Appendix H: *Introduction to Object-Oriented Concept*

Object-oriented technology partially rooted from knowledge representation techniques namely **Frame** and **Semantic Net** commonly used in Artificial Intelligence. Therefore, object-oriented technology is very useful in software abstraction and modeling. This is proven with the introduction of **UML (*Unified Modeling Language*)** as the industry standard today.

Today, object-oriented technology is not only adopted by software developers as methodology, it also being use extensively in business modeling.

What is an Object?

From the OO perspective, object is simply something that exists and made sense to us in the process of modeling.

We should be able to uniquely identify each of the objects in our world based on their ***Identity***.

Object can be physical or logical. Both physical and logical objects exist in the world that we are in. The different is that the physical objects are with physical appearance, whereas the logical objects do not have it. For example, computer is a physical object, whereas security policy is a logical object.

Object can be simple or complex under certain context. Simple object is the object that we take it as whole, whereas we are interested to study other sub-objects that form the Complex object. For example, car is a simple object for the parking system, but it is a complex object for the manufacturing process. Both parking system and manufacturing process are the contexts.

Abstraction

It is not necessary to know every single details of the object in our world in order to understand them or solving problem. In the process of modeling the real world, normally we are only interested in certain aspects of the real world for our purpose. The ***Abstraction*** is the process of focusing the essential aspects of the world that will help us in achieving our objective. Abstraction is the human ability in dealing with complexity.

How to Understand an Object?

The process of understanding an object is basically refers to three aspects of the object. Namely:

1) Attributes
2) Behavior
3) Relationships

We can identify the Attributes and Behavior aspects of the object by observing the individual object, whereas the Relationships aspect required us to relate the object with other objects.

Classification

There are many objects in our world. Inherently human equip with the ability in classifying things/objects. In order to recognize and understand them systematically we classification them base on similarities. Listed below are the methods we normally use:

1) Classification base on same attribute set
2) Classification base on same attribute value
3) Classification base on same behavior
4) Classification base on same relationship

Because of we can refer to the aspects of attributes and behavior of the individual object without relating to other objects, therefore, we can use both method 1 and 3 to classify objects. This form of classification is called ***Encapsulation***. In other word, Encapsulation is a specific form of classification. The result of encapsulation is called ***Class***. The aspect of relationship will be handled outside the class because it is relating two of more classes.

Attributes

This is the most obvious aspect of the object, especially the physical objects. The attribute is holding certain value(s) that will represent the ***State*** or ***Status*** of the object. For example, student x is an object. The name, CGPA, and age, are the example of the attributes for the student x. The attribute value of name for student x is “Tong Sam Pah”, age is 16, and CGPA is 2.1. The value 2.1 for CGPA indicates that the student x is in the weak (state), and 16 for age indicates s/he is young (state).

Behavior

Behavior is the general description of how an object replies to certain operations. For example, Ali is strange (Behavior) because of the way (***Method***) he walk (***Operation***).

Behavioral aspect normally describe with the “Adjectives”, it is high-level description. Sometimes it is more convenience to refer to **what** are the ***Operations*** that can be applied to the object. These operations normally can be referred as **verbs**. And the ways **how** these operations being carried out are called ***Methods.*** Method is the implementation of Operation.

Relationship

The relationship of the object must involve other object(s). Because of this reason, we relate different classes of objects to capture the relationship.

Basically there are three types of relationships:

1) Is-A or Kind-Of
2) Consists-Of or Part-Of
3) Ordinary relationship

Let say we have 2 classes of object X and Y. Mention below is the way to distinguish the types of relationship for these classes. Go through each type challenge in sequence.

Challenge 1: Is-A or Kind-Of

Ask two questions:

Q1) Is class X Is-A/Kind-Of class Y?
Q2) Is class Y Is-A/Kind-Of class X?

4 possibilities:

Q1	Q2	Result	Remark
True	True	Not adequate	This is normally happen when X and Y are referring to the same class of object, or we call this as **synonym**. The development team needs to decide to select only one. For example, when X is Client and Y is Customer.
True	False	Is-A Exist	X is a specific case of Y, or Y is more general than X
False	True	Is-A Exist	Y is a specific case of X, or X is more general than Y
False	False	Is-A not exist	Try the next challenge

The generic class is called the ***Superclass*** of the other class, and the specific class is called ***Subclass*** of the other class. We only can use the superclass/subclass with referring to two classes with the Is-A relationship.

Multiplicity does not make any sense to this type of relationship. We will discuss more about multiplicity in other form of relationship.

Challenge 2: Consist-Of or Part-Of

When we say X Consists-Of Y, this implies that Y is Part-Of X.

Ask two questions:

Q1) Is class X Consists-Of Y?
Q2) Is class Y Consists-Of X?

