MODULE 9

OBJECT ORIENTED SYSTEM MODELLING

Contents

OBJECT ORIENTED SYSTEM MODELLING

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. How do we do this? The main purpose of this module is to answer this question.

LEARNING GOALS

At the end of this module you will know:

1. What are objects.
2. Why is it necessary to identify objects in an application.
3. How to identify objects given requirement specifications.
4. How objects are used to model information system.

LEARNING UNIT 1

Objects and their properties

DESIRABLE PROPERTIES OF COMPONENTS

Each subsystem or component must

- Have clearly defined responsibility
- Act 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

OBJECT ORIENTED MODELLING

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 will be simplified. It also 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.
And a member of a class is an object instance.

CHARACTERSTICS OF OBJECTS

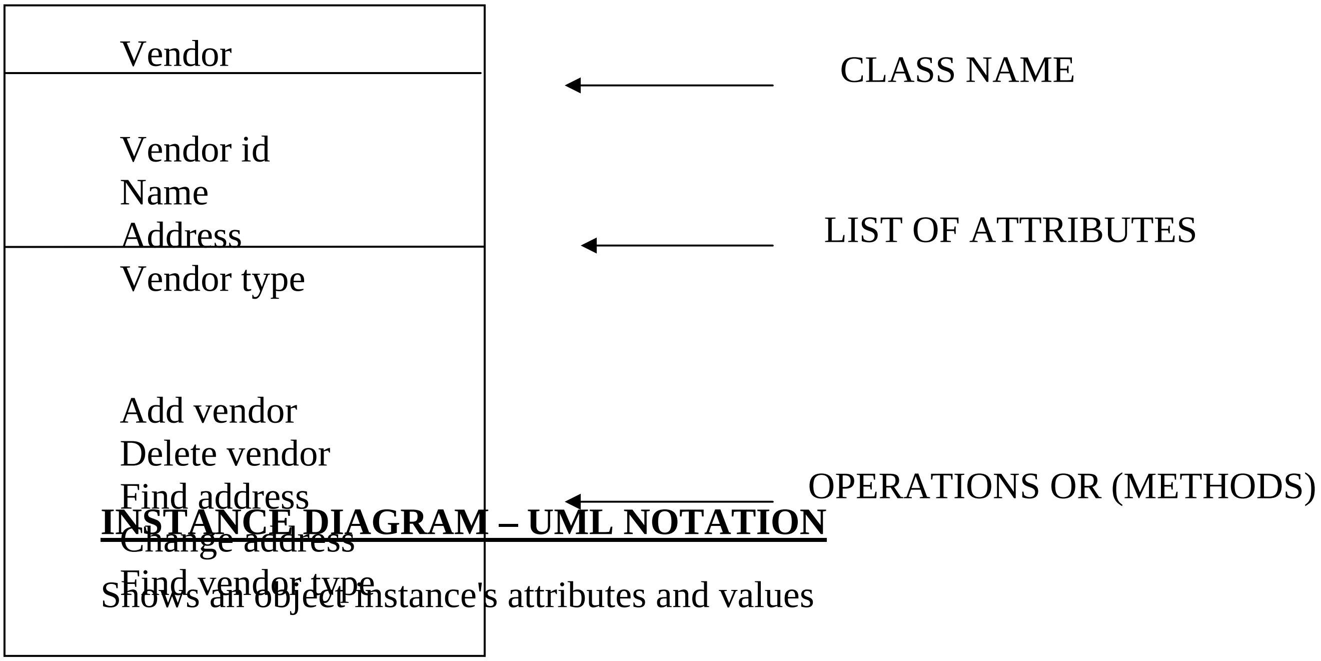
- All objects have attributes
  Example : student : Name
  Roll no
  Address
  Year
  Department
- All objects have a state
  Example Ticket: reserved, waiting list
  Student: present, absent

- All objects have set of OPERATIONS which can be performed on them. Operations determine object behavior
  Example : Admit student
  Cancel ticket

CLASS DIAGRAM – UML NOTATION

Universal Modelling Language (UML) is an industry standard notation to represent a class

