USN	1	P	E	1	7	M	C	A		

PESIT Bangalore South Campus

Hosur road, 1km before Electronic City, Bengaluru -100

Department of MCA

INTERNAL ASSESSMENT TEST – 1

Date : 19/08/19 **Max Marks**: 40
Subject & Code : Object Oriented Modeling & Design(17MCA51) **Time** : 8:30 am – 10:00 am
Name of faculty : Ms. Richa Sharma

Note: ***Note: Answer any FIVE full questions, choosing one full question from each Part***

	PART-A	
1 OR 2	Explain the concept of Generalization and Inheritance with suitable example. OR Explain a) Object Orientation b) Three Models of OO Approach	8
	PART-B	
3 OR 4	a) Discuss Class Diagram b) Discuss Attribute scope with an example OR Explain in detail navigation of class models with OCL constructs.	8
	PART -C	
5 OR 6	a) Discuss Ordering and Bags & Sequences with example. b) Define Multiplicity and Association End names with the help of example. OR Discuss in detail various object and class concepts with example.	8
	PART-D	
7 OR 8	Explain various OO themes in detail. OR Discuss in detail the concept of Link and Association with the help of example .	8
	PART- E	
9 OR 10	Explain advanced concepts of object and class with suitable example. OR Discuss N-ary associations and Association Ends with suitable example.	8

USN	1	P	E	1	7	M	C	A		

PESIT Bangalore South Campus

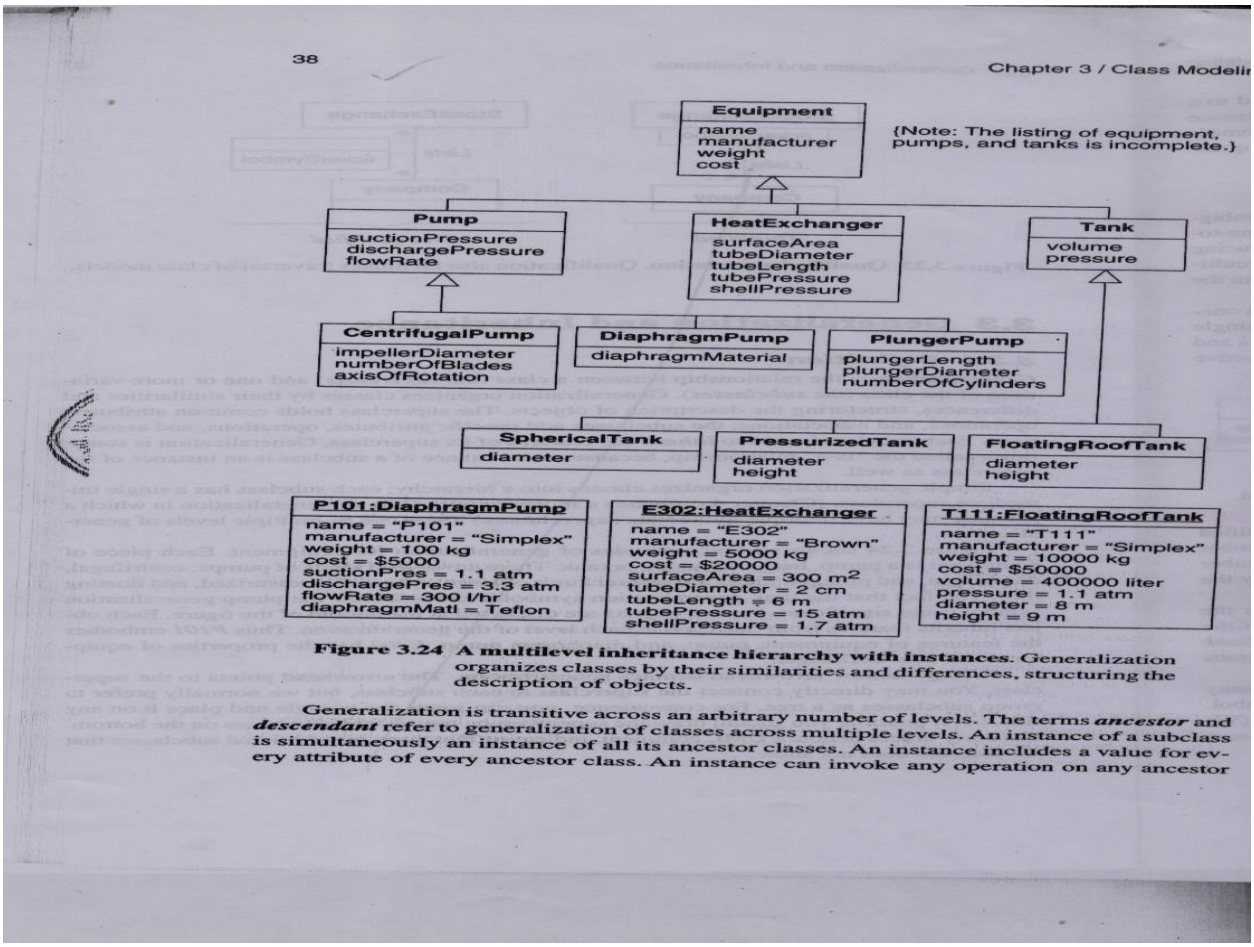
Hosur road, 1km before Electronic City, Bengaluru -100

