Chapter 5 – Domain Modeling

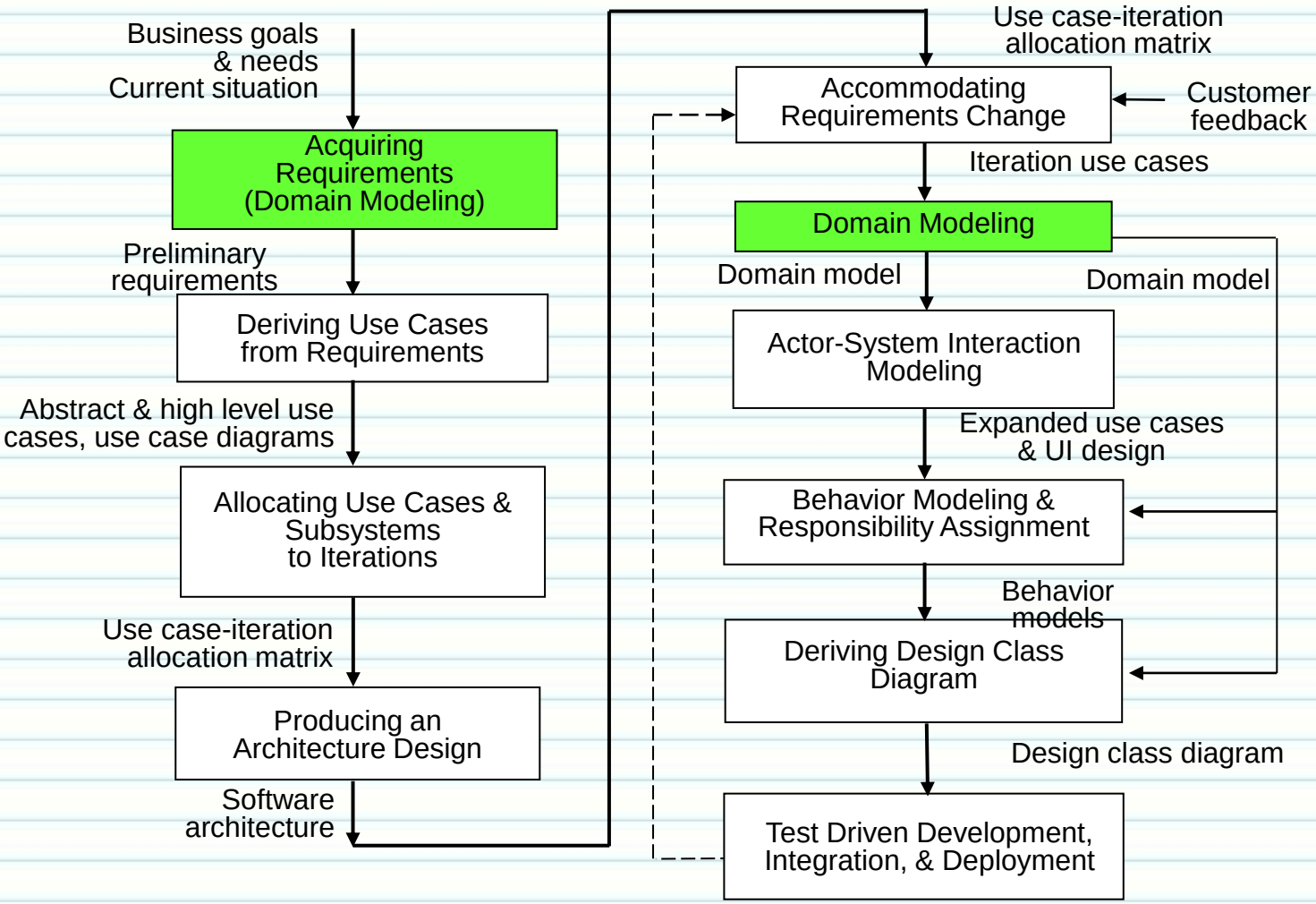
Dr. Michael F. Siok, PE, ESEP
UT Arlington
Computer Science and Engineering

Key Takeaway Points

- Domain modeling is a conceptualization process to help the development team understand the application domain

- Five easy steps:
  1. Collecting information about the application domain
  2. Brainstorming
  3. Classifying brainstorming results
  4. Visualizing the domain model using a UML class diagram
  5. Performing inspection and review

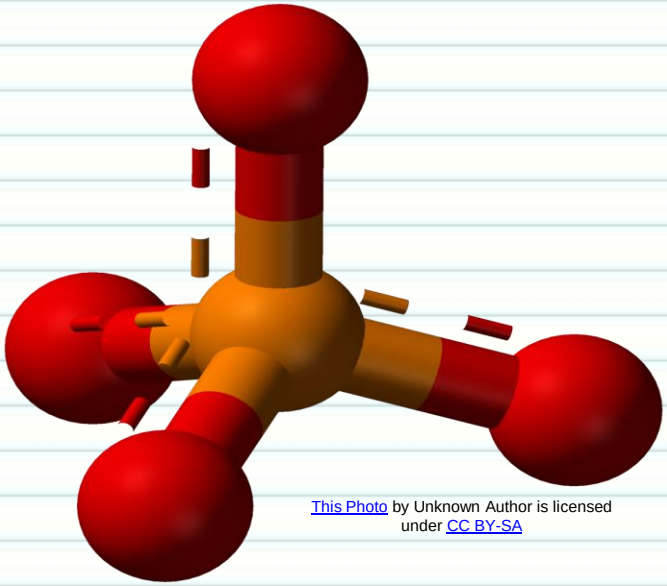
Domain Modeling in our Methodology Context



What Is a Model?

