# Object Oriented Methods with UML

Introduction to Class Diagram

Lecture -3

Presented By

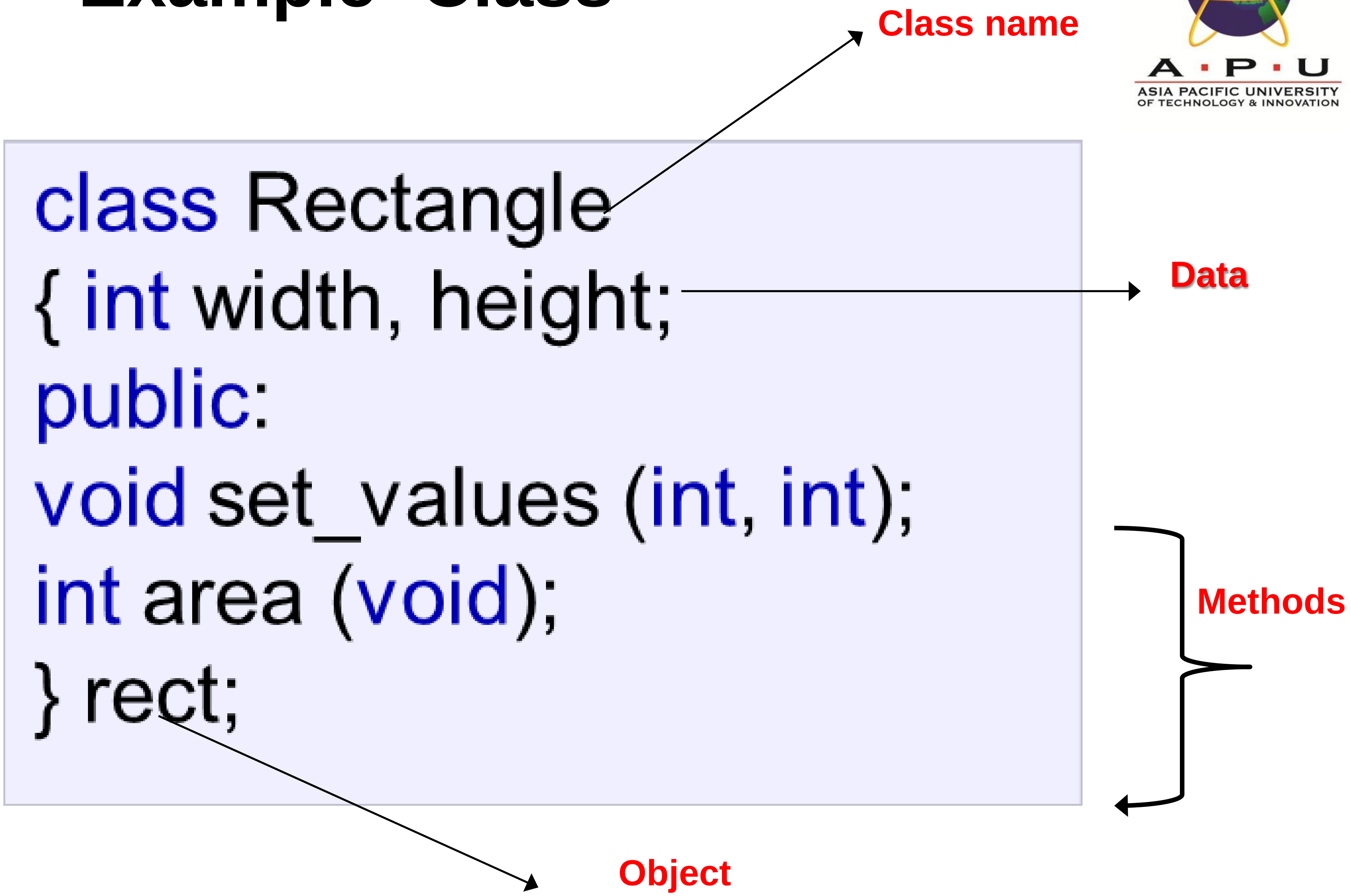
Dr.A.Bazila Banu

# What is a class?

- Central feature of C++ that supports OOP .
- It combines data representation and methods for manipulating the data.
- Members of class
    - Data
    - Methods

```
class Rectangle
{ int width, height;
public:
void set_values (int, int);
int area (void);
} rect;
```

