SOOAD

UNIT 4

OBJECT ORIENTED ANALYSIS & DESIGN

UNIT -4 Use Case, Class & Object Diagram

- Use - Case Diagram
- Class Diagram
- Object Diagram

CLASS DIAGRAM

UNIT -4 Class Diagram

- **Class Diagram**
  - Analysis and Design version of Class Diagram
  - Elements of Class Diagram
  - Guidelines for design of Class Diagram

UNIT -4 Class Diagram

- It shows the static structure of the system being modelled.
- It shows the static structural aspect of the system.
- It represents the classes and relationship in the system.
- A UML class diagram is made up of:
  - A set of classes and
  - A set of relationships between classes

UNIT -4 Class Diagram

- Purpose of Class Diagrams
  - Shows static structure of classifiers in a system
  - Understanding requirements
  - Diagram provides a basic notation for other structure diagrams prescribed by UML
  - Helpful for developers and other team members too
  - Business Analysts can use class diagrams to explore business concepts in the form of domain model

UNIT -4 Class Diagram

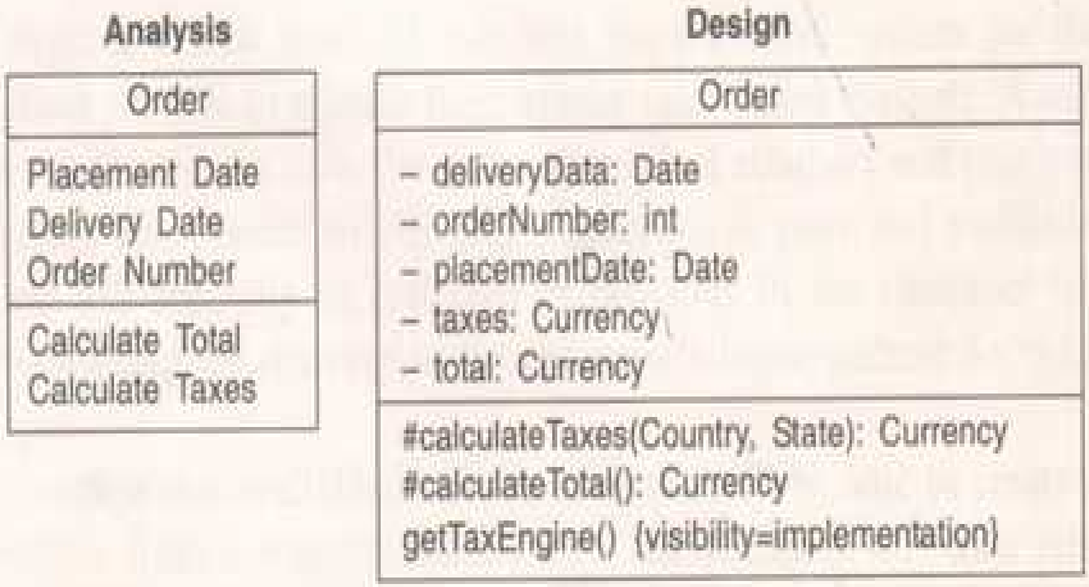
- **Analysis and Design Versions of a Class Diagram**
  - It can be two types:
    - Analysis class diagram
      - Represent domain analysis model
    - Design Class Diagram
      - Represent detailed design model

UNIT -4 Class Diagram

- During the analysis phase, the class diagram is used to represent the conceptual model which depicts the detailed understanding of the problem space for the system.
- Class diagrams only model the static structure of the system.
- It will only show the case and not actual instances of class.

UNIT -4 Class Diagram

- **Analysis and Design Versions of a Class Diagram**



Analysis

Order

Placement Date
Delivery Date
Order Number

Calculate Total
Calculate Taxes

Design

Order

– deliveryData: Date
– orderNumber: int
– placementDate: Date
– taxes: Currency
– total: Currency

#calculateTaxes(Country, State): Currency
#calculateTotal(): Currency
getTaxEngine() {visibility=implementation}

FIGURE 4.1 Analysis and design version of class diagram.

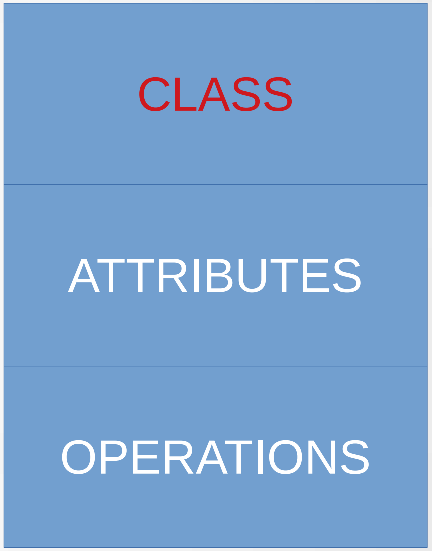
UNIT -4 Class Diagram

- **Elements of Class Diagram**
  - Class
  - Relationships
  - Association Class
  - Interface
  - Package

UNIT -4 Class Diagram

CLASS

- Classes have
  - Attributes (member variables)
  - Operations (member functions)
  - Relationships
- Representation of class diagram



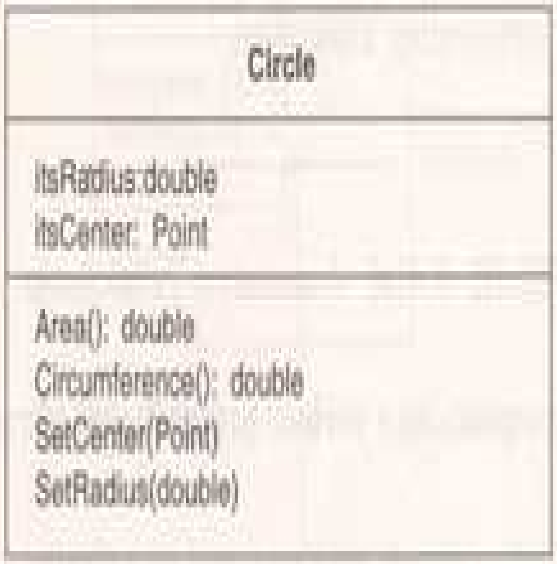
CLASS

ATTRIBUTES

OPERATIONS

UNIT -4 Class Diagram

- CLASS
  - It does not show evey attribute and operations.



Circle

ItsRadius:double
itsCenter: Point

Area(): double
Circumference(): double
SetCenter(Point)
SetRadius(double)

FIGURE 4.3 Example class circle.

UNIT -4 Class Diagram

- **Elements of Class Diagram**
  - **Guidelines for identifying classes**
    - Look for **nouns and noun phrases** in the problem statement
    - Some classes are **implicit or taken from** general knowledge.
    - All class must **make sense in the application domain**
    - **Carefully choose** and define the class **names**
    - Finding **classes is an incremental and iterative** process
    - Remove **redendant,irrelevent, adjective and attribute classes from the list of classes identified.**

UNIT -4 Class Diagram

- **Elements of Class Diagram**
  - **Guidelines for identifying attributes**
    - **Look for the nouns** followed by prepositional phrases
    - **Look for the adjectives or adverbs**
    - **Attributes may not be fully described** in the problem statement.
    - **Do not discover excess attributes.**

UNIT -4 Class Diagram

- **Elements of Class Diagram**
  - **Guidelines for identifying Operations**
    - Operations **correspond to the methods** reponsible for managing the value of attributes.
      - Example: getBalance(), setBalance()
    - Operation **also sometimes correpond to queries about association of the objects**
      - Example: deposit(), withdraw()

UNIT -4 Class Diagram

- **Relatonships**
  - The types of relationships in a class diagram are listed:
    - **Association**
    - **Bidirectional Association**
    - **Undirectional Association**
    - **Reflesive Association**
    - **Generalization**
    - **Aggregation**
    - **Composition**
    - **Dependency**
    - **Multiplicity**

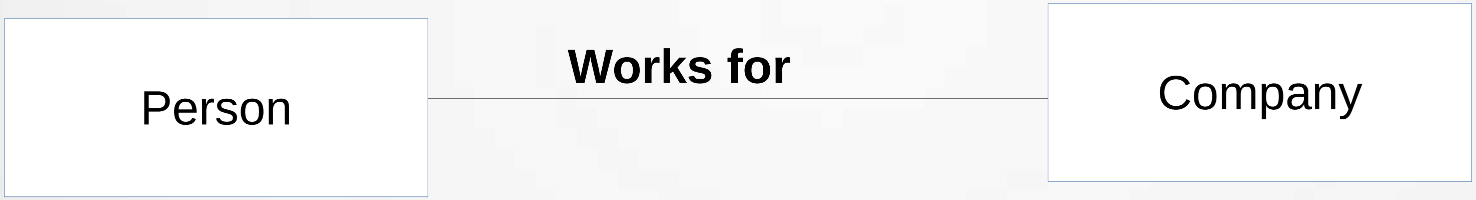
UNIT -4 Class Diagram

- **Relationships – Association**
  - Association represents a physical or conceptual connection between two or more classes.
  - A reference from one class to another is an association .

Association Relationship

UNIT -4 Class Diagram

- **Relationships – Bidirectional Association**
  - Associaton are always assumed to be bidirectional.
  - It means that **both classes are aware of each other and their relationship.**



Person

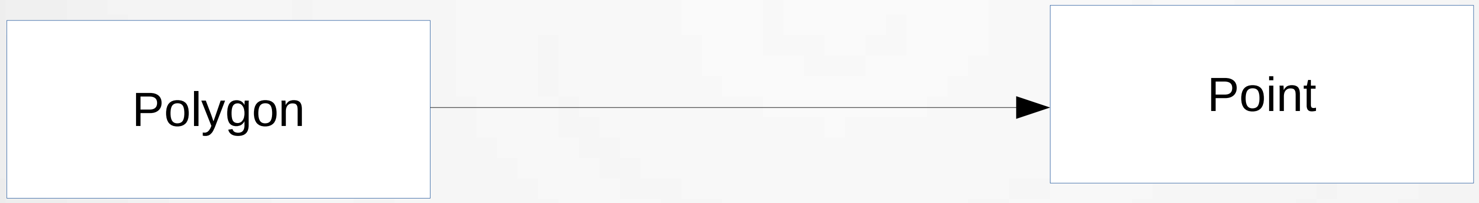
Works for

Company

Bidirectional Association

UNIT -4 Class Diagram

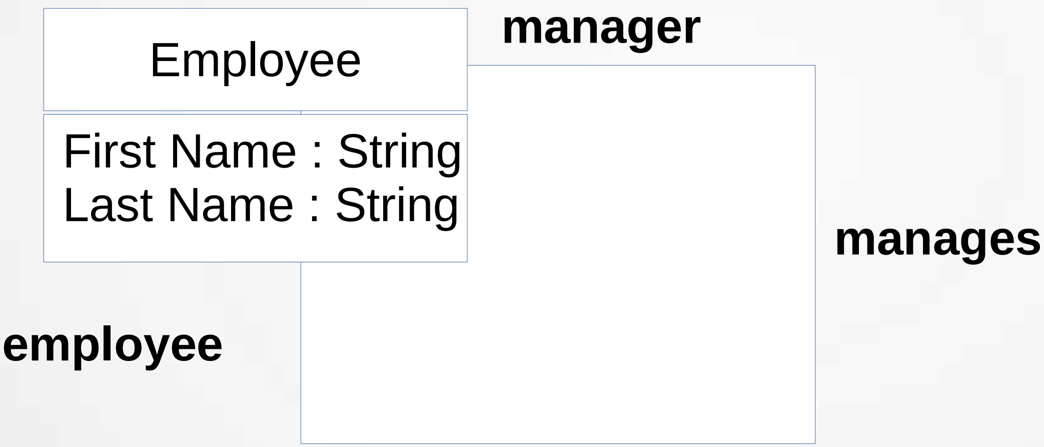
- **Relatonships – Unidirectional Association**
  - two classes are related, **but only one class knows that the relationship exists.**



Unidirectional Association

UNIT -4 Class Diagram

- **Relatonships – Reflexive Association**
  - **A class can also be associated with itself, using a reflexive asociation.**



Employee

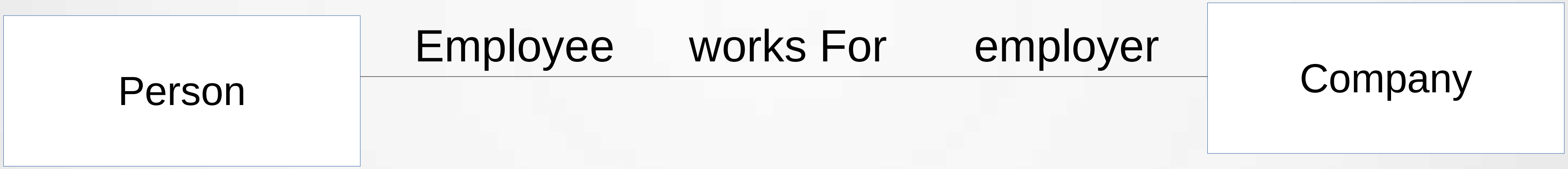
First Name : String
Last Name : String

Reflexive Association

  - Each connection of an association to a class is called an **association end**

UNIT -4 Class Diagram

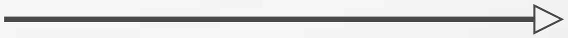
- **Relationships – Reflexive Association**



Use of Association end name

UNIT -4 Class Diagram

- **Relationships – Generalization**
  - Generalization is a relationship **between a class (Superclass) and one or more variation of the class (subclass)**
  - The instanace of a subclass is fullyconsistent with the instance of a super class and contain additionl information.
  - A generalization is used to indicate **inheritance.**



Generalization Relationship

UNIT -4 Class Diagram

- **Relationships – Generalization**

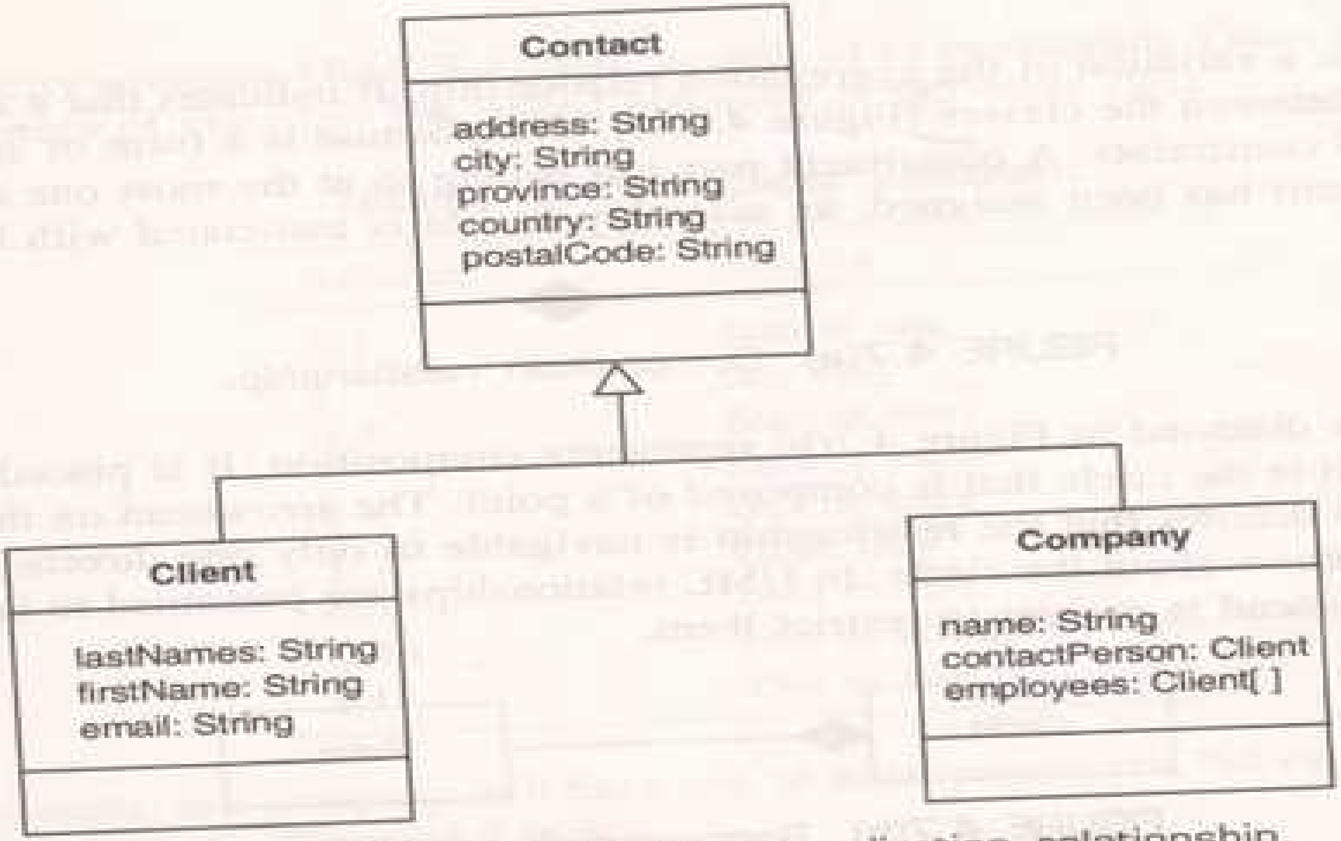
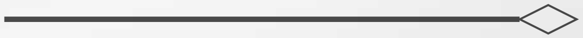


FIGURE 4.5[b] Example of generalization relationship.