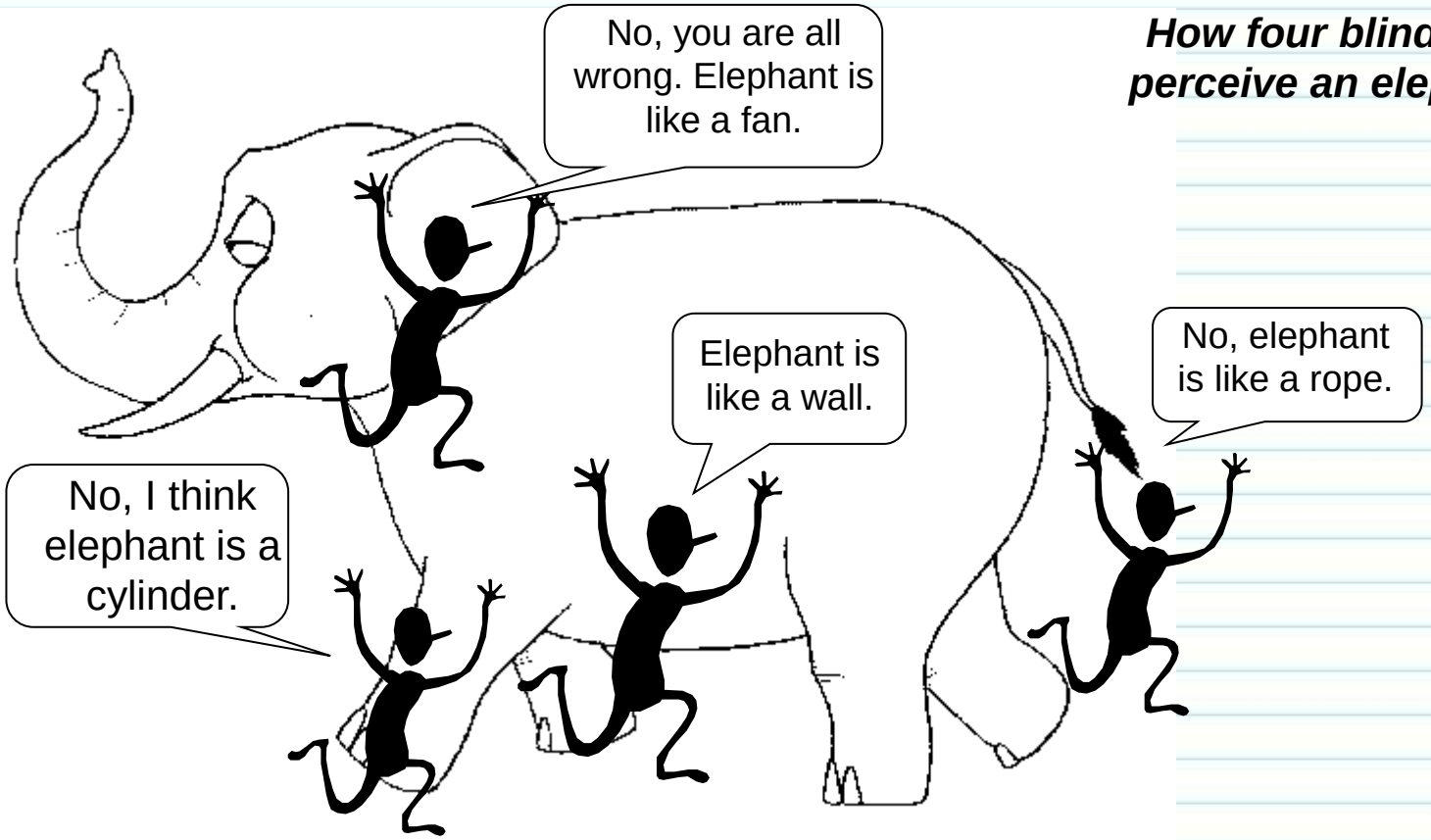
- A conceptual representation of something

- A schematic description of a system, theory, or phenomenon that accounts for its known or inferred properties and may be used for further study of its characteristics. (Dictionary Definition)



Why Do We Need Models?

An ancient Chinese story.
How four blind men perceive an elephant.



We perceive the world differently due to differences in backgrounds and viewpoints. Modeling facilitate collective understand of the application.

Why Do We Need Models?

Because the team members and users need to communicate their perceptions about a piece of reality

A model facilitates team members and users communication of their perception and design ideas

Why Do We Need Models?

Because we need models during the maintenance phase to perform enhancement maintenance

Domain Modeling

- What is it?
  - A process that helps the team understand the application or application domain
  - DM enables the team to establish a common understanding
- Why?
  - Software engineers need to work in different domains or different projects. They need domain knowledge to develop the system.
  - Software engineers come from different backgrounds, which affect their perception of the application domain.
- How?
  - Collect domain information, perform brainstorming and classification, and visualize the domain knowledge using a UML class diagram

Domain Modeling

- A domain model defines application domain concepts in terms of classes, attributes, and relationships

- The construction of the domain model
  - Helps the development team or the analyst understand the application and the application domain
  - Lets the team members communicate effectively their understanding of the application and the application domain
  - Improves the communication between the development team and the customer/user in some cases
  - Provides a basis for the design, implementation, and maintenance

- Domain model is represented by UML class diagrams (without showing the operations)

Domain Modeling in the OO Paradigm

- The OO paradigm views the real world as consisting of:
  - Objects
    - that relate to each other
    - interact with each other

- The basic build blocks and starting point are Objects

Important Object-Oriented Concepts

Class --- a class is a type

- an abstraction of objects with similar properties and behavior
- an intentional definition of a collection of objects

Attribute --- defines properties of class of objects

Operation --- defines behaviors of class of objects

Object --- an instance of a class

Encapsulation --- defining/storing together properties and behavior of a class/object

Information hiding --- shielding implementation detail to reduce change impact to other part of a program

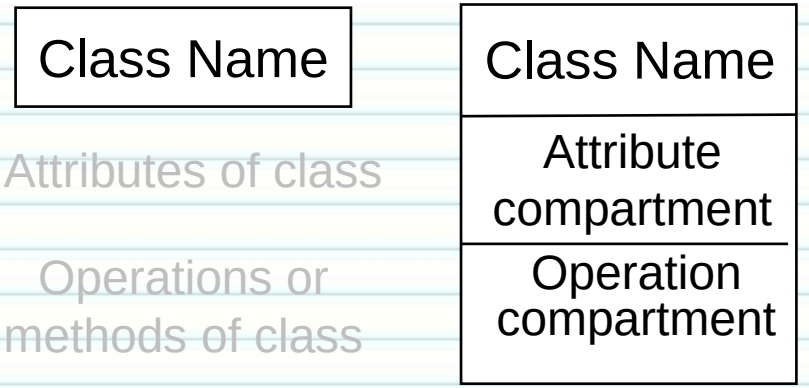
Polymorphism --- one thing can assume more than one form

Representing a Domain Model as a UML Class Diagram

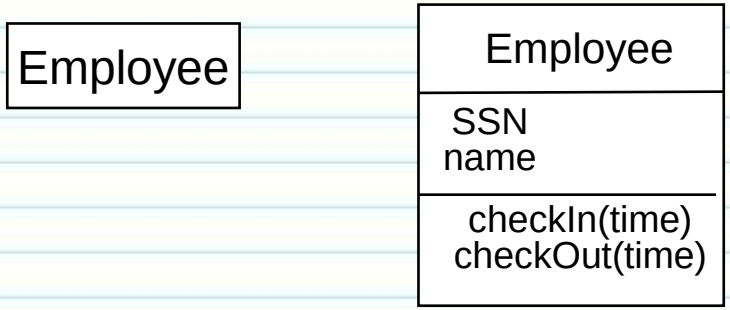
- The UML class diagram is a structural diagram
  - It shows the classes, their attributes and operations, and relationships between the classes
- The Domain model is represented by a class diagram *without showing the operations*

