MODULE 9

OBJECT-ORIENTED SYSTEM MODELLING

WORKED EXAMPLES

13.1What is object-oriented modelling?

Object-oriented modelling is a methodology of analyzing requirements of a system with the aim of identifying subsystems with the following desirable properties:

(a) Each subsystem should have clearly specified responsibility of performing a part of overall task.

(b) Other parts of the subsystem should not have to know how a subsystem performs the task assigned to it, rather they should only know what task a subsystem does

(c) Each subsystem should be self-contained and independent

(d) Each subsystem should know what other subsystems do and how to send requests to them for assistance so that it can cooperate with them to get its own job done

(e) Subsystem should hide from outside world the data it uses

(f) The subsystem should be designed to be reusable

13.2 Why is object-oriented modelling used in practice?

Object-oriented modelling is used in practice as it

- Facilitates changing of system to improve functionality during the system life time
- Facilitates reuse of code of each of the subsystems used to design the large system
- Facilitates integrating subsystems into a large system
- Facilitates design of distributed systems

13.3 When is objected-oriented modelling particularly useful?

An object-oriented modelling is particularly useful in the following situations:

- It is required to change an existing system by adding new functionality
- While designing large system and it is found that it can be designed as a collection of existing reusable objects

13.4 Define an object.

An object is an entity, which is uniquely identifiable and permanent relative to the life time cycle of an application. It may be tangible or intangible. Examples of tangible objects are bus, student etc. Examples of intangible objects are bank account, queue data structure etc.

13.5 What is the difference between a class and an object?

A class can be termed as a group of objects having similar behavior and similar attributes. A class is a template or blueprint, which defines all the properties and attributes that an object belonging to it possess. An object is a particular instance of a class. Each object has values assigned to its attributes.

13.6What are subclasses and superclasses? Give examples of each of these

Subclasses are classes which have some common attributes and operations inherited from parent class. The parent class is called superclass.

There may be more than one superclass for a subclass. Though the operations in subclasses are same as superclasses but they may be interpreted in a different way. A subclass overrides the functionality of superclass.

For example, if class furniture is a superclass then its subclasses are tables, chairs, cots etc.

A vehicle is a superclass and its subclasses may be cycle, scooter, car, buses etc.

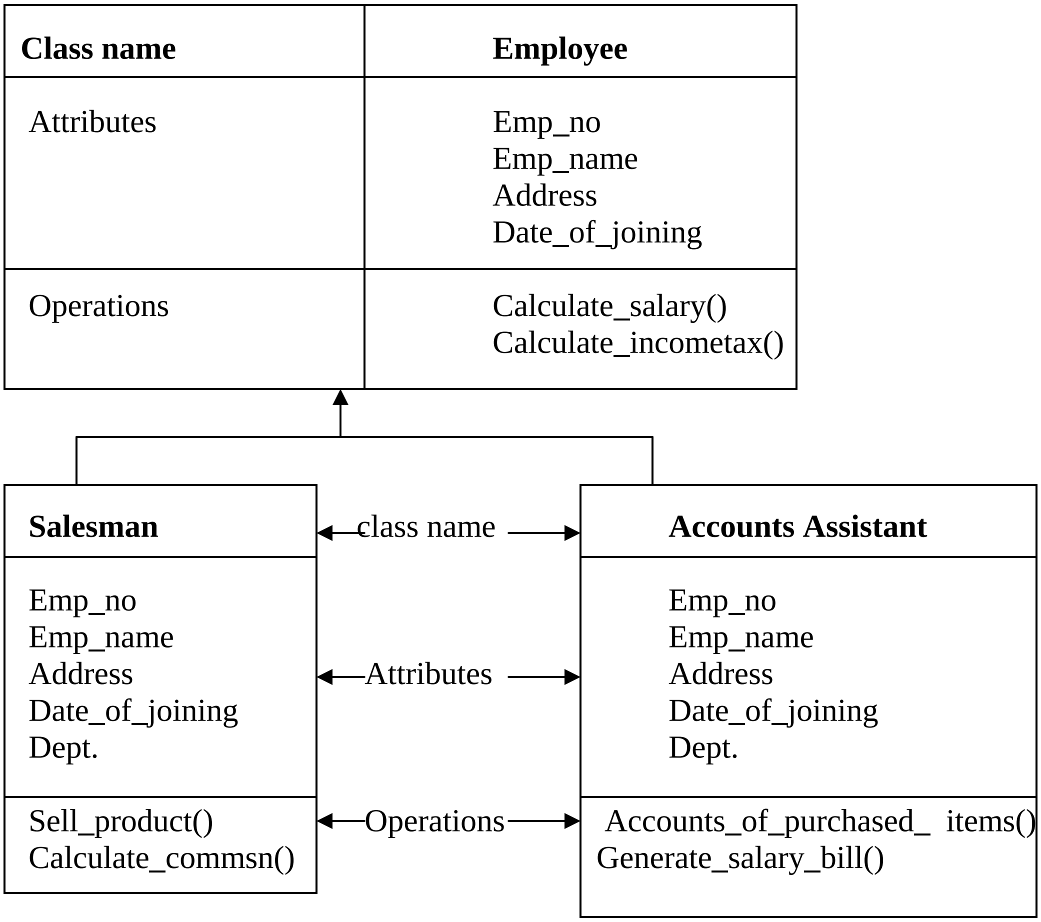
13.7 What do you understand by information hiding? Why is it resorted to in designing information systems? How is it achieved?

