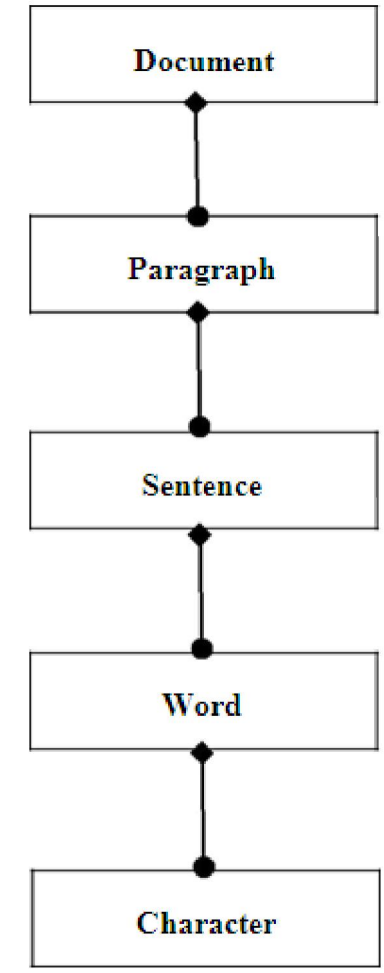
Figure 2.12

5. Aggregation can be fixed, variable or recursive.
6. In a fixed aggregation number and subtypes are fixed i.e. predefined.
7. In a variable aggregation number of parts may vary but number of levels is finite.
8. A recursive aggregate contains, directly or indirectly, an instance of the same aggregate. The number of levels is unlimited. For example, as shown in Figure 2.13, a computer program is an aggregation of blocks, with optionally recursive compound statements. The recursion terminates with simple statement. Blocks can be nested to arbitrary depth.



5. One more example of aggregation is shown in Figure 2.14. A company is composed of zero, one or more divisions. A division is composed of zero, one or more sections.





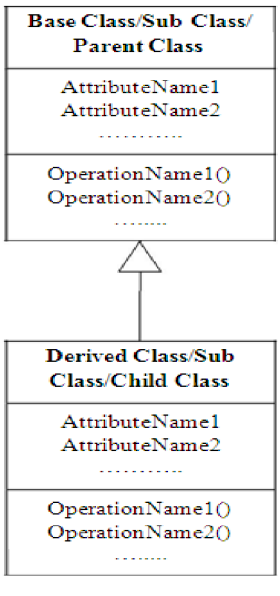
6. Another example of aggregation is shown in Figure 2.15. A document is composed of zero, one or more paragraphs. A paragraph is composed of zero, one or more sentences. A sentence is composed of one or more words.

7. A word is composed of one or more characters.

Q9) Explain inheritance in detail. (6M)

Ans:

1. The inheritance concept was invent ted in 1967 for Simula. Inheritance is a way to form new classes using classes that have already been defined.
2. Inheritance is intended to help reuse existing code with little or no modification.
3. The new classes, known as derived classes (or child classes or sub classes), inherit attributes and behavior of the pre-existing classes, which are referred to as base classes (or parent classes or super classes) as shown in Figure 2.16.
4. The inheritance relationship of sub- and super classes gives rise to a hierarchy.



5. Inheritance is a "is-a" relationship between two classes. For example, Student is a Person; Chair is Furniture; Parrot is a Bird etc. in all these examples, first class (i.e. Student, Chair, Parrot) inherits properties from the second class (i.e. Person, Furniture, Bird).

6. Example of an inheritance: Manager is an Employee. Manager class inherits features from Employee class as shown in Figure 2.17.

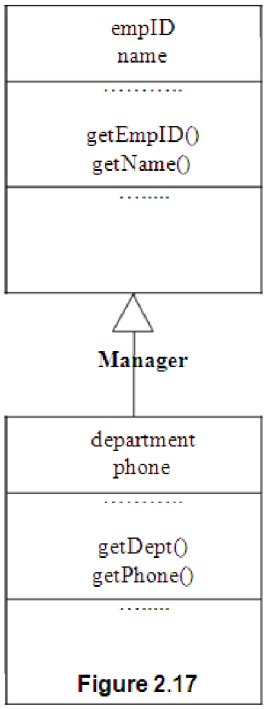


Figure 2.17

Q10) Explain Inheritance for Specialization detail. (6M)

Ans:

Inheritance for Specialization

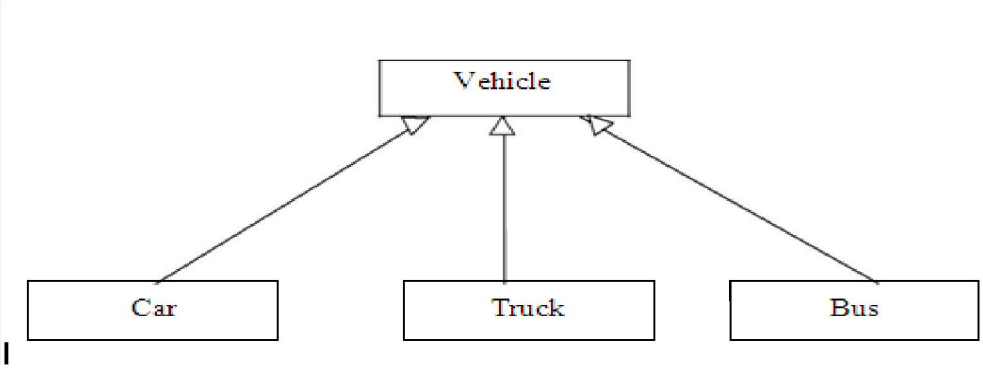
1. One common reason to use inheritance is to create specializations of existing classes. In specialization, the derived class has data or behavior aspects that are not part of the base class. For example, Square is a Rectangle. Square class is specialization of Rectangle class. Similarly, Circle is an Ellipse.

2. Herealso, Circle class is specialization of Ellipse class. Another example, a BankAccount class might have data members such as accountNumber, customerName and balance. An InterestBearingAccount class might inheritBankAccount and then add data member interestRate and interestAccrued along with behavior for calculating interest earned.

3. Another form of specialization occurs when a base class specifies that it has a particular behavior but does not actually implement the behavior. Each non-abstract, concrete class which inherits from that abstract class must provide an implementation of that behavior. This providing of actual behavior by a subclass is sometimes known as implementation or reification.

4. For example, there is a class Shape having operation area(). The operation area() cannot be implemented unless we have concrete class. So, Shape class is abstract class. Rectangle is a Shape. Now, Rectangle is a concrete class, which can implement the operation area().

Q11) Explain Inheritance for Generalization in detail. 6M **Ans:**



1. Generalization is reverse of specialization. For instance, a "fruit" is a generalization of "apple", "orange", "mango" and many others. One can consider fruit to be an abstraction of apple, orange, etc.

2. Conversely, since apples are fruit (i.e., an apple is-a fruit), apples may naturally inherit all the properties common to all fruit, such as being a fleshy container for the seed of a plant.

3. Another example: Vehicle is a generalization of Car, Truck, Bus etc. as shown in Figure 2.18. Car, Truck, Bus etc. share some properties such as “number of wheels”, speed, capacity etc. these common properties are abstracted out and put into another class say Vehicle, which comes higher in the hierarchy.

Q12) Explain types of inheritance in detail. (6M)

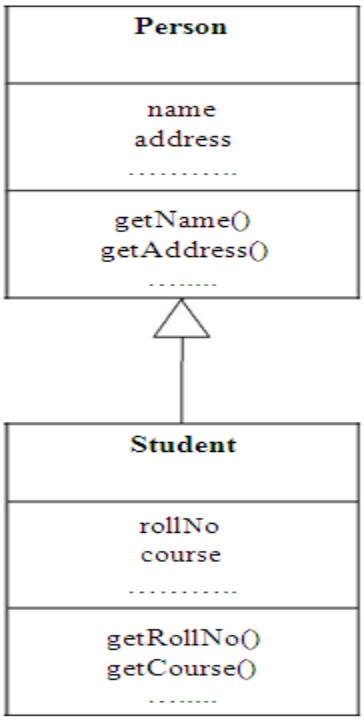
Ans:

There are many ways a derived class inherits properties from the base class.

Following are the types of inheritance:

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance
- Multipath Inheritance
- Hybrid Inheritance

Single Inheritance

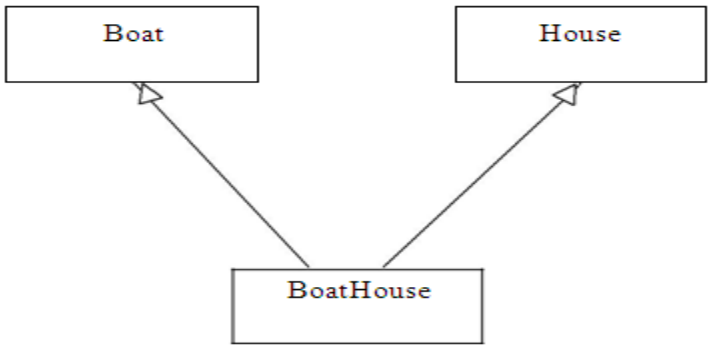


When a (derived) class inherits properties (data and operations) from a single base class, it is called as single inheritance. For example, Student class inherits properties from Person class as shown in Figure 2.20.

Multiple Inheritance

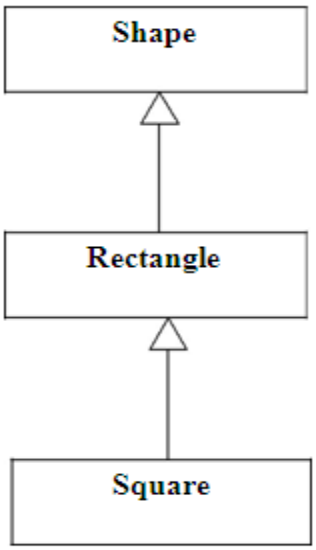
When a (derived) class inherits properties (data and operations) from more than one base class, it is called as multiple inheritance. For example,

BoatHouse class inherits properties from both Boat class and House class as shown in Figure 2.21.