UML Class Diagram: Notion and Notation

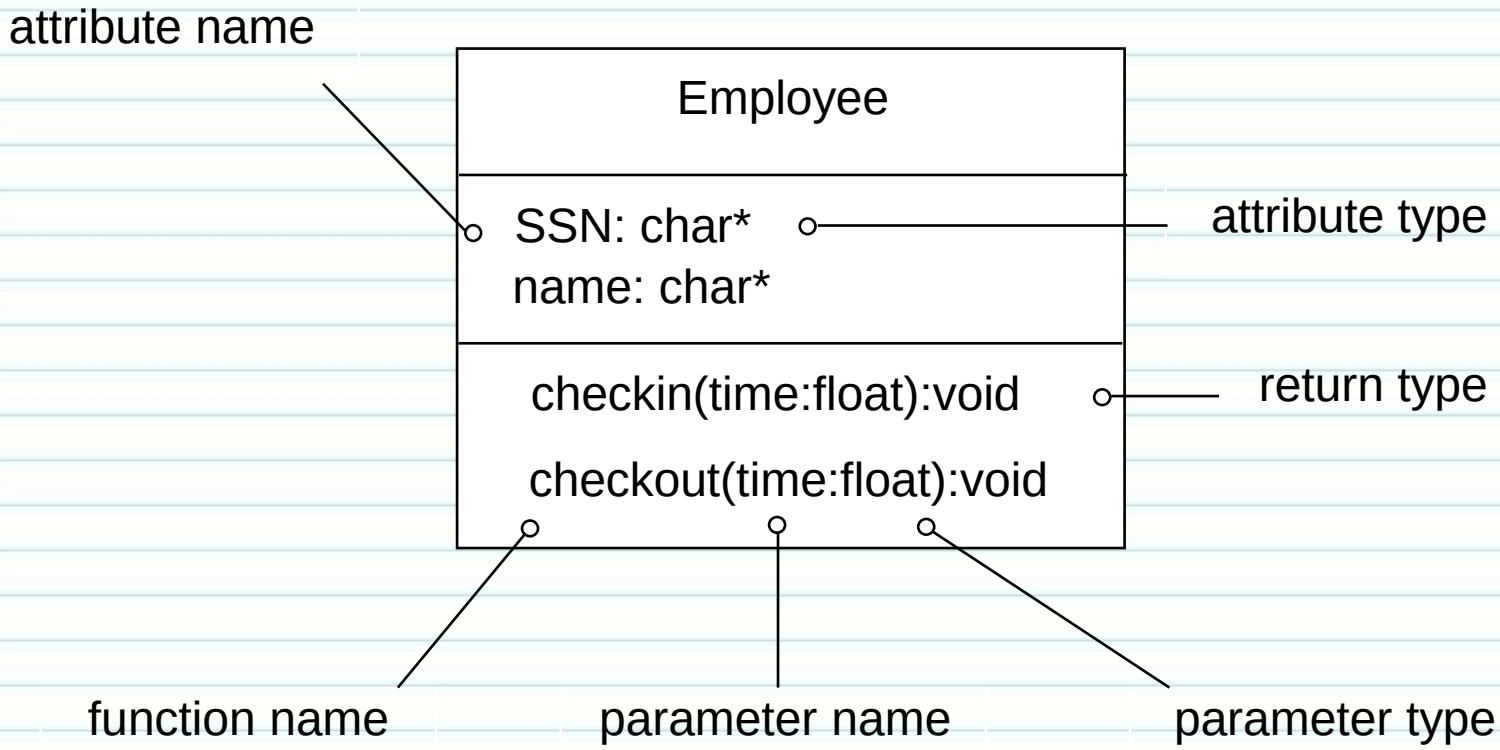
Class: a type (in OO)



Example:



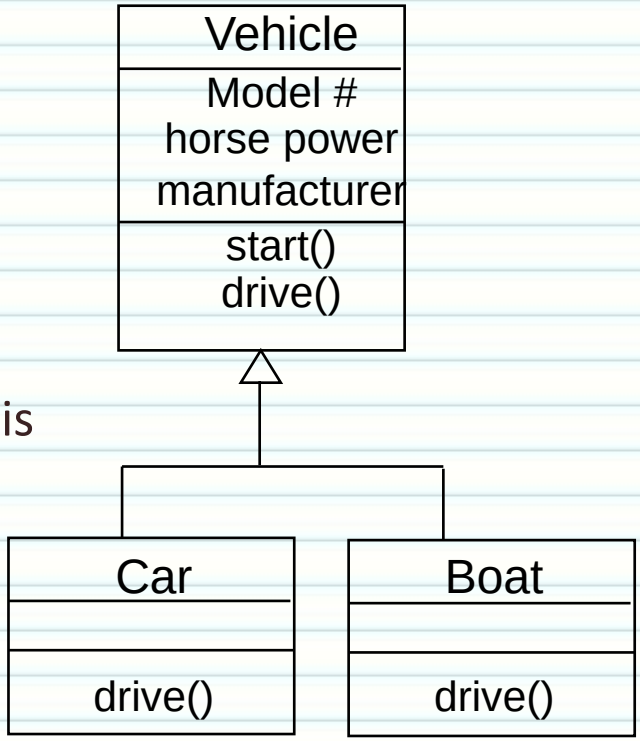
Representing Type in UML



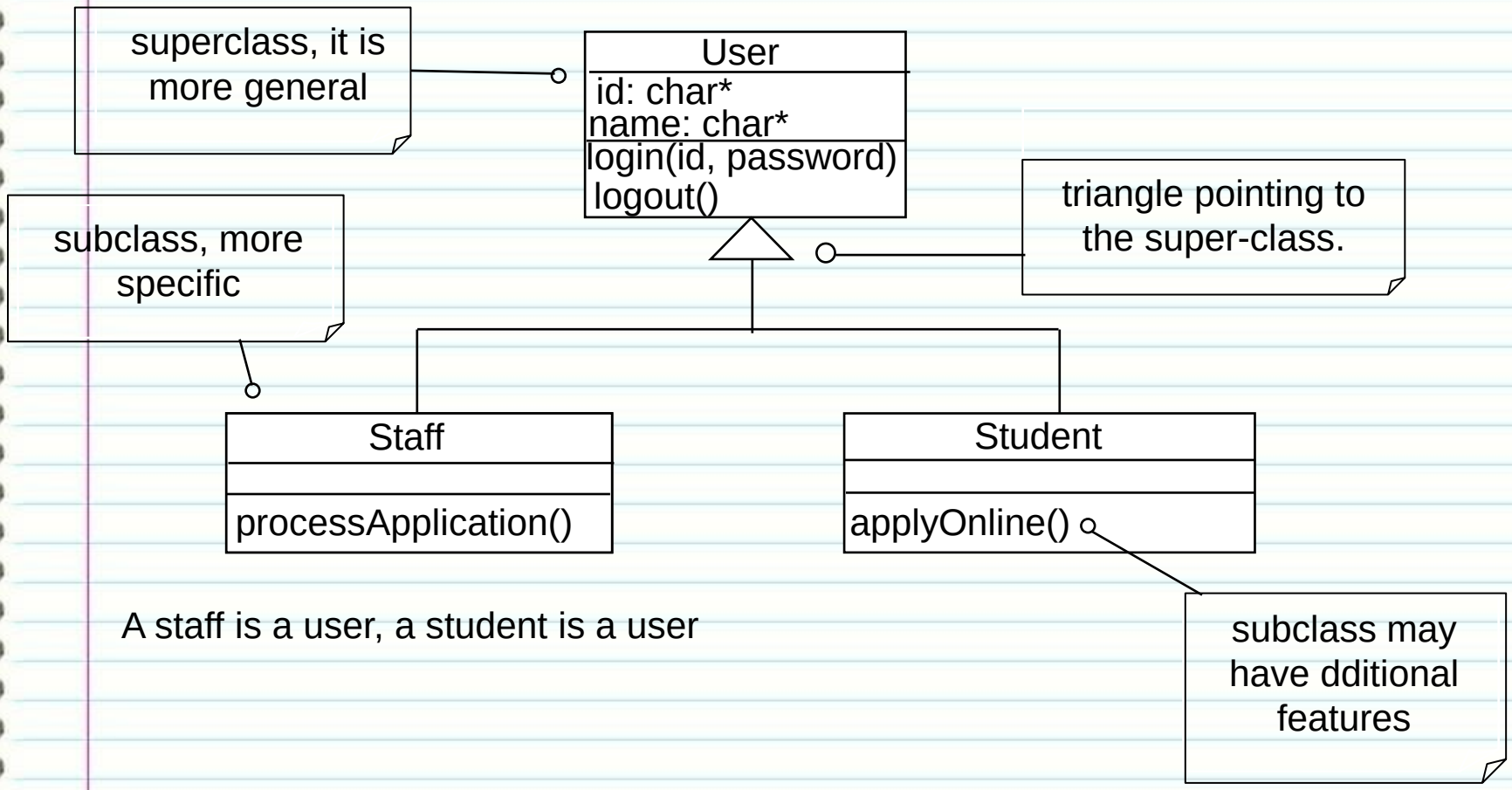
general syntax name **:** type

Inheritance Relationship

- Expresses the generalization / specialization relations between concepts
- One concept is more general/specialized than the other
- Example: vehicle is a generalization of car, car is a specialization of vehicle
- It is also called “IS-A” relation



Example: Inheritance



A staff is a user, a student is a user

Features (i.e., attributes and operations) defined for the super-class are automatically defined for the subclasses

Object and Attribute

- A noun/noun phrase can be a class or an attribute, how do we distinguish?
- This is often a challenge.
- Rules to apply:

An object has an *"independent existence*" in the application/application domain; an attribute does not

Example: "Number of seats“: class or attribute?

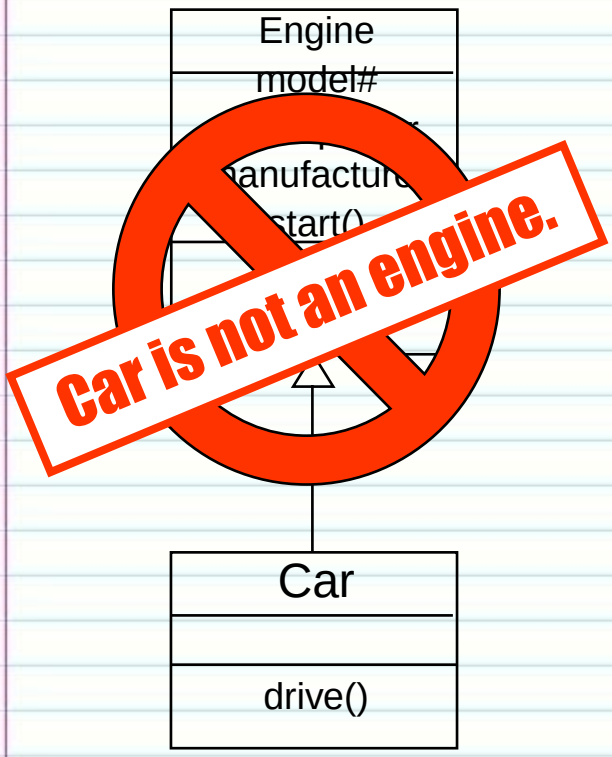
Attribute, because "number of seats" cannot exist without referring to a car, airplane, or classroom as in
"number of seats of a car"
"number of seats of an airplane"
"number of seats of a classroom"

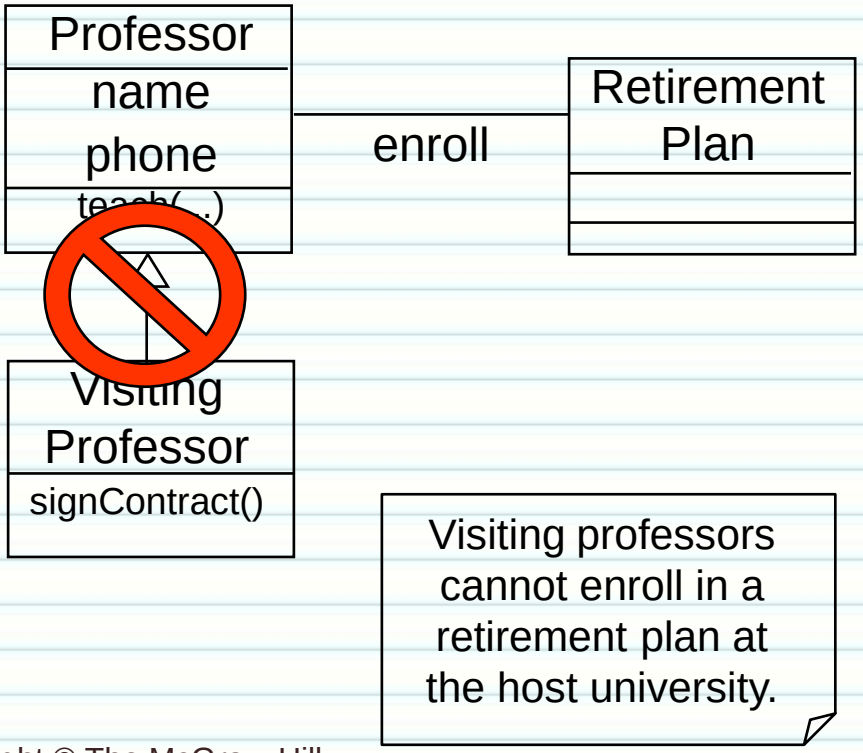
Object and Attribute

- Rules to apply:
  - Attributes describe objects or store state information of objects
  - You can enter an attribute (value) from the keyboard, but you cannot enter an object from the keyboard
  - Objects must be created by invoking a constructor (either explicitly or implicitly)

