**Module-1**

**Chapter-1**

**Introduction**

* 1. **Object**

- Object is a combination of data structure and behavior in a single entity eg: gray bicycle

- The characteristics of object are:

* **Identity:** Identity means data quantized into discrete distinguishable objects. Each object has its own inherent identity. Eg: paragraph in a document, white queen in a chess
* **Classification:** Object with same attributes and operation are grouped into a class. Class has attributes and operations. Eg: paragraph, chess piece
* **Polymorphism:** Same operation behaves differently in different classes. Eg: Consider move operation. It is different in window and chess piece classes
* **Inheritance:** Sharing of attributes and operations among classes based on hierarchical relationship. A class is divided into subclass. Subclass inherits the properties of superclass and add its own unique properties. Eg: Scrolling window and fixed window are subclass of window and inherits the properties of window such as visible region . Scrolling window adds scroll bar and offset

**1.2 Object oriented development**

- Development means describing the life cycle of software: analysis, design and implementation. Object oriented development means identification and organization of application domain concepts

**1.2.1 Modeling Concepts**

- Object oriented design helps the software developer to work and think in terms of application domain

- Object oriented design is a conceptual process independent of programming language until the final stage

**1.2.2 Object oriented methodology**

- Methodology means building a model of application domain and adding implementation details during the design of a system and the approach is called Object Modeling Technique(OMT)

- The stages of methodology are:-

* **Analysis**:- In analysis the analyst builds a model ie, the analyst tries to understand the problem. In analysis we focus on what the system must do and not how it is done. It contains only application domain concepts and not implementation concepts
* **System design**:- The system designer makes decision on the overall architecture. The target system is divided into subsystems based on analysis structure and proposed architecture. The designer decides what performance characteristics are to optimized
* **Object design**:- The object designer makes a design model based on analysis model and also contains implementation details. The designer adds details to design model based on strategy formed in system design. Object design focus on the algorithm to implement the class
* **Implementation**:- The classes and relationships developed during object design are translated into programming language

**1.2.3 Three models**

- OMT methodology has 3 models:

* **Object model**:- Described the static structure of an object and its relationship. The object model contains object diagram. Object diagram is a graph whose node represents object classes and arc represents its relationships

object

object

**Fig 1.1 Object Diagram**

Relationship

* **Dynamic model**:- It is related with time and describes the aspects of system. It contains state diagram. State diagram is a graph whose node represent state and arc represent transitions

transition

Fig 1.2 State Diagram

* **Functional model**:- It represents functional relationship and describes data transformations. It contains data flow diagram. It is a graph whose node represents process and arc represents data flow

Fig 1.3 Data Flow Diagram

Data flow

**1.2.4 Object oriented themes**

- **Abstraction**:- Means act of representing essential features without including background details. It focus on what an object is and not how it will be implemented

- **Encapsulation**:- Wrapping up of data and functions into a single unit. It hides the internal implementation details from the user

- **Combined data and behavior**:- Same as polymorphism. It has the ability to take more than one form

**- Sharing**:- Object oriented techniques promote sharing at different levels. Same as inheritance. Superclass share its features with subclass. Advantage: Reusability

- **Emphasis on object structure not procedure structure**:- It means focusing on what an object is and not how it is implemented

- **Synergy**:- Characteristics of object such as identity, classification, polymorphism, inheritance are used together to attain the goal easily

**Chapter-3**

**Object Modeling**

**3.1 Objects and Classes**

**3.1.1 Objects**

- Combination of data structures and behavior

- Object has two purpose

* Understands real world problems
* Provides basis of computer implementation
* Object has four characteristics:
* **Identity:** Identity means data quantized into discrete distinguishable objects. Each object has its own inherent identity. Eg: paragraph in a document, white queen in a chess
* **Classification:** Object with same attributes and operation are grouped into a class. Class has attributes and operations. Eg: paragraph, chess piece
* **Polymorphism:** Same operation behaves differently in different classes. Eg: Consider move operation. It is different in window and chess piece classes
* **Inheritance:** Sharing of attributes and operations among classes based on hierarchical relationship. A class is divided into subclass. Subclass inherits the properties of superclass and add its own unique properties. Eg: Scrolling window and fixed window are subclass of window and inherits the properties of window such as visible region . Scrolling window adds scroll bar and offset
* Object instance refer to exactly one thing
* Object class refer to group of similar thing
* Eg: Joe smith, gray bicycle