Department of MCA

PART-A

1. Explain the concept of Generalization and Inheritance with suitable example.

Generalization is the relationship between a class (the superclass) and one or more variations of the class (the subclass). Generalization organizes classes by their similarities and differences, structuring the description of objects. A large hollow arrowhead denotes generalization. The arrowhead points to the superclass.



A generalization set name is an enumerated attribute that indicates which aspect of an object is being abstracted by a particular generalization. You should generalize only one aspect at a time.

Use of Generalization: Three purposes:

1. To support Polymorphism

2. To structure the description of the objects

3. To enable reusability of code

2.(a) Object Orientation

- It is the identification and organization of application concepts, rather than final representation in a prog. Language.
- OOD approach encourages software developers to work and think in terms of the application domain through most of the software engineering life cycle.
- It is a conceptual process independent of a programming language until the final stage.

Here we represent a methodology for OOD and a graphical notation for representing OOD concept. The methodology consists of: -

- Building a model of an Application
- Design(Adding implementation details)

We call this approach the Object Modeling Technique.

(b) Three models of OO approach:

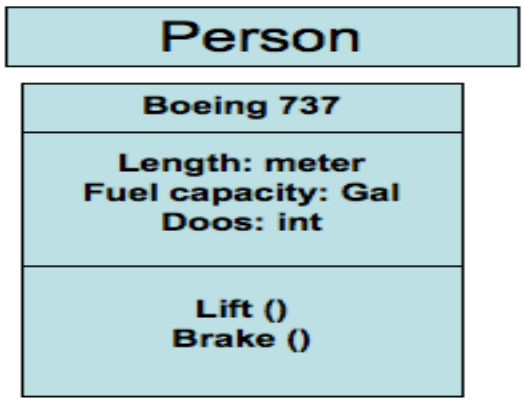
- The class model- represents the static, structural,"data" aspects of a system.
- The state model – represents the termporal, behavioral, "control" aspects of a system.
- The Interaction model – represents the collaboration of individual objects, the "interaction " aspects of a system.

PART-B

3.(a) Discuss Class Diagram

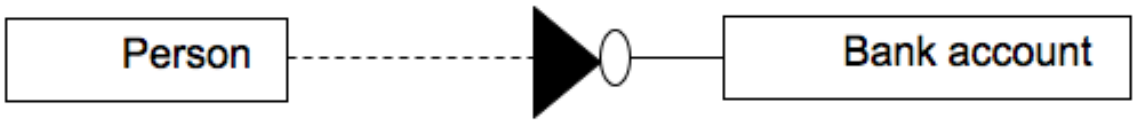
- Class describes a group of objects with the same properties (attributes), behavior (operations), kinds of relationships.
- Class model captures the static structure of a system by characterizing the objects in the system, the relationships between the objects and the attributes and operations for each class of objects.

- **Class Notation: Static Structure:**
- A class is drawn as a rectangle with three component separated by horizontal lines.
- The top name compartment holds the class name and other general properties of the class.
- Attributes are in the middle and last compartment holds a list of operation.
- Either or both the attribute and operation compartments may be suppressed.
- A separate line is not drawn for missing compartment if it is suppressed.