UNIT -4 Class Diagram

- **Relationships – Aggregation**
  - Aggregation is a kind of association in which an aggregate object is made up of constituent parts.
  - **When a class is formed as a collection of other classes, it is called an aggregation.**
  - It is also **called a "has a" relationship"**
  - **Identify a-part-of-relation between objects from the prepositional phrases like part of, contained in**



Aggregation Relationship

UNIT -4 Class Diagram

- **Relationships – Aggregation**

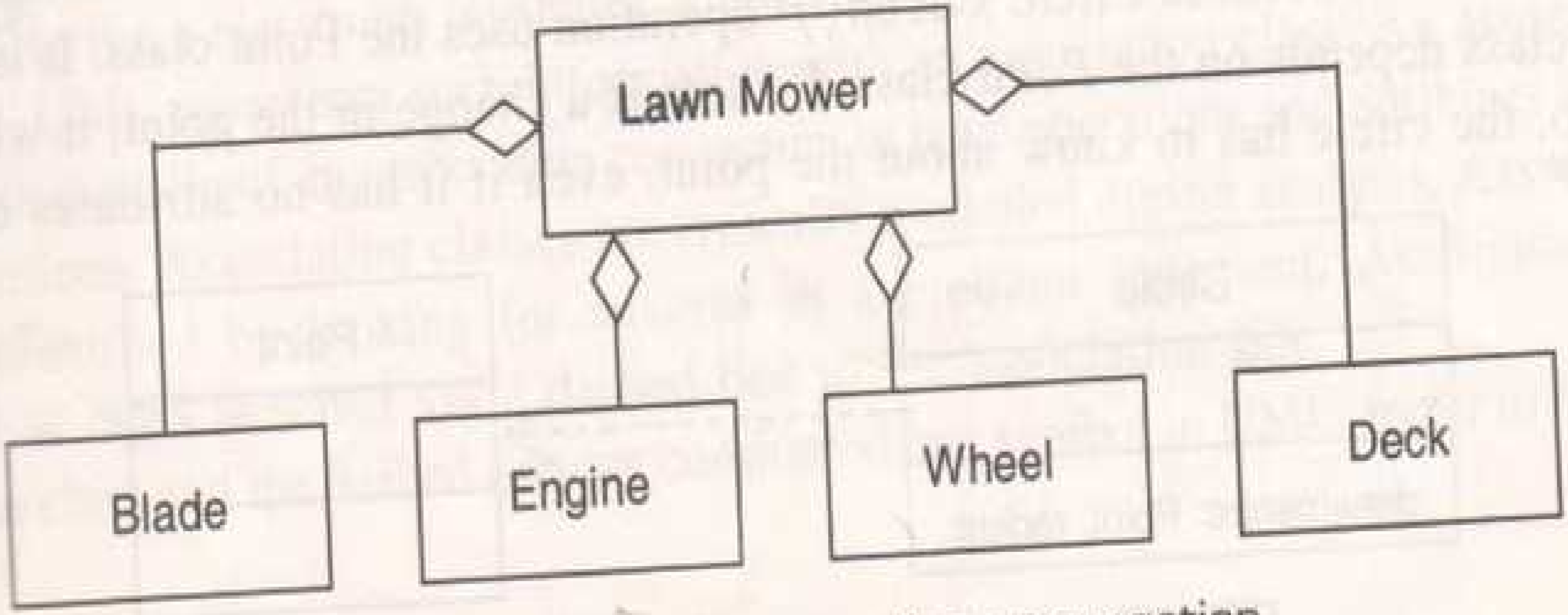
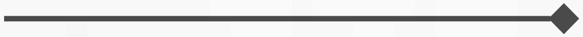


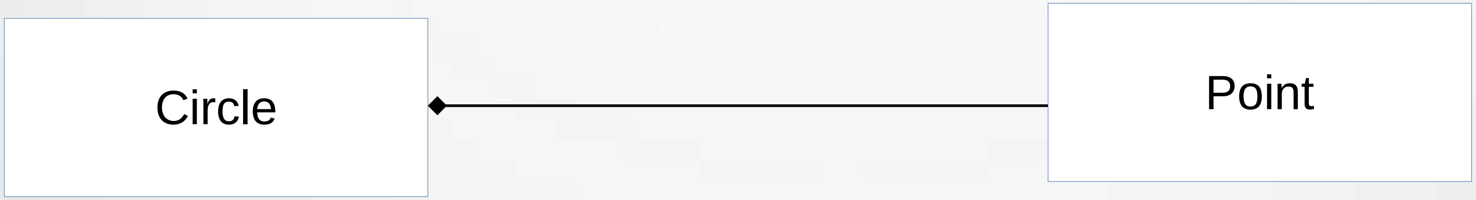
FIGURE 4.6(b) Representing aggregation.

UNIT -4 Class Diagram

- **Relationships – Composition**
  - Composition is a **variation of the aggregation relationship.**
  - **It indicate a strong life cycle is associated between the classes.**



Composition Relationship



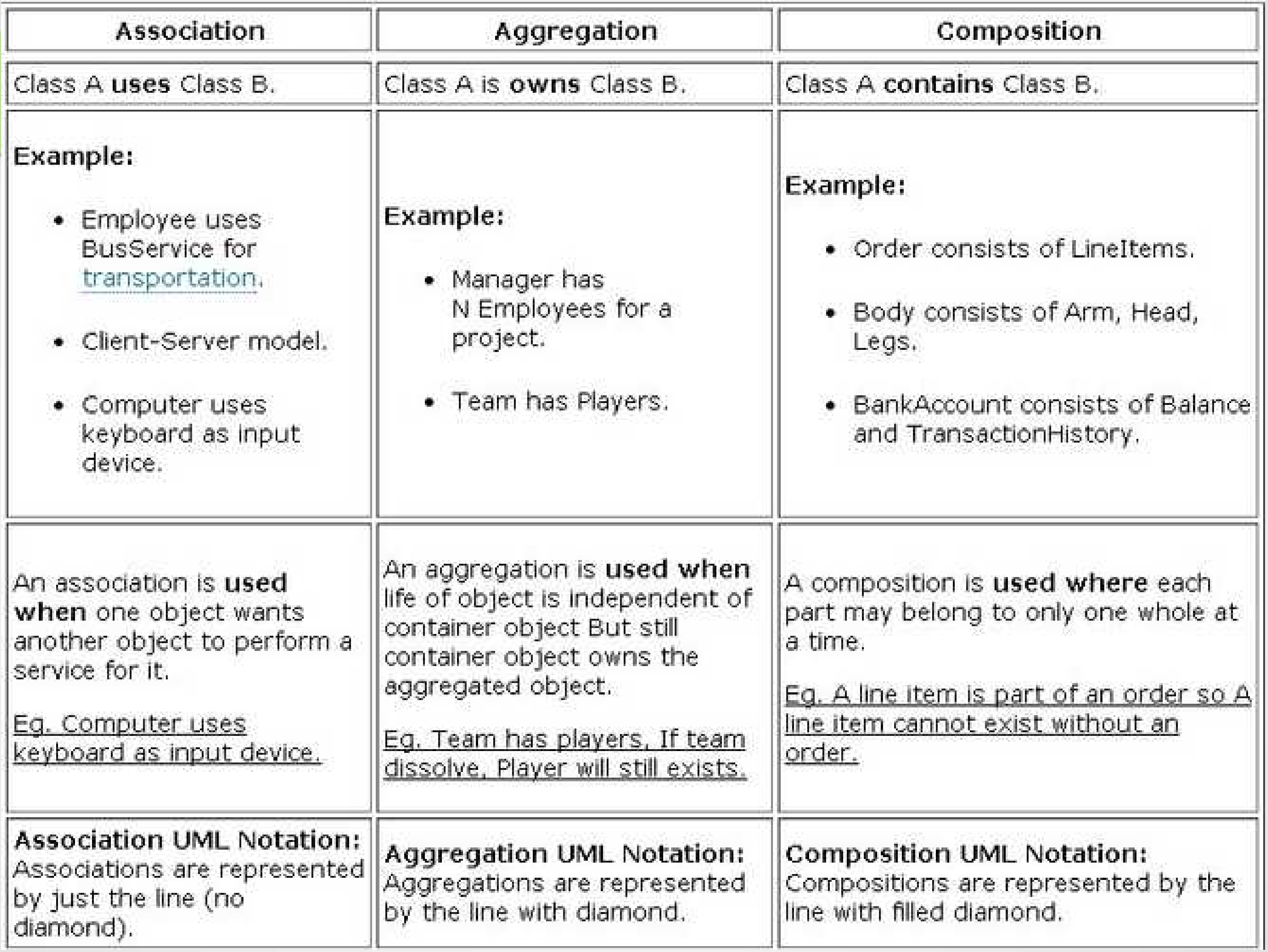
Circle

Point

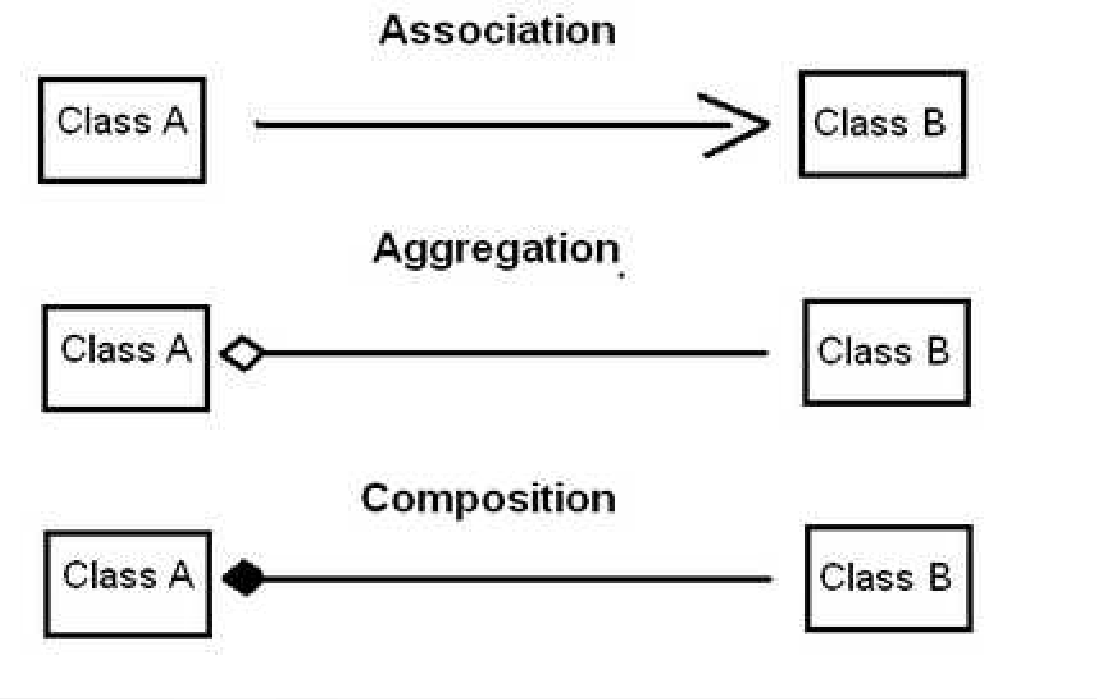
Representing Composition

UNIT -4 Class Diagram

- **Relationships – Composition**
  - **Strong form of containment or aggregation.**
  - **Aggregration is a whole/part relationship means circle is whole and point is part of the circle.**
  - **Composition indicates that lifetime of point is dependent upon the circle. This means that if the circle is destroyed, the point will be destroyed with it.**

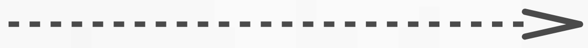


Association	Aggregation	Composition
Class A **uses** Class B.	Class A is **owns** Class B.	Class A **contains** Class B.
Example: - Employee uses BusService for transportation. - Client-Server model. - Computer uses keyboard as input device.	**Example:** - Manager has N Employees for a project. - Team has Players.	**Example:** - Order consists of LineItems. - Body consists of Arm, Head, Legs. - BankAccount consists of Balance and TransactionHistory.
An association is **used when** one object wants another object to perform a service for it. Eg. Computer uses keyboard as input device.	An aggregation is **used when** life of object is independent of container object But still container object owns the aggregated object. Eg. Team has players, If team dissolve, Player will still exists.	A composition is **used where** each part may belong to only one whole at a time. Eg. A line item is part of an order so A line item cannot exist without an order.
Association UML Notation: Associations are represented by just the line (no diamond).	**Aggregation UML Notation:** Aggregations are represented by the line with diamond.	**Composition UML Notation:** Compositions are represented by the line with filled diamond.

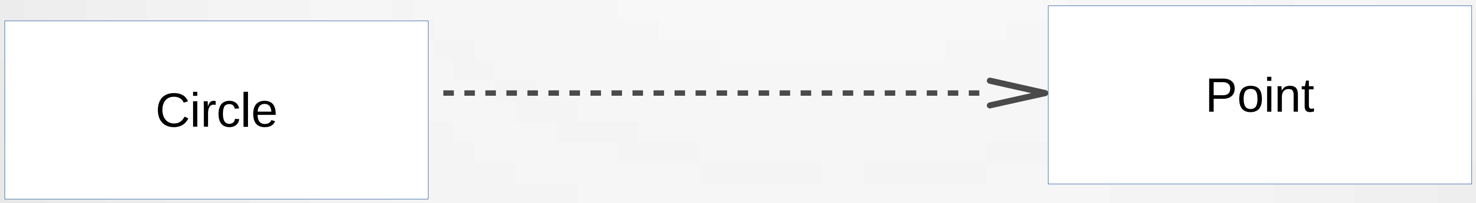

Association
Class A
Class B
Aggregation
Class A
Class B
Composition
Class A
Class B

UNIT -4 Class Diagram

- **Relationships – Dependency**
  - A dependency is a **weak relationship between two classes** and is represented by a dotted line with open arrowhead.



Dependency Icon



Circle

Point

Representing Dependency

UNIT -4 Class Diagram

- **Relationships – Multiplicity**
  - The symbols of multiplicity indicate how the instance of one class are linked instances of the other class.