# Example -Class



```
class Rectangle
{ int width, height;
public:
void set_values (int, int);
int area (void);
} rect;
```

# Access Specifiers in Class

- ***Data hiding*** is one of the important features of Object Oriented Programming which allows preventing the functions of a program to access directly the internal representation of a class type.

- The access restriction to the class members is specified by
  - **Private** :Variable or function cannot be accessed, or even viewed from outside the class
  - **Public** :Accessible from anywhere outside the class but within a program.
  - **Protected** : Accessed in child classes which are called derived classes.

## How to identify a class?

- ***Noun Phrase Approach***

  - ***Read through the Use cases, Interviews and Requirement Specification to find the***

    ***"Noun Phrases***

# Noun Phrase Approach

- Identify Tentative classes
  - Look for nouns and noun phrases in use cases
  - All classes must make sense in application domain

- Select classes from relevant categories

- Eliminate the following classes
  - **Adjectives**
  - **Attributes**
  - **Irrelevant classes**
  - **Redundant**

# Example-Banking

Account
Account Balance
Amount
ATM card
ATM machine
Bank
Bank Client
Card
Cash
Check
Checking

Checking a/c
Client
Client's a/c
Currency
Dollar
Envelope
Invalid PIN
Message

Money
Password
PIN
PIN code
Record
Savings a/c
Business a/c
Transaction

## Eliminate Adjectives

- No adjectives to eliminate

## Eliminate Attributes

Account
~~Account Balance~~
~~Amount~~
ATM card
ATM machine
Bank
Bank Client
~~Card~~
~~Cash~~

~~Check~~
Checking
Checking a/c
Client
Client's a/c
~~Currency~~
~~Dollar~~
~~Envelope~~
~~Invalid PIN~~
Message

~~Money~~
~~Password~~
~~PIN~~
~~PIN code~~
~~Record~~
Savings a/c
Business a/c
Transaction

