Learning Objectives

- Objects and their properties
- Identifying objects in an application
- Modeling systems with object
Motivation

- Information Systems are becoming very complex.
- We thus need methods to design complex systems.
- Main method is to break up a large system into a number of cooperation components and designing each component or subsystem separately.
- Question: How do we do this?
- The main purpose of this module is to answer this question.
Desirable Properties Of Components

Each subsystem or component must

• Have clearly defined responsibility
• Acts when requested by an "order"
• How the component does its task need not be known to other components
• What the component does should be known
Desirable Properties Of Components (Contd)

• Components must be general enough to be reusable
• Variety of components should be reduced-this is facilitated by allowing components to inherit properties of other components
• Another aid to generalize the function of a component is to allow generic commands which make components do their task
• This is called POLYMORPHISM
Object Oriented Modeling

Use of component oriented design

• Facilitates changes in the system at low cost

• Promotes reuse of components

• Problem of integrating components to configure
  large system simplified

• Simplifies design of distributed systems
Object And Their Properties

- All tangible entities in an application can normally be modelled as objects
  For example: A student, a cycle, a train ticket

- Some intangible entities may also be modelled as objects
  For example: a bank account, stack data structure

- Objects with similar meaning and purpose grouped together as CLASS

- A member of a class is an object instance
Characteristics Of Objects

- All objects have attributes

  Example: student: Name
  Roll no
  Address
  Year
  Department
Characteristics Of Objects

- All objects have a state

  Example Ticket: reserved, waiting list
  Student: present, absent

- All objects have set of OPERATIONS which can be performed on them

  Operations determine object behavior

  Example: Admit student
  Cancel ticket
Universal Modeling Language (UML) is an industry standard notation to represent a class.

Example of UML notation for a Class

<table>
<thead>
<tr>
<th>Vendor</th>
<th>CLASS NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor id</td>
<td>LIST OF ATTRIBUTES</td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>Vendor type</td>
<td></td>
</tr>
<tr>
<td>Add vendor</td>
<td>OPERATIONS OR (METHODS)</td>
</tr>
<tr>
<td>Delete vendor</td>
<td></td>
</tr>
<tr>
<td>Find address</td>
<td></td>
</tr>
<tr>
<td>Change address</td>
<td></td>
</tr>
<tr>
<td>Find vendor type</td>
<td></td>
</tr>
</tbody>
</table>
Instance Diagram – UML Notation

- Shows an object instance's attributes and values

Example

A 2546 : VENDOR
VENDORNAME = AD SINGH & CO
VENDOR TYPE = DISTRIBUTOR
VENDOR ADDRESS = 5, MALL
ROAD, KANPUR 208001

Vendor id

Class name

Object name and its Class name

Attributes and their values
Operation Types On Objects

- Constructor - creating new instances of a class
  Deleting existing instance of class
  Example: add new vendor

- Query - accessing state without changing value
  - has no side effects
  Example: find vendor address
Operation Types On Objects

- Update - changes value of one or more attributes
  - affect state of object
  - has side effects
    example: change address of vendor

Implementation of operations on objects called methods
Implementation Of Classes

TERMINOLOGY USED IN OBJECT ORIENTED MODELLING

- **ABSTRACTION**
  Picking necessary operation and attributes to specify objects

- **ENCAPSULATION**
  Hiding implementation details of methods from outside world
Implementation Of Classes (Contd.)

ENCAPSULATION ALSO KNOWN AS INFORMATION HIDING
INFORMATION HIDING ALLOWS IMPROVEMENT OR MODIFICATION OF METHODS USED BY OBJECTS WITHOUT AFFECTING OTHER PARTS OF A SYSTEM
View Of Objects
As Contractors

1) Objects can be thought of contractors who carry out assigned contracts for clients

2) Clients need not know how the contractor carries out its contracts

3) Contractors can modify/improve methods they use to carry out contracts without “informing” clients

4) External interface presented to clients remain same
Inheritance

- New classes are created from current classes by using the idea of inheritance
- New classes inherit attributes and/or operations of existing classes
- Inheritance allows both generalisation and specialisation in modeling
- Specialisation - given student class, arts students and science student are two subclasses
  - Subclasses inherit properties of parents and in addition may have their own special attributes and operations
## Example Of Inheritance

<table>
<thead>
<tr>
<th>Class name</th>
<th>College student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Roll no</td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Address</td>
</tr>
<tr>
<td></td>
<td>Year of study</td>
</tr>
<tr>
<td>Operations</td>
<td>Admit</td>
</tr>
<tr>
<td></td>
<td>Promote</td>
</tr>
</tbody>
</table>

### Science student
- Roll no
- Name
- Address
- Year of study
- Department
- Laboratory name

### Arts student
- Roll no
- Name
- Address
- Year of study
- Department

### Class Name

### Attributes

### Operations
- Admit
- Promote
- Calculate laboratory fee
- Calculate field trip fee
Generalisation/Specialisation

Given a class Eye surgeon we can generalize it to surgeons which will inherit most of the attributes and operations of the eye surgeon.