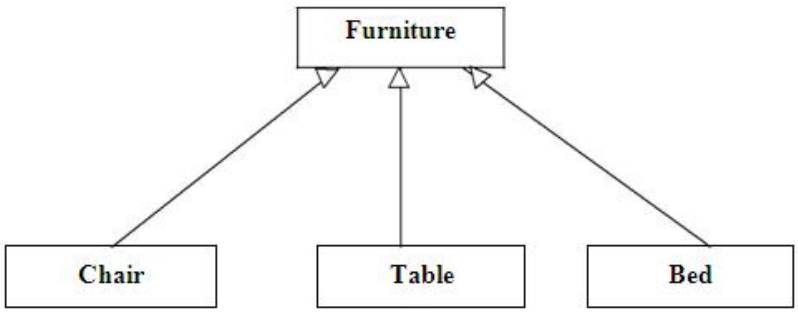
Multilevel Inheritance

When a (derived) class inherits properties (data and operations) from another derived class, it is called as multilevel inheritance. For example, Rectangle class inherits properties from Shape class and Square inherits properties from Rectangle class as shown in Figure.



Hierarchical Inheritance

When more than one (derived) class inherits properties (data and operations) from a single base class, it is called as hierarchical inheritance. For example, Chair class, Table class and Bed class all inherit properties from Furniture class as shown in Figure 2.23.



Multipath Inheritance

When more than one inheritance paths are available between two classes in the inheritance hierarchy, it is called as multipath inheritance. For example, Carnivorous and Herbivorous class inherit properties from Animal class.

Omnivorous class inherits properties from Carnivorous and Herbivorous classes. So, there are two alternative paths available from Animal class to Omnivorous class as shown in Figure 2.24.

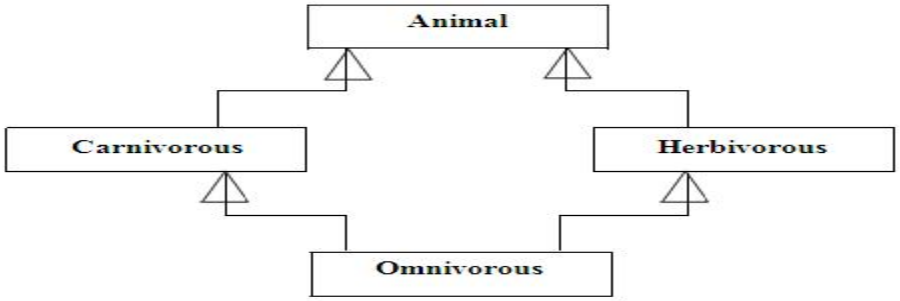
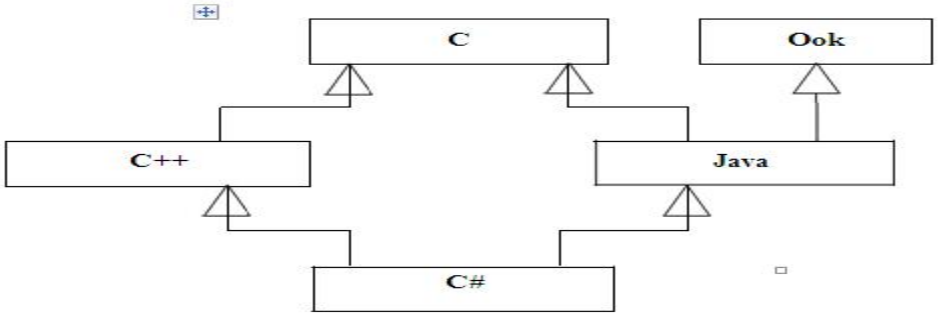


Figure 2.24



Hybrid Inheritance

Mixture of single, multiple, hierarchical and multilevel inheritance forms hybrid inheritance as shown in Figure 2.25.

Q13) Explain difference between module and sheet. (6M)

Ans:

Grouping Constructs

There are two grouping constructs: module and sheet.

Module is logical construct for grouping classes, associations and generalizations. An object model consists of one or more modules. The module name is usually listed at the top of each sheet.

A sheet is a single printed page. Sheet is the mechanism for breaking a large object model into a series of pages. Each module is contained in one or more sheets. Sheet numbers or sheet names inside circle contiguous to a class box indicate other sheets that refer to a class.