## Identify Redundant Classes

**Account**
**Client's a/c**
ATM card
ATM machine
Bank
**Bank Client**
**Client**

**Checking**
**Checking a/c**

Savings a/c
Business a/c
Transaction

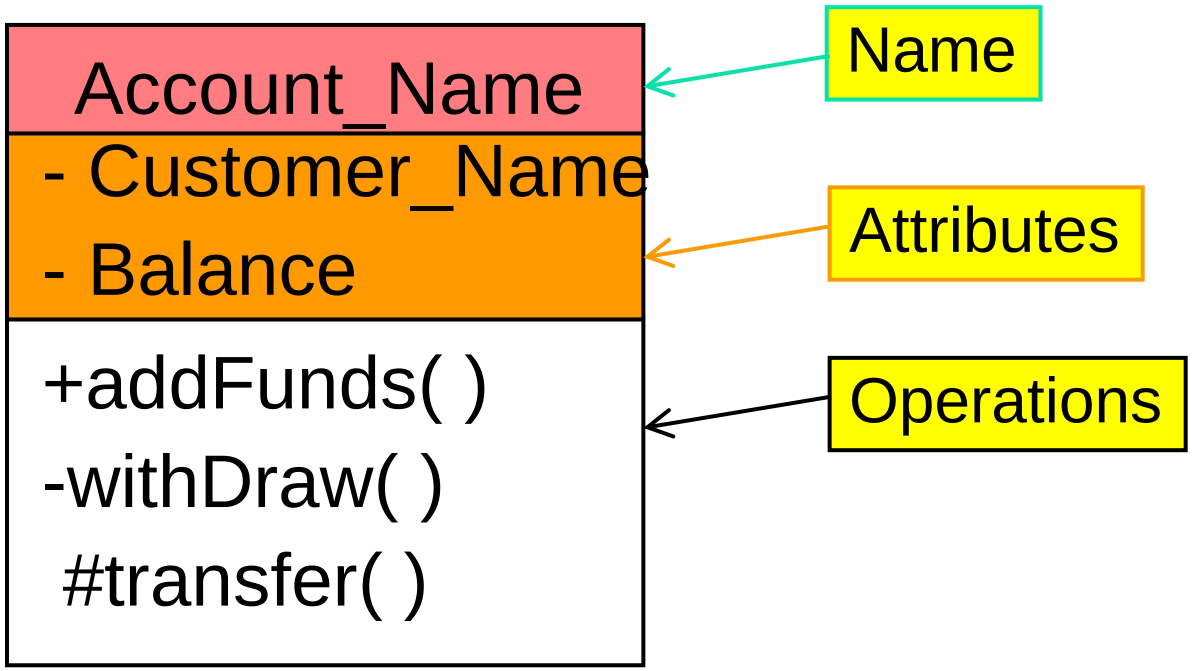
## Selected List of Classes

- Account
- ATM card
- ATM machine
- Bank Client
- Bank
- Savings Account
- Business Account
- Transaction

## Class Diagram

- A class diagram depicts classes and their interrelationships
- Used for describing structure and behavior in the use cases
- Provide a conceptual model of the system in terms of entities and their relationships
- Used for requirement capture, end-user interaction

# Class Diagram



# Class Diagram -Notations

- Each class is represented by a rectangle subdivided into three compartments
  - **Name**
  - **Attributes**
  - **Operations**

- Modifiers are used to indicate visibility of attributes and operations.
  - ‘+’ is used to denote *Public* visibility (everyone)
  - ‘#’ is used to denote *Protected* visibility (friends and derived)
  - ‘-’ is used to denote *Private* visibility (no one)

- By default, attributes are hidden and operations are visible.

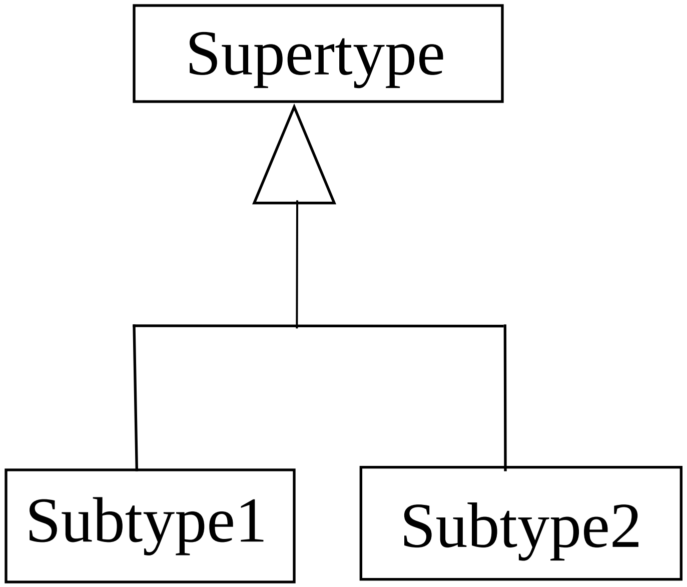
## Relationships in Class Diagram

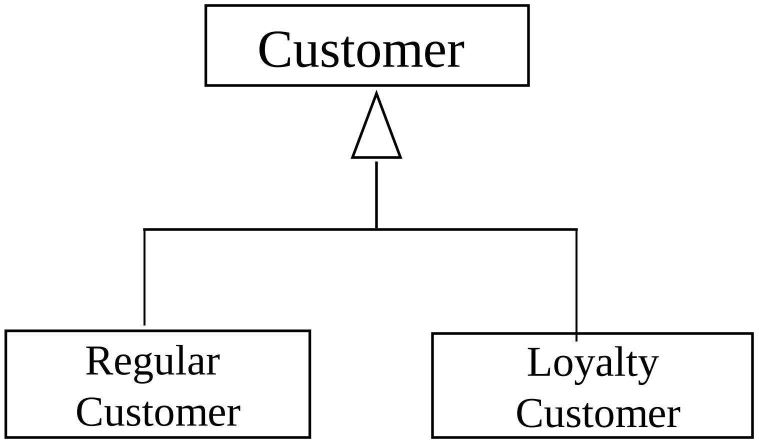
- There are two kinds of Relationships
  - Generalization (parent-child relationship)
  - Association (student enrolls in course)