A general class School, will inherit many properties of middle school, primary school.

Given a class Doctor we can obtain subclasses: Surgeon, Physician, General Practitioner, Consulting Doctor. All these will inherit many properties of doctor and will have their own new attributes and operations.
Polymorphism

- By polymorphism we mean ability to manipulate objects of different distinct classes knowing only their common properties.
- Consider classes hospital & school.

For both the operation admit will be meaningful.
- They will be interpreted differently by each class.
- Advantage of polymorphism is ease of understanding by a client.
- A client gives a generic request - each contractor interprets and executes request as appropriate to the circumstances.
Identifying Objects

- Simple method
  - identify nouns in Requirements specification. These are potential objects
  - Identify verbs in requirements specification. These are potential operations
Criteria For Picking Objects

1) We remind that an object class has many objects as members.
2) Wherever there is no possibility of confusion we use them synonymously.
3) Objects should perform assigned services. In other words they must have responsibilities specified by us.
4) Objects must have relevant attributes which are necessary to perform service. Attributes must have Non-Null values.
Criteria For Picking Objects

5) A class must be essential for functioning of the system

6) Must have common set of attributes and operations
   which are necessary for all occurrences of the objects in
   the class

7) Objects should be independent of implementation of the
   system.
How To Select Objects

1) Potential objects selected from word statement primarily by examining noun phrases
2) All Noun phrases need not be objects
3) If there are some objects whose attributes do not change during the functioning of a system we reject them
   - They are probably external entities
4) We will illustrate selecting objects using examples
Example 1 – Word Statement

ESSENTIALS OF AN ADMISSION PROCESS TO A UNIVERSITY ARE

- Applicants send applications to a university registrar’s office
- A clerk in the registrar's office scrutinizes applications to see if mark list is enclosed and fee paid
- If scrutiny successful applications passed on to the relevant department
Example 1 – Word Statement

- Departmental committee scrutinizes applications sent to it. Applications are ranked. Depending on the seats available decides to admit, wait list or reject. The application is returned with the message to the registrar’s office clerk.
- Registrar's office clerk informs the applicant the result of his applications
Example 1 – Identification Of Objects

POTENTIAL OBJECTS

1. APPLICANT
2. APPLICATION
3. REGISTRAR’S OFFICE CLERK
4. DEPARTEMENTAL (COMMITTEE)

- How to select relevant objects?
- Decision based on answers to following questions
- Does it have attributes?
- Are operations performed on the attributes?
Example 1 – Identification Of Objects

ANSWERS FOR EXAMPLE 1

1. Applicant has attributes. However no operations performed on it. It is not an object in this problem.

2. Application has attributes operations are performed using attributes of application. Result conveyed to applicant. Admit it as an object.
Example 1 – Identification Of Objects (Contd.)

Registrar’s office clerk has attributes, performs operations on application, attributes and not on clerk’s attributes. Thus reject.

4. Department taken as potential object. It has attributes. Operations are performed using attributes. Operations are performed using attributes of application object and also using attributes of department. Thus admit department as an object.
<table>
<thead>
<tr>
<th>CLASS NAME</th>
<th>APPLICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTES</td>
<td>APPLICATION NUMBER</td>
</tr>
<tr>
<td></td>
<td>APPLICANT NAME</td>
</tr>
<tr>
<td></td>
<td>APPLICANT ADDRESS</td>
</tr>
<tr>
<td></td>
<td>MARKS SHEET</td>
</tr>
<tr>
<td></td>
<td>FEE PAID RECEIPT</td>
</tr>
<tr>
<td></td>
<td>DEPT. APPLIED CODE</td>
</tr>
<tr>
<td></td>
<td>APPLN STATUS</td>
</tr>
<tr>
<td></td>
<td>CLERK CODE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCRUTINIZE</td>
</tr>
<tr>
<td>SEND APPLICATION TO DEPT</td>
</tr>
<tr>
<td>SEND RESPONSE</td>
</tr>
<tr>
<td>ADMIT/W.L/REJECT TO APPLICANT</td>
</tr>
<tr>
<td>ATTRIBUTES</td>
</tr>
<tr>
<td>----------------------------------</td>
</tr>
<tr>
<td>SCRUTINIZE APPLICATION</td>
</tr>
</tbody>
</table>
Example 2: Receiving Items Ordered

ABSTRACT OF WORD STATEMENTS

- Receiving office receives several items from vendors
- Receiving office checks delivery note against orders and detects excess/deficient deliveries if any
- Discrepancy note (if any) sent to purchase office
- Receiving office sends items received note to inspection office
- Inspection office physically inspects items received and accepts good items. Bad items returned to vendor

Candidate objects underlined
Example 2: Receiving Items Ordered (Contd.)

