MODULE 9

OBJECT ORIENTED SYSTEM MODELLING

Learning Units

9.1 Objects and their properties

9.2 Identifying objects in an application

9.3 Modelling systems with object
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
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
• 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
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
9.1.4

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

- All objects have a state
  
  **Example**: Ticket: reserved, waiting list
  Student: present, absent
CHARACTERSTICS OF OBJECTS

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

  Operations determine object **behavior**

  **Example**: Admit student

  Cancel ticket
Universal Modelling 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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Add vendor</th>
<th>OPERATIONS OR (METHODS)</th>
</tr>
</thead>
<tbody>
<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>
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
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
TERMINOLOGY USED IN OBJECT ORIENTED MODELLING

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

- **ENCAPSULATION**
  Hiding implementation details of methods from outside world

- **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
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 modelling

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 Name Address Year of study</td>
</tr>
<tr>
<td>Operations</td>
<td>Admit Promote</td>
</tr>
</tbody>
</table>

- **Science student**
  - Roll no
  - Name
  - Address
  - Year of study
  - Department
  - Laboratory name
  - Admit
  - Promote
  - Calculate laboratory fee

- **Arts student**
  - Roll no
  - Name
  - Address
  - Year of study
  - Department
  - Admit
  - Promote
  - Calculate field trip fee
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.
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
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.
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
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
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.

3. 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.
### ATTRIBUTES AND OPERATIONS PERFORMED
### BY IDENTIFIED OBJECTS

<table>
<thead>
<tr>
<th>CLASS NAME</th>
<th>APPLICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATTRIBUTES</strong></td>
<td></td>
</tr>
<tr>
<td>APPLICATION NUMBER</td>
<td></td>
</tr>
<tr>
<td>APPLICANT NAME</td>
<td></td>
</tr>
<tr>
<td>APPLICANT ADDRESS</td>
<td></td>
</tr>
<tr>
<td>MARKS SHEET</td>
<td></td>
</tr>
<tr>
<td>FEE PAID RECEIPT</td>
<td></td>
</tr>
<tr>
<td>DEPT. APPLIED CODE</td>
<td></td>
</tr>
<tr>
<td>APPLN STATUS</td>
<td></td>
</tr>
<tr>
<td>CLERK CODE</td>
<td></td>
</tr>
<tr>
<td><strong>OPERATIONS</strong></td>
<td></td>
</tr>
<tr>
<td>SCRUTINIZE</td>
<td></td>
</tr>
<tr>
<td>SEND APPLICATION TO DEPT</td>
<td></td>
</tr>
<tr>
<td>SEND RESPONSE</td>
<td></td>
</tr>
<tr>
<td>ADMIT/W.L/REJECT TO APPLICANT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NAME</th>
<th>DEPARTEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATTRIBUTES</strong></td>
<td></td>
</tr>
<tr>
<td>DEPARTMENT CODE</td>
<td></td>
</tr>
<tr>
<td>DEPARTMENT NAME</td>
<td></td>
</tr>
<tr>
<td>COURSE</td>
<td></td>
</tr>
<tr>
<td>NO OF STUDENTS TO BE ADMITTED</td>
<td></td>
</tr>
<tr>
<td>NO ON WAIT LIST</td>
<td></td>
</tr>
<tr>
<td>MIN. ENTRY QUALIFICATION</td>
<td></td>
</tr>
<tr>
<td>STATUS OF APPLICATION</td>
<td></td>
</tr>
<tr>
<td><strong>OPERATIONS</strong></td>
<td></td>
</tr>
<tr>
<td>SCRUTINIZE APPLICATION</td>
<td></td>
</tr>
<tr>
<td>SEND APPLICATION STATUS</td>
<td></td>
</tr>
</tbody>
</table>

---

9.2.8 System Analysis And Design © V. Rajaraman 26 of 41
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
- 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

As no operations on these

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

#### CLASS : DELIVERY NOTE

Attributes:
- Receiving clerk id
- Order no
- Vendor code
- Delivery date
- Item code
- Qty supplied
- Units

#### CLASS : ORDER TO VENDOR

Attributes:
- Order no
- Vendor code
- Item code
- Item name
- Qty ordered
- Units
- Price/Unit
- Order date
- Delivery period
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

### CLASS: RECEIVING OFFICE

**Attributes:** Derived as shown in the previous slide

**Operations:**
- Compare order no, item code, qty, etc in delivery note with that in order to vendor
- Send discrepancy note (if any) to purchase office and vendor. If no discrepancy send delivery note to purchase
- Send delivery note to inspection office (object)
### OTHER RELEVANT OBJECTS

**CLASS : STORES OFFICE**

<table>
<thead>
<tr>
<th>Attributes : Attributes of inspection office + qty in stock</th>
</tr>
</thead>
<tbody>
<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>
CLASS: INSPECTION OFFICE

Attributes: Derived attributes from delivery note + no of items accepted

Operations:
- Send information an accepted items to store and accounts
- Send discrepancy note (if any) to purchase office and vendor
### CLASS: ACCOUNTS OFFICE

**Attributes:** Derived from inspection office attributes + price/unit of item

**Operations:**
- Calculate amount to be paid
- Print cheque
- Request vendor object for vendor address
- Print vendor address label
- Dispatch payment to vendor
- Intimate Purchase office of payment
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.
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>CONTRACTS WITH COLLABORATORS :</td>
</tr>
</tbody>
</table>

Develop a graph to show interaction between classes
For Example 1 of last learning unit the CRC model is given below

<table>
<thead>
<tr>
<th><strong>Class</strong> : APPLICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Super class</strong> : None</td>
</tr>
<tr>
<td><strong>Sub class</strong> : None</td>
</tr>
<tr>
<td><strong>Collaborators</strong> : DEPARTEMENT</td>
</tr>
</tbody>
</table>

**Description** : This class represents applications received for admission to a university

**Private Responsibilities** :
Scrubitize : 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
### Class: DEPARTMENT

<table>
<thead>
<tr>
<th>Super class</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub class</td>
<td>None</td>
</tr>
</tbody>
</table>

**Collaborators**: APPLICATION

**Description**: This class represents departments whose responsibility is to admit, reject or place an applicant on a waiting list on application.

**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 EXAMPLE 2

1. **Applicant**
   - Admit/reject/wait list

2. **CLASS APPLICATION**
   - Examine application
   - Application status

3. **CLASS DEPARTMENT**

COLLABORATION GRAPH FOR EXAMPLE 2

1. **Vendor**
   - Delivery
   - Payment

2. **CLASS RECEIVING OFFICE**
   - Inspect
   - Delivery copy
   - Discrepancy note

3. **CLASS PURCHASE OFFICE**
   - Payment copy

4. **CLASS ACCOUNTS OFFICE**
   - Payment copy

5. **CLASS STORES OFFICE**
   - Make payment
   - Update Inventory

**System Analysis And Design** © V. Rajaraman