4 possibilities:

Q1	Q2	Result	Remark
True	True	Seldom happen	This situation normally happens when both X and Y referring to the same class. We call this as **recursive aggregation/composition.** For example, 1) Chicken and Egg: The chicken has egg and egg has chicken 2) Parent-Child Window: The MDI under the GUI environment, Windows might consist of other windows.
True	False	Consists-Of Exist	X is the container Object
False	True	Consists-Of Exist	Y is the container Object
False	False	Consists-Of not exist	Try the next challenge

If the Consists-Of/Part-Of relationship exists, there are 2 possible variation of this relationship.

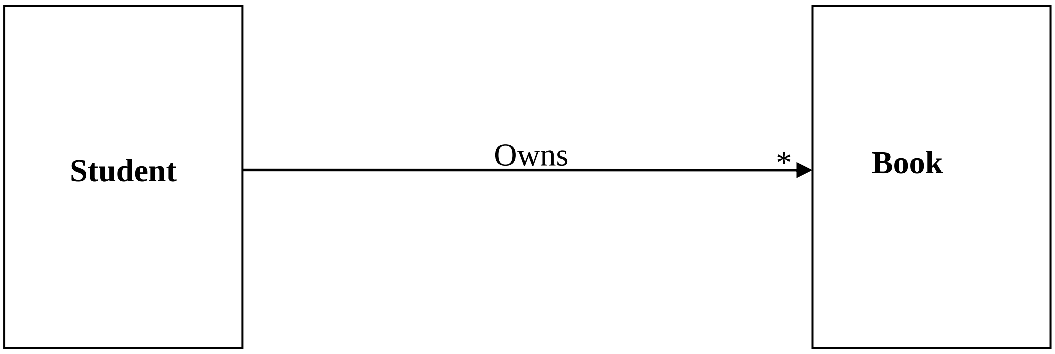
1) ***Composition*** – "Must has", e.g., "Classroom" Consists-Of "Table"
2) ***Aggregation*** – "Can have", e.g., "Classroom" Consists-Of "Air-Con."

Multiplicity for this type of relationship is 1-to-Many or 1-to-1. The container object is always 1.

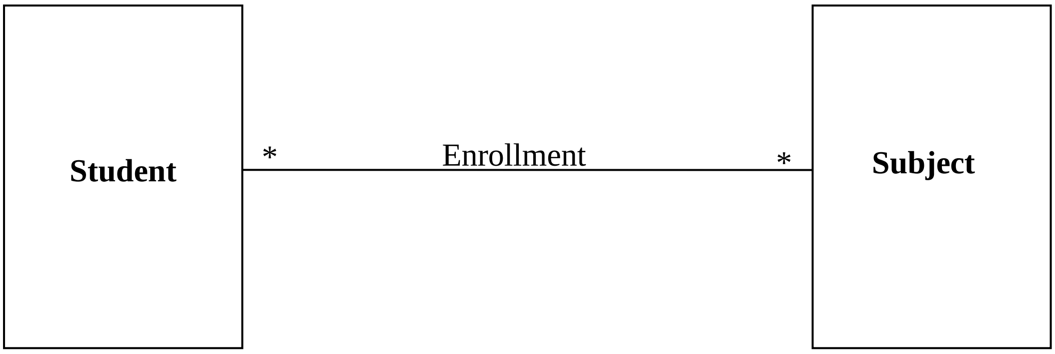
Challenge 3: Ordinary Relationship

If both Is-A and Consists-Of challenges fail, then the relationship is considered **ordinary**. Few characteristics can be observed:

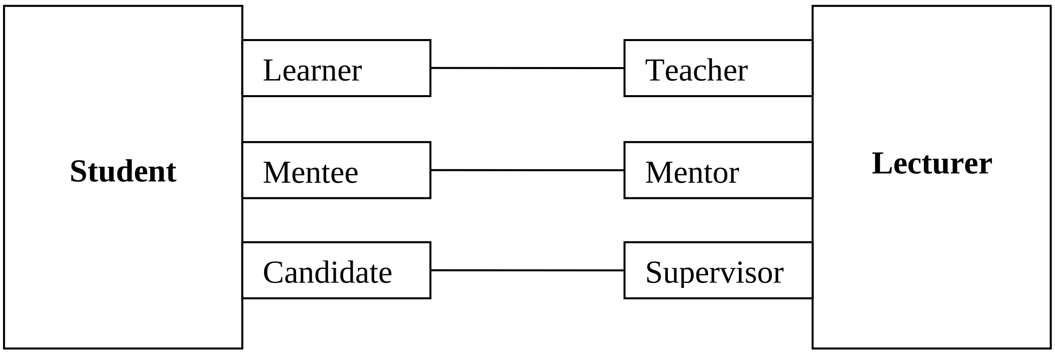
1) The name is need for the relationship, or the **roles** for both classes.
2) The **multiplicity** can be many-to-many.
3) Relationship can be directional (lecturer teaches student, and student learn from lecturer) or bidirectional (friendship, enrollment)



Example of using Name (Directional)



Example of using Name (Bidirectional)



Example of using roles

Polymorphism

Two classes might response differently to the same operation. This is called **Polymorphism**. For example, Ellipse is the superclass of Circle. Circle might use different method to the common operation "Rotate" because rotation at the center point make not different to the circle objects.