- Items accepted note sent to stores office
- Discrepancy note sent to purchase office
- Stores office updates inventory based on items accepted note
- Stores office sends taken into stock report to the accounts office for payment to vendor
- Accounts office sends payments to vendors

Candidate objects underlined
Picking Relevant Objects

POTENTIAL OBJECTS (UNDERLINED IN LAST PPT) ARE:

1. RECEIVING OFFICE     2. ITEMS     3. VENDORS
4. DELIVERY NOTE     5. ORDERS     6. DISCREPANCY NOTE
7. PURCHASE OFFICE     8. ITEMS RECEIVED NOTE
9. INSPECTION OFFICE   10. ACCEPTED ITEMS NOTE
11. STORES OFFICE     12. INVENTORY  13. GOODS TAKEN IN STOCK REPORT
14. ACCOUNTS OFFICE
15. PAYMENT VOUCHER
OBJECTS NOT RELEVANT TO THIS APPLICATION

Items
Orders
Inventory
Goods taken in stock
Payment voucher

RELEVANT OBJECTS

Receiving office – Even though its own attributes are not relevant, its functional attributes are important. These are:

- Delivery note and order to vendor

It thus derives its attributes from these
Relevant Objects

- **VENDORS**
  
  No operations on this object are needed in this application. However, its attributes are necessary as the Accounts office makes payment to vendors.

<table>
<thead>
<tr>
<th>CLASS : VENDORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTES :</td>
</tr>
<tr>
<td>_Vendor code</td>
</tr>
<tr>
<td>Vendor name</td>
</tr>
<tr>
<td>Vendor address</td>
</tr>
</tbody>
</table>

VENDOR is actually an external object. We have thus given only attributes relevant to this application. In general design one would usually define this object more comprehensively.
# Attributes Of Delivery Note And Order To Vendor

<table>
<thead>
<tr>
<th>CLASS : DELIVERY NOTE</th>
<th>CLASS : ORDER TO VENDOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes :</td>
<td>Attributes :</td>
</tr>
<tr>
<td>Receiving clerk id</td>
<td>Order no</td>
</tr>
<tr>
<td>Order no</td>
<td>Vendor code</td>
</tr>
<tr>
<td>Vendor code</td>
<td>Item code</td>
</tr>
<tr>
<td>Delivery date</td>
<td>Item name</td>
</tr>
<tr>
<td>Item code</td>
<td>Qty ordered</td>
</tr>
<tr>
<td>Qty supplied</td>
<td>Units</td>
</tr>
<tr>
<td>Units</td>
<td>Price/Unit</td>
</tr>
<tr>
<td></td>
<td>Order date</td>
</tr>
<tr>
<td></td>
<td>Delivery period</td>
</tr>
</tbody>
</table>
Receiving Office Object

Receiving office is selected as an object. Its attributes are attributes derived from delivery note and order to vendor.

The class diagram is given below:

```
CLASS
RECEIVING OFFICE

Is Part of
DELIVERY
NOTE

Is Part of
ORDER TO
VENDOR
```

## Receiving Office Object

<table>
<thead>
<tr>
<th>CLASS : RECEIVING OFFICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes : Derived as shown in the previous slide</td>
</tr>
<tr>
<td>Operations :</td>
</tr>
<tr>
<td>▪ Compare order no, item code, qty, etc in delivery note with that in order to vendor</td>
</tr>
<tr>
<td>▪ Send discrepancy note (if any) to purchase office and vendor. If no discrepancy send delivery note to purchase</td>
</tr>
<tr>
<td>▪ Send delivery note to inspection office(object)</td>
</tr>
</tbody>
</table>
Other Relevant Objects

<table>
<thead>
<tr>
<th>CLASS : STORES OFFICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes : Attributes of inspection office + qty in stock</td>
</tr>
<tr>
<td>Operations :</td>
</tr>
<tr>
<td>- Update inventory by adding no of items accepted to qty in stock</td>
</tr>
<tr>
<td>- Send advice to accounts object to make payment for qty accepted</td>
</tr>
</tbody>
</table>
Next Object Is
Inspection Office

<table>
<thead>
<tr>
<th>Class</th>
<th>Inspection Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Derived attributes from delivery note + no of items accepted</td>
</tr>
<tr>
<td>Operations</td>
<td>- Send information an accepted items to store and accounts</td>
</tr>
<tr>
<td></td>
<td>- Send discrepancy note( if any) to purchase office and vendor</td>
</tr>
</tbody>
</table>
Other Objects Are