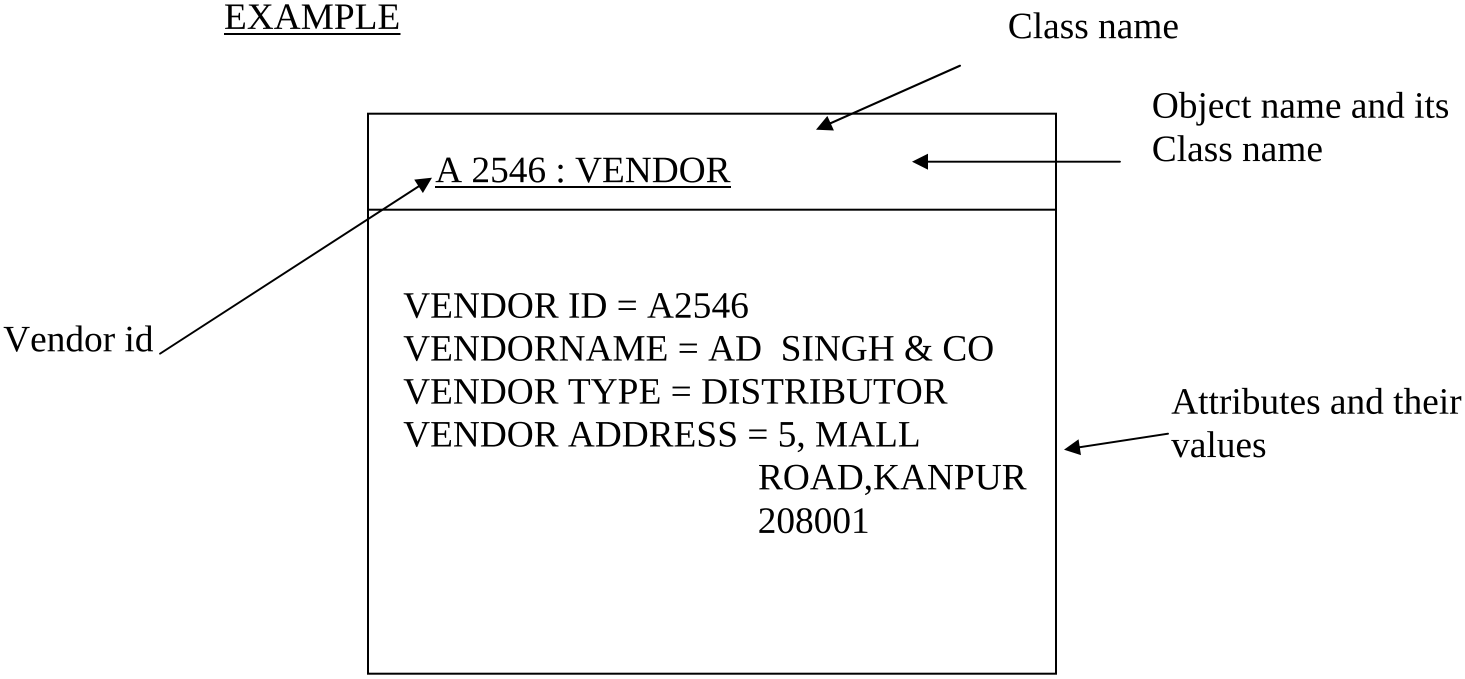
Example of UML notation for a Class



INSTANCE DIAGRAM – UML NOTATION

Shows an object instance's attributes and values

EXAMPLE



OPERATION TYPES ON OBJECTS

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

- Query - accessing state without changing value, has no side effects
  Example : find vendor address

- Update - changes value of one or more attributes & affect state of object
  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, it is 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