Generalization

The process of deriving a superclass from the existing classes called ***Generalization***.

Specialization

The process of deriving subclass from existing class called ***Specialization***. There are 3 typical reasons for specialization:

1) Specialization for Extension – Subclass has something more
2) Specialization of Restriction – Subclass lacking of something
3) Specialization for Overriding – Subclass has something different

Instantiation

The process of deriving object from existing class called ***Instantiation***. When we use the term ***Instance*** on object xyz, we must relate the object to the class. If class X is the subclass of class Y, all the Instances of X also the instances of Y. These instances will normally ***Inherits*** whatever we define in both classes X and Y. In this case, the object xyz is the ***Direct Instance*** of X, and it is the ***Indirect Instance*** of Y. For example, Ali is a Malaysian, and if Asian generally is friendly, therefore Ali is friendly too because of Malaysian is an Asian.

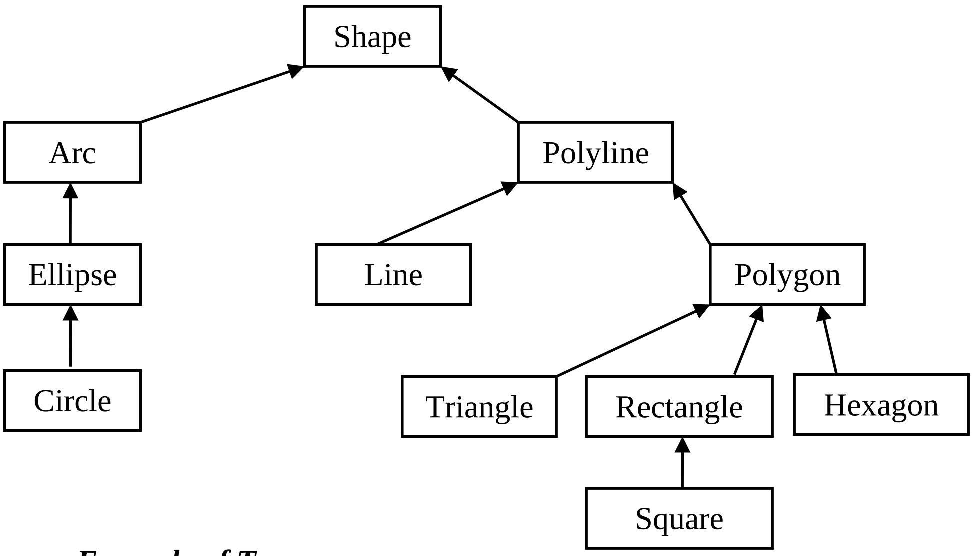
Abstract Class

Sometime a class is not completely defined under certain circumstances. This class will be referred by its existing subclasses. Therefore, direct Instantiation is not possible. This type class is called ***Abstract Class***. This normally happen during the process of generalization that cause new more general classes to group common aspects of existing classes.

Taxonomy

In order to represent the relationship between different species of living thing (such as Orchid, Japanese Koi Fish), biologists build ***Taxonomy*** about them. In OO

Technology, we borrow this term to describe the IS-A relationships among different classes in the system.



Example of Taxonomy

Foundation Class

When we place the classes in taxonomy, the class on the top are more general then the classes at the bottom, or the classes at the bottom are more specific then the classes on the top. The more general classes are more reusable then the more specific classes and we call them as ***Foundation Classes***. There many example of foundation classes that we can use to build software, such as MFC (Microsoft Foundation Classes) from Microsoft, Object Window from Borland, Swing from Sun Microsystem.

Important Keywords

1	Object
2	Identity
3	Attributes
4	Attributes Value
5	State
6	Behavior
7	Relationship
8	Operation
9	Method
10	Abstraction
11	Classification
12	Encapsulation
13	Class
14	Superclass
15	Subclass
16	Abstract Class
17	Taxonomy
18	Generalization
19	Specialization
20	Multiplicity
21	Inheritance
22	Multiple Inheritance
23	Instantiation
24	Instance
25	Direct and Indirect Instance
26	Information Hiding
27	Polymorphism
28	Aggregation
29	Composition
30	Foundation Classes