Class Interface Notation:

- Class interface Notation is used to describe the externally visible behavior of a class.
- The UML Notation for an interface is a small circle with the name of the interface connected to the class.
- A class that requires the operations in the interface may be attached to the circle by a dashed arrow.
- For example, a person object may need to interact with the Bank account object to get the balance**.**



4. Explain in detail navigation of class models with OCL constructs.

- Navigation lets us exercise a model and uncover hidden flaws and omissions.
- Considering bank example
- What transactions occurred for a credit card account within a time interval?
- What volume of transactions were handled by an institution in the last year?
- What is the total maximum credit for a customer for all accounts?
- UML incorporates a language that can express these kind of questions the OCL.

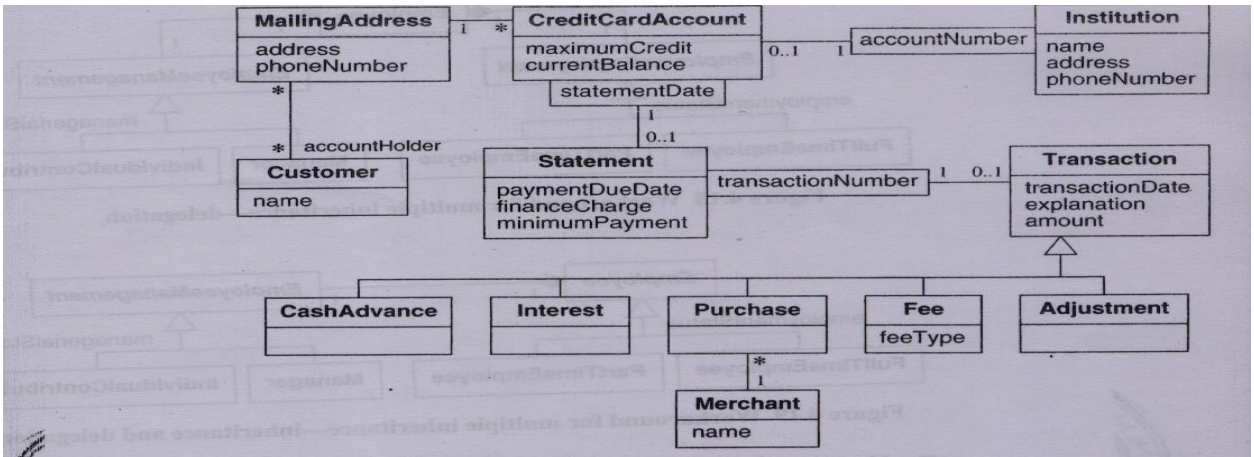


Figure 3.27 Class model for managing credit card accounts

- How many credit card accounts does a customer currently have?
- What is the total maximum credit for a customer, for all accounts?

The UML incorporates a language that can express these kinds of questions—the ***Object Constraint Language (OCL)*** [Warmer-99]. The next two sections discuss the OCL, and Section 3.5.3 then expresses the credit card questions using the OCL. By no means do we cover the complete OCL; we just cover the portions relevant to traversing class models.

3.5.1 OCL Constructs for Traversing Class Models

The OCL can traverse the constructs in class models.

- **Attributes**. You can traverse from an object to an attribute value. The syntax is the source object, followed by a dot, and then the attribute name. For example, the expression *aCreditCardAccount.maximumCredit* takes a *CreditCardAccount* object and finds the value of *maximumCredit*. (We use the convention of preceding a class name by "a" to refer to an object.) Similarly, you can access an attribute for each object in a collection, returning a collection of attribute values. In addition, you can find an attribute value for a link, or a collection of attribute values for a collection of links.
- **Operations**. You can also invoke an operation for an object or a collection of objects. The syntax is the source object or object collection, followed by a dot, and then the operation. An operation must be followed by parentheses, even if it has no arguments, to

OCL constructs for traversing class models

- Attributes- syntax - source object, followed by dot and then the attribute name

  aCreditcardaccount.maximumcredit finds the value of maximum credit.