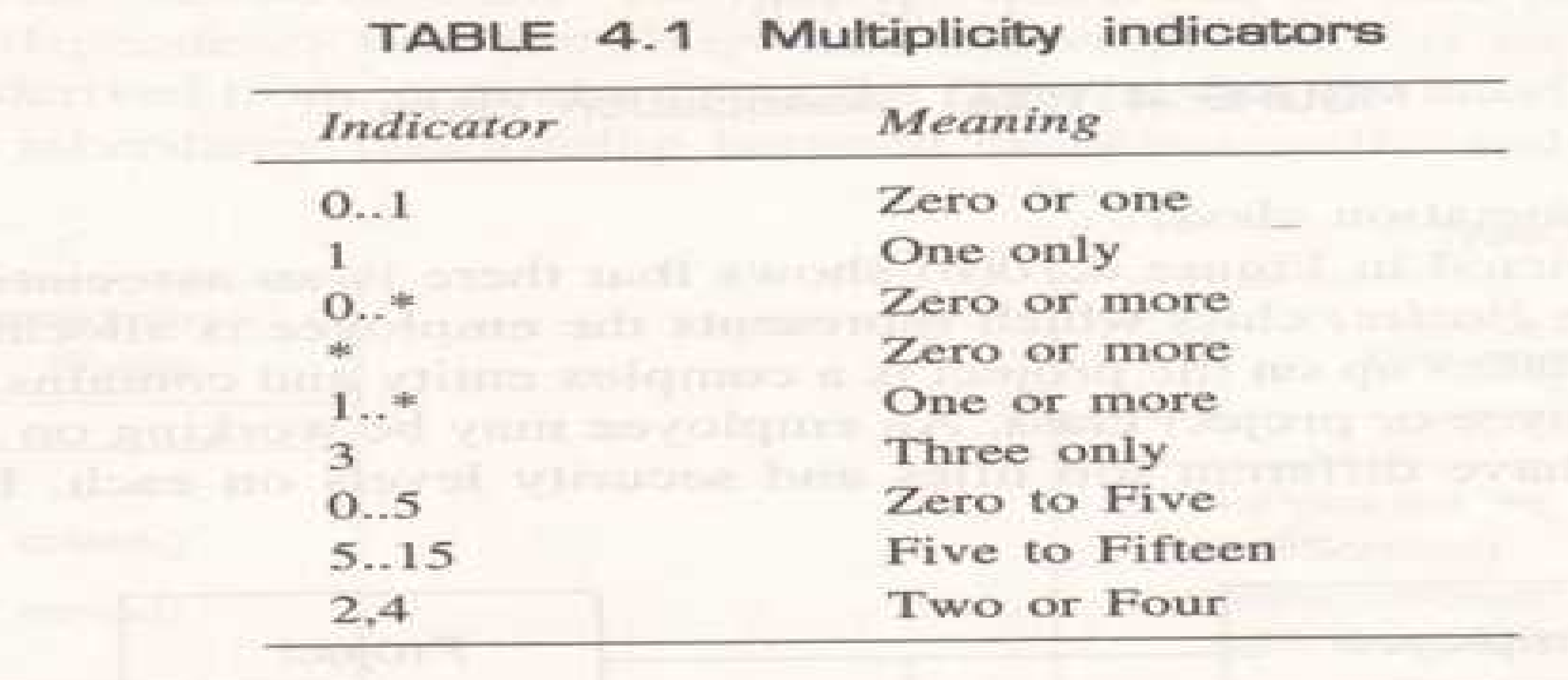
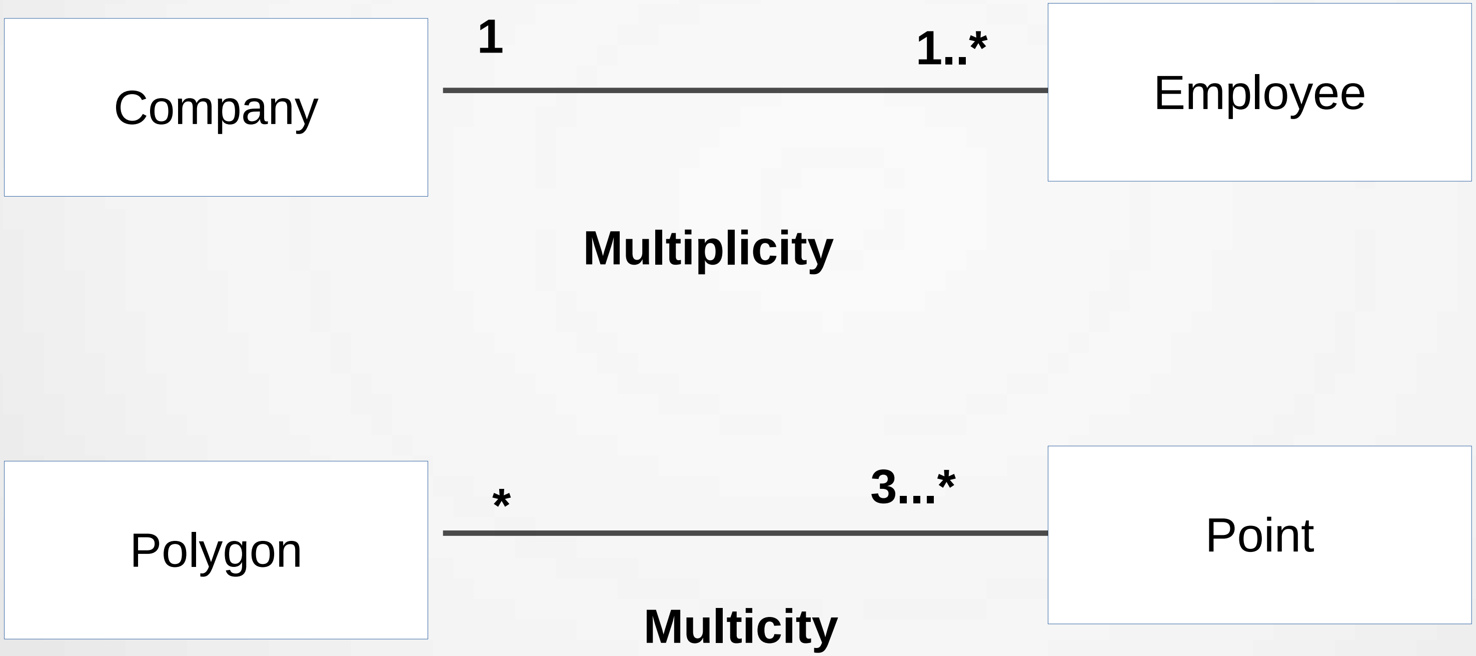


TABLE 4.1 Multiplicity indicators

Indicator	*Meaning*
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more
1..*	One or more
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen
2,4	Two or Four

UNIT -4 Class Diagram

- **Relatonships – Multiplicity**



Company

1

1..*

Employee

Multiplicity

Polygon

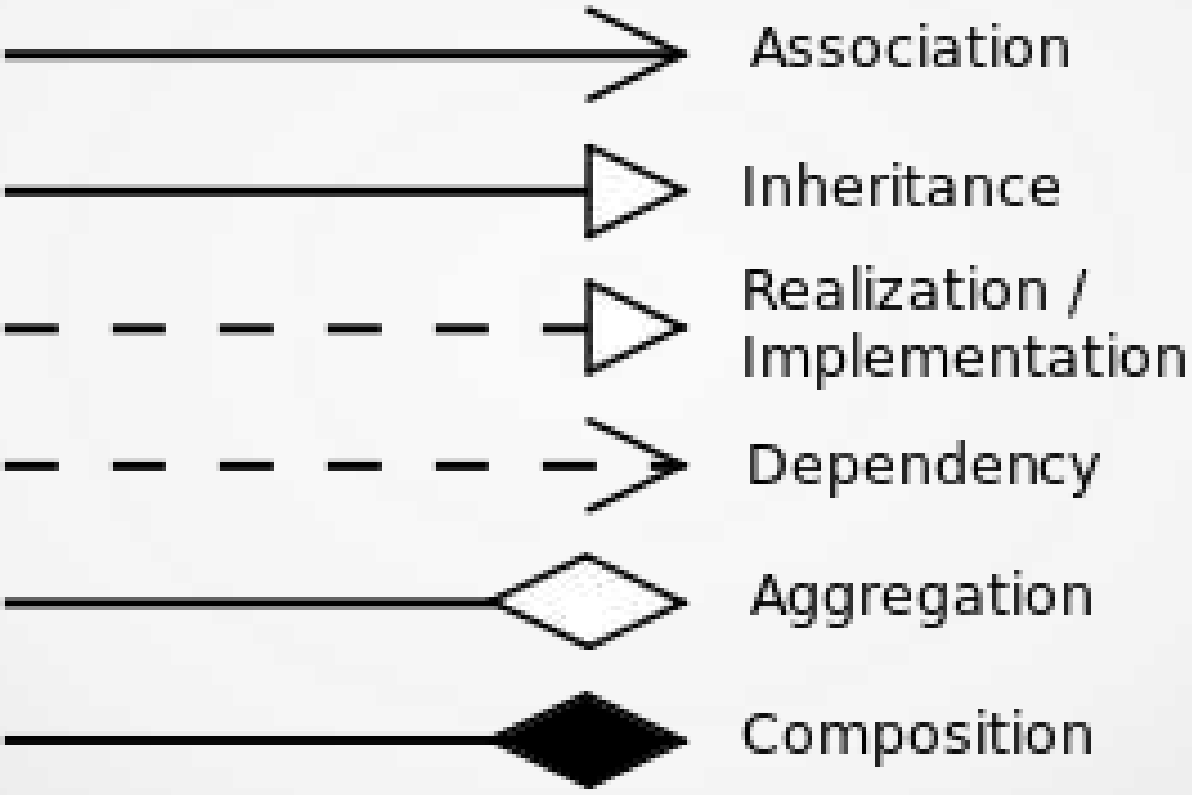
*

3...*

Point

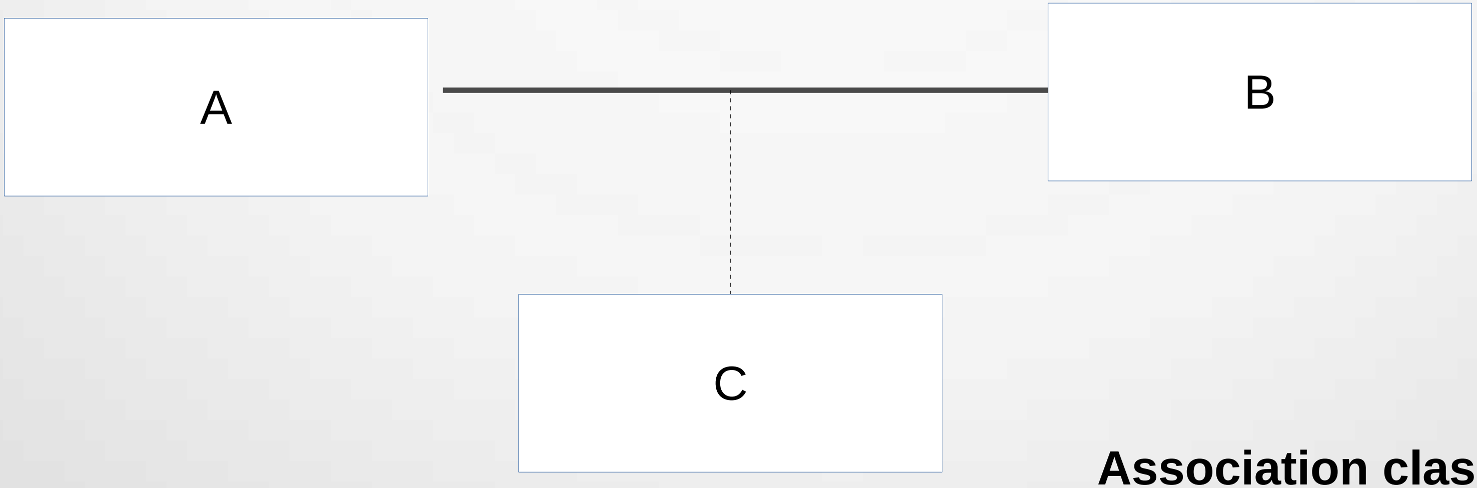
Multicity

UNIT -4 Class Diagram



UNIT -4 Class Diagram

- **Association Class**
  - An association class is a **construct that allows an association connection to have operations and attributes and participate in associations.**
  - Association classes are depicted as class attached via a dashed line to the association line.



Association class

UNIT -4 Class Diagram

- **Association Class**

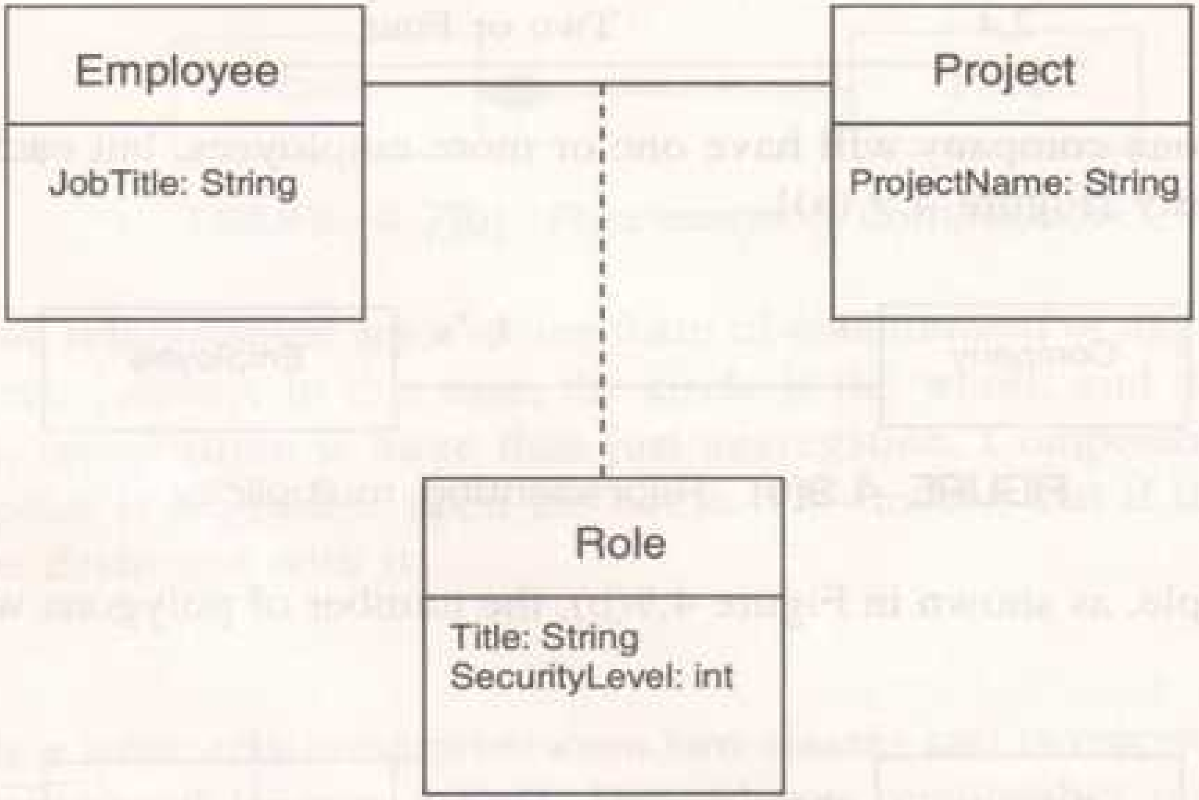
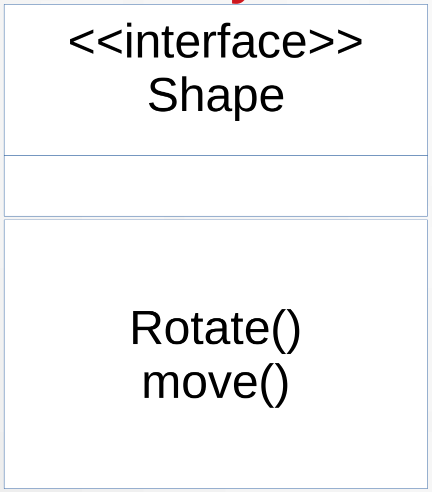


FIGURE 4.10[b] Representing association class.