<table>
<thead>
<tr>
<th>Class</th>
<th>Accounts Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Derived from inspection office attributes + price/unit of item</td>
</tr>
<tr>
<td>Operations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Calculate amount to be paid</td>
</tr>
<tr>
<td></td>
<td>• Print cheque</td>
</tr>
<tr>
<td></td>
<td>• Request vendor object for vendor address</td>
</tr>
<tr>
<td></td>
<td>• Print vendor address label</td>
</tr>
<tr>
<td></td>
<td>• Dispatch payment to vendor</td>
</tr>
<tr>
<td></td>
<td>• Intimate Purchase office of payment</td>
</tr>
</tbody>
</table>
Object Oriented Modelling-crc Method

Steps in object oriented modelling

1) Find objects and their classes
2) Determine responsibilities of each object
3) State responsibilities, that is, actions. It can carry out on its own using its knowledge
4) Determine objects with whom they collaborate.
Object Oriented
Modelling-crc Method (Contd.)

5) State contracts each object assigns to its collaborations
6) A collaborator either performs a requested action or gives information
7) Document each class – its responsibilities, its collaborators and their responsibilities
8) Develop an object interaction/collaboration graph
Crc Team Idea

CRC TEAM: user's representative
    System analyst(s)
    project coordinator

RESPONSIBILITY: Identify objects
    Specify responsibility
    Specify collaborators and their responsibilities

Prepare a card for each class called class index cards
Crc Methodology

1. Make CRC Card for each class

CRC CARD

<table>
<thead>
<tr>
<th>CLASS NAME :</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPER CLASSES AND SUBCLASSES :</td>
</tr>
<tr>
<td>SHORT DESCRIPTION OF CLASS :</td>
</tr>
<tr>
<td>COLLABORATORS :</td>
</tr>
<tr>
<td>PRIVATE RESPONSIBILITIES OF CLASS :</td>
</tr>
<tr>
<td>CONTACTS WITH COLLABORATORS :</td>
</tr>
</tbody>
</table>

Develop a graph to show interaction between classes
Crc Model - Example

For Example 1 of last learning unit the CRC model is given below

<table>
<thead>
<tr>
<th>Class</th>
<th>APPLICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super class</td>
<td>None</td>
</tr>
<tr>
<td>Sub class</td>
<td>None</td>
</tr>
<tr>
<td>Collaborators</td>
<td>DEPARTMENT</td>
</tr>
<tr>
<td>Description</td>
<td>This class represents applications received for admission to a university</td>
</tr>
</tbody>
</table>
Private Responsibilities:
Scrutinize: Applications are scrutinized to see if fee is paid and marks sheet is enclosed. If yes, applications is sent to department class. Else a rejected letter is sent to the applicant.

Contract(s) and Collaborator(s):
Forward application to department: When it passes scrutiny else send reject to applicant
Send letter to applicant: When Department notifies decision (Admit, Reject, Waitlist) send appropriate letter to the applicant
Crc Model – Example (Contd)

<table>
<thead>
<tr>
<th>Class</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super class</td>
<td>None</td>
</tr>
<tr>
<td>Sub class</td>
<td>None</td>
</tr>
<tr>
<td>Collaborators</td>
<td>APPLICATION</td>
</tr>
<tr>
<td>Description</td>
<td>This class represents departments whose responsibility is to admit, reject or place an waiting list on application</td>
</tr>
</tbody>
</table>
Private Responsibilities:
Rank order applications based on selection criteria. Mark in application: admitted, rejected or in waiting list depending on available seats.

Contract(s) and Collaborator(s):
Send reply to application class on admitted, rejected or wait list.
Collaboration Graph For Example2

Vendor

Delivery

Payment

CLASS PURCHASE OFFICE

CLASS RECEIVING OFFICE

Inspect

CLASS INSPECTION OFFICE

Update Inventory

Delivery copy

Discrepancy note

CLASS ACCOUNTS OFFICE

Make payment

CLASS STORES OFFICE

Payment copy

Make payment

Update Inventory
9.1 Computer systems are designed by
   a. simplifying requirements of system
   b. breaking of the system into smaller self-contained co-operating subsystems
   c. breaking up the systems into independent parts
   d. modular design

9.2 Functions and procedures are
   a. not useful in designing computer systems
   b. old fashioned and they are not useful
   c. useful in designing computer systems
   d. have side effects which require special care if they are used as subsystems

9.3 A subsystem of a complex system must specify
   a. what task it performs
   b. how it performs a task
   c. with which subsystems it co-operates
   d. how it co-operates with other systems

9.4 A subsystem of a complex system must
   a. ii, iii
   b. ii, iv
   c. iii, iv

9.5 A subsystem of a complex system
   (i) should be reusable in other complex system
   (ii) must not be able to inherit the properties of other subsystems
   (iii) must have clearly specified responsibilities
   (iv) must know the stimuli to which it should respond
   a. i, ii, iii
   b. ii, iii, iv
   c. i, iii, iv
   d. i, ii, iv