- Associations can be further classified as
  - Aggregation
  - Composition

## OO Relationships: **Generalization** -

- -Inheritance is a required feature of OO Model.



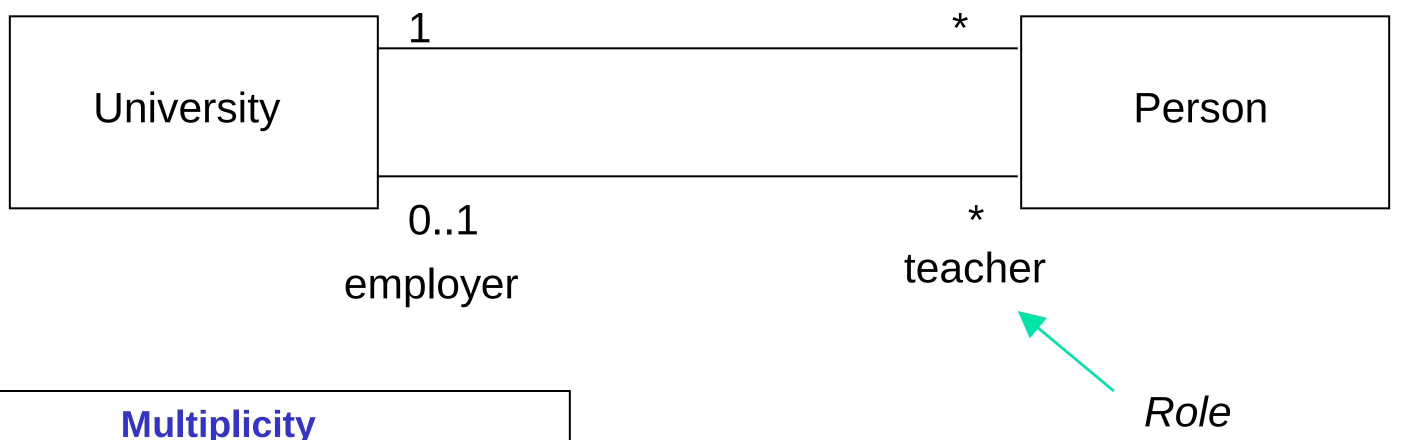


# OO Relationships: **Association**

- Represent relationship between instances of classes
  - Student enrolls in a course
  - Courses have students
  - Courses have exams
  - Etc.

- Association has two ends
  - Role names (e.g. enrolls)
  - Multiplicity (e.g. One course can have many students)
  - Navigability (unidirectional, bidirectional)

# Association



**Multiplicity**

| Symbol | Meaning |
|---|---|
| 1 | One and only one |
| 0..1 | Zero or one |
| M..N | From M to N (natural language) |
| * | From zero to any positive integer |
| 0..* | From zero to any positive integer |
| 1..* | From one to any positive integer |

**Role**

*"A given university groups many people; some act as students, others as teachers. A given student belongs to a single university; a given teacher may or may not be working for the university at a particular time."*

# Association

## Association

Models the part–whole relationship

I) **Composition**

Models the part–whole relationship but, in addition, Every part may belong to only one whole, and If the whole is deleted, so are the parts