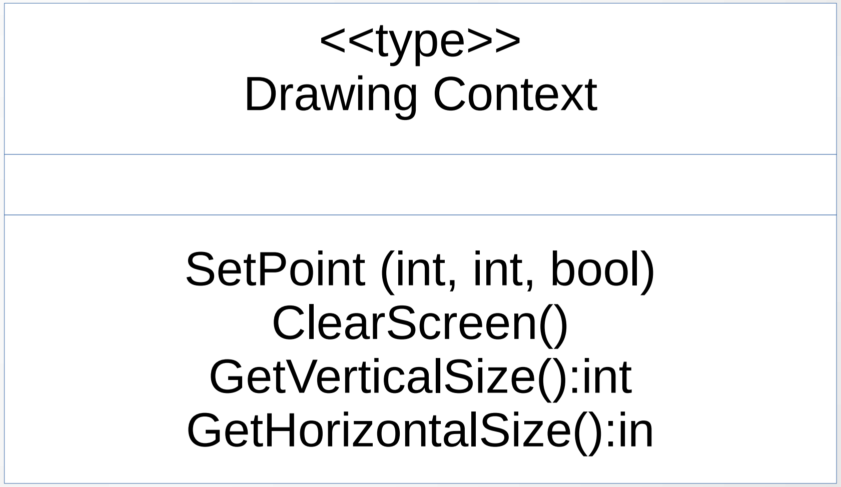
UNIT -4 Class Diagram

- **Interface**
  - An Interface is a variation of a class.
  - An interface provides only a definition of business functionality of a system.


<<interface>>
Shape

Rotate()
move()

Interface Icon


<<type>>
Drawing Context

SetPoint (int, int, bool)
ClearScreen()
GetVerticalSize():int
GetHorizontalSize():in

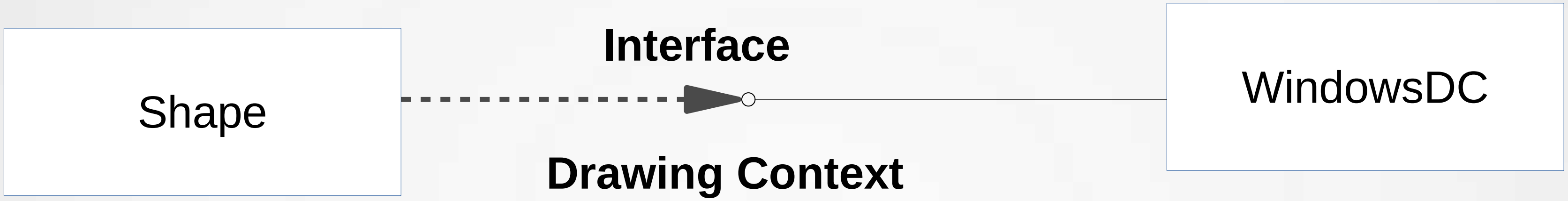
Interface Example

UNIT -4 Class Diagram

- **Interface**
  - The only difference between classes and interfaces is that the methods are only declared in interface and will be implemented in the class implementing the interface.
  - These are classes having pure virtual functions.
  - The icon for interface is just like a class with special denotation called a stereotype.
  - The two surrounding characters “<< >>” are called guillemets.
  - No member variables and and all member functions are virtual.

UNIT -4 Class Diagram

- **Interface**



LOLLYPOP NOTATION TO REPRESENT AN INTERFACE

UNIT -4 Class Diagram

- **Interface**

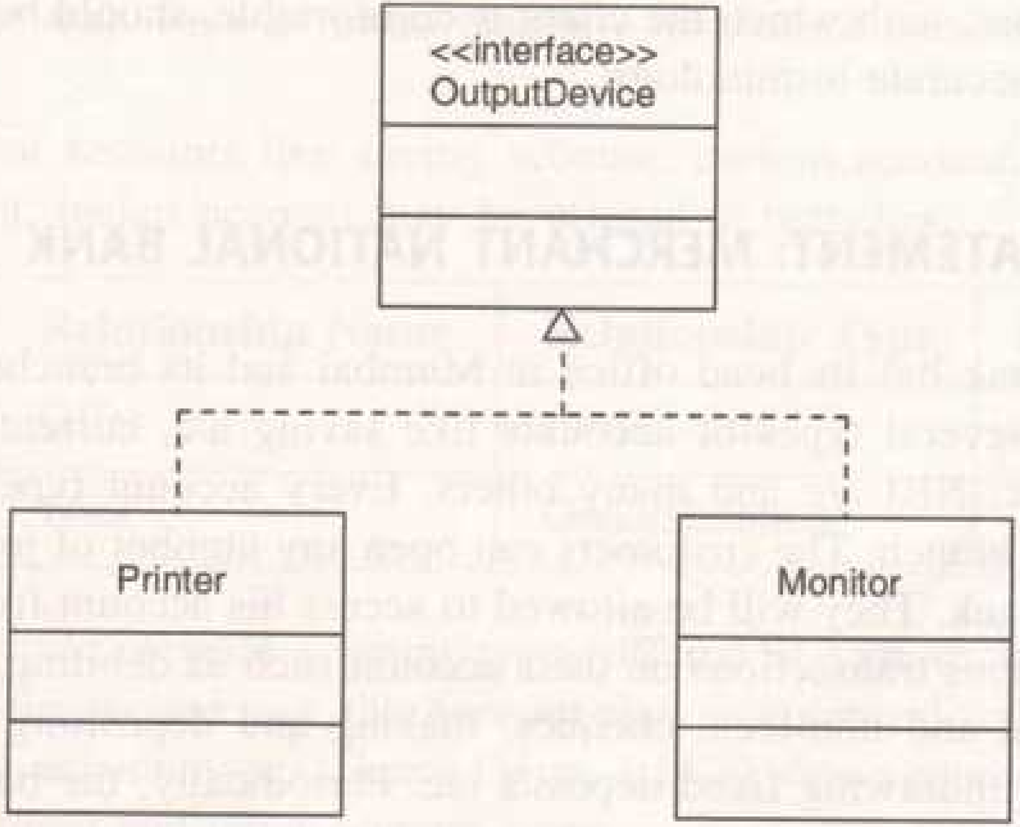
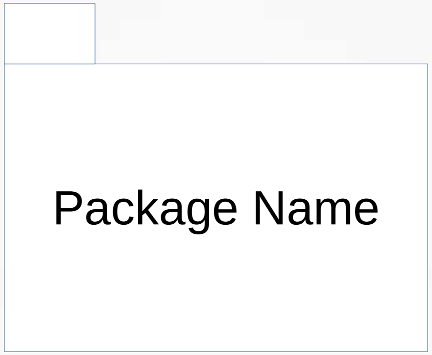


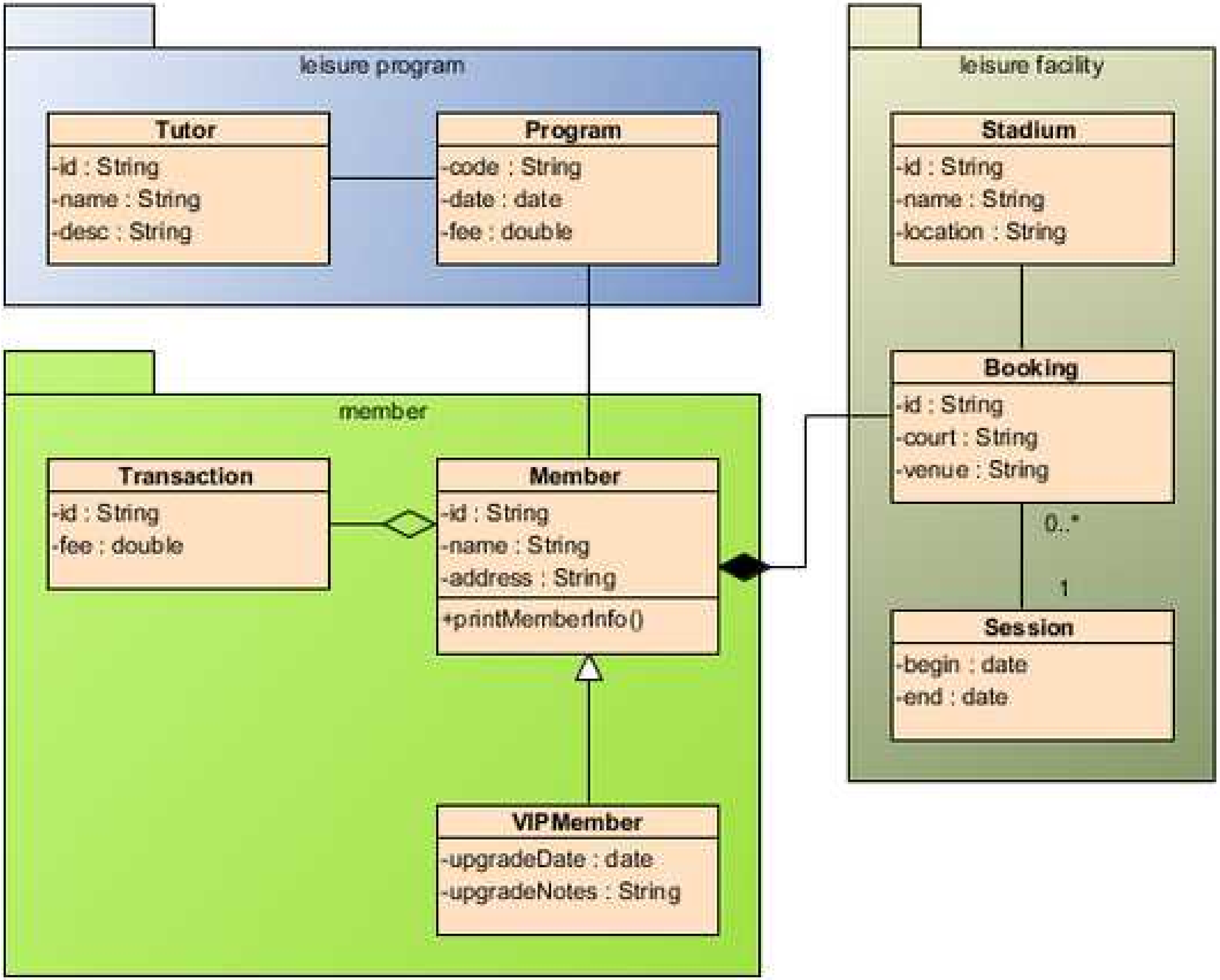
FIGURE 4.12 Representing interface output design.

UNIT -4 Class Diagram

- **Package**
  - A Package provides the **ability to group classes and/or interfaces that are either similar in nature or related.**

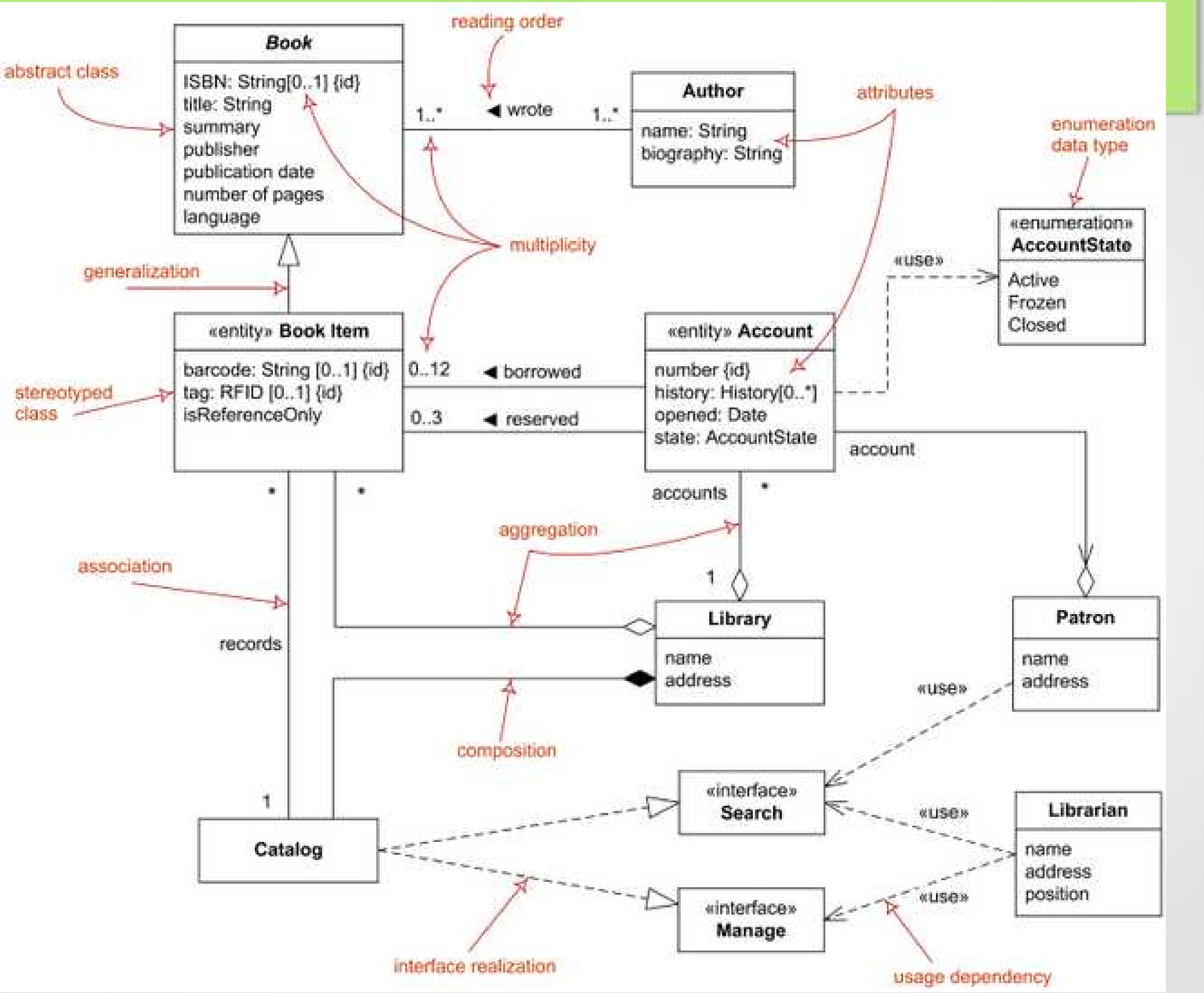


Package Icon


leisure program
Tutor
-id : String
-name : String
-desc : String
Program
-code : String
-date : date
-fee : double
member
Transaction
-id : String
-fee : double
Member
-id : String
-name : String
-address : String
+printMemberInfo()
VIPMember
-upgradeDate : date
-upgradeNotes : String
leisure facility
Stadium
-id : String
-name : String
-location : String
Booking
-id : String
-court : String
-venue : String
0..*
1
Session
-begin : date
-end : date

GUIDELINES FOR DESIGN OF A CLASS DIAGRAM

- Requires different approach in different cirumstances.
- During analysis, a domain analysis class diagram will be drawn first.
- Identify responsibilities of each domain class
- Visibility should be shown in design models.
- Should follow lanuage naming conventions.
- Association class is modelled on the anlaysis diagram and association name is not specified in that case.
- Draw a dashed line for association class.
- Class name should be singular
- Class names should be as per client’s requirements.