Q14) Explain Object Oriented Methodologies. (6M)

Ans:

1. Object Oriented Methodology (OOM) is a new system development approach encouraging and facilitating reuse of software components. With this methodology, a computer system can be developed on a component basis, which enables the effective reuse of existing components and facilitates the sharing of its components by other systems.
2. By using OOM, higher productivity, lower maintenance cost and better quality can be achieved.
3. OOM requires that object-oriented techniques be used during the analysis, design and implementation of the system. This methodology makes the analyst to determine what the objects of the system are, how they behave over time or in response to events, and what responsibilities and relationships an object has to other objects. Object-oriented analysis has the analyst look at all the objects in a system, their commonalties, difference, and how the system needs to manipulate the objects.
4. During design, overall architecture of the system is described. During implementation phase, the class objects and the interrelationships of these classes are translated and actually coded using the programming language. The databases are created and the complete system is made operational.

Q15) Explain Advantages of Object Oriented Methodology. (6M)

Ans:

Advantages of Object Oriented Methodology

- As compared to the conventional system development techniques, OOM provides many benefits.
- The systems designed using OOM are closer to the real world as the real world functioning of the system is directly mapped into the system designed using OOM. Because of this, it becomes easier to produce and understand designs.

- The objects in the system are immune to requirement changes because of data hiding and encapsulation features of object-orientation. Here, encapsulation we mean a technique that allows the programmer to hide the internal functioning of the objects from the users of the objects. Encapsulation separates the internal functioning of the object from the external functioning thus providing the user flexibility to change the external behavior of the object making the programmer code safe against the changes made by the user.

- OOM designs encourage more reusability. The classes once defined can easily be used by other applications. This is achieved by defining classes and putting them into a library of classes where all the classes are maintained for future use. Whenever a new class is needed the programmer first looks into the library of classes and if it is available, it can be used as it is or with some modification. This reduces the development cost & time and increases quality.
- Another way of reusability is provided by the inheritance feature of the object-orientation. The concept of inheritance allows the programmer to use the existing classes in new applications i.e. by making small additions to the existing classes can quickly create new classes. This provides all the benefits of reusability discussed in the previous point.

- As the programmer has to spend less time and effort so he can utilize saved time (due to the reusability feature of the OOM) in concentrating on other aspects of the system.

Tulsiramji Gaikwad-Patil College of Engineering and Technology
Wardha Road, Nagpur-441 108
NAAC Accredited

Department of Information Technology

Session 2018-2019 (Even Semester)

Fourth Semester **Subject: Object Oriented Methodology**

Unit – II

Syllabus

Dynamic modeling, events and states, nested state diagrams, concurrency, advanced dynamic modeling concepts, functional models, data flow diagram, constraints, a sample functional module

Q1) What is dynamic modeling? Explain in detail. (4m)

Ans:

1. Temporal relationships are difficult to understand .A system can be best understood by first examining its staitic behavior.

2. Then next we examine the changes to the objects and their relationships over time. Those aspects of a system that are concerned with time and changes are the dynamic model."

3. Control is that aspect of a system that describes the sequences of operations that occur in response to external stimuli without consideration of what the operations do , what they operate on, or how they are implemented.

4. The major dynamic modeling concepts are:

- events, states
- state diagrams

Q2) Differentiate between states and event. (4m)

Ans:

Event	States
"An event is something that happens at appoint in time."	A state is an abstraction of the attribute values and links of an object.
An event has no duration	A state has duration.
an event separates two states	A State separates 2 events.
Eg:-Flight 123 departs from Chicago.	For example, Water is liquid when temperature of water is greater than zero degrees Celsius and below 100 degree Celsius.
Events represent points in time	states represent intervals of time.
Past events are eventually hidden by subsequent events.	The state of an object depends on the past sequence of events it has received

Q3) Explain scenerio event trace diagram for ATM. (6m)

Ans:

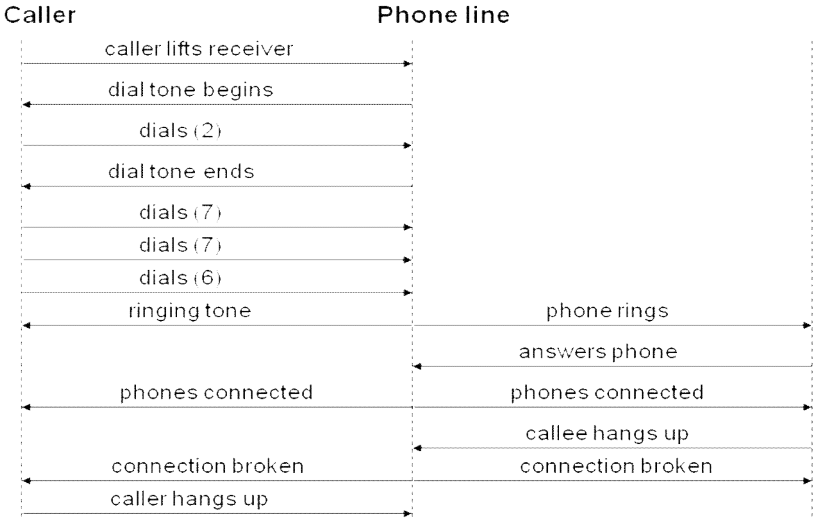
Scenerio

1. A scenario is a sequence of events that occurs during one particular execution of system. The scope of a scenario can vary.
2. A scenario can be thought of as a historical record of executing a system or a thought expression of executing a proposed system.