9.6 By polymorphism of a subsystem we mean
   a. it should be reusable
   b. it should have polymorphic data types
   c. it should accept generic commands and interpret appropriately
d. it should morph polygons

9.7 **The advantages of object-oriented modelling are**

a. i, ii  

b. i, iii  

c. ii, iii  

d. i, iv  

9.8 **Objects are**

(i)tangible entities  

(ii)intangible entities  

(iii)transient entities  

(iv) uniquely identifiable  

a. i, ii  

b. i, ii, iii  

c. i, ii, iii, iv  

d. i, ii, iv  

9.9 **A class is**

a. a group of objects  

b. template for objects of a particular type  

c. a class of objects  

d. a classification of objects  

9.10 **All objects have**

(i) attributes  

(ii) states  

(iii)a set of operations  

(iv) a unique identity  

a. i, ii, iii  

b. ii, iii, iv  

c. i, iii, iv  

d. i, ii, iii, iv  

9.11 **In UML diagram of a class**

a. state of object cannot be represented
b. state is irrelevant
c. state is represented as an attribute
d. state is represented as a result of an operation

9.12 Attributes are assigned value
   a. when operations are performed on an object
   b. when instances of objects are defined
   c. when methods are invoked
   d. when classes are identified

9.13 The following are intangible entities which can be defined as objects
   (i) a motor car
   (ii) a bank account
   (iii) an aircraft
   (iv) a linked list
   a. i, ii
   b. ii, iv
   c. iii, iv
   d. ii, iii, iv

9.14 A query operation on a object
   a. has side effect
   b. has no side effects
   c. changes the state of an object
   d. is not allowed

9.15 An instance of an object is created by a
   a. query operation
   b. update operation
   c. constructor operation
   d. open operation

9.16 An update operation in an object instance
   a. updates the class
   b. has no side effects
   c. deletes an instance
   d. alters values of attribute(s) of an object instance

9.17 In object-oriented design
a. operations and methods are identical
b. methods specify algorithms whereas operations only state what is to be done
c. methods do not change values of attributes
d. methods and constructor are same

9.18 **By abstraction in object-oriented modelling we mean picking**

a. only attributes appropriate to model an object
b. only operations
c. both operation and attributes with operations appropriate to model an object
d. the appropriate abstract data type

9.19 **By encapsulation in object-oriented modelling we mean**

a. encapsulating data and programs
b. hiding attributes of an object from users
c. hiding operations on object from users
d. hiding implementation details of methods from users of objects

9.20 **Encapsulation in object-oriented modelling is useful as**

a. it allows improving methods of an object independent of other parts of system
b. it hides implementation details of methods
c. it allows easy designing
d. encapsulates attributes and operations of object

9.21 **Objects may be viewed as**

a. clients in a system
b. servers in a system
c. as both clients and servers in a system
d. neither as clients nor as servers in a system

9.22 **Inheritance in object-oriented system is used to**

a. create new classes from existing classes
b. add new operations to existing operations
c. add new attributes to existing attributes
d. add new states to existing states

9.23 **Inheritance in object-oriented modelling can be used to**
a. generalize classes
b. specialize classes
c. generalize and specialize classes
d. create new classes

9.24 When a subclass is created using inheritance the resulting class
a. may have only attributes of parent class
b. may have only operations of parent class
c. may have new operations only in addition to those in parent class
d. may have new attributes and new operations in addition to those of the parent class

9.25 By polymorphism in object-oriented modelling we mean
a. the ability to manipulate objects of different distinct classes
b. the ability to manipulate objects of different distinct classes knowing only their common properties
c. use of polymorphic operations
d. use of similar operations to do similar things

9.26 A polymorphic operation
a. has same name
b. has same name but uses different methods depending on class
c. uses different methods to perform on the same class
d. uses polymorphic method

9.27 Given a word statement of a problem potential objects are identified by selecting
a. verb phrases in the statement
b. noun phrases in the statement
c. adjectives in the statement
d. adverbs in the statement

9.28 Given a word statement of problem potential operations appropriate for objects are identified by selecting
a. verb phrases in the statement
b. noun phrases in the statement
c. adjectives in the statement
d. adverbs in the statement
9.29 **Objects selected to model a system**

(i) must be essential for functioning of the system  
(ii) must have all attributes which are invariant during operations of a system  
(iii) must have attributes relevant for performing services of object  
(iv) must be able to perform assigned services

a. i, ii, iii  
b. ii, iii, iv  
c. i, iii, iv  
d. i, ii, iii, iv

9.30 **An object is selected for modelling a system provided**

a. its attributes are invariant during operation of the system  
b. its attributes change during operation of the system  
c. it has numerous attributes  
d. it has no attributes relevant to the system