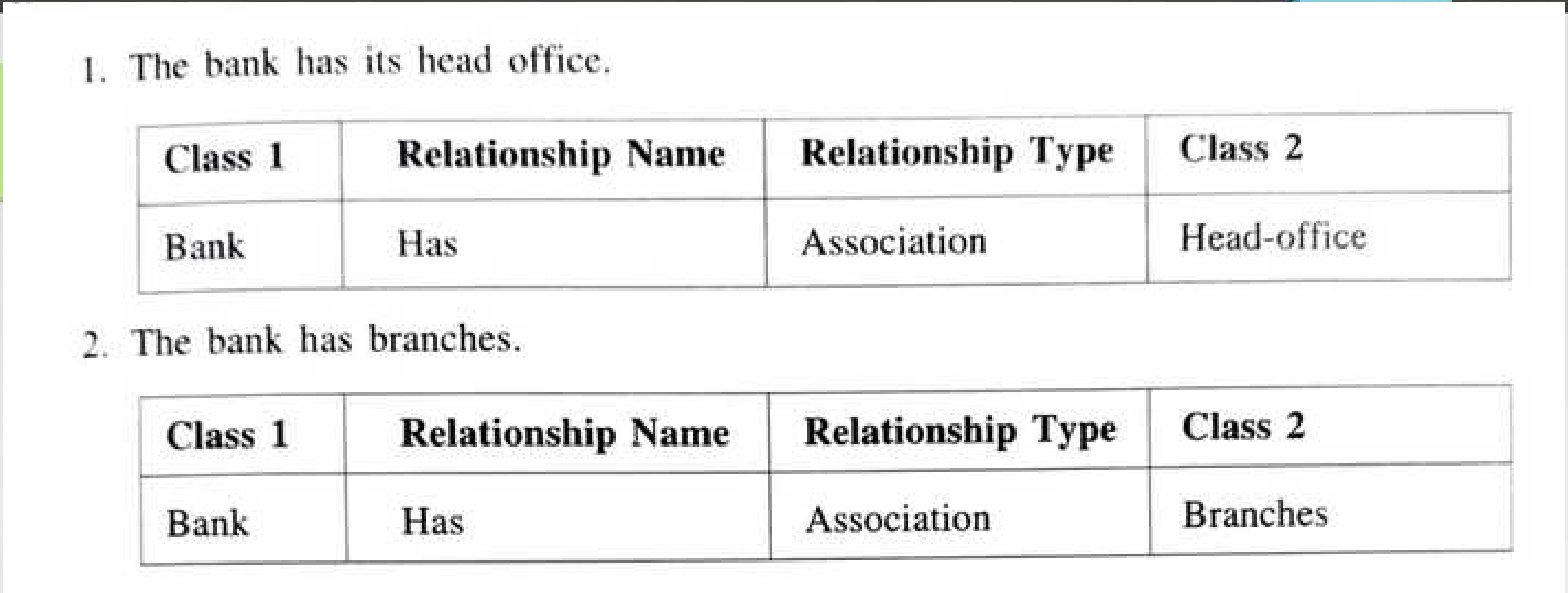

reading order
abstract class
Book
ISBN: String[0..1] {id}
title: String
summary
publisher
publication date
number of pages
language
1..*
◀ wrote
1..*
Author
name: String
biography: String
attributes
enumeration data type
«enumeration» AccountState
Active
Frozen
Closed
multiplicity
generalization
«use»
«entity» Book Item
barcode: String [0..1] {id}
tag: RFID [0..1] {id}
isReferenceOnly
stereotyped class
0..12
◀ borrowed
0..3
◀ reserved
«entity» Account
number {id}
history: History[0..*]
opened: Date
state: AccountState
account
accounts
*
*
*
aggregation
association
1
Library
name
address
Patron
name
address
records
composition
«use»
1
Catalog
«interface» Search
«use»
Librarian
name
address
position
«interface» Manage
«use»
interface realization
usage dependency

CASE STUDY: MERCHANT NATIONAL BANK

- Merchant national bank has its head office in Mumbai and its branches in several parts of India. The bank has several types of accounts like saving a/c, current a/c, PPF a/c, fixed deposit a/c, locker a/c, NRI a/c, and many others. Every account type may or may not be offered for a particular branch. The customers can open any number of joint or single accounts in any branch of the bank. They will be allowed to access his account from any of the branch.

  The customers do various transactions on their account such as debiting, crediting, depositing and withdrawing local and non-local cheqes, making and depositing drafts, operating the locker, making and withdrawing fixed deposits, etc.

  Periodically, the bank calculates interest as per the current rates of interest and credits it to the accounts of the account holders. From time to time, the head office requires management reports from the branches. The management of the branch office also requires daily, weekly and monthly, quaterly and annual reports of the operations.



1. The bank has its head office.

Class 1	Relationship Name	Relationship Type	Class 2
Bank	Has	Association	Head-office

2. The bank has branches.

Class 1	Relationship Name	Relationship Type	Class 2
Bank	Has	Association	Branches

Both the above examples are simple associations with 1:1 and 1:* multiplicity respectively as shown in Figure 4.14(a).

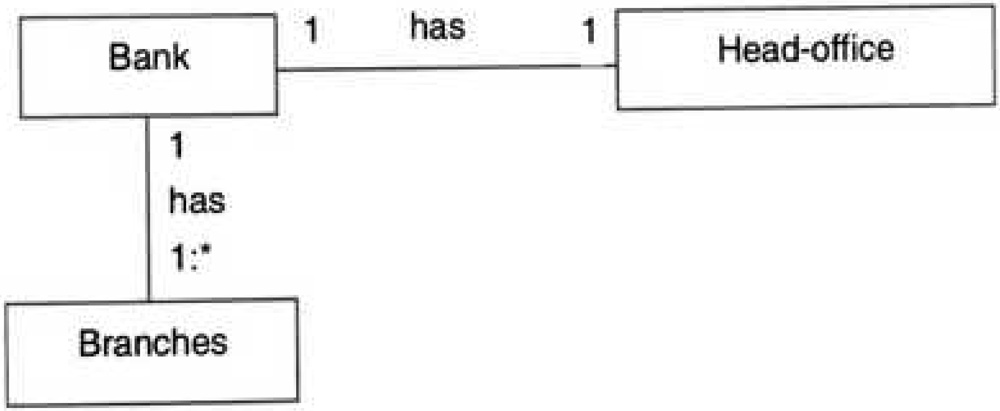
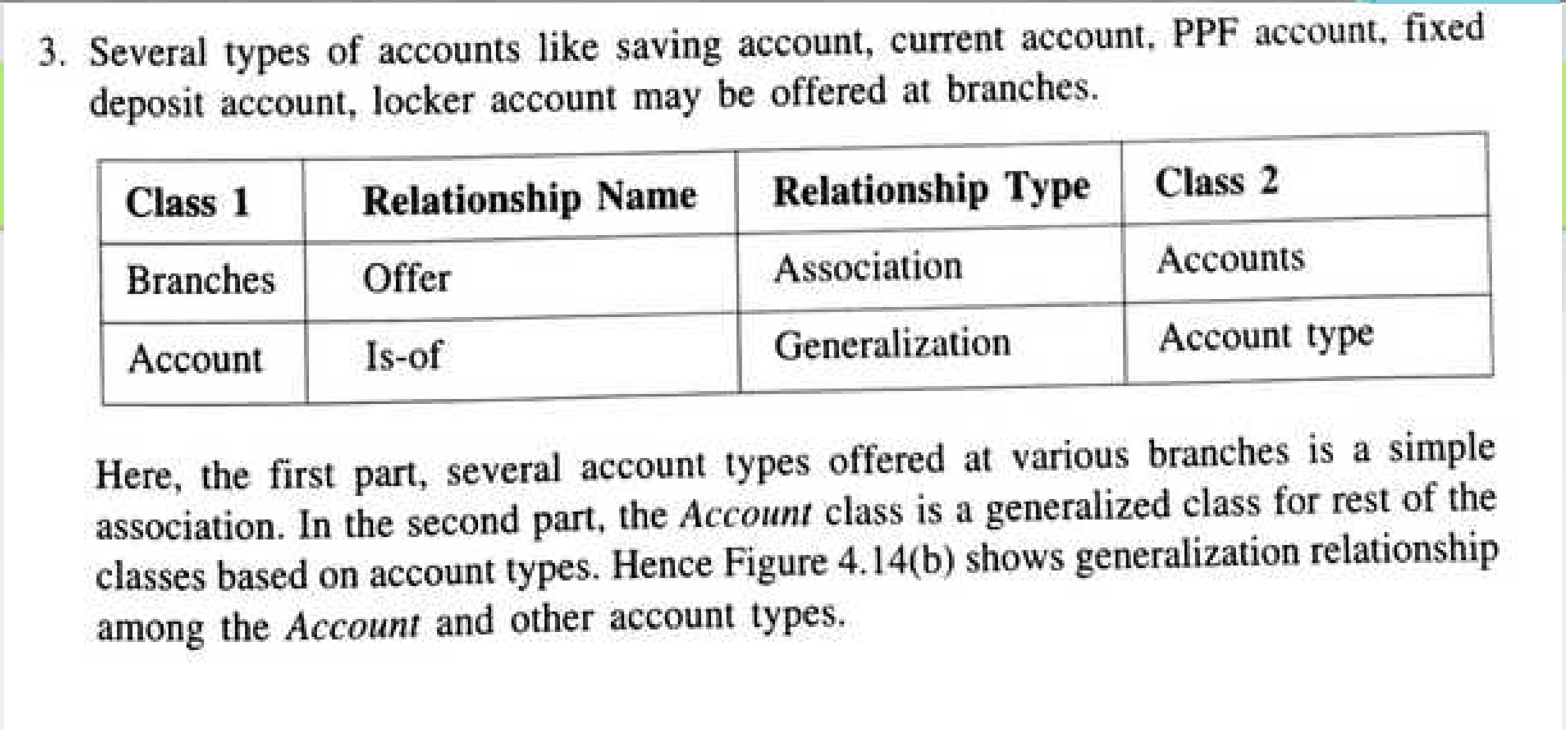


FIGURE 4.14(a) Part of class diagram—Merchant National Bank.



3. Several types of accounts like saving account, current account, PPF account, fixed deposit account, locker account may be offered at branches.

Class 1	Relationship Name	Relationship Type	Class 2
Branches	Offer	Association	Accounts
Account	Is-of	Generalization	Account type

Here, the first part, several account types offered at various branches is a simple association. In the second part, the *Account* class is a generalized class for rest of the classes based on account types. Hence Figure 4.14(b) shows generalization relationship among the *Account* and other account types.

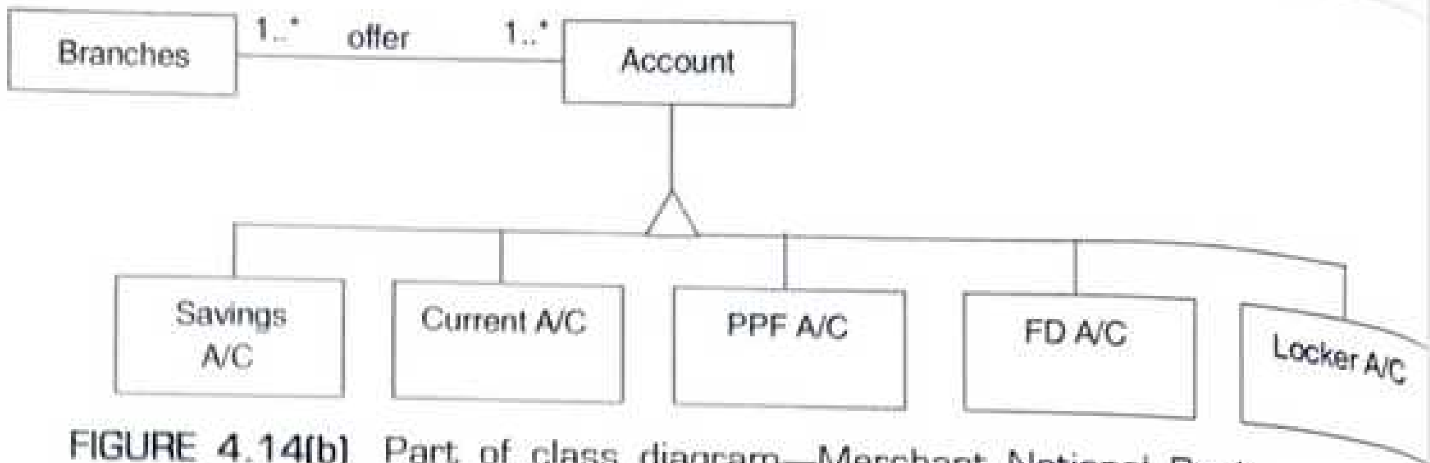
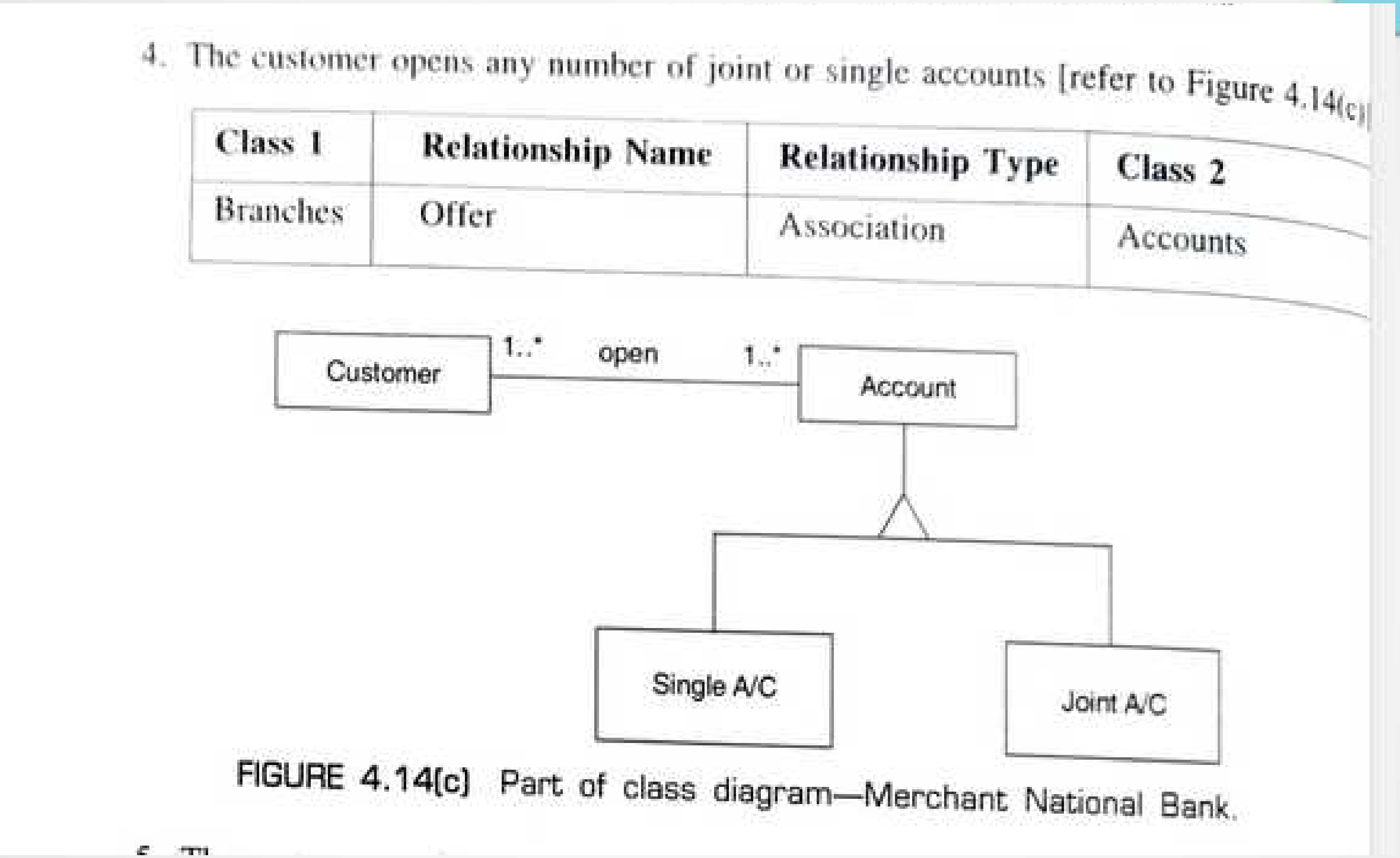


FIGURE 4.14(b) Part of class diagram—Merchant National Bank.