Information hiding is the way of hiding the implementation details of an object from the outside world. Any changes in the way the operations are performed by an object is not visible to the client and vice-versa. An object can change the code used by it without affecting the functionality of client. It is achieved by specifying the operations which an object can perform and publicising them.

13.8 What do you mean by inheritance in object-oriented systems? Why is it useful? Give an example of inheritance.

Inheritance is a technique by which the properties and attributes of a class are inherited by a subclass. With inheritance we can refine the subclasses by adding some new attributes and functionality.

For Example, the Employee class shown below is the superclass and the classes Salesman and Accounts_Assistant are inherited classes.



13.9What do you understand by the term polymorphism in object-oriented system? Why is it useful? Give an example of polymorphism.

By polymorphism we mean , the ability to manipulate objects of different distinct classes using only knowledge of their common property.

For example: An operation reservation would be interpreted appropriately by a class describing a train using rules relevant to the railway. The same operation reservation would be interpreted differently by another class representing an airline.

13.11Pick objects and model the following requirements statement using the object. “A magazine is printed monthly and posted to its subscribers. Two months before the expiry of subscription, a reminder is sent to the subscribers. If subscription is not received within a month, another reminder is sent. If renewal subscription is not received upto two weeks before the expiry of the subscription, the subscriber’s name is removed from the mailing list and subscriber is informed”.

Class: Magazine Publishing Company

Attributes:

Magazine_id, Sub_id
Company_address
Magazine_name

Operations:

Request_subscn(magazine_id)

Class: Subscription Accounts

Attributes:

Magazine_id
Sub_id
Sub_address
Subcn_exp_date
Subscn_amount

Operations:

Send_first_reminder(sub_id)
Send_second_reminder(sub_id)
Delete_subscriber(sub_id)
Update_subscriber_account(sub_id)

Magazine Publishing Company Class

Class: Magazine Publishing Company

Superclass: None

Subclass: None

Collaborators: Subscription Accounts

Description: The class represents the company, which prints many magazines monthly and sends magazine with magazine_id to the subscription accounts class

Private responsibilities:

Request_subscn_accounts processing

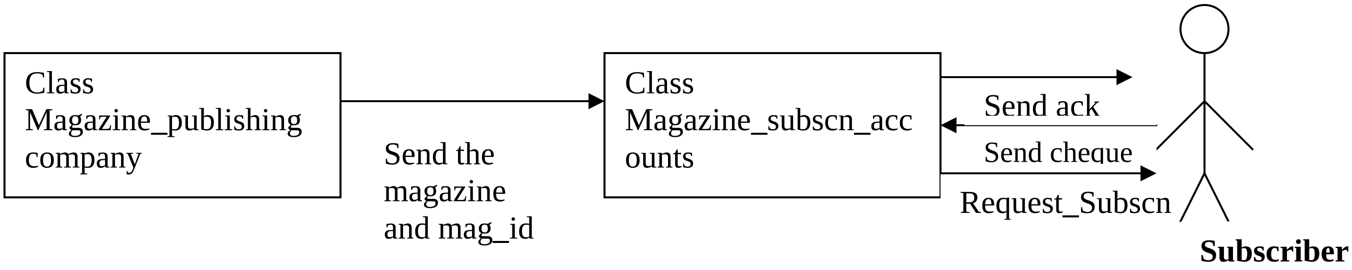
Delete the subscriber's name from the mailing list and send acknowledgement