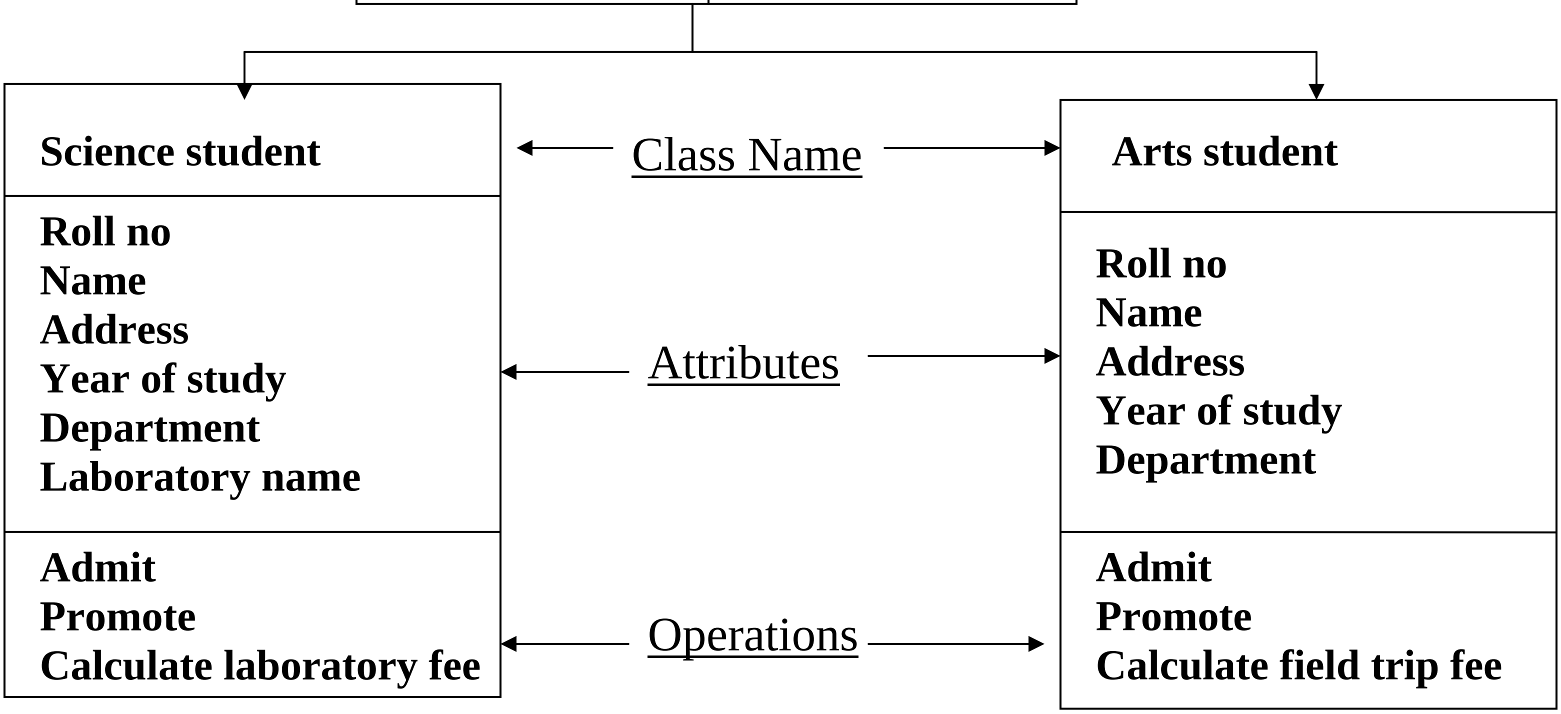
INHERITANCE

New classes are created from current classes by using the idea of inheritance.They 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

Class name	College student
Attributes	**Roll no** **Name** **Address** **Year of study**
Operations	**Admit** **Promote**



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, but 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

LEARNING UNIT 2

Identifying objects in an application

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.

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 are 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

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

- 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

POTENTIAL OBJECTS

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

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’s 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

APPLICATION

ATTRIBUTES

APPLICATION NUMBER
APPLICANT NAME
APPLICANT ADDRESS
MARKS SHEET
FEE PAID RECEIPT
DEPT. APPLIED CODE
APPLN STATUS
CLERK CODE

OPERATIONS

SCRUTINIZE
SEND APPLICATION TO DEPT
SEND RESPONSE
ADMIT/W.L/REJECT TO APPLICANT

DEPARTEMENT

ATTRIBUTES

DEPARTMENT CODE
DEPARTMENT NAME
COURSE
NO OF STUDENTS TO BE ADMITTED
NO ON WAIT LIST
MIN. ENTRY QUALIFICATION
STATUS OF APPLICATION