Two Tests for Inheritance

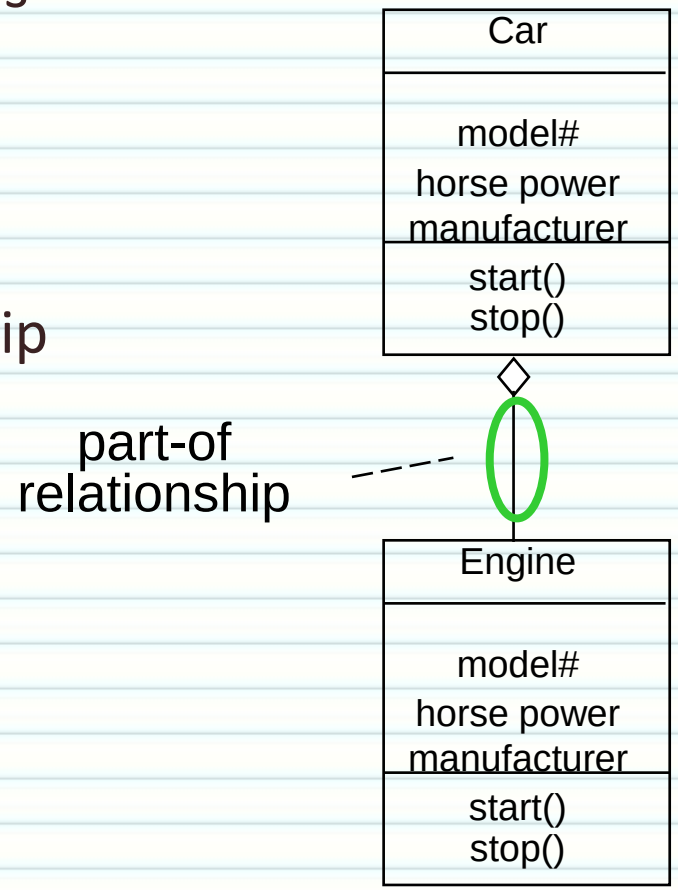
- IS-A test: every instance of a subclass is also an instance of the superclass.
- Conformance test: relationships of a superclass are also relationships of subclasses.





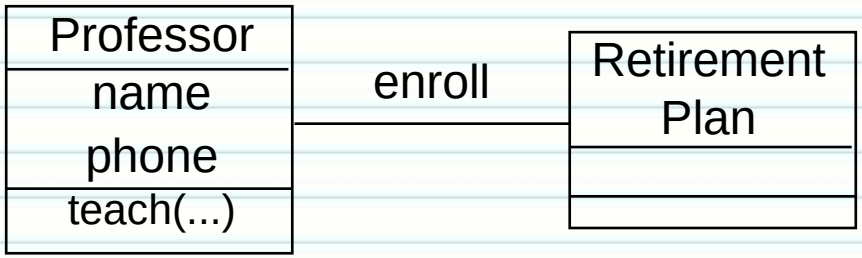
Aggregation Relationship

- Expresses the fact that one object is part of another object
  - Example: engine is part of a car
- It is also called “part-of” relationship