- Operations- Syntax – source object, followed by dot and then the operation followed by paraentheses.

aCreditcardaccount.check()

- for collection operation use sourcecode -> operation
- Simple associations – dot notations is to traverse an association to target end.
- The target end may be indicated by an association end name or where there is no ambiguity, a class name

  aCustomer.mailingaddress –gives a set of addresses for acustomer

  aCreditcardaccount.mailingaddress – gives a single address
- **Qualified associations-** a more precise traversal.

  aCreditcardaccount.statement[30nov1999]

  finds the stt for a credit card account with stt date.
- **Association classes-** link of an assn class.
- **Generalizations** – Traversal of a generelization hierarchy is implicit for the OCL notation
- **Filters-** to filter the objects in a set. Ocl has several filters. Most common one is select operation.
- aStatement.transaction->select(amount>$100)

Building OCL expressions

- OCL comes from combining primitive constructs into expressions.
- With the OCL a traversal from an object through a single association yields a singleton or a set.
- If a traversal is through multiple associations can yield a bag.
- Examples for OCL expressions
- What transactions occurred for a credit account within a time interval?

aCreditCardAccount.Statement.Transaction->

Select(aStartDate <= transactionDate and transactionDate <= anEndDate)

- How many credit card accounts does a customer currently have?

  aCustomer.MailingAddress.CreditCardAccount->size()

- What is the total maximum credit for a customer for all accounts?

  aCustomer.MailingAddress.CreditCardAccount.maximumCredit->sum()

- What volume of transactions were handled by an instruction last year?

anInstitution.CreditCardAccount.Statemetn.Transaction-> select(aStartdate <= transactionDate and transactionDate<= anEnddate).amount->()

PART-C

5.(a) Discuss Ordering and Bags & Sequences with example.

Ordering: Often the objects on a "many" association end have no explicit order, and you can regard them a set. For example, A workstation screen containing a number of overlapping windows. Each window on a screen occurs at most once. The windows have an explicit order, so only the top most window is visible at any point on the scree, by writing "{ordered}" next to the appropriate association end.

Bags & Sequences: A bag is a collection of elements with duplicates allowed.

A sequence is an ordered collection of elements with duplicates allowed. Like the ordered indication {bag} and {sequence} are permitted only for binary association.

A sequence association is an ordered bag, while an ordered association is an ordered set.

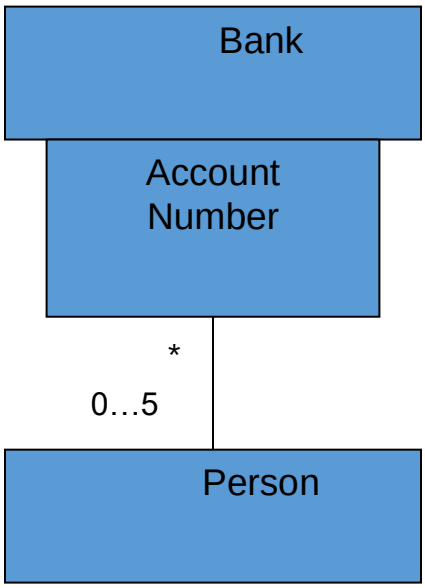
(b) Define Multiplicity and Association End names with the help of example.

- Specifies the number of instances of one class that may relate to a single instance of an associated class
- It is one or many.
- It is a constraint on the size of collection.
- It specifies that an object may be associated with

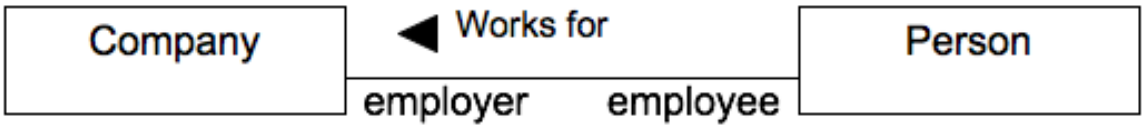
  multiple objects.

- Lower bound and upper bound.

– ***** denotes unlimited upper bound.



The end of an association, where it connects to a class is called the association role.



6. Discuss in detail various object and class concepts with example.

- An object is a concept, abstraction or thing with identity that has meaning for an application.
- All objects have identity and are distinguishable
- Object is an instance of a class.
- Class describes a group of objects with the same properties (attributes), behavior (operations), kinds of relationships.

- Class model captures the static structure of a system by characterizing the objects in the system, the relationships between the objects and the attributes and operations for each class of objects.