9.31 **An object is considered an external entity in object-oriented modelling**

a. its attributes are invariant during operation of the system  
b. its attributes change during operation of the system  
c. it has numerous attributes  
d. it has no attributes relevant to the system

9.32 **Object-oriented system modelling using CRC method gives**

a. Java programs for the system  
b. C++ programs for the system  
c. Classes of the system, their responsibilities and collaborating classes  
d. Objective C programs for the system

9.33 **The expansion of the acronym CRC is**

a. Collecting Responsibilities Classes  
b. Collaborating with Relevant Classes  
c. Class Responsibilities and Collaborators  
d. Creating Relevant Classes

9.34 **In CRC based design a CRC team consists of**

(i) one or two user’s representatives
(ii) several programmers  
(iii) project coordinators  
(iv) one or two system analysts  

a. i, ii  
b. i, iii  
c. i, iii, iv  
d. i, ii, iii, iv  

**9.35 A class index card contains besides class name**

(i) superclasses and subclasses  
(ii) short description of class  
(iii) collaborators  
(iv) private responsibilities of class  
(v) contract(s) with collaborators  

a. i, ii, iii  
b. i, iii, iv, v  
c. i, ii, iii, iv  
d. i, ii, iii, iv and v  

**9.36 The CRC modeling primarily requires**

(i) identifying classes and their responsibilities  
(ii) identifying collaborators of each class and their responsibilities  
(iii) developing a collaboration graph  

a. i, ii  
b. i, iii  
c. ii, iii  
d. i, ii, iii
Key to Objective Questions

9.1 b 9.2 d 9.3 c 9.4 c 9.5 c 9.6 c
9.7 a 9.8 d 9.9 b 9.10 d 9.11 c 9.12 b
9.25 b 9.26 b 9.27 b 9.28 a 9.29 c 9.30 b
9.31 a 9.32 c 9.33 c 9.34 c 9.35 d 9.36 d
SUMMARY of Module 9

1. Complex systems are designed by breaking down such systems into a number of cooperating subsystems which are reasonably independent and self-contained.

2. One method of design is to use what are known as objects as subsystems.

3. The main motivation for using object-oriented modelling is the realisation that individual objects may be modified without affecting the other parts of the system.

4. Objects also promote their reuse in many systems.

5. An object is an entity which is uniquely identifiable and is permanent relative to the life time scale of an application. All objects have attributes and state. A set of operations are defined for each object which determines their behaviour.

6. Objects collaborate by responding to requests from other objects to carry out specified operations.

7. Objects having similar meaning and purpose may be grouped together to form a class. A class may be considered as a template for objects of a particular type.

8. Special properties of objects are encapsulation, inheritance and polymorphism.

9. Encapsulation is the hiding of implementation details of an object from the clients who use the services provided by an object.

10. A class A may have attributes and operations which are identical to another class B in addition to its own attributes and operations. In such a case class A is said to inherit operations and attributes of class B.

11. Using inheritance a hierarchy of classes may be constructed where children inherit some of the properties of their parents in addition to their own properties.

12. An important concept in modelling using objects is polymorphism.
Polymorphism is the ability to manipulate objects of different distinct classes using only knowledge of their common property.

13. To model an information system using objects the requirements specification is examined and all nouns in it are picked as potential objects.

14. From among potential objects the relevant ones appropriate for modelling are the ones which have specific responsibilities and essential for the functioning of the system.

15. Having identified objects and their responsibilities the next step is to find the objects with whom they collaborate.

16. A document is prepared for each class giving its own responsibilities, its collaborators and their responsibilities.

17. A graph showing pictorially the classes and their mutual collaboration completes object oriented modeling.
Worked Examples

9.1 What 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

9.2 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

9.3 When is object-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

9.4 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.

9.5 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.

9.6 What 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.

9.7 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.

**9.8 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.

<table>
<thead>
<tr>
<th>Class name</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>Emp_no Emp_name Address Date_of_joining</td>
</tr>
<tr>
<td>Operations</td>
<td>Calculate_salary() Calculate_incometax()</td>
</tr>
</tbody>
</table>

**Salesman**

<table>
<thead>
<tr>
<th>Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp_no Emp_name Address Date_of_joining Dept.</td>
<td></td>
</tr>
<tr>
<td>Operations</td>
<td>Sell_product() Calculate_commsn()</td>
</tr>
</tbody>
</table>

**Accounts Assistant**

<table>
<thead>
<tr>
<th>Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp_no Emp_name Address Date_of_joining Dept.</td>
<td></td>
</tr>
<tr>
<td>Operations</td>
<td>Accounts_of_purchased_items() Generate_salary_bill()</td>
</tr>
</tbody>
</table>

**9.9 What 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.

**9.10** Pick 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, Subcn_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**