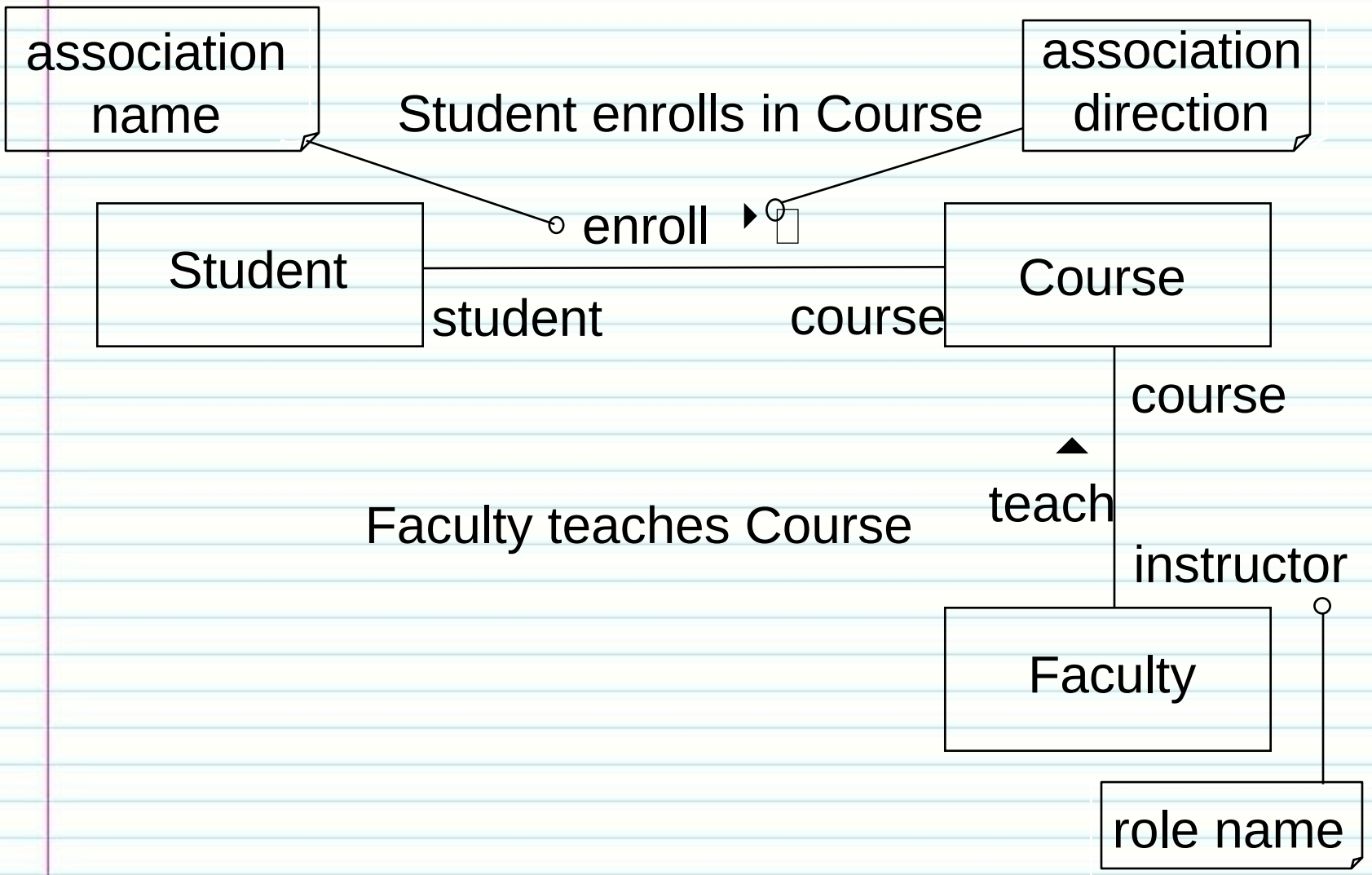
Association Relationship

- Expresses a general relationship other than inheritance and aggregation
  - These can be application-specific relationships between two concepts
- Example: "instructor teaches course" "user has account"