OPERATIONS

SCRUTINIZE APPLICATION
SEND APPLICATION STATUS

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

PICKING RELEVANT OBJECTS

POTENTIAL OBJECTS 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

CLASS : VENDORS

ATTRIBUTES :
Vendor code
Vendor name
Vendor address

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

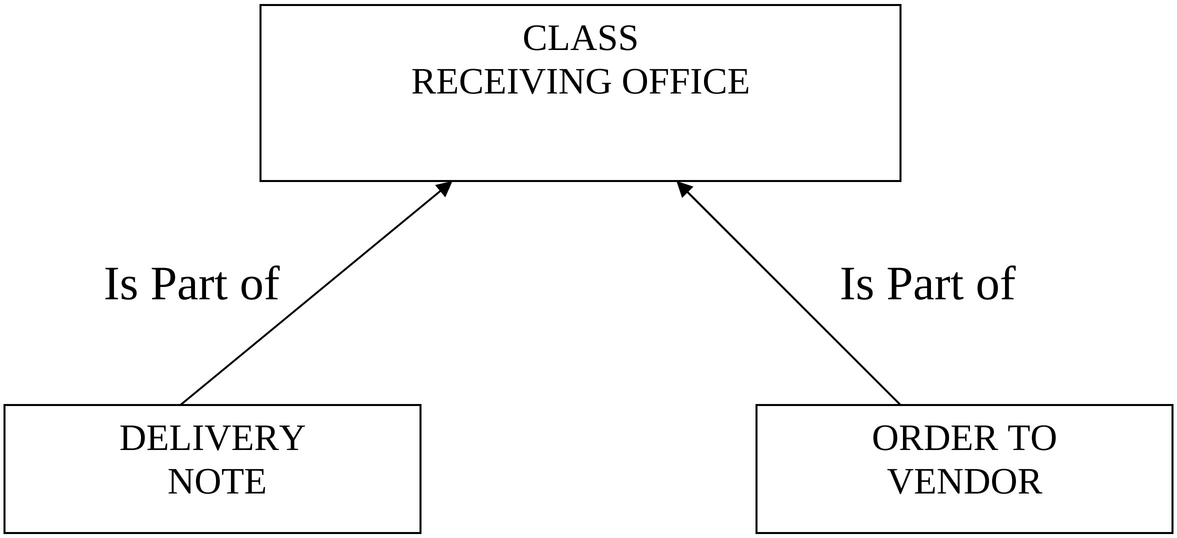
CLASS : ORDER TO VENDOR

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



RECEIVING OFFICE OBJECT

CLASS : RECEIVING OFFICE

Attributes : Derived as shown above

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

Attributes : Attributes of inspection office + qty in stock

Operations :

- Update inventory by adding no of items accepted to qty in stock
- Send advice to accounts object to make payment for qty accepted

CLASS : INSPECTION OFFICE

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

Operations :

- Send information on 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

LEARNING UNIT 3

Modelling systems with object

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.
5) State contracts each object assigns to its collaborators
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

CLASS NAME :
SUPER CLASSES AND SUBCLASSES :
SHORT DESCRIPTION OF CLASS :

Develop a graph to show interaction between classes

CRC MODEL – EXAMPLE1

Class : APPLICATION

Super class : None

Sub class : None

Collaborators : DEPARTEMENT

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

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)