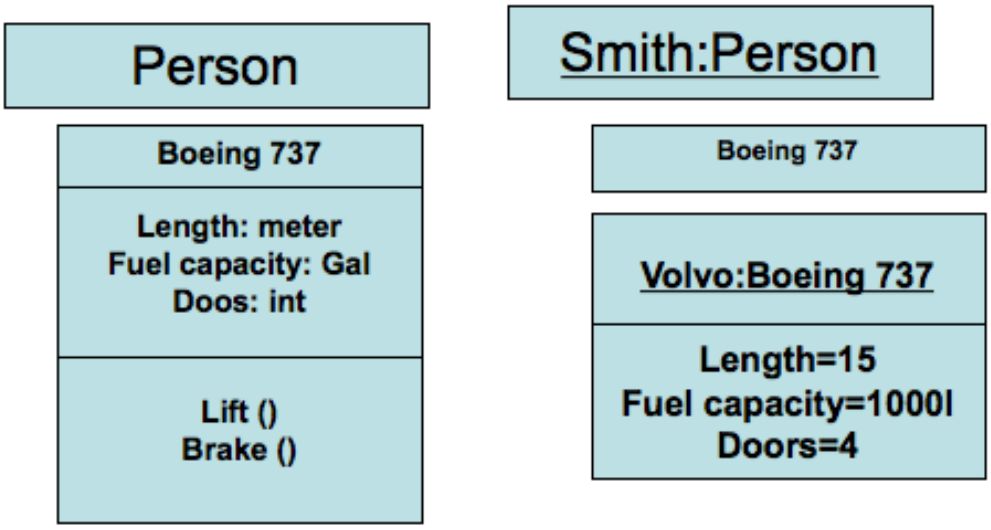
CLASS MODELING

- **Class Notation: Static Structure:**
- A class is drawn as a rectangle with three component separated by horizontal lines.
- The top name compartment holds the class name and other general properties of the class.
- Attributes are in the middle and last compartment holds a list of operation.
- Either or both the attribute and operation compartments may be suppressed.
- A separate line is not drawn for missing compartment if it is suppressed.

Object Diagram:

- Object is a concept, abstraction or thing with identity that has meaning for an application.
  - Static Object diagram is an instance of a class diagram. Notation is the same for an object diagram and a class diagram.
  - It shows individual objects and their relationships.
  - It is helpful for documenting test cases and discussing examples
- Operation is a function or procedure that may be applied to or by objects in a class.
- The same operation may apply to many different classes-poly
- A method is the implementation of an operation for class.
- When an operation has methods on several classes, all have the same signature- the number and types of arguments and the type of result value.
- Ex print should not have filename as an argument for one and file pointer for another.

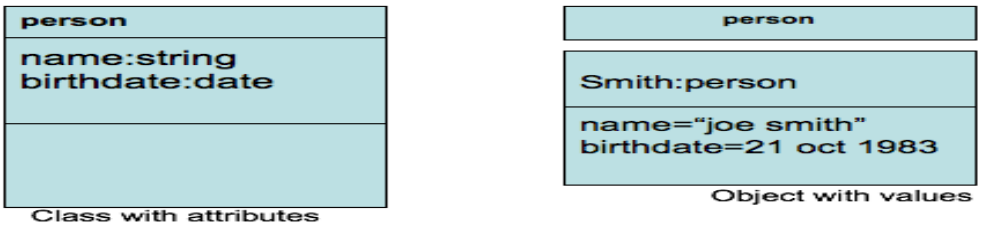
PESIT Bangalore South Campus
Hosur road, 1km before Electronic City, Bengaluru -100
Department of MCA



Values and attributes

- Value is a piece of data.
- Attribute is a named property of a class that describes a value held by each object of a class.
- Object is to class as value is to attribute.
- Attributename:datatype=defaultvalue
- Operationname(argumentlist):resulttype

.