(Person)

Joe Smith

**Fig 3.1 Object**

**3.1.2 Class**

- Class is a group of objects with similar attributes and behavior

- Eg: Person, animal

- Each object knows its class

Person

**Fig 3.2 Class**

**3.1.3 Object Diagram**

- Object diagram is a graph whose node represent object classes and arc represents relationship

object

object

**Fig 3.3 Object Diagram**

Relationship

* Two types
* **Class diagram**:- Describes class. Represented using rectangular box
* **Instance diagram**:- Describes object. Represented using rounded box

**3.1.4 Attributes**

- Data held by objects in a class

- Eg: Name, age, weight of person class

- Donot have identity

Person

Name: string

Age: integer

**Fig 3.4 Attributes**

**3.1.5 Operations and Methods**

- Transformations that are applied on or by objects in a class

- Eg: Open, close, hide are operations of class window

- Methods are implementations of objects for a class

- eg: class file uses method print. It can print binary files, ASCII files etc

- Listed third in class box

File

Person

change job

change address

size in bytes

file name

print

name

age

**Fig 3.5 Operations and Method**

**3.2 Links and Association**

- Links and association establish connection between objects and classes

- Link is a physical or conceptual connection between objects

- Link shows relationship between two or more objects

(country)

India

(city)

Delhi

Has\_capital

**Fig 3.6 Link**

* Association is a group of links with common structure and behavior

works\_for

Person

Company

employs

**Fig 3.7 Association**

* Association can be bidirectional (forward and inverse)
* Association can be binary, ternary or higher order

Project

Language

Person

**Fig 3.8 Ternary Association**

* Person who are programmers uses language to do a project

(Language)

C

(Project)

CAD

(Person)

Mary

(Language)

Cobol

(Project)

Accounting system

**Fig 3.9 Ternary Link**

**3.2.1 Multiplicity**

- Multiplicity means how many instances of one class may relate to single instances of an associated class

- Described as one or many

- One to one

(country)

India

(city)

Delhi

Has\_capital

**Fig 3.10 one-to-one multiplicity**

* Country has one capital city
* Many-to-many

Line

Point

intersects

**Fig 3.11 many-to-many multiplicity**

* Line may have zero or more intersection points. An intersection point may be associated with two or more lines
* Zero or one (optional)

Workstation

Window

console

**Fig 3.12 zero or one multiplicity**

* Workstation has one window as console to receive error messages

**3.2.2 Importance of association**

- Supported by programming language

- Not violating encapsulation

**3.3 Advanced link and association concepts**

**3.3.1 Link attributes**

- Link attribute is the property of link in association. In this fig salary and job title are the link attributes

**Fig 3.13 Link attribute**

salary

job title

Works-for

name

address

Company

Person

name

address

**3.3.2 Modeling an association as a class**

Authorized on

User

Workstation

Authorization

priority

privilege

start session

home directory

Directory

**Fig 3.14 Modeling association as a class**

**3.3.3 Role names**

- Role is one end of association

- Role has a role name

- Role name is the name that uniquely identifies one end of association

employee employer

works-for

Company

Person

**Fig 3.15 Rolename**

- Person assumes the role of employee with respect to company. Company assumes the role of employer with respect to person

**3.3.4 Ordering**

- Objects are ordered

{ordered}

Visible-on

Screen

Window

**Fig 3.16 Ordering**

* Screen contain number of overlapping window
* So windows are ordered and only top-most window is visible
* Ordered set is indicated by writing ordered

**3.3.5 Qualification**

* Reduce multiplicity

Directory

File

**Fig 3.17 Many to one multiplicity**

* Qualifiers are used to reduce multiplicity
* Here file name is a qualifier

Directory

File

File name

**Fig 3.18 One to one multiplicity**

**3.3.6 Aggregation**

- Aggregation represents part of or part whole relationship

- Aggregation has two properties

* Transitivity- A -> B

B -> C

A -> C

* Antisymmetric- A = B

B = A

Document

Paragraph

Sentence

**Fig 3.19 Aggregation**

**3.4 Generalization and Inheritance**

- Generalization is the relationship between class and one or more refined versions of it