caller lifts receiver
dial tone begins
caller dials digit(5)
dial tone ends
caller dials digit (5)
caller dials digit (5)
caller dials digit (1)
caller dials digit (2)
caller dials digit (3)
caller dials digit (4)
called phone begins ringing
ringing tone appears in calling phone
called party answers
called phone stops ringing
ringing tone disappears in calling phone
phones are connected
called party hangs up
phones are disconnected
caller hangs up

Event trace diagram

1. The next step after writing a scenario is to identify the sender and receiver objects of each event.
2. "The sequence of events and the objects exchanging events can both be shown in an augmented scenario called event trace diagram."
3. The diagram shows each object as a vertical line and each event as a horizontal arrow from the sender object to the receiver object. Time increases from top to bottom.

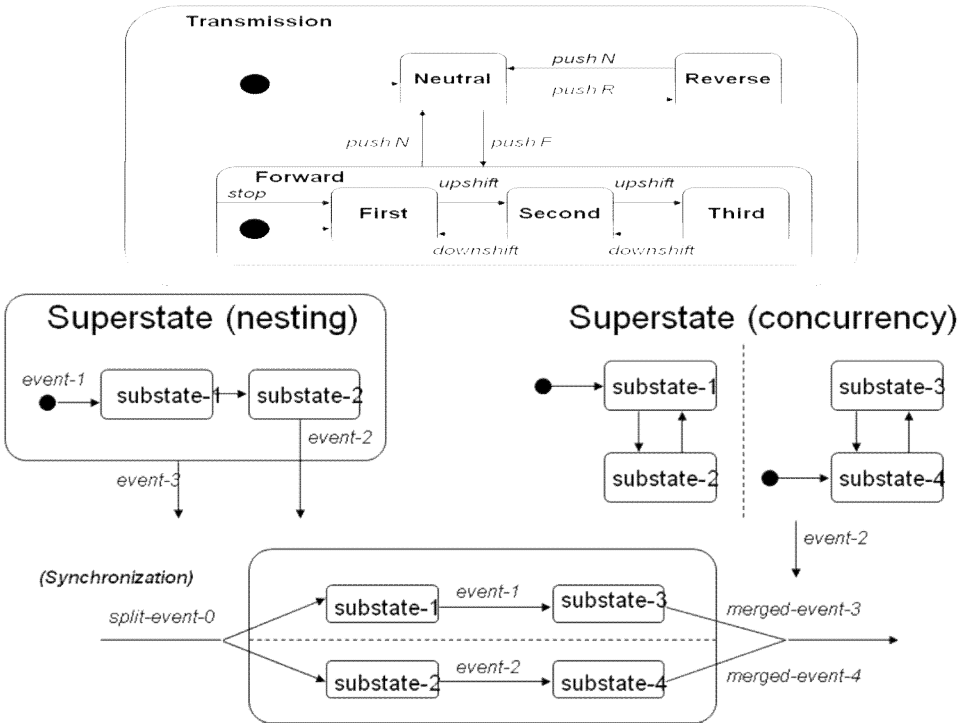


Q4) Explain in shot event generalization with suitable example. (3m)

Ans:

State generalization

1. A nested state diagram is actually a form of generalization on states.
2. The states in the nested diagrams are all refinements of the state in the high level diagrams, but in general the states in a nested state diagram may interact with other states.
3. States may have substates that inherit the transitions of superstates, unless overridden.



d) Event Generalization

Events can be organized into generalization hierarchy with inheritance of event attributes.

Q5) Explain the following terms in brief:-

- **Processes**
- **Dataflows**
- **Actors**
- **Data stores**
- **EtryExit** **(6m)**

Ans:

1. Processes
    1. A process transforms data values. The lowest level processes are pure functions without side effects.
    2. An entire data flow graph is a high level process. A process may have side effects if it contains non-functional components. A process is drawn as an ellipse.

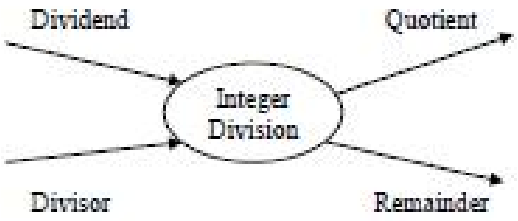


Fig: process

2. Data Flows
    1. A data flow connects the output of an object or process to the input of another object or process.
    2. It represents an intermediate data value within a computation. A data flow is drawn as an arrow.
    3. Sometimes an aggregate data value is split into its components, each of which goes to a different process. Flows on the boundary of a data flow diagram are its inputs and outputs.