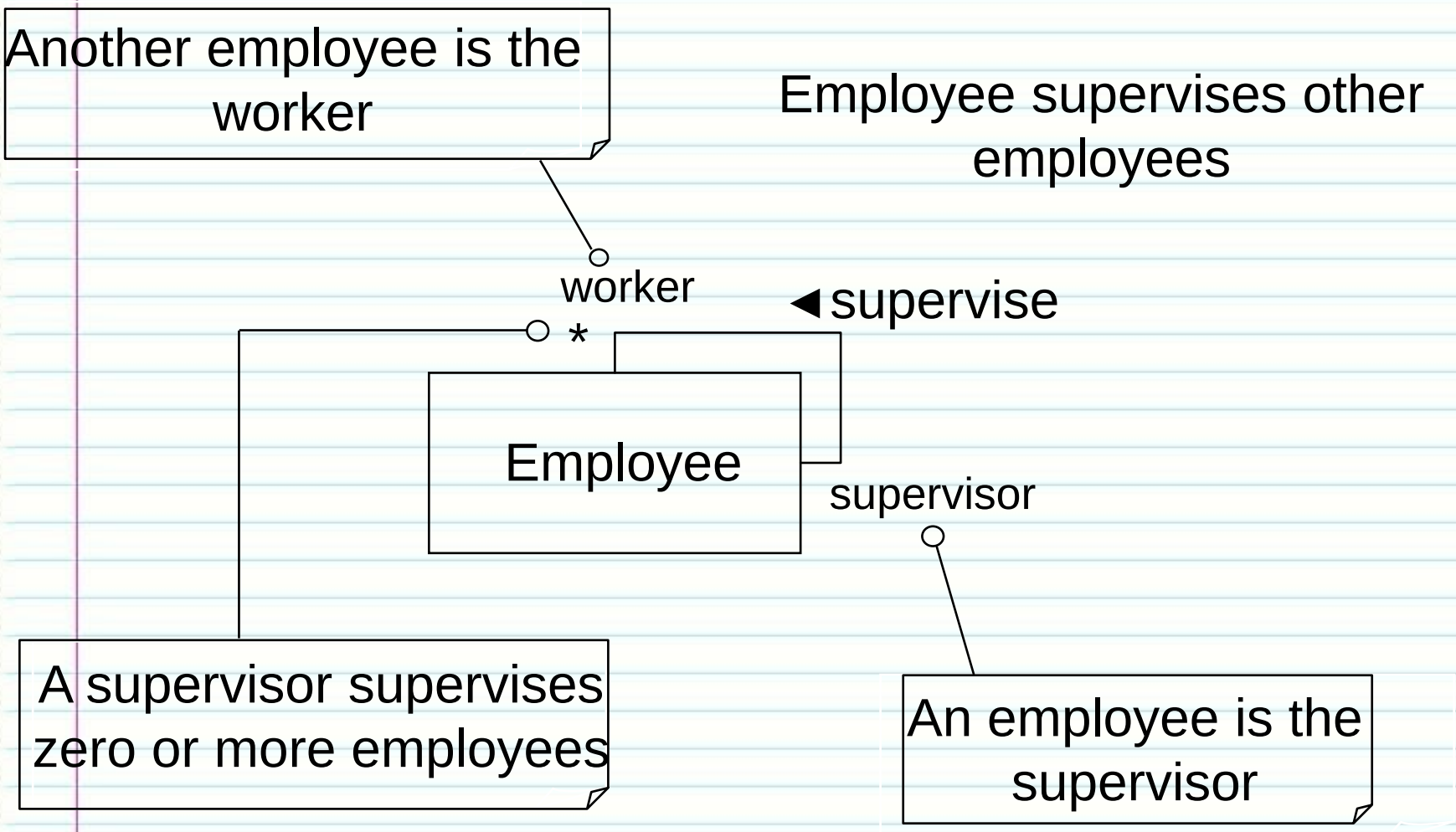


Enroll is not an inheritance or aggregation relationship

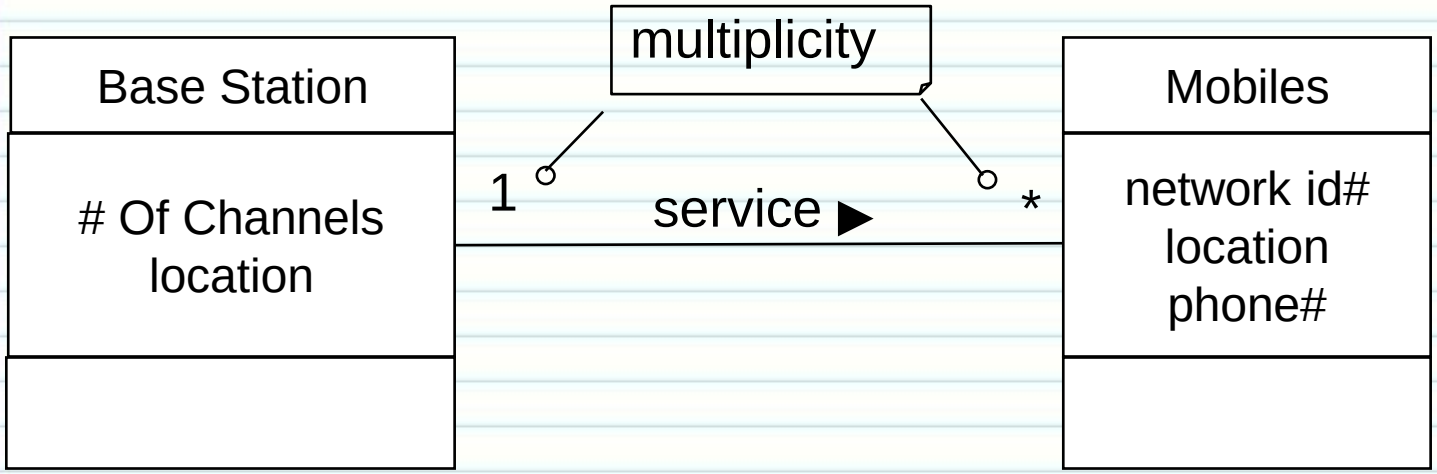
Role and Association Direction



Role and Multiplicity



Multiplicity Assertion/Constraint

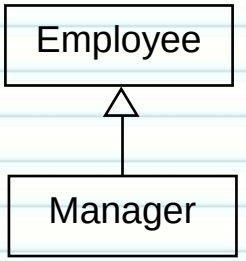
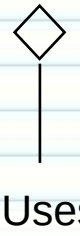
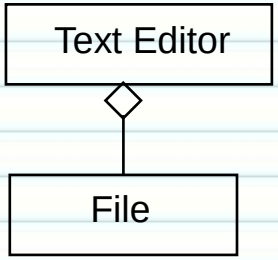
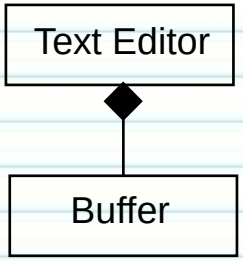


One base station services zero or more mobiles and every mobile is serviced by exactly one base station.

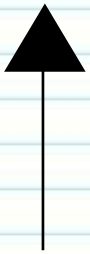
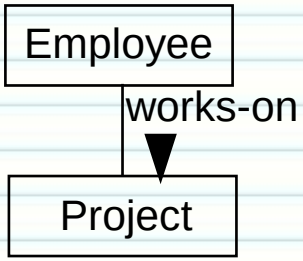
Other multiplicity constraints:

1	exactly one (default)	1..*	one or more
0..1	zero or one	m..n	m to n
, 0..	zero or more	n	exactly n

Summary: Relationships in UML Class Diagram

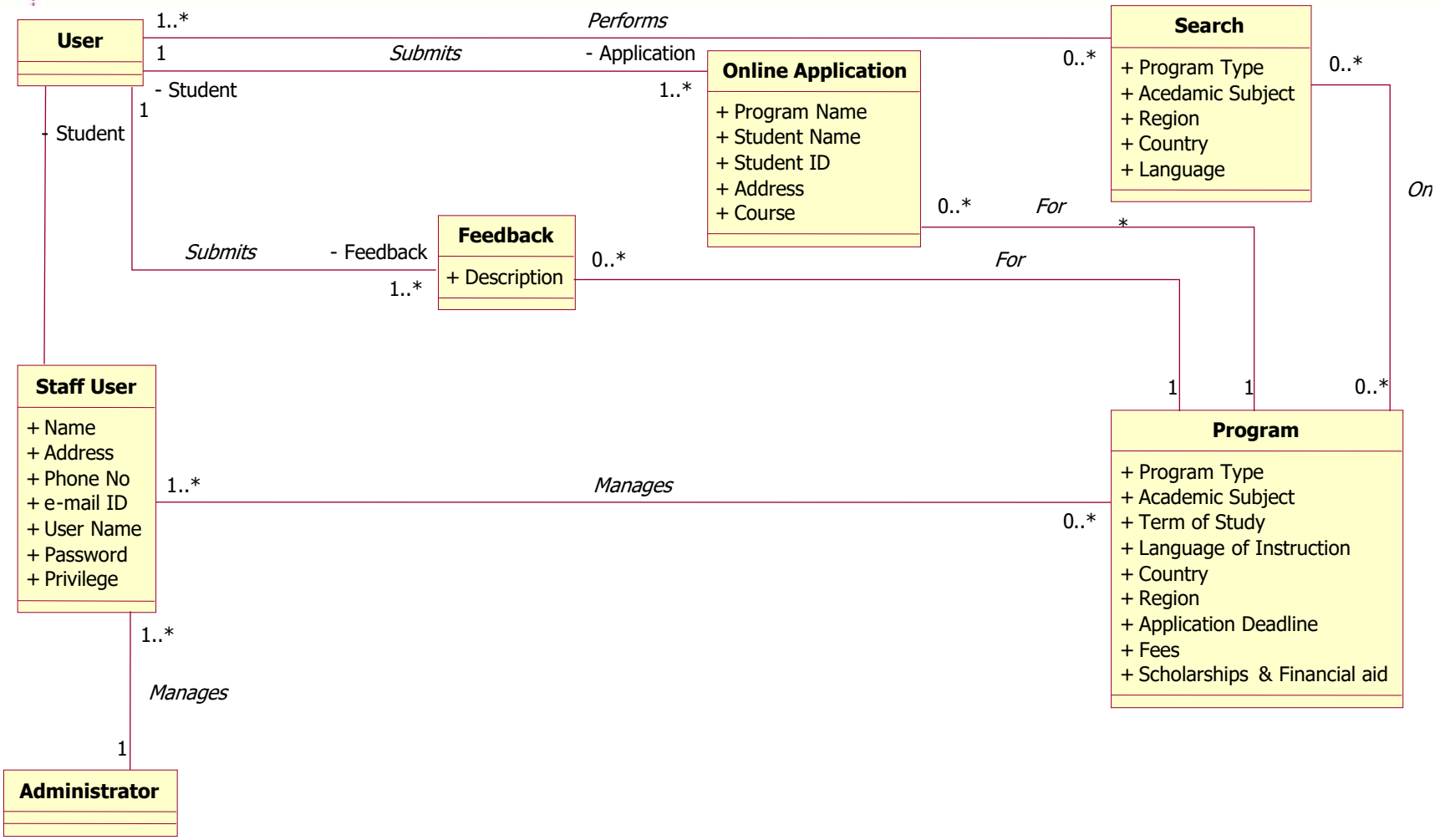
Notion	Notation		Example
Inheritance (between classes) IS-A relationship		pointing from subclass to super-class	
Aggregation (between classes) part-of relationship		File still exists even when the Text Editor is terminated	
Composition (between classes) aggregate exclusively owns the part		Buffer has no meaning without the Text Editor	

Relationships in UML Class Diagram

Notion	Notation	Example
Association (between classes) general relationship	 solid triangle shows the direction of association	

Note: there are more relationships in UML but we only need these three for domain modeling