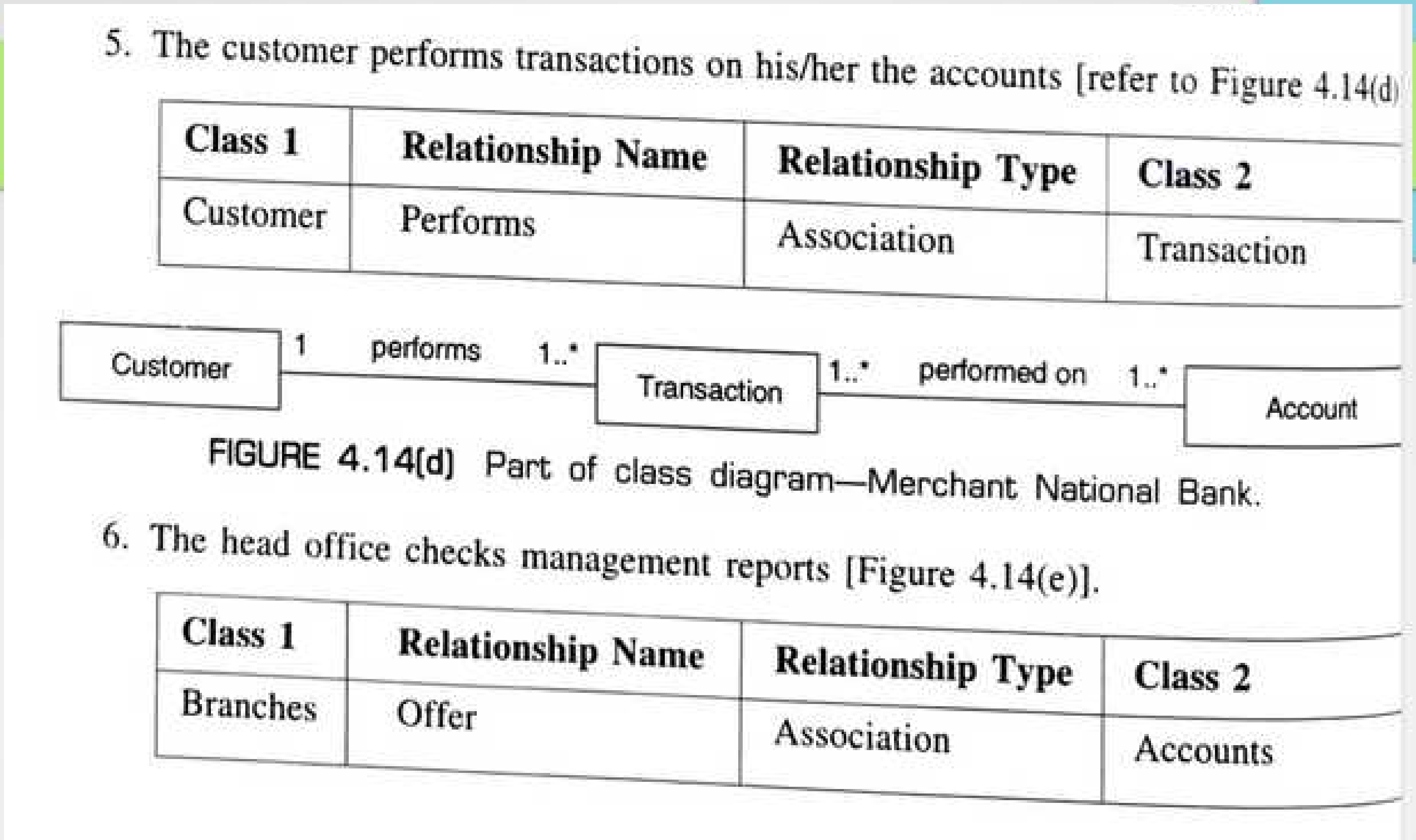


4. The customer opens any number of joint or single accounts [refer to Figure 4.14(c)]

Class 1	Relationship Name	Relationship Type	Class 2
Branches	Offer	Association	Accounts

Customer
1..*
open
1..*
Account
Single A/C
Joint A/C

FIGURE 4.14(c) Part of class diagram—Merchant National Bank.



5. The customer performs transactions on his/her the accounts [refer to Figure 4.14(d)

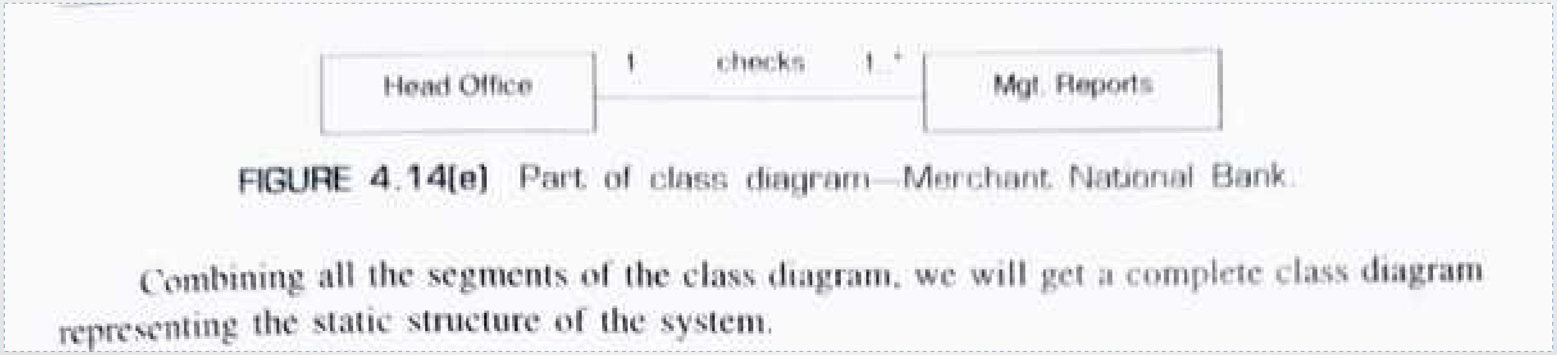
Class 1	Relationship Name	Relationship Type	Class 2
Customer	Performs	Association	Transaction

Customer
1
performs
1..*
Transaction
1..*
performed on
1..*
Account

FIGURE 4.14(d) Part of class diagram—Merchant National Bank.

6. The head office checks management reports [Figure 4.14(e)].

Class 1	Relationship Name	Relationship Type	Class 2
Branches	Offer	Association	Accounts



Head Office
1
checks
1..*
Mgt. Reports

FIGURE 4.14(e) Part of class diagram—Merchant National Bank.

Combining all the segments of the class diagram, we will get a complete class diagram representing the static structure of the system.

UNIT -4 Class Diagram – Case Study

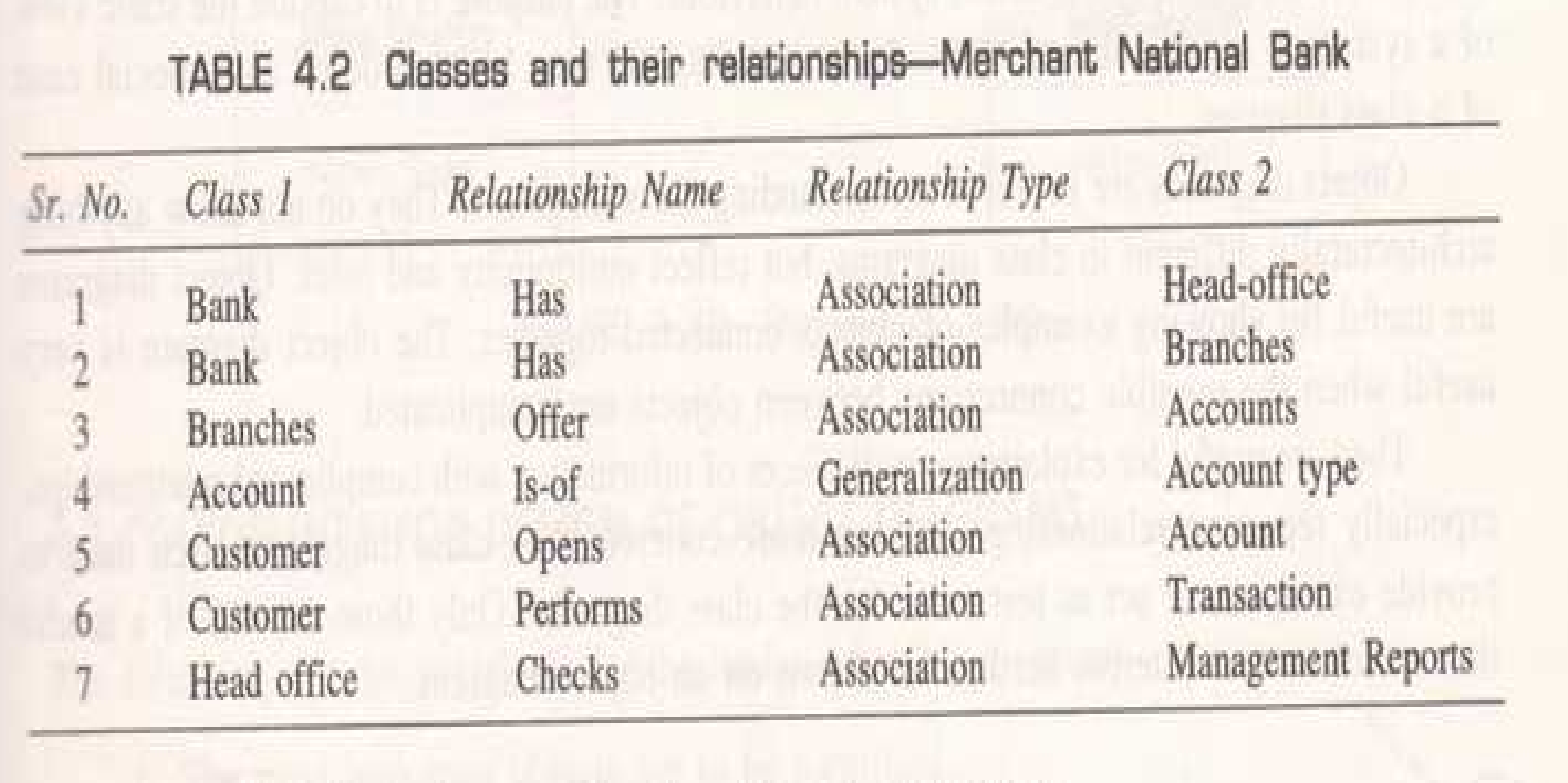


TABLE 4.2 Classes and their relationships—Merchant National Bank

Sr. No.	Class 1	Relationship Name	Relationship Type	Class 2
1	Bank	Has	Association	Head-office
2	Bank	Has	Association	Branches
3	Branches	Offer	Association	Accounts
4	Account	Is-of	Generalization	Account type
5	Customer	Opens	Association	Account
6	Customer	Performs	Association	Transaction
7	Head office	Checks	Association	Management Reports

UNIT -4 Class Diagram – Case Study

Figure 4.15 shows the complete class diagram of the bank.

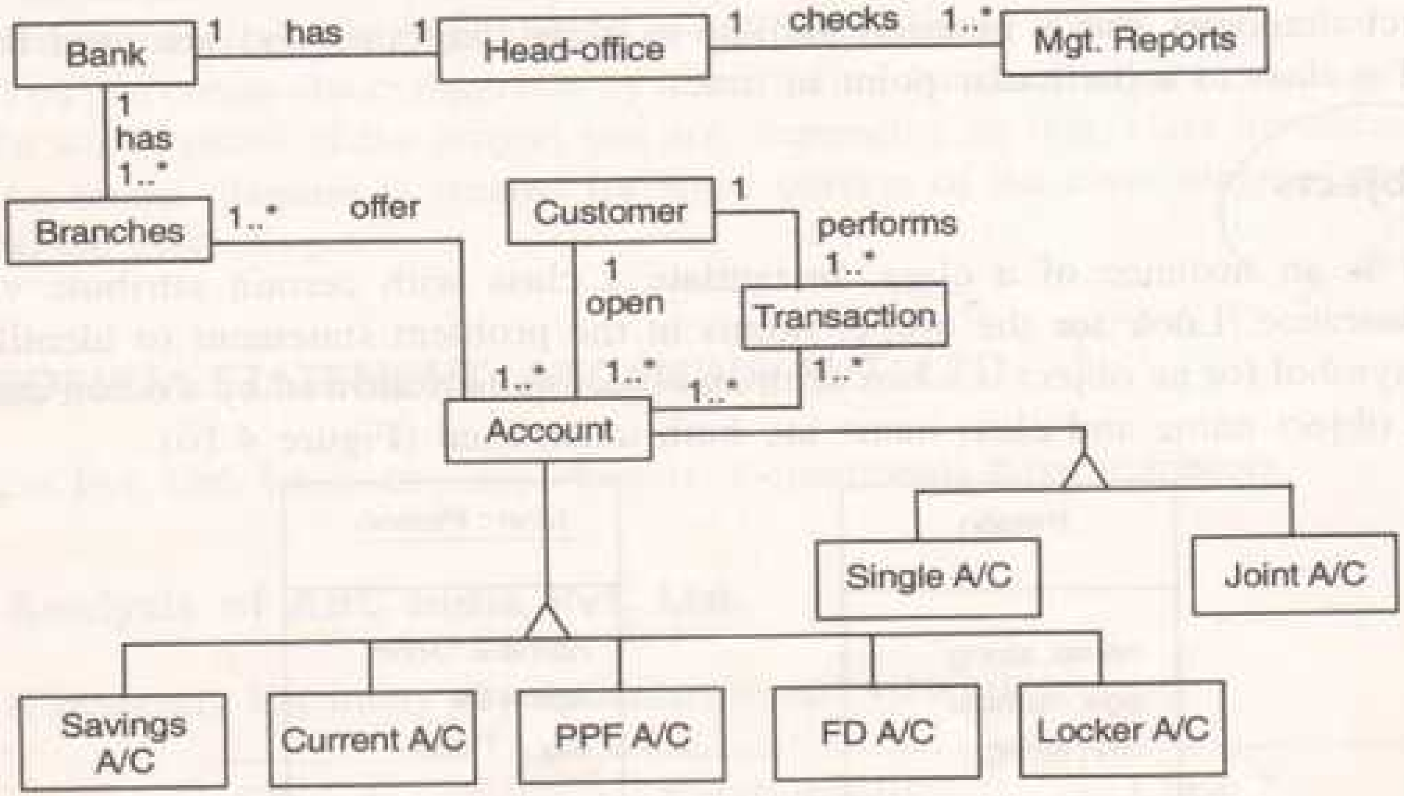


FIGURE 4.15 Complete class diagram—Merchant National Bank.

CASE STUDY 2.

STUDENT LOAN SYSTEM

- A University gives loans to students. Before getting a loan, there is an evaluation process afer which if the loan is approved, agreement is reached. A transaction records each step of the evaluation process, and another transaction records teh overall loan agreeement. A student can take any number of loans, but only one can be active at any time. Each loan is initiated by a seperate transaction. Then, the student repays the loan with a series of repayments. Each repayment transaction is recorded. After the complete settlement, finally the loan account is closed.

  Two output functions are desired:

  1. an inquiry function that prints out the loan balance for any student

STUDENT LOAN SYSTEM

2. a repayment acknowledgement sent to each student after payment is received by the university.

The university loan office decides to implement the student loans on a single processor. Inquires should be processed as soon as they are received. However, repayment acknowledgements need only be processed at the end of each day.

For the above application,create appropriate diagrams.