PESIT Bangalore South Campus
Hosur road, 1km before Electronic City, Bengaluru -100
Department of MCA

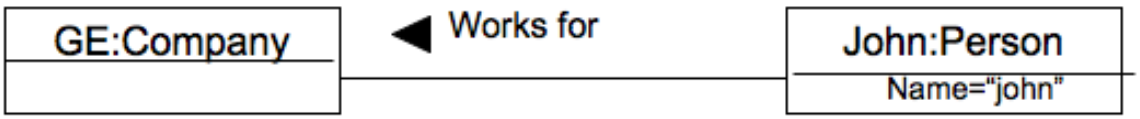
PART-D

7. Explain various OO themes in detail.

- **Abstraction-** Focusing on essential features of an application. What an object is and what it does?
- **Encapsulation- Information hiding.** It separates the external aspects of an object that are accessible to other objects from the internal implementation details that are hidden from other objects. Prevents interdependence.
- **Combining data & behavior-**Polymorphism, a concept of class hierarchy.
- **Sharing –** Inheritance and reusability of the code**.**
- **Emphasis on the essence of an object-** Focuses on what an object is rather than how it is used. OO emphasizes more on data structure and less on procedure.
- **Synergy-** Identity, Classification, Polymorphism and Inheritance characterize OO languages. Each of these concepts can be used in isolation but together they complement each other synergistically.

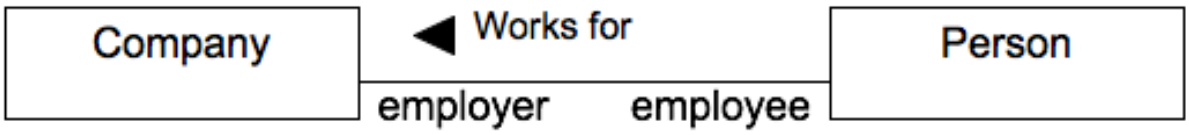
8. Discuss in detail the concept of Link and Association with the help of example.

- A link is a physical or conceptual connection among objects.
- Links usually relate two objects but some links 3 or more.
- Link is a line between objects. ie relationship among object
- An association is a descriptions of group of links with common structure and common semantics. It connects related classes by a line.
- Many types of associations can be used.
- Reference is an attribute in one object that refers to another object.



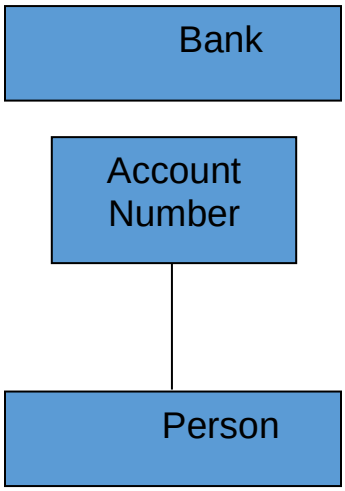
Binary Association Notation & Association Role/ end

- A Binary association is drawn as a solid path connecting two classes
- An association may have an association name.
- The association name may have an optional black triangle in it, the point of the triangle indicating the direction in which to read the name.
- The end of an association, where it connects to a class is called the association role.
  - UML uses the term association navigation or navigability.
  - Association is represented by an open arrow.
  - Here navigation is in only one direction BK a/c class can know about person class and not the reverse.



Qualifier:

- A Qualifier is an association attribute.
-

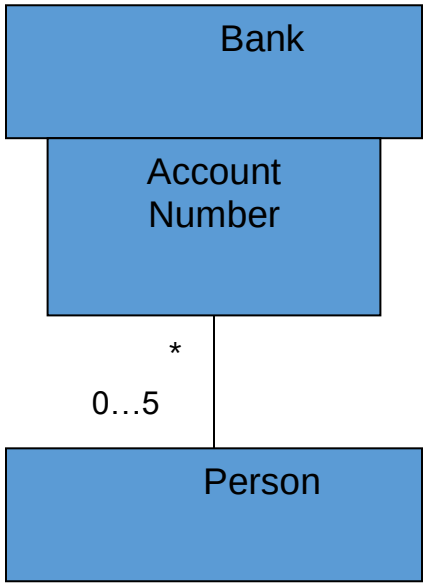


Multiplicity:

- Specifies the number of instances of one class that may relate to a single instance of an associated class
- It is one or many.
- It is a constraint on the size of collection.
- It specifies that an object may be associated with