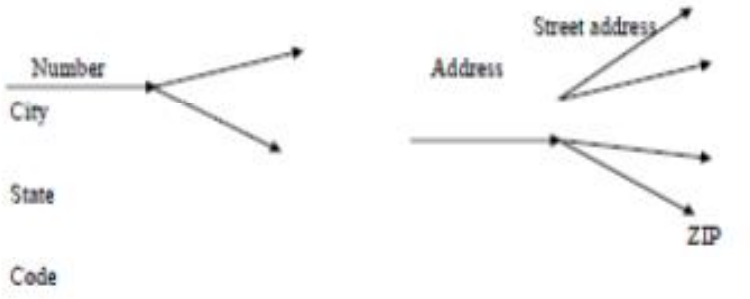
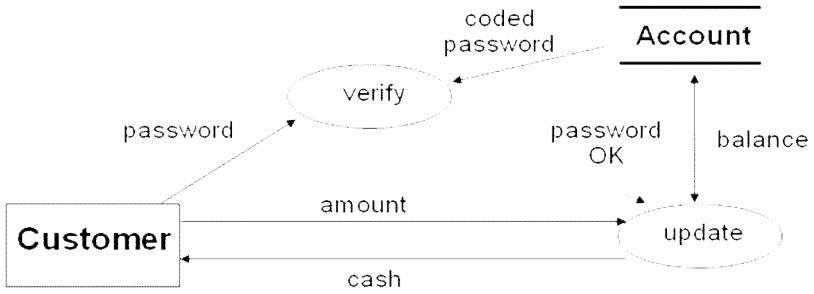


Fig: Data flows to copy a value and split an aggregate value

3. Actors
    1. An actor is an active object that drives the data flow graph by producing or consuming values.
    2. Actors are attached to the inputs and outputs of a data flow graph. The actors are lie on the boundary of the data flow graph. An actor is drawn as a rectangle.

4. Data Stores
    1. A data store is a passive object within a data flow diagram that stores data for later access.
    2. A data store is drawn as a pair of parallel lines containing the name of the store. Input arrows indicate information or operations that modify the stored data.
    3. Output arrows indicate information retrieved from the store. Both actors and data stores are objects.

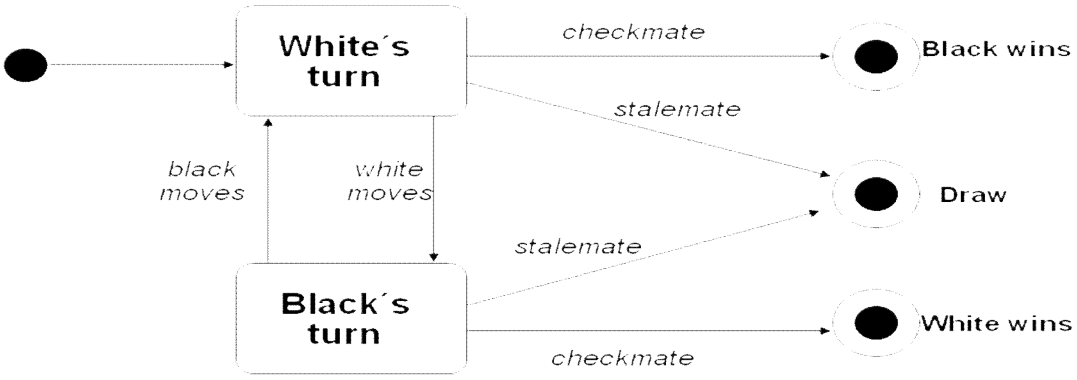


4. Entry and Exit Actions

1. As an alternative to showing actions on transitions, actions can be associated with entering or exiting a state. Exit actions are less common.
2. Of all the action on the incoming transition will be executed first, then the entry action, followed by the activity within the state, followed by the exit action and finally the action on the outgoing action.
3. The activities can be interrupted by events causing transitions out of state but entry and exit can"t be interrupted. The entry/exit actions are useful in state diagrams .A state can be expressed in terms of matched entry-exit actions.

Q6) Explain one shot diagram for a simple chess game. 3m

Ans:



Q7) Explain and draw entry and exit actions for opening and closing the door. (4m)

Ans:

Entry Point: Sometimes you won't want to enter a sub-machine at the normal initial state. For example, in the sub-machine shown in Figure 3.10, it would be normal to begin in the "Initializing" state, but if for some reason it wasn't necessary to perform the initialization, it would be possible to begin in the "Ready" state by transitioning to the named entry point.