<table>
<thead>
<tr>
<th>Class: Magazine Publishing Company</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Superclass</strong>: None</td>
</tr>
<tr>
<td><strong>Subclass</strong>: None</td>
</tr>
<tr>
<td><strong>Collaborators</strong>: Subscription Accounts</td>
</tr>
<tr>
<td><strong>Description</strong>: The class represents the company, which prints many magazines monthly and sends magazine with magazine_id to the subscription accounts class</td>
</tr>
<tr>
<td><strong>Private responsibilities</strong>: Request_subscn_accounts processing, Delete the subscriber’s name from the mailing list and send acknowledgement</td>
</tr>
<tr>
<td><strong>Contract(s) with collaborator(s)</strong>: Dispatch the receipt after receiving the money</td>
</tr>
</tbody>
</table>

**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

---

**Class:** Magazine_publishing company  
**Collaboration graph**  
Class `Magazine_publishing company`  
Send the magazine and `mag_id`  
Class `Magazine_subscn_accounts`  
Send `ack`  
Send cheque  
Request_Subscn  
Subscriber

### 9.11 Give 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
9.1 **Computer systems are designed by**
   a. simplifying requirements of system
   b. breaking of the system into smaller self-contained co-operating subsystems
   c. breaking up the systems into independent parts
   d. modular design

9.2 **Functions and procedures are**
   a. not useful in designing computer systems
   b. old fashioned and they are not useful
   c. useful in designing computer systems
   d. have side effects which require special care if they are used as subsystems

9.3 **A subsystem of a complex system must specify**
   a. what task it performs
   b. how it performs a task
   c. with which subsystems it co-operates
   d. how it co-operates with other systems

9.4 **A subsystem of a complex system must**
   a. ii, iii
   b. ii, iv
   c. iii, iv

9.5 **A subsystem of a complex system**
   (i) should be reusable in other complex system
   (ii) must not be able to inherit the properties of other subsystems
   (iii) must have clearly specified responsibilities
   (iv) must know the stimuli to which it should respond
   a. i, ii, iii
   b. ii, iii, iv
   c. i, iii, iv
   d. i, ii, iv
9.6 By polymorphism of a subsystem we mean
   a. it should be reusable
   b. it should have polymorphic data types
   c. it should accept generic commands and interpret appropriately
   d. it should morph polygons

9.7 The advantages of object-oriented modelling are
   a. i, ii
   b. i, iii
   c. ii, iii
   d. i, iv

9.8 Objects are
   (i) tangible entities
   (ii) intangible entities
   (iii) transient entities
   (iv) uniquely identifiable
   a. i, ii
   b. i, ii, iii
   c. i, ii, iii, iv
   d. i, ii, iv

9.9 A class is
   a. a group of objects
   b. template for objects of a particular type
   c. a class of objects
   d. a classification of objects

9.10 All objects have
   (i) attributes
   (ii) states
   (iii) a set of operations
   (iv) a unique identity
   a. i, ii, iii
   b. ii, iii, iv
9.11 **In UML diagram of a class**
   a. state of object cannot be represented
   b. state is irrelevant
   c. state is represented as an attribute
   d. state is represented as a result of an operation

9.12 **Attributes are assigned value**
   a. when operations are performed on an object
   b. when instances of objects are defined
   c. when methods are invoked
   d. when classes are identified

9.13 **The following are intangible entities which can be defined as objects**
   (i) a motor car
   (ii) a bank account
   (iii) an aircraft
   (iv) a linked list
   a. i, ii
   b. ii, iv
   c. iii, iv
   d. ii, iii, iv

9.14 **A query operation on a object**
   a. has side effect
   b. has no side effects
   c. changes the state of an object
   d. is not allowed

9.15 **An instance of an object is created by a**
   a. query operation
   b. update operation
   c. constructor operation
   d. open operation
9.16 An update operation in an object instance
   a. updates the class
   b. has no side effects
   c. deletes an instance
   d. alters values of attribute(s) of an object instance

9.17 In object-oriented design
   a. operations and methods are identical
   b. methods specify algorithms whereas operations only state what is to be done
   c. methods do not change values of attributes
   d. methods and constructor are same

9.18 By abstraction in object-oriented modelling we mean picking
   a. only attributes appropriate to model an object
   b. only operations
   c. both operation and attributes with operations appropriate to model an object
   d. the appropriate abstract data type

9.19 By encapsulation in object-oriented modelling we mean
   a. encapsulating data and programs
   b. hiding attributes of an object from users
   c. hiding operations on object from users
   d. hiding implementation details of methods from users of objects

9.20 Encapsulation in object-oriented modelling is useful as
   a. it allows improving methods of an object independent of other parts of system
   b. it hides implementation details of methods
   c. it allows easy designing
   d. encapsulates attributes and operations of object

9.21 Objects may be viewed as
   a. clients in a system
   b. servers in a system
c. as both clients and servers in a system
d. neither as clients nor as servers in a system