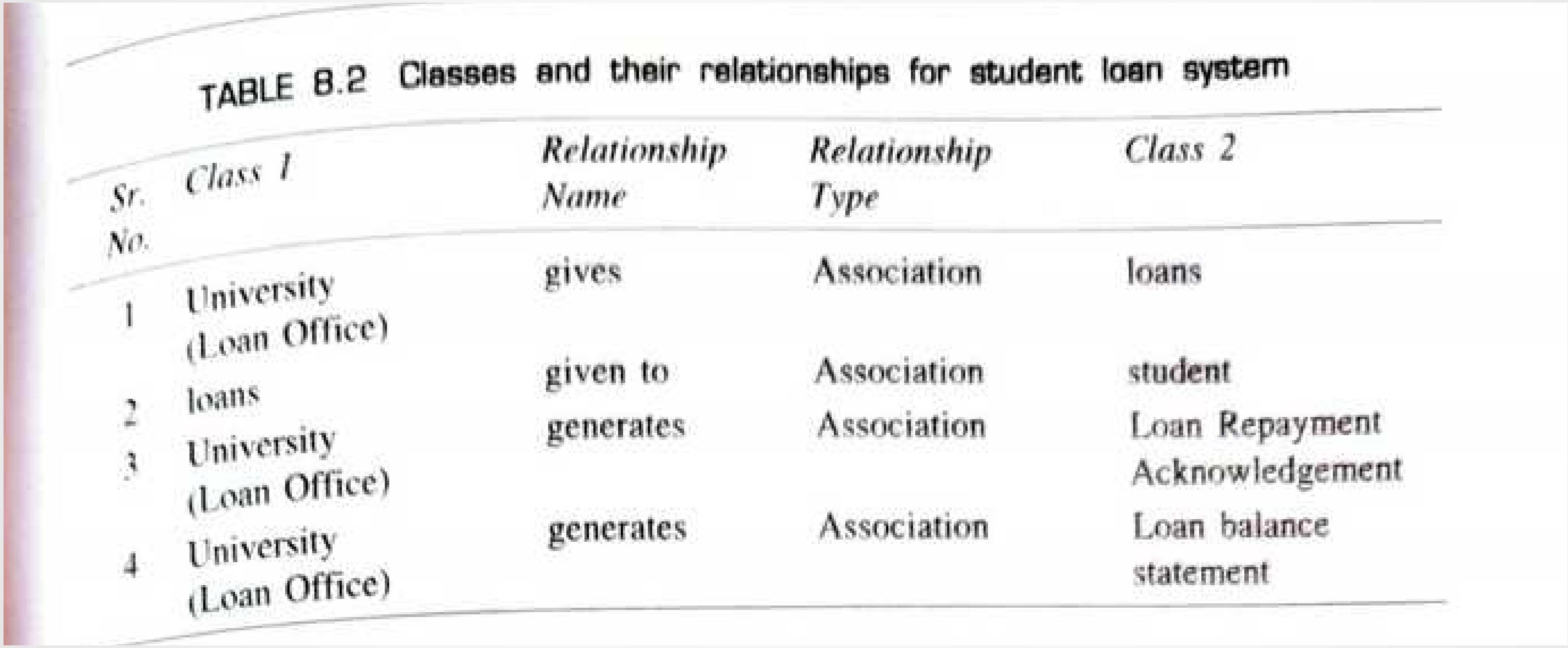
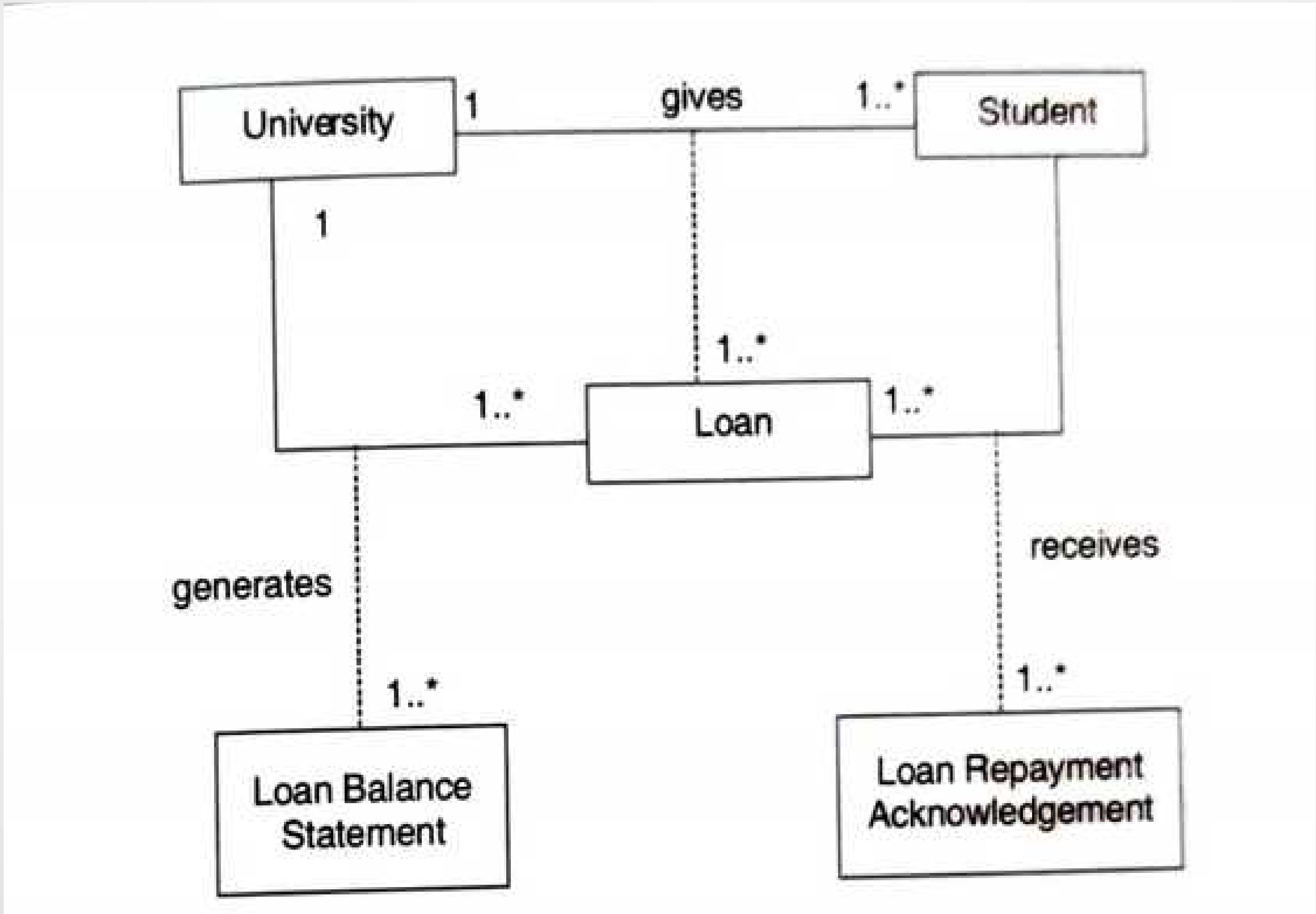


TABLE 8.2 Classes and their relationships for student loan system

Sr. No.	*Class 1*	*Relationship Name*	*Relationship Type*	*Class 2*
1	University (Loan Office)	gives	Association	loans
2	loans	given to	Association	student
3	University (Loan Office)	generates	Association	Loan Repayment Acknowledgement
4	University (Loan Office)	generates	Association	Loan balance statement


University
1
gives
1..*
Student
1
1..*
1..*
Loan
1..*
receives
generates
1..*
1..*
Loan Balance Statement
Loan Repayment Acknowledgement

OBJECT DIAGRAM

UNIT -4 Object Diagram

- **Object Diagram**
  - Introduction
  - Elements of Object Diagram
  - Guidelines for design of Object Diagram
  - Case Study

UNIT -4 Object Diagram

- **Object Diagram**
  - Object Diagram **shows complete or partial view of the structure of a modelled system** at a specific time.
  - Object **diagrams sometimes referred to as instance diagrams.**
  - Object diagrams represent an **instance of a class diagram.**
  - Object diagram can be **good option for explaining complex relationship between** classes.
  - **Object diagrams are derived from class diagrams** so object diagrams are dependent upon class diagram.

UNIT - 4 Object Diagram

- Object diagrams are used to render a set of objects and their relationships as an instance hence it is more close to the actual syystem behaviour.
- Do not show anything architecturally different to class diagrams, but reflect mulitiplicity and roles.
- Useful for explaining small information i.e. recursive relationships

UNIT -4 Object Diagram

- **Elements of Object Diagram**
  - **Objects**
    - The object is an instance of class.
    - The UML symbol for an object is a box with an object name followed by a colon and the class name.
    - The object name and the class name are underlined.

UNIT -4 Object Diagram

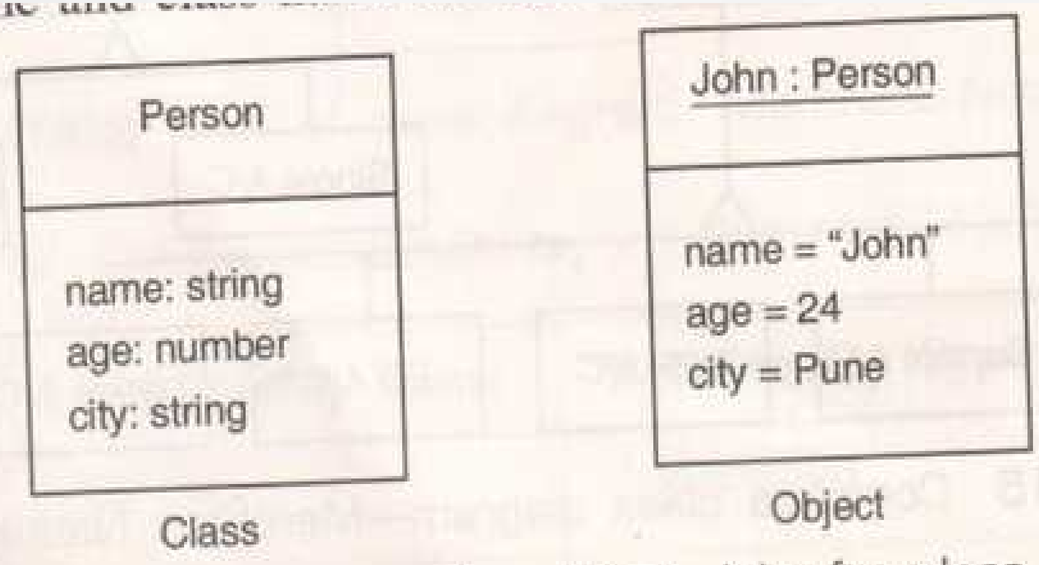


FIGURE 4.16 Representing object *John* for class *Person*.

UNIT -4 Object Diagram

- **Elements of Object Diagram**
  - **Links**
    - A Link is a physical or conceptual connection among objects.
    - UML notation for a link is a line between objects.

UNIT -4 Object Diagram

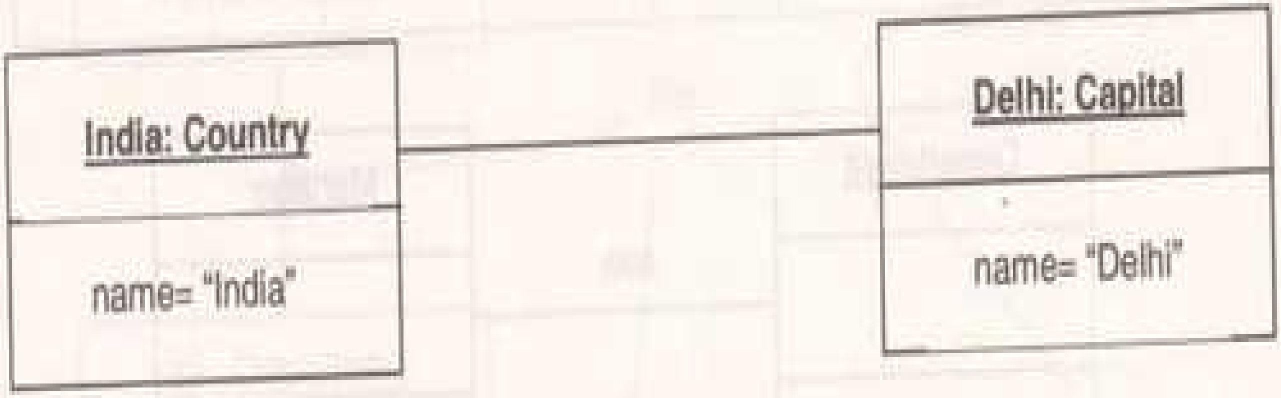


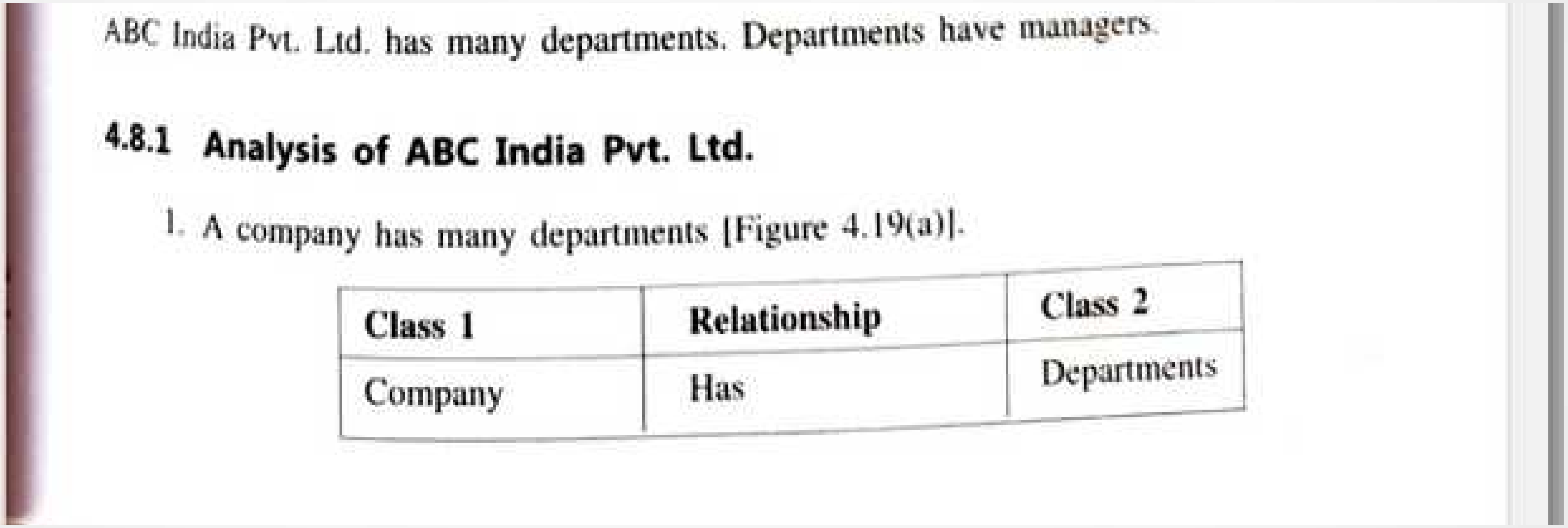
FIGURE 4.18 Representing link.

UNIT -4 Object Diagram

- **Guideline for Design of Object Diagrams**
  - Identify important objects
  - Link among objects should be clarified
  - Values of attributes need to be captured to include in the object diagram.
  - Proper notes should be added where required.
  - Can create object diagrams by instantiating the classes that exist in a class diagram.
  - In which phase of project, depending on that, class instances are created.
  - Object diagram is created for a portion of class diagram.

Problem Statement: ABC India Pvt. Ltd.

ABC India Pvt. Ltd has many departments. Departments have Managers.



ABC India Pvt. Ltd. has many departments. Departments have managers.