- Class being refined is called superclass and refined versions are called subclass

- Subclass inherits the features of superclass

- Generalization has is-a relationship

- Each subclass not only inherits the properties of superclass but adds its own attributes and operation

- The notation of generalization is a triangle connecting superclass and subclass

- In the fig equipment is the superclass and it has attributes name, weight, cost and manufacturer

- Pump and tank are the sub class of class equipment

- Pump has attributes suction pressure and flow rate

- Tank has attributes volume and pressure

pressure

volume

Pump

suction pressure

flow rate

Tank

name

weight

cost

manufacturer

Equipment

**Fig 3.20 Generalization**

**3.4.1 Use of generalization**

- Useful for modeling and implementation

- Facilitates modeling by structuring classes

- Captures what is similar and different about classes

- Generalization refer to relationship among classes

- Inheritance refer to mechanism of sharing attributes and operation using generalization relationship

- Specialization refers to fact that subclass refine or specialize the superclass

**3.4.2 Overriding Features**

- Subclass may override a superclass feature by defining a feature with same name

- Overriding feature refines and replaces the overridden feature

- Override should preserve attribute type, number and type of arguments to operation and operation return type

**3.5 Grouping Constructs**

**3.5.1 Module**

- Module is a logical construct for grouping classes, associations and generalization

- Captures one view of situation

- Eg: Electrical plumbing and ventilation modules are different view of building

- Object model consist of one or more modules

- Modules enables to partition object model into manageable pieces

- Module provide an intermediate unit of packaging between an entire object model and basic building blocks of class and association

- Module name is usually listed at top of each sheet

- There is no special notation for modules

- Same class may be referenced in different modules

- There should be fewer links between modules

**3.5.2 Sheet**

- Sheet is a mechanism for breaking large object model down into series of pages

- Sheet is a single printed page

- Each module consists of one or more sheets

- Sheet is not a logical construct

- Each sheet has a title and a name or number

- Each association and generalization appears on multiple sheet

- Class may appear on multiple sheets

- Sheet numbers or sheet names inside circles contiguous to class box indicate other sheets that refer to class

**3.6 Sample Object Model**

- Refer text Pg no: 58 for diagram

- Fig shows object model of workstation window management system

- Class Window defines common parameters of all kinds of window defined by attributes

x1,y1,x2,y2 and operations are display, undisplay raise and lower

- Panel, canvas and text window are varieties of window

- Canvas is a region for drawing graphics

- Canvas inherits the properties of class window and adds its own properties and defined by

attributes cx1,cy1,cx2,cy2

- Canvas contains set of elements shown by association to class Shape

- All shapes have color and line width

- Shapes can be lines, ellipse or polygon

- Polygon consist of ordered set vertices

- Ellipse and polygon are closed shapes which have fill color and fill pattern

- Canvas windows have operations to add elements and to delete elements

- Text window is a kind of scrolling window which is specified by x-offset and y-offset

- Text window contains string and operations insert and delete characters

- Scrolling canvas is a special kind of canvas that supports scrolling; It is both Canvas and

Scrolling window. This is an example of multiple inheritance

- Panel contains set of Panel item each identified by a unique item name

- Panel item come in three kinds: buttons, choice items and text items

- Button has string which appear on screen and has attribute depressed

- Choice item allows user to select one of the predefined choice each of which is choice entry

containing string to be displayed

- Current choice must be one of allowable choice and labeled as {subset}

- When a panel item is selected by user it generated an event

- All kind of panel item have notify event association

- Text item also inherit notify event from superclass Panel item

**Chapter-4**

**Advanced Object Modeling**

**4.1 Aggregation**

- Aggregation is a form of association in which object is made of components

- Aggregation represents part of or part whole relationship

- Aggregation has two properties

* Transitivity- A -> B

B -> C

A -> C

* Antisymmetric- A = B

B = A

Document

Paragraph

Sentence

**Fig 4.1 Aggregation**

**4.1.1 Aggregation versus association**

- Aggregation:- special form of association

Not an independent concept

- Association:- shows relationship between two or more class

An independent concept

Department

Division

Company

Works-for

Person

**Fig 4.2 Aggregation versus association**

**4.1.2 Aggregation versus generalization**

- Aggregation:- part of relationship

and relationship

- Generalization:- is a relationship

or relationship

Wiring

Switch

Base

Lamp

Incandescent lamp

Fluroscent lamp

Starter

Socket

**Fig 4.3 Aggregation versus association**

**4.1.3 Recursive aggregates**

- Aggregation can be fixed, variable or recursive

- Fixed aggregate has a fixed structure eg: Lamp

- Variable aggregate has finite number of levels but the number of parts may vary