II) **Aggregation**

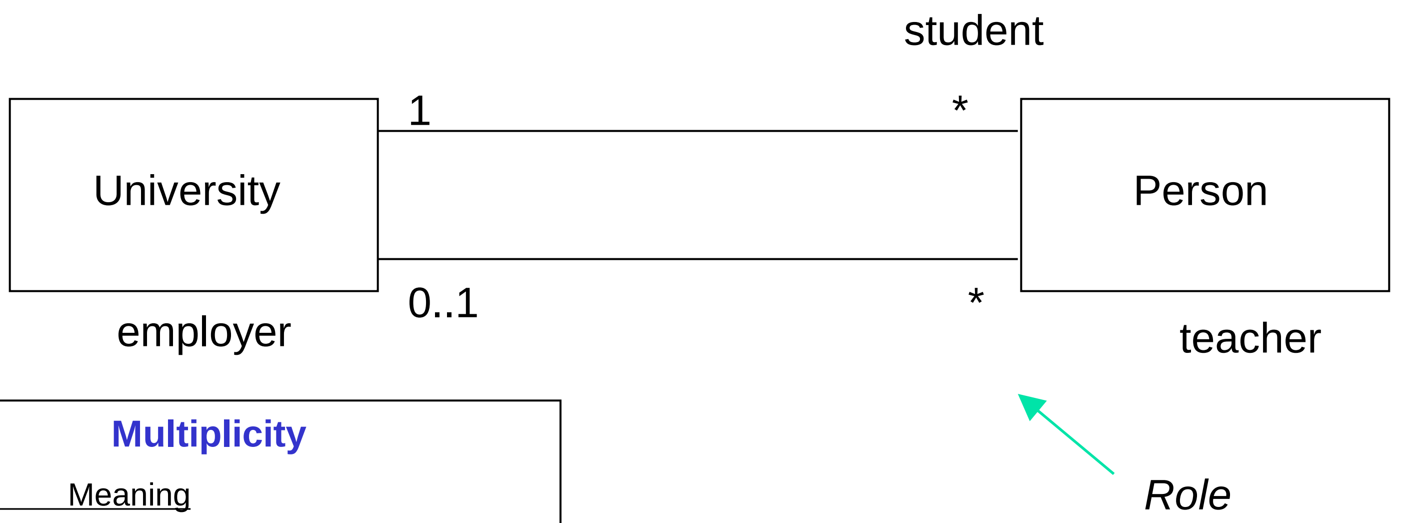
Expresses a relationship among instances of related classes.

## Association

- Represent relationship between instances of classes
  - Student enrolls in a course
  - Courses have students
  - Courses have exams
  - Etc.

- Association has two ends
  - Role names (e.g. enrolls)
  - Multiplicity (e.g. One course can have many students)
  - Navigability (unidirectional, bidirectional)