Contract(s) with collaborator(s):

Dispatch the receipt after receiving the money

Subscription Accounts Class

Class: Subscription Accounts
Superclass: None
Subclass: None
Collaborators: Magazine Publishing Company
Description: This class processes subscription accounts
Private responsibilities: Send Subscription, reminders, update subscribers
Contractor(s) and Collaborators: Send confirmation along with money for subscription Send acknowledgement after receiving magazine



Collaboration graph

13.12Give brief requirements specification for a bus ticket reservation system. Model it using objects.

For making reservation for a bus a passenger has to fill-up a form, which has field specifications like name, address, date of journey, destination etc. After filling up the form the ticket is issued from the counter after checking for the availability of the seats. A passenger has an option to cancel his ticket.

The two classes taken here are ticket class and bus class. The ticket class sends a message to bus class regarding the availability of seats and gets the confirmation from the bus class.

Class:	Ticket
Attributes:	Bus_No Ticket_No Date_of_issue Date_of_journey Starting_Point Destination Passenger_Name Fare Departure_Time Checkin_Time

Operation:	Issue_ticket()
	Cancel_ticket()
	Query()

Class:	Bus
Attributes:	Bus_No No_of_Seats Availability Route_No
Operation:	Confirm()

Class:	Passenger
Attributes:	Passenger_Name Address Ph. No Date_of_Journey Bus_No Route_No
Operation:	Booking_Ticket()

Ticket Class

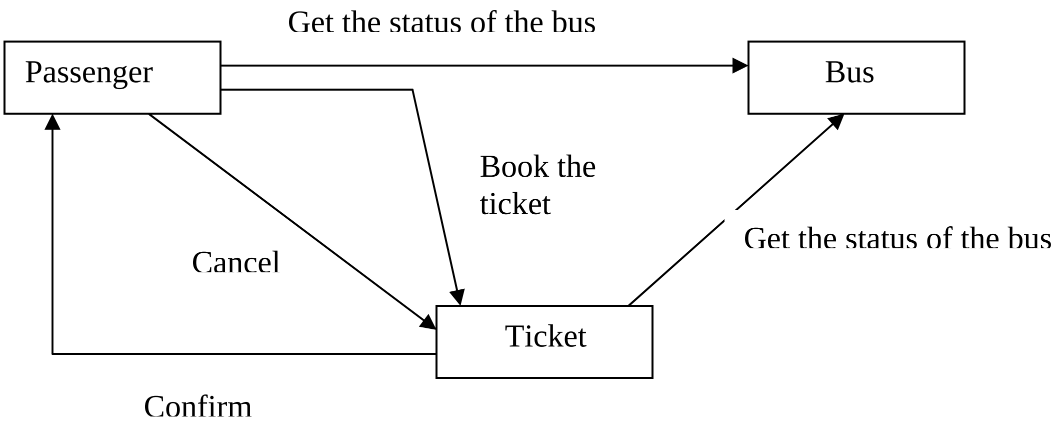
Class: Ticket
Superclass: None
Subclass: None
Collaborators: Bus, Passenger
Description: Issue ticket to the passenger
Private responsibilities: Issue the ticket to the passenger if it is confirmed Cancel the ticket and return the money
Contractor(s) and collaborator(s) Send the query to Bus class for getting the status

Passenger Class

Class: Passenger
Superclass: None
Subclass: None
Collaborator(s): Bus
Description: This class gets the details about availability of bus and seats
Private responsibilities:
Contractor(s) and Collaborator(s): Send query to the bus class to get the bus no. and the availability of bus and seats

Bus Class

Class: Bus
Superclass: None
Subclass: None
Collaborator(s): passenger, Ticket
Description: Send the information regarding status of bus and seats
Private responsibilities:
Contractor(s) and collaborator(s): Send the confirmation regarding the availability of seats and bus to the passenger and ticket class



Collaboration graph