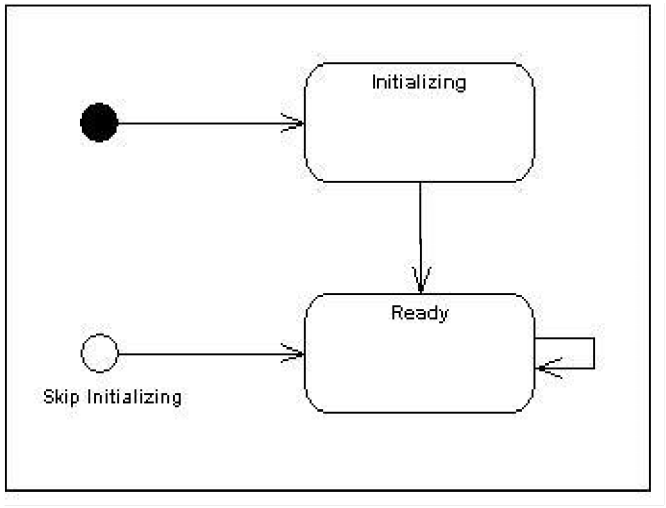


Figure 3.10

Exit Point: In a similar manner to entry points, it is possible to have named alternative exit points. The Figure 3.11 gives an example where the state executed after the main processing state depends on which route is used to transition out of the state.

Q7) Explain events and states in detail. (8m)

Ans:

Events and States:

1. "The attribute values and links held by an object are called its state. Over time, the objects stimulate each other, resulting in a series of changes to their states." **"**An individual stimulus from one object to another is an event."
2. The response of an event depends on the state of the object receiving it, and can include a change of state or the sending of another event to the original sender or to a third object.
3. "The pattern of events, states and state transitions for a given class can be abstracted and represented as a state diagram."
4. The dynamic model consists of multiple state diagrams, one state diagram for each class with important dynamic behavior, and shows the pattern of activity for an entire system. The state diagrams for the various classes combine into a single dynamic model via shared events.
5. "An event is something that happens at appoint in time." Eg:-Flight 123 departs from Chicago.
6. An event has no duration. One event may logically precede or follow another, or the two events may be unrelated.Eg:-Flight 123 must depart Chicago before it can arrive in San Francisco; the two events are casually related.

7. “Two events that are casually unrelated are said to be concurrent; they have no effect on each other.” Concurrent events can occur in any order.
8. An event is a one way transmission of information from one object to another. An object sending an event to another object may expect a reply, but the reply is a separate event under the control of the second object, which may or may not choose to send it.
9. Every event is a unique occurrence but we group them into event classes and give each event class a name to indicate common structure and behavior.
10. Eg:-Flight 123 departs from Chicago. Flight 456 departs from Rome.
11. Both are instances of the event class airplane flight departs. Some events are signals, but most event classes have attributes indicating the information they convey.eg:-airplane flight departs has attributes airline, flight number and city.
12. The time at which each event occurs is an implicit attribute of an event. An event conveys information from one object to another. Some classes of events may be simply signals that something has occurred, while other classes of events convey data values. The data values conveyed by events are its attributes, like data values held by its objects. Attributes are shown in the parentheses after the event class name.

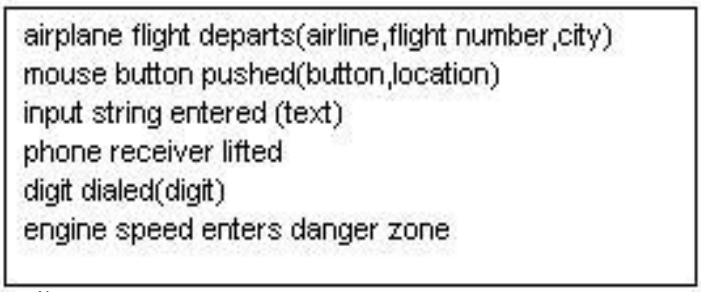


Fig.2.1.1 Event classes and attributes

13. Events include error conditions as well as normal occurrences. eg:-motor jammed.

State

1. A state is an abstraction of the attribute values and links of an object. The set of values are grouped together into a state according to properties that affect the gross behavior of the object.

2. For example, let us consider the state of a bank. It can either in a solvent or an insolvent state depending on whether it assets exceeds its liabilities.

3. A state specifies the response of the object to input events. The response to an event may vary quantitatively depending on the exact value of its attributes .The response is the same for values within the same states but different for values in different states.

4. The response of an object to an event may include an action or a change of state by the object.

5. Eg:-if a digit is dialed in state dial tone the phone line drops the dial tone and enters dialing.

6. A state corresponds to the interval between two events received by an object. Events represent points in time; states represent intervals of time.

7. For example, in the phone example, after the receiver is lifted and before the first digit is dialed, the phone line is in Dial Tone state.

8. The state of an object depends on the past sequence of events it has received, but in
9. most cases past events are eventually hidden by subsequent events. For example, Events that has happened before the phone is hung up have no effect on future behavior.

10. A state has duration. A state is often associated with a continuous activity .eg:-ringing of phone. Events and states are duals i.e. an event separates two states and a state separates 2 events. A state is often

associated with the value of an object satisfying some condition. For example, Water is liquid when temperature of water is greater than zero degrees Celsius and below 100 degree Celsius.

11. Both events and states depend on the level of abstraction used. For example, travel agent planning an itinerary would treat each segment as a single event, but a flight status board will treat it as 2 events departure and arrival.