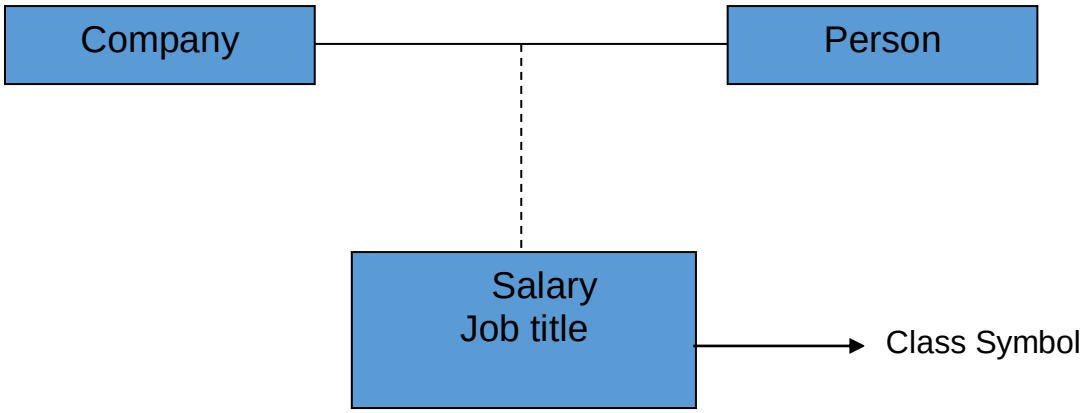
  multiple objects.

- Lower bound and upper bound.
  - * denotes unlimited upper bound.
  -



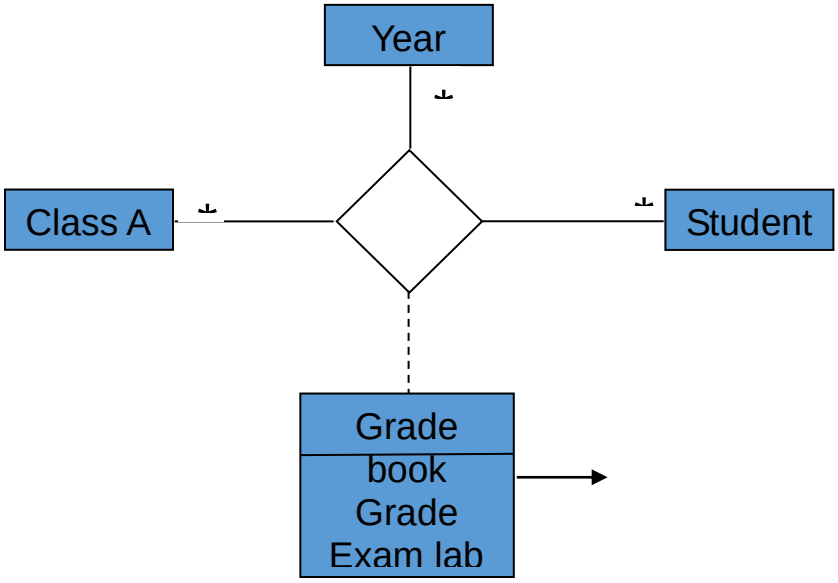
Association Class:

- An association class is an association that also has class properties.
- An association class is shown as a class symbol attached by a dashed line to an association path.
- It lets you specify identity and navigation paths precisely



N-Ary Association:

- An N-ary association is an association among more than two classes.
- Represented by (Diamond) Symbol.
- Multiplicity may be indicated.
- Association class with attributes, operations/associations.