4.8.1 Analysis of ABC India Pvt. Ltd.

1. A company has many departments [Figure 4.19(a)].

Class 1	Relationship	Class 2
Company	Has	Departments

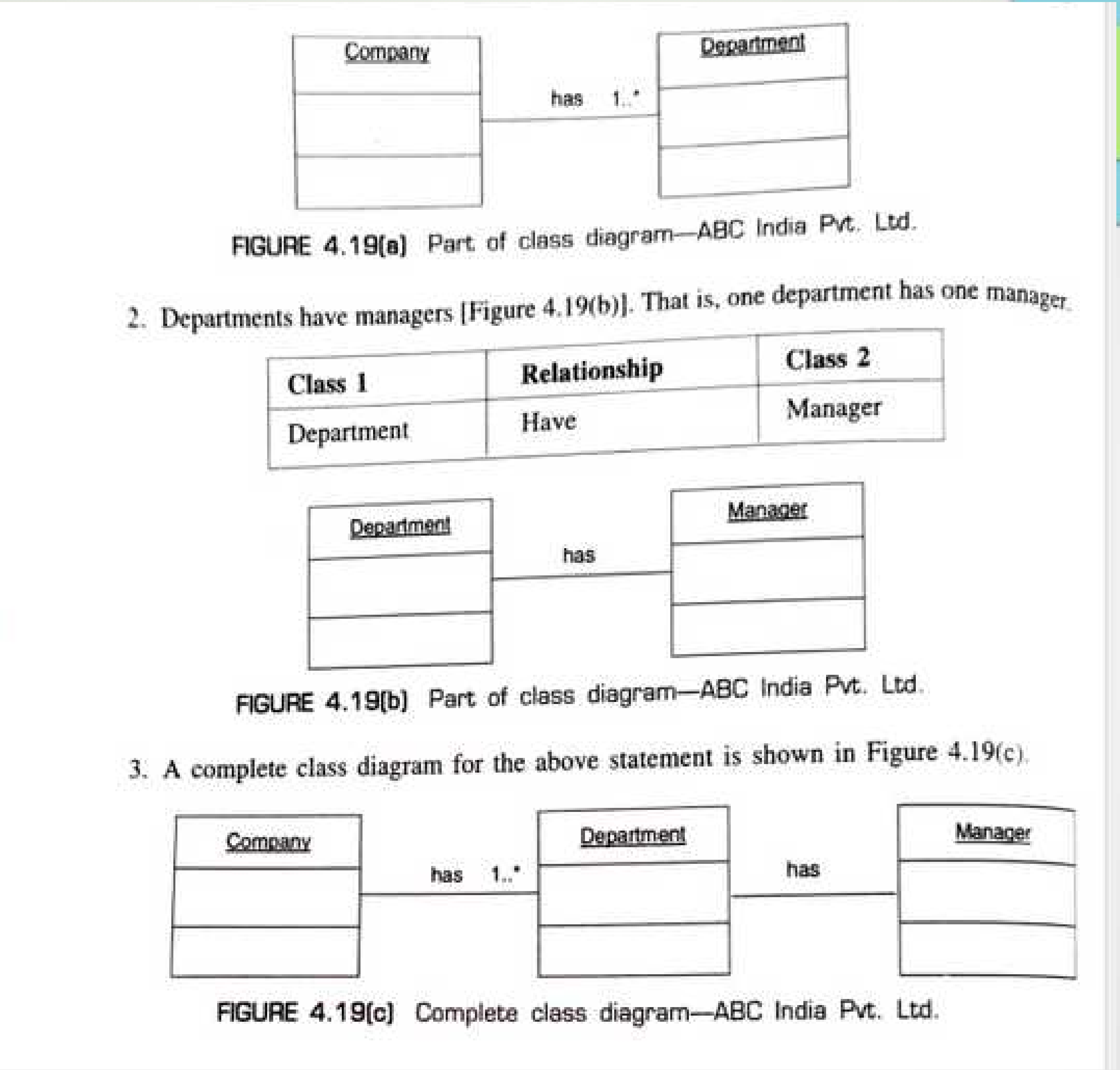


FIGURE 4.19(a) Part of class diagram—ABC India Pvt. Ltd.

2. Departments have managers [Figure 4.19(b)]. That is, one department has one manager.

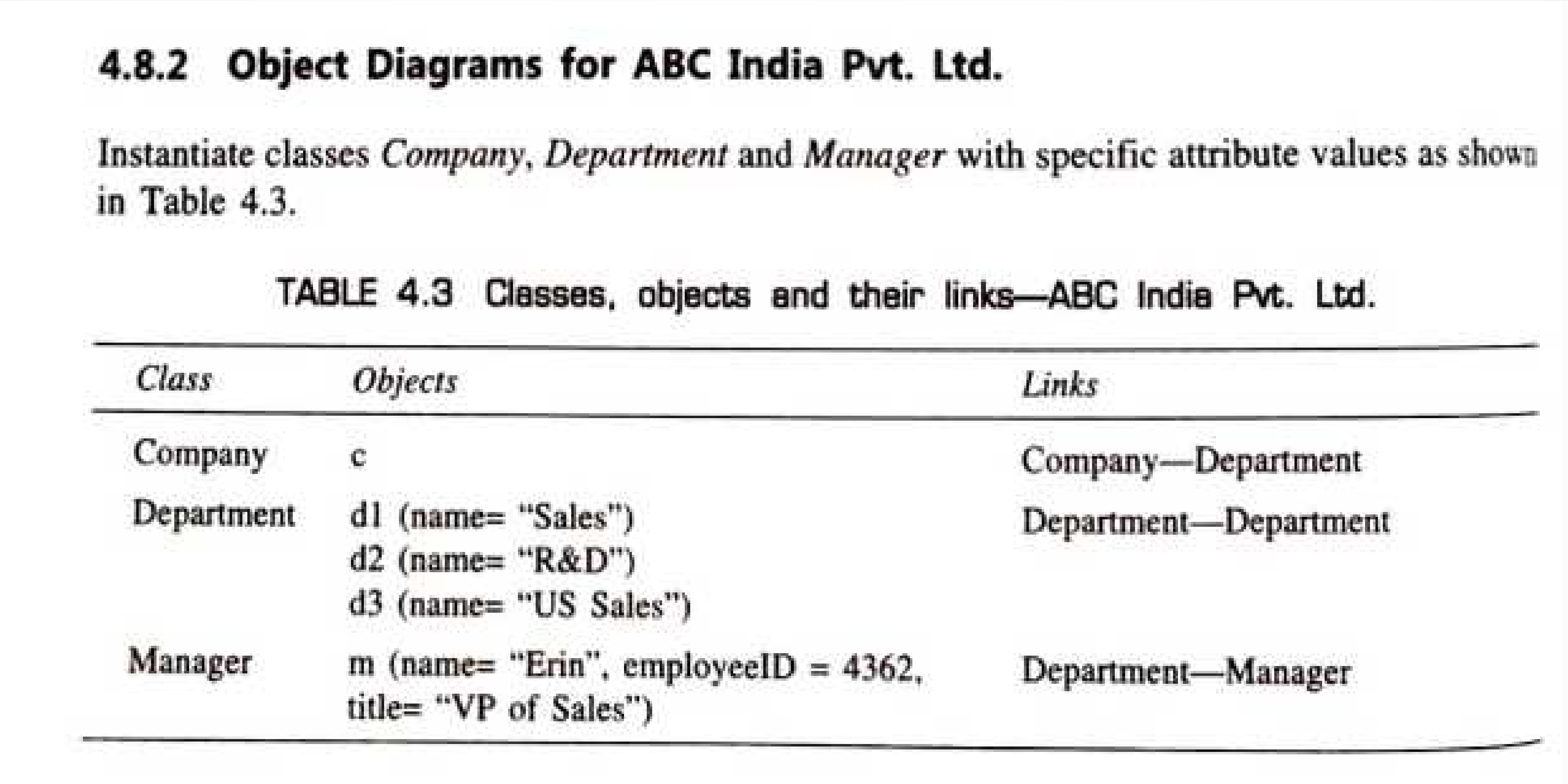
Class 1	Relationship	Class 2
Department	Have	Manager

FIGURE 4.19(b) Part of class diagram—ABC India Pvt. Ltd.

3. A complete class diagram for the above statement is shown in Figure 4.19(c).

FIGURE 4.19(c) Complete class diagram—ABC India Pvt. Ltd.

UNIT -4 Object Diagram Case Study - 1



4.8.2 Object Diagrams for ABC India Pvt. Ltd.

Instantiate classes *Company*, *Department* and *Manager* with specific attribute values as shown in Table 4.3.

TABLE 4.3 Classes, objects and their links—ABC India Pvt. Ltd.

Class	*Objects*	*Links*
Company	c	Company—Department
Department	d1 (name= "Sales") d2 (name= "R&D") d3 (name= "US Sales")	Department—Department
Manager	m (name= "Erin", employeeID = 4362, title= "VP of Sales")	Department—Manager

UNIT -4 Object Diagram Case Study

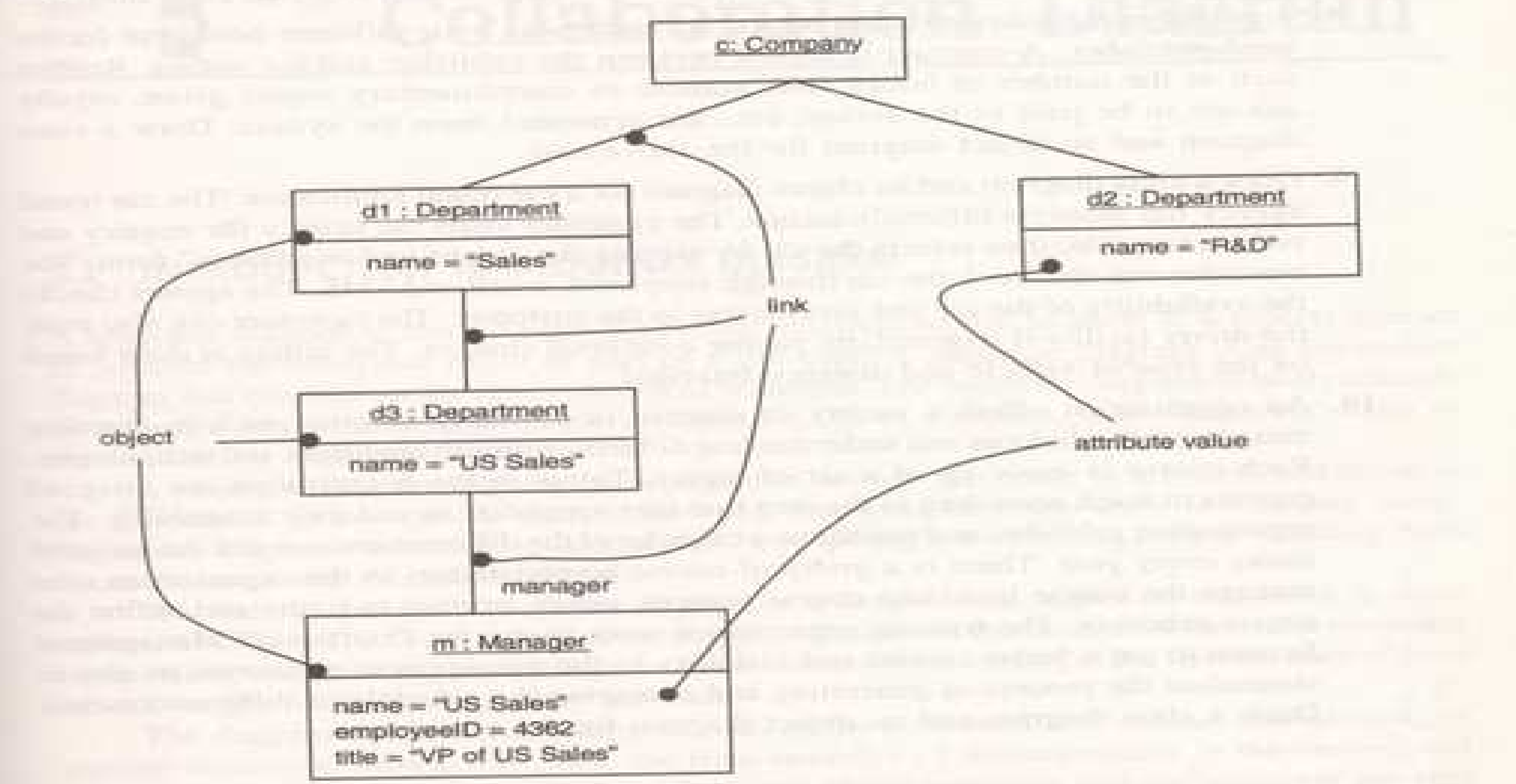


FIGURE 4.20 Object diagram—ABC India Pvt. Ltd.

CASE 2: STUDENT LOAN SYSTEM

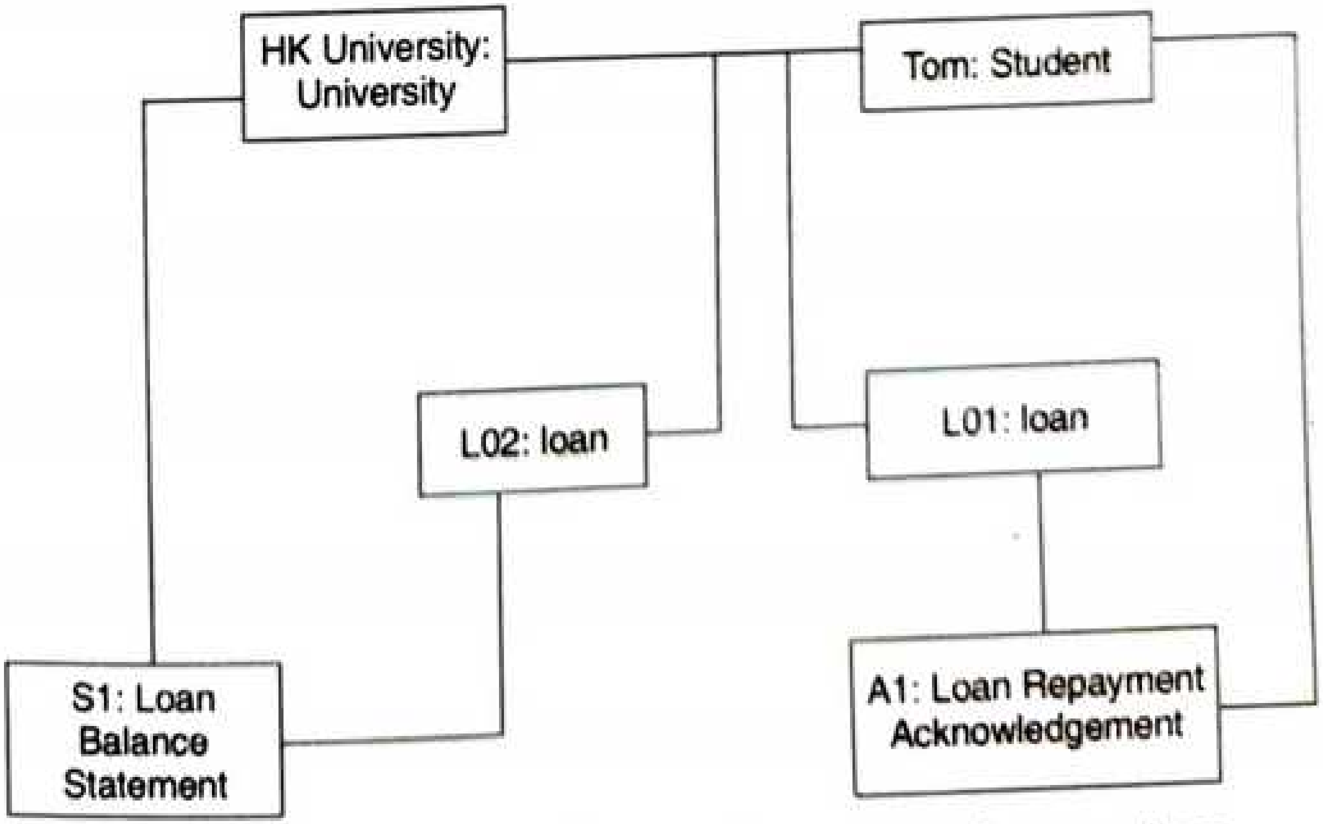

HK University: University
Tom: Student
L02: loan
L01: loan
S1: Loan Balance Statement
A1: Loan Repayment Acknowledgement

FIGURE 8.13 Object diagram for student loan system.