# Association: Multiplicity and Roles



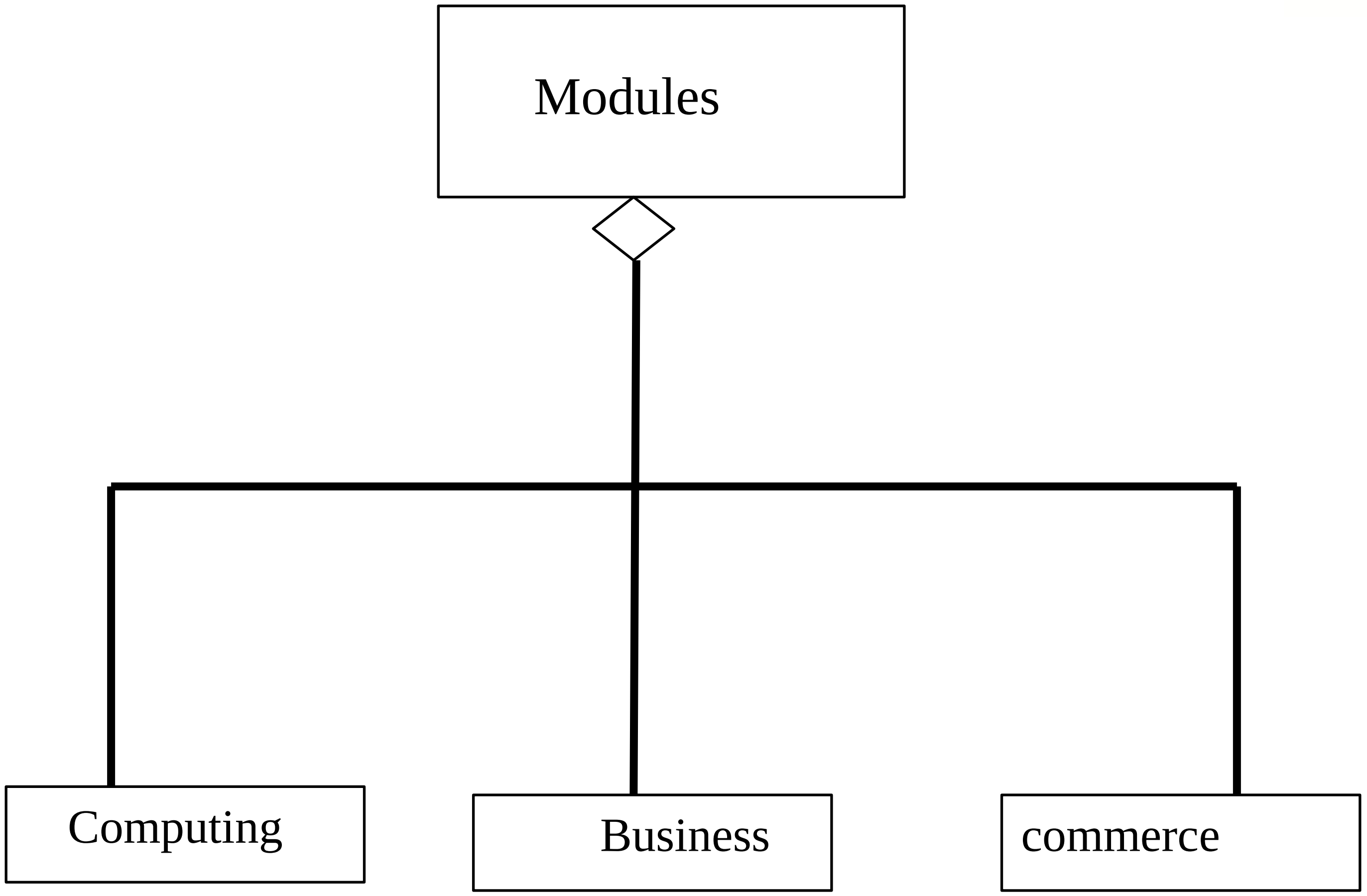
**Multiplicity**

| Symbol | Meaning |
|---|---|
| 1 | One and only one |
| 0..1 | Zero or one |
| M..N | From M to N (natural language) |
| * | From zero to any positive integer |
| 0..* | From zero to any positive integer |
| 1..* | From one to any positive integer |