Class : DEPARTMENT

Super class : None

Sub class : None

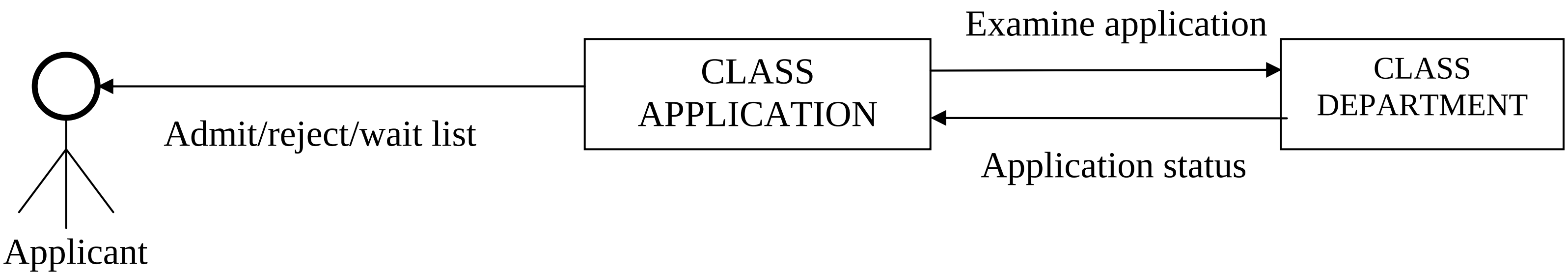
Collaborators : APPLICATION

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

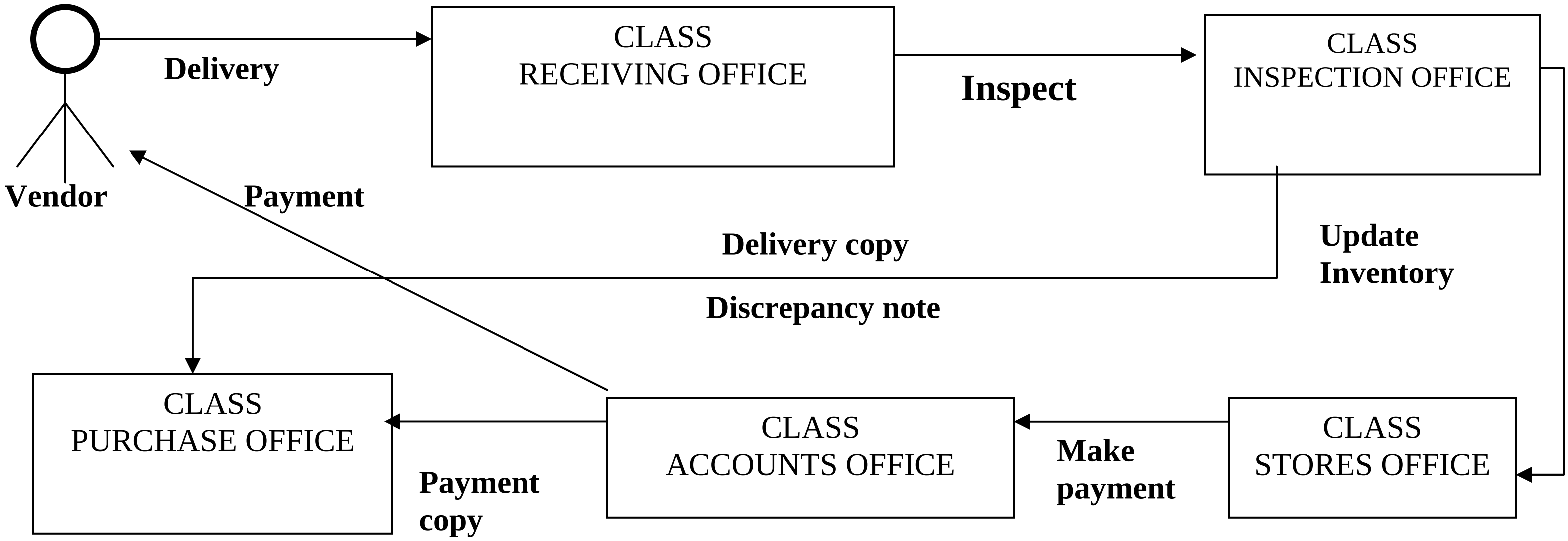
Private Responsibilities :
Rank order applications based on selection criteria mark’s in application:admitted,rejected or in waiting list depending o available seats

Contract(s) and Collaborator(s):
Send reply to applicationclass on admitted, rejected or wait list

COLLABORATION GRAPH



COLLABORATION GRAPH FOR EXAMPLE2 : Receiving items ordered by a company



REFERENCES

1. Most of the material in this module has been taken from the book "Analysis and Design of Information Systems", 2^{nd} Edition, V.Rajaraman, Prentice Hall of India, New Delhi, 2003. Chapter 13, Object Oriented System Modelling (pp.180-199)

2. There are several standard books on object oriented modeling for those who want to study deeper. Some of these are:
   (i) I.Jacobson *et.al* "Object Oriented Software Engineering", Pearson Education Asia, 1998
   (ii) David Colemen *et.al* "Object Oriented Development", Prentice Hall, Inc., N.J., WA,1999
   (iii) J.Rumbaugh *et.al* "Object Oriented Design and Applications", Benjamin Cumming, U.S.A., 1991