- Eg:many divisions per company, many department per division

- recursive aggregate contains an instance of same kind of aggregate and the number of levels is unlimited

- Eg: Block is a part of program. Compound statement and simple statement inherits the properties of block. Compound statement can also be divided in to block

Program

Block

Simple statement

Compound statement

**Fig 4.4 Recursive aggregates**

**4.1.4 Propagation of operations**

- Propagation is the automatic application of an operation to network of objects

copy

copy

owns

Character

Paragraph

Document

Person

copy

copy

copy

**Fig 4.5 Propagation of operation**

- Eg: Person owns multiple documents. Each document is composed of paragraph. Each paragraph is composed of characters. Copying a paragraph copies all the charactersin it

**4.2 Abstract classes**

- Abstract classes may be used as base class

- Objects of abstract classes cannot be created

- Objects must be inherited

- Class that has no direct instance but whose descendent class have direct instance is called abstract class

- Concrete class is a class that is instantiable ie, it can have direct instance

- Concrete class may have abstract subclasses

- Concrete class describes concept completely



**Fig 4.6 Object model defining abstract and concrete class**

Worker

Painter

Baker

Butcher

**Fig 4.7 Concrete class**

-Butcher, baker and candlestick maker are concrete class because they have direct instances

-Worker is also concrete class because some occupation may not be further specified

Exempt Employee

compute pay

monthly rate

compute pay

Salaried Employee

weekly rate

Hourly Employee

compute pay

Employee

compute pay {abstract}

year to date earning

**Fig 4.8 Abstract class**

hourly rate

overtime rate

* Class Employee is an example of abstract class
* All employees must be either hourly, salaried or exempt
* Abstract classes are frequently used to define methods to be inherited by subclasses

**4.3 Generalization as extension and restriction**

- Instance of a class is an instance of all ancestors of the class

- Subclass may add new feature. This is called extension

- In fig 4.7 class Employee is extended with three subclasses that inherit Employee features and adds new features of their own

- Subclass may contain ancestor attributes. This is called restriction

**4.3.1 Overriding operation**

- Overriding is done for many reasons

* Overriding for extension
* Overriding for restriction
* Overriding for optimization
* Overriding for convenience

**4.4 Multiple Inheritance**

- Permits a class to have more than one superclass and to inherit features from parents

- Advantage: Greater power in specifying classes and reuse

- Class with more than one superclass is called join class

Vehicle

WaterVehicle

LandVehicle

AmphibiousVehicle

Boat

Car

**Fig 4.9 Multiple inheritance from overlapping classes**

* AmphibiousVehicle is both LandVehicle and WaterVehicle
* AmphibiousVehicle is a join class
* Solid triangle indicates overlapping classes
* Hollow triangle indicates disjoint classes

Hourly Employee

Employee

Salaried Employee

Exempt Employee

Vested Employee

Unvested Employee

pay status

pension status

**Fig 4.10 Multiple inheritance from disjoint classes**

**4.4.1 Accidental Multiple Inheritance**

- Instance that happens to participate in two overlapping classes

- Any instance of join class is inherently an instance of all ancestors of join class

Person

University Member

Staff

Student

Faculty

Instructor

**Fig 4.11 Workaround for accidental multiple inheritance**

**4.4.2 Workarounds**

**- Delegation using aggregation of roles:** A superclass with multiple independent generalization can be recast as an aggregate in which each component replaces a generalization. This approach replaces a single object having a unique ID by group of related objects that compose an extended object

Employee

Hourly Employee

Salaried Employee

Exempt Employee

Vested Employee

Unvested Employee

Employee Payroll

Employee Pension

pension status

pay status

**Fig 4.12 Multiple inheritance using delegation**

**-**EmployeePayroll becomes a superclass of HourlyEmployee, SalariedEmployee and ExemptEmployee. EmployeePension becomes a superclass of VestedEmployee and UnvestedEmployee. Employee can be modeled as an aggregation of EmployeePayroll and EmployeePension