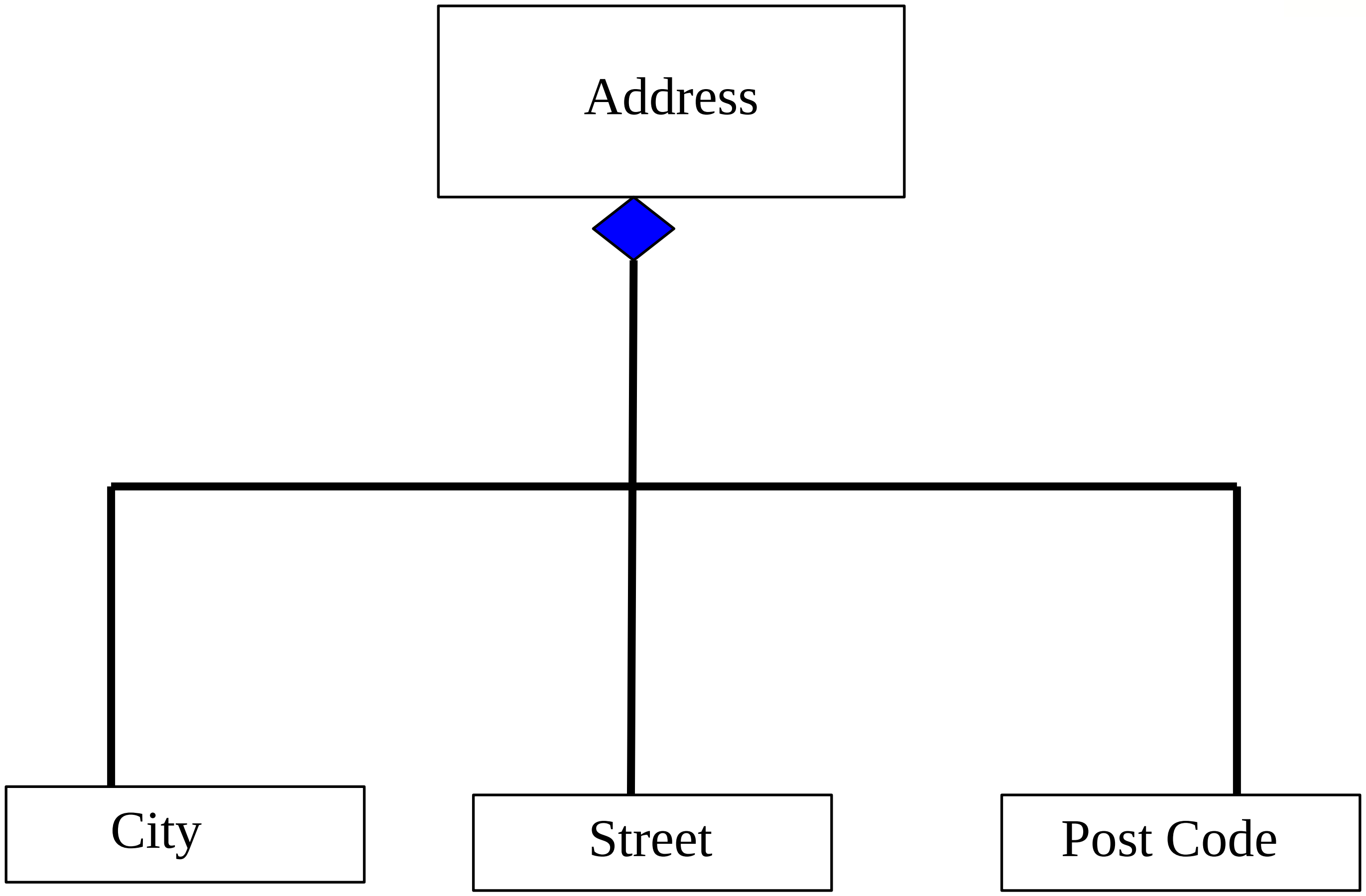
**Role**

*"A given university groups many people; some act as students, others as teachers. A given student belongs to a single university; a given teacher may or may not be working for the university at a particular time."*