9.22 Inheritance in object-oriented system is used to
a. create new classes from existing classes
b. add new operations to existing operations
c. add new attributes to existing attributes
d. add new states to existing states

9.23 Inheritance in object-oriented modelling can be used to
a. generalize classes
b. specialize classes
c. generalize and specialize classes
d. create new classes

9.24 When a subclass is created using inheritance the resulting class
a. may have only attributes of parent class
b. may have only operations of parent class
c. may have new operations only in addition to those in parent class
d. may have new attributes and new operations in addition to those of the parent class

9.25 By polymorphism in object-oriented modelling we mean
a. the ability to manipulate objects of different distinct classes
b. the ability to manipulate objects of different distinct classes knowing only their common properties
c. use of polymorphic operations
d. use of similar operations to do similar things

9.26 A polymorphic operation
a. has same name
b. has same name but uses different methods depending on class
c. uses different methods to perform on the same class
d. uses polymorphic method

9.27 Given a word statement of a problem potential objects are identified by selecting
9.28 Given a word statement of problem potential operations appropriate for objects are identified by selecting

- verb phrases in the statement
- noun phrases in the statement
- adjectives in the statement
- adverbs in the statement

9.29 Objects selected to model a system

(i) must be essential for functioning of the system
(ii) must have all attributes which are invariant during operations of a system
(iii) must have attributes relevant for performing services of object
(iv) must be able to perform assigned services

- i, ii, iii
- ii, iii, iv
- i, iii, iv
- i, ii, iii, iv

9.30 An object is selected for modelling a system provided

- its attributes are invariant during operation of the system
- its attributes change during operation of the system
- it has numerous attributes
- it has no attributes relevant to the system

9.31 An object is considered an external entity in object-oriented modelling

- its attributes are invariant during operation of the system
- its attributes change during operation of the system
- it has numerous attributes
- it has no attributes relevant to the system
9.32 Object-oriented system modelling using CRC method gives
   a. Java programs for the system
   b. C++ programs for the system
   c. Classes of the system, their responsibilities and collaborating classes
   d. Objective C programs for the system

9.33 The expansion of the acronym CRC is
   a. Collecting Responsibilities Classes
   b. Collaborating with Relevant Classes
   c. Class Responsibilities and Collaborators
   d. Creating Relevant Classes

9.34 In CRC based design a CRC team consists of
   (i) one or two user’s representatives
   (ii) several programmers
   (iii) project coordinators
   (iv) one or two system analysts
   a. i, ii
   b. i, iii
   c. i, iii, iv
   d. i, ii, iii, iv

9.35 A class index card contains besides class name
   (i) superclasses and subclasses
   (ii) short description of class
   (iii) collaborators
   (iv) private responsibilities of class
   (v) contract(s) with collaborators
   a. i, ii, iii
   b. i, iii, iv, v
   c. i, ii, iii, iv
   d. i, ii, iii, iv and v

9.36 The CRC modeling primarily requires
   (i) identifying classes and their responsibilities
(ii) identifying collaborators of each class and their responsibilities
(iii) developing a collaboration graph

a. i, ii
b. i, iii
c. ii, iii
d. i, ii, iii
Key to Objective Questions

9.1  b  9.2  d  9.3  c  9.4  c  9.5  c  9.6  c
9.7  a  9.8  d  9.9  b  9.10 d  9.11 c  9.12 b
9.25 b  9.26 b  9.27 b  9.28 a  9.29 c  9.30 b
9.31 a  9.32 c  9.33 c  9.34 c  9.35 d  9.36 d
Question Bank

9.1 What is object oriented modelling?

9.2 Why is object oriented modelling used in practice?

9.3 When is object oriented modelling particularly useful

9.4 Define an object

9.5 What is the difference between a class and an object

9.6 What are subclasses and superclasses. Give examples of each of these.

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

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

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

9.10 How do you select objects from a requirement specification. Given the following requirement statement, select potential objects.

A list of employers with their basic pay is sent to a clerk. He calculates the gross pay using standard allowances which are known for each pay slab. Deduction statements such as loan repayment, subscription to association etc., are also sent to another clerk who matches these slips with the slips of gross pay and calculates net pay. This step is used by another clerk to write out pay cheques for each employee and sent to respective employees. The total pay bills computed is also computed".

9.11 Pick objects and model the following requirements statement using objects. "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 up to two weeks before the expiry of the subscription, the subscriber's name is removed from the mailing list and the subscriber informed.

9.12 Give a brief requirement specification for a bus ticket reservation system. Model it using objects.
References


2. There are several standard books on object oriented modeling for those who want to dwell deeper. Some of these are:

   (i) I. Jacobson et.al “Object Oriented Software Engineering”, Pearson Education Asia, 1998