- **Inherit the most important class and delegate the rest**: Join class is treated as an aggregation of remaining superclass. This approach preserves identity and inheritance across generlization

Hourly Employee

Salaried Employee

Exempt Employee

Vested Employee

Unvested Employee

Employee Payroll

Employee Pension

pension status

pay status

**Fig 4.13 Multiple inheritance using inheritance and delegation**

Employee

pay status

Exempt Employee

Salaried Employee

Hourly Employee

pension status

pension status

pension status

Exempt Unvested Employee

Exempt Vested Employee

Salaried Unvested Employee

Salaried Vested Employee

Hourly Unvested Employee

Hourly Vested Employee

**Fig 4.14 Multiple inheritance using nested generalization**

**-Nested generalization:** This approach multiplies out all possible combinations. In fig 4.14 hourly employee, salaried employee and exempt employee add two subclasses for vested and unvested employees. This preserves inheritance but duplicates declaration and code and violates the spirit of object oriented programming

**4.5 Metadata**

- Data that describes other data

- Provides information about certain item content

- Relational database management systems use metadata

- Eg:An image may include metadata that describes how large the picture is, the color, depth, image resolution, when the image was created

- A person can define database tables for storing information

**4.5.1 Patterns and metadata**

- Class describes a set of object instance

(Person)

Joe

24

- Instantiation relates a class to its instances

- Pattern describes examples of pattern

Person

(Person)

Mary

20

name

age

**Fig 4.15 Notation for instantiation**

**4.5.2 Class descriptors**

- Class descriptors have features and have their own classes called metaclasses

- Class attribute describes a value common to entire class

- Class operations is an operation on class

- Class features are indicated using dollar sign

Window

Size: rectangle

$default size: rectangle

Display

$get-highest-priority-window

**Fig 4.16 Class with class features**

**4.6 Candidate key**

- Minimal set of attributes that uniquely identifies an object or link

- Class or association may have one or more candidate keys

- Object ID is always a candidate key

- Candidate key is a logical concept

- Candidate key is delimited to object model with braces

- Many to many association requires both related objects to identify each link

- One to many association has a single candidate key

Country

Person

Company

Person

Company

City

{Candidate key: {Candidate key: {Candidate keys:

(person,company)} (person)} (country)

(city)}

**Many to many association One to many association Optional to one association**

**Fig 4.17 Comparison of multiplicity with candidate keys for binary association**

**4.7 Constraints**

- Functional relationships between entities of object model

- Entity include objects, classes, attributes, links and associations

- Eg: No employees salary can exceed the salary of employees boss

Employee

salary

{salary < boss.salary}

**Fig 4.18 Constraints on objects**

**` 4.7.1 Constraints on links**

{ordered}

Office

Person

Country

**Fig 4.19 Constraints on association links**

**4.7.2 General Constraints**

- General constraints must be expressed with natural language or equation

- Dotted line is drawn between classes involved in constraint and specify the details with comment in braces

Member-of

Person

Country

{subset}

Chair-of

**Fig 4.20 Subset constraint between associations**

**4.7.3 Derived objects, links and attributes**

- Derived object is defined as a function of one or more objects

- Derived object is completely determined by other objects

- Notation for derived entity is a slash or diagonal line

- Eg: age can be derived form birth date and current date

Current date

Person

Birthdate

/age

{Age = currentdate – birthdate }

**Fig 4.21 Derived attribute**

Machine

NetOffset

offset

offset

Offset

Part

Assembly

{offset = assembly – machine.offset \* part-assembly.offset

**Fig 4.22 Derived object and association**

**4.7.4 Homomorphisms**

- Homomorphism maps between two associations

- Homomorphism involves four relationships among four classes

- Homomorphism maps links of one general association (u) into links of another general association (t) as many to one mapping

- Two instantiation relationships map elements of one class into another: r is a many to one mapping from class B to class A and s is a many to one mapping from class D to class C

r

B

A

{u(b,d) t(b,r,d,s

s

t

u

**D**

C