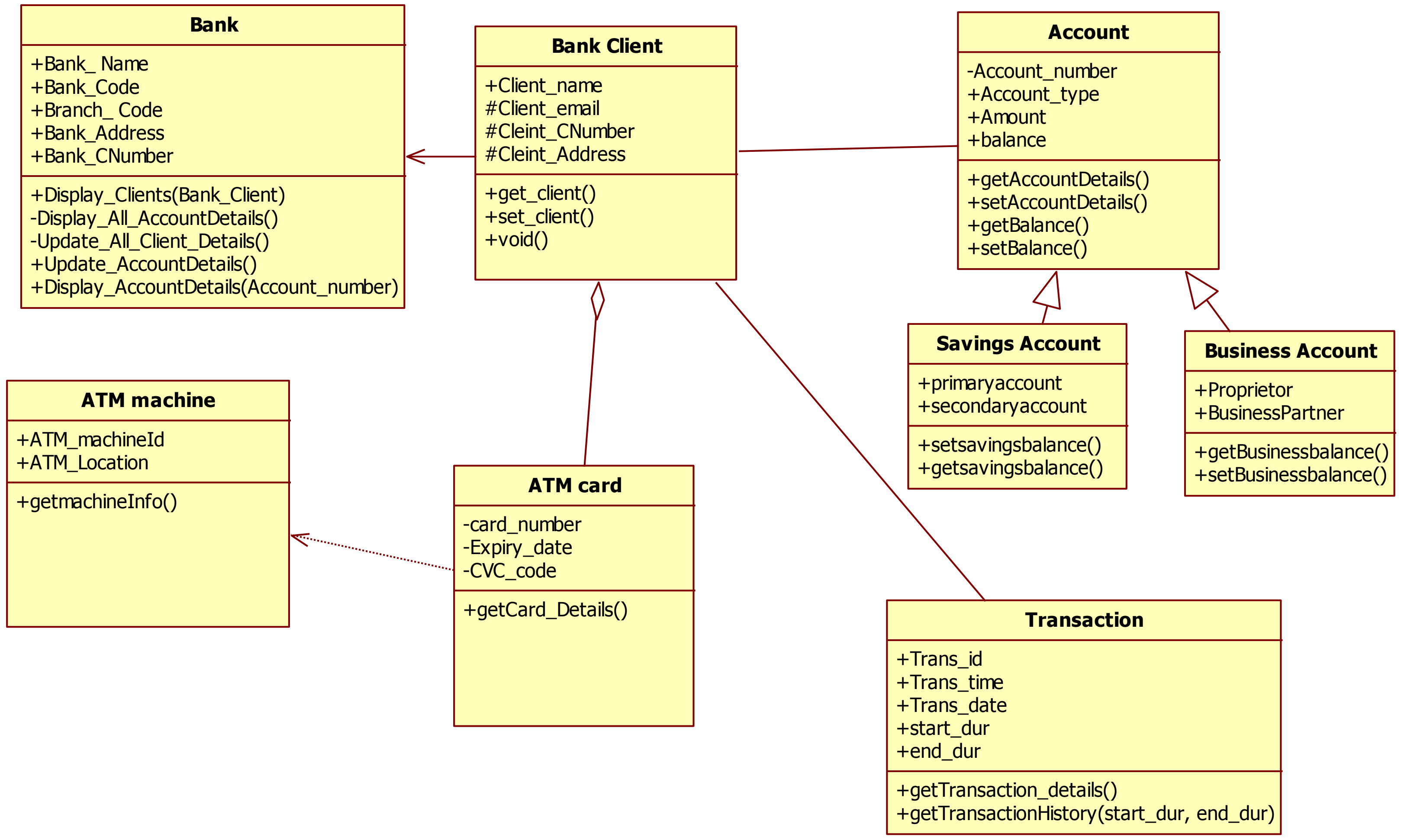
## Depiction of Aggregation Relationship



# Depiction of Composition Relationship



# Class Diagram -Example



Bank
+Bank_ Name
+Bank_Code
+Branch_ Code
+Bank_Address
+Bank_CNumber
+Display_Clients(Bank_Client)
-Display_All_AccountDetails()
-Update_All_Client_Details()
+Update_AccountDetails()
+Display_AccountDetails(Account_number)
Bank Client
+Client_name
#Client_email
#Cleint_CNumber
#Cleint_Address
+get_client()
+set_client()
+void()
Account
-Account_number
+Account_type
+Amount
+balance
+getAccountDetails()
+setAccountDetails()
+getBalance()
+setBalance()
Savings Account
+primaryaccount
+secondaryaccount
+setsavingsbalance()
+getsavingsbalance()
Business Account
+Proprietor
+BusinessPartner
+getBusinessbalance()
+setBusinessbalance()
ATM machine
+ATM_machineId
+ATM_Location
+getmachineInfo()
ATM card
-card_number
-Expiry_date
-CVC_code
+getCard_Details()
Transaction
+Trans_id
+Trans_time
+Trans_date
+start_dur
+end_dur
+getTransaction_details()
+getTransactionHistory(start_dur, end_dur)