PESIT Bangalore South Campus

Hosur road, 1km before Electronic City, Bengaluru -100

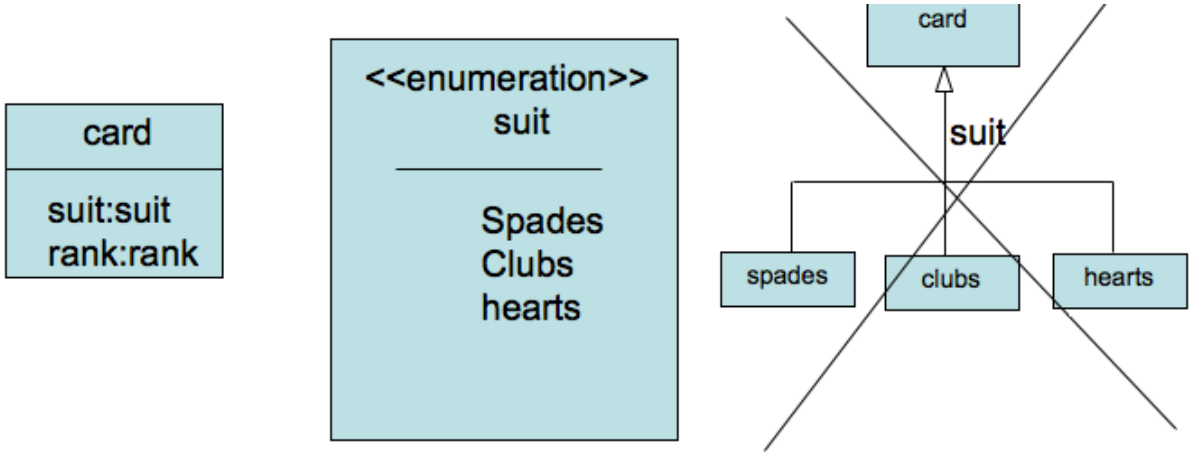
Department of MCA

PART-E

9. Explain advanced concepts of object and class with suitable example.

Enumerations

- Enumeration is a data type that has a finite set of values.
- Do not use a generalization to capture the values of enumerated attribute.
- It is a list of values.



- **visibility refers to the ability of a method to reference a feature another class and has possible values of public, private, protected.**

  + - public, - private # protected

Several issues when choosing visibility

- Comprehension-consider only public features. Can ignore private, protected
- Extensibility- many classes depend on public methods.
- Since fewer classes depend on private, protected and package methods, u can change them.
- Context-private, protected and package methods may rely on preconditions or state information created by other methods in the class.

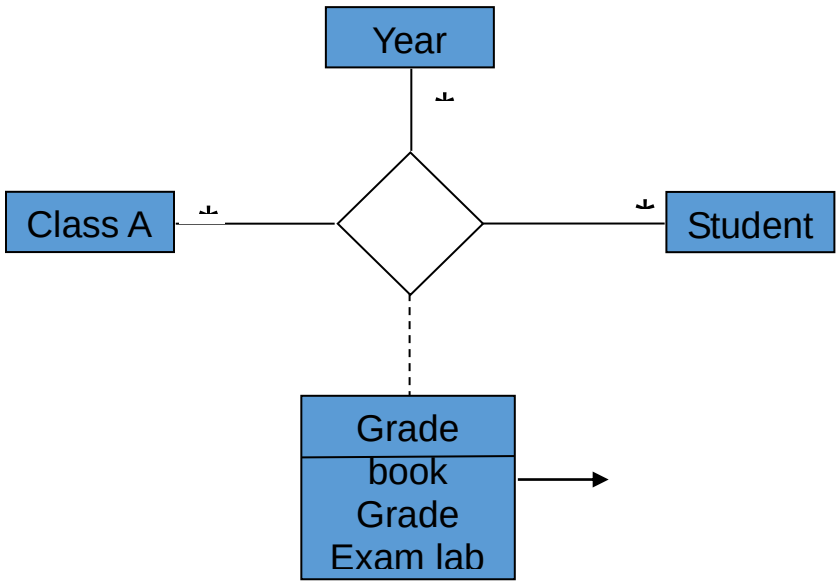
Association ends

- Association end name- facilitate navigation
- Multiplicity – can specify for each association end
- Ordering – the objects for a "many" association end are usually just a set.
- Qualification- one or more qualifier attributes can disambiguate the objects for "many" association end
- Aggregation-only binary association can be aggregation. One association end can be aggregate and other constituent
- Changeability- specifies update status of an association end.
- It can be updated or intialized
- Navigability-association may be traversed in either direction
- Visibility-association ends may be public, private or protected.

10. Discuss N-ary associations and Association Ends with suitable example.

N-Ary Association:

- An N-ary association is an association among more than two classes.
- Represented by (Diamond) Symbol.
- Multiplicity may be indicated.
- Association class with attributes, operations/associations.



Association End names:

- Association end name- facilitate navigation
- Multiplicity – can specify for each association end
- Ordering – the objects for a "many" association end are usually just a set.
- Qualification- one or more qualifier attributes can disambiguate the objects for "many" association end
- Aggregation-only binary association can be aggregation. One association end can be aggregate and other constituent
- Changeability- specifies update status of an association end.
- It can be updated or intialized
- Navigability-association may be traversed in either direction
- Visibility-association ends may be public, private or protected.

PESIT Bangalore South Campus

Hosur road, 1km before Electronic City, Bengaluru -100

Department of MCA