12. A state can be characterized in various ways.

Q8) Write short note on

1. Controlling Operations

2. **Conditions** **(8m)**

Ans:

Conditions

1. "A condition is a Boolean function of object values. For example, when we say the temperature was below freezing point from Nov to March.

2. Here the condition to be temperature below freezing point .It has duration. A state can be defined in terms of a condition; conversely being in a state is a condition.

3. Conditions can be used as guards on transitions "A guarded transition fires when its events occur but only if the condition is true.

4. For example, let us say a person goes out in the morning (event), if the temperature is below freezing (condition) , then put on your gloves(next state).

5. A guarded condition on a transition is shown as a Boolean expression in brackets following event name.

6. Now that we have learnt about states, events, etc. Let us go into the advanced concepts. Let us first discuss on the operations

Controlling Operations

1. State diagrams could be of little use if they just described the patterns of the events. A behavior description of an object must specify what the object does in response to events .Operations attached to states / transitions are performed in response to corresponding states or events.
2. "An ***activity*** is an operation that takes time to complete."
3. An activity is associated with a state.
4. Activities include continuous as well as sequential events.
5. A state may control a continuous activity that persists until an event terminates it by calling a transition from the state.

- do: activity within state box indicates that activity starts an entry to the state and stops on exit. In the case of sequential events also it is the same thing.
- If an event causes a transition from the state before the activity is complete, it terminates prematurely.

6. "An ***action*** is an instantaneous operation." it is associated with an event
7. Actions can also represent internal control operations, such as setting attributes or generating other events.
8. Notation („/" on transition) for action and the name (or description of the action following the name of the event that causes it.

Q8) Write short note on concurrency. (6m)
Ans:

Concurrency

a) Aggregation Concurrency

A dynamic model describes a set of concurrent objects. A state of an entire system can"t be represented by a single state in a single object. It is the product of the states of all objects in it. A state diagram for an assembly is a collection of state diagram, one for each component. By using aggregation we can represent concurrency. Aggregate state corresponds to combined state of all component diagrams.

b) Concurrency within an object

Concurrency within a single composite state of an object is shown by partitioning the composite state into sub diagrams with dotted lines.

Q9) Explain Advanced dynamic modeling concepts. (8m)
Ans:

Advanced dynamic modeling concepts.

i) Entry and Exit Actions

1. As an alternative to showing actions on transitions, actions can be associated with entering or exiting a state.
2. Exit actions are less common. What about the order of execution of various actions and events associated with states and events?

3. Of all the action on the incoming transition will be executed first, then the entry action, followed by the activity within the state, followed by the exit action and finally the action on the outgoing action.
4. The activities can be interrupted by events causing transitions out of state but entry and exit can"t be interrupted. The entry/exit actions are useful in state diagrams.
5. A state can be expressed in terms of matched entry-exit actions. ii)

Internal Actions

1. An event can cause an action to be performed without causing a state change .

2. The event name is written inside state box followed by „/" . Now to differentiate between a self-transition and an internal action. Self-transition causes an entry and exit action to be executed but an internal action does not.

Automatic Transition

1. A state performs a sequential activity. When it is over, the transition to another fires. An arrow without an event name indicates an automatic transition that fires when

2. the activity associates with the source state is completed.

3. If there is no activity, the unlabeled transitions fires as soon as the state is entered. Such transitions are called as „*lambda*" transitions.

iv) Sending Events

1. An object can perform action of sending an event to another object. A system interacts by events. An event can be directed to a single or a group of objects. Any and all objects can accept it concurrently.

2. "If a state can accept events from more than one object, the order in which concurrent events are received may affect final state. This is called a race condition. Unwanted race conditions should be avoided.

v) Synchronization of Concurrent Activities

Sometimes an object must perform two or more activities concurrently. Both activities must be completed before the object can progress to its next state. The target state occurs after both events happen in any order.

Q10) Explain functional modelling in detail. (8m)

Ans:

1. The functional model specifies the results of a computation without specifying how or when they are computed. That means it specifies the meaning of the operations in the object model and the actions in the dynamic model.
2. Spreadsheet is a kind of functional model. The purpose of the spreadsheet is to specify values in terms of other values.
3. The functional model consists of multiple data flow diagrams. A Data Flow Diagram (DFD) shows the functional relationships of the values computed by a system, including the input values, output values, and internal data stores. Or we can say that a DFD is a graph showing the flow of data values from their sources in objects through processes that transform them to their destinations in other objects.
4. A DFD contains **processes** that transform data, **data flows** that move data, **actor** objects that produce and consume data, and **data store** objects that store data passively.

Q11) Explain Relation of Functional to Object and Dynamic Models. (4m)

Ans:

Relation of Functional to Object and Dynamic Models.

The functional model shows what has to be done by a system. The leaf processes are the operations on objects. The processes in the functional model correspond to operations in the object model Processes in the functional model show objects that are related by function.A process is usually implemented as a method.