Example Domain Model

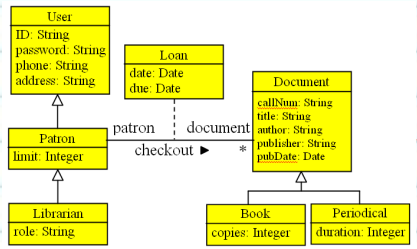


Domain Modeling Steps

1.Collecting application domain information

2. Brainstorming

3. Classifying brainstorming result



4. Visualizing the domain model

5. Reviewing the domain model.

Steps for Domain Modeling

- 1) Collecting application domain information
  - focus on the functional requirements
  - consider other requirements and documents
  - consider business descriptions

- 2) Brainstorming
  - list important application domain concepts
  - list their properties/attributes
  - list their relationships

Steps for Domain Modeling

3) Classifying the domain concepts into:
   - Classes
   - Attributes / attribute values
   - Relationships
     - Association, inheritance, aggregation

4) Visualizing the result using a UML class diagram

5) Review the Domain Model

Brainstorming: Rules to Apply

- The team members get together to identify and list domain-specific concepts and terms using:
  1. nouns / noun phrases
  2. "X of Y" expressions (e.g., color of car)
  3. transitive verbs
  4. adjectives
  5. numeric
  6. possession expressions (has/have, possess, etc.)
  7. "constituents / part of" expressions
  8. containment / containing expressions
  9. "X is a Y" expressions

Classifying Brainstorming Result

1.	nouns/noun phrases	⇒	class or attributes
2.	"X of Y" expressions	⇒	X is an attribute of Y
		⇒	X is part of Y
		⇒	X is a role in an association
3.	transitive verbs	⇒	association relationships
4.	adjectives	⇒	attribute values
5.	numeric	⇒	attribute / multiplicity values
6.	possession expressions (has/have, possess, etc.)	⇒	aggregation or attribute
7.	"consist of/part of" expression	⇒	aggregation relationships
8.	containment / containing expressions	⇒	association or aggregation
9.	"X is a Y" expressions	⇒	inheritance

Objects have independent existence, attributes do not.

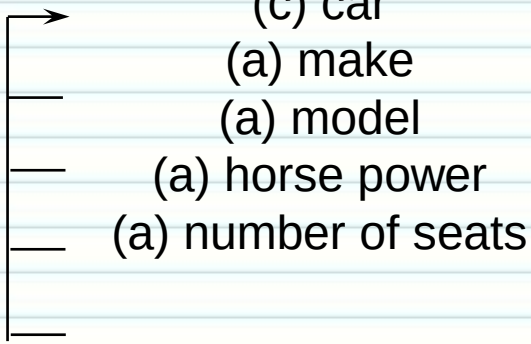
Example

noun/noun phrases

numeric

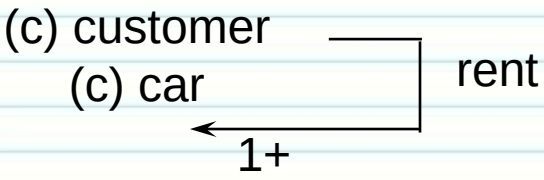
A car has make, model, horse power, number of seats ...

possession expression



Car has independent existence. Make, model, horse power, and number of seats do not.

A customer can rent one or more cars ...



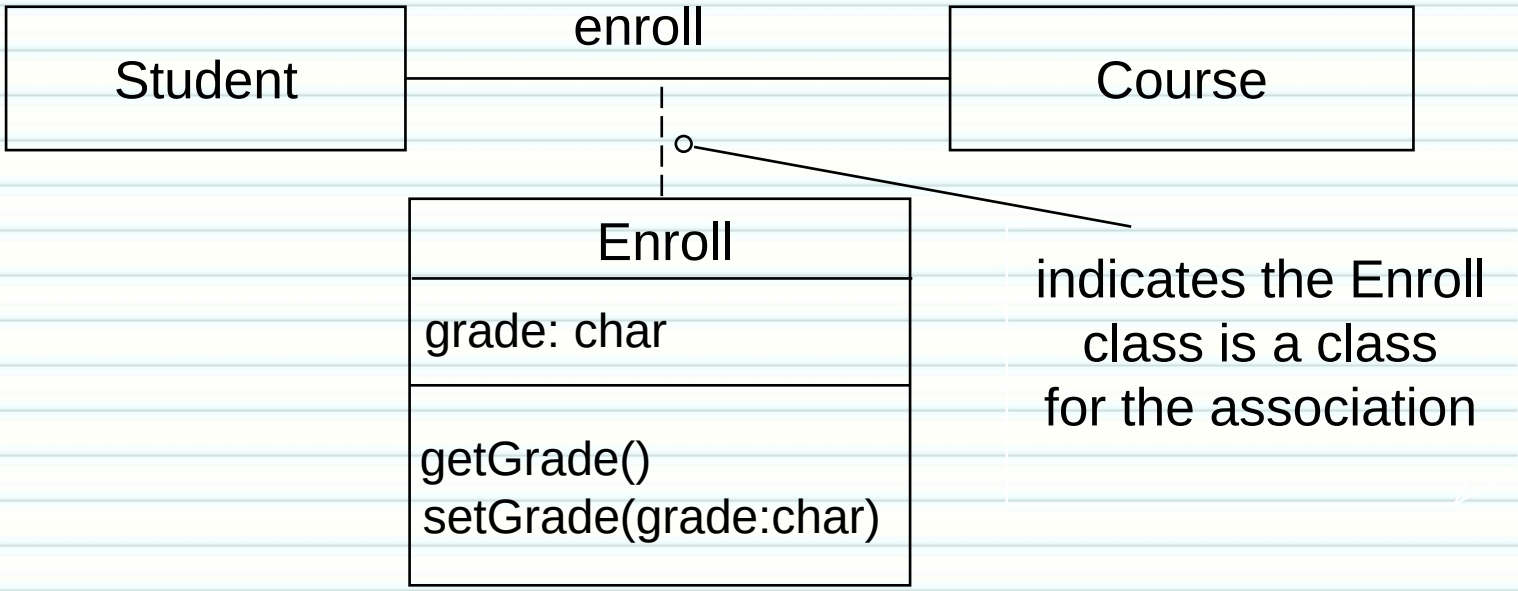
Association Classes

- An association class is used to model an association as a class
  - Association classes often occur in many-to-one and many-to-many associations where the association itself has attributes
- As an example, consider a many-to-many association between classes Person and Company.
  - The association could have properties of salary, jobClassification, startDate, and others
  - In this case, the association is more correctly modeled as an association class with attributes rather than trying to fold the attributes into one of the classes in the association
- Here are some pointers to consider when modeling with association classes
  - You cannot attach the same class to more than one association
    - an association class **is the association**
  - The name of the association is usually omitted since it is considered to be the same as that of the attached class
    - Distinguish between the use of an association class as a modeling technique and the implementation of the association class
      - There can be several ways to implement an association class

Association Class

An association class defines properties and operations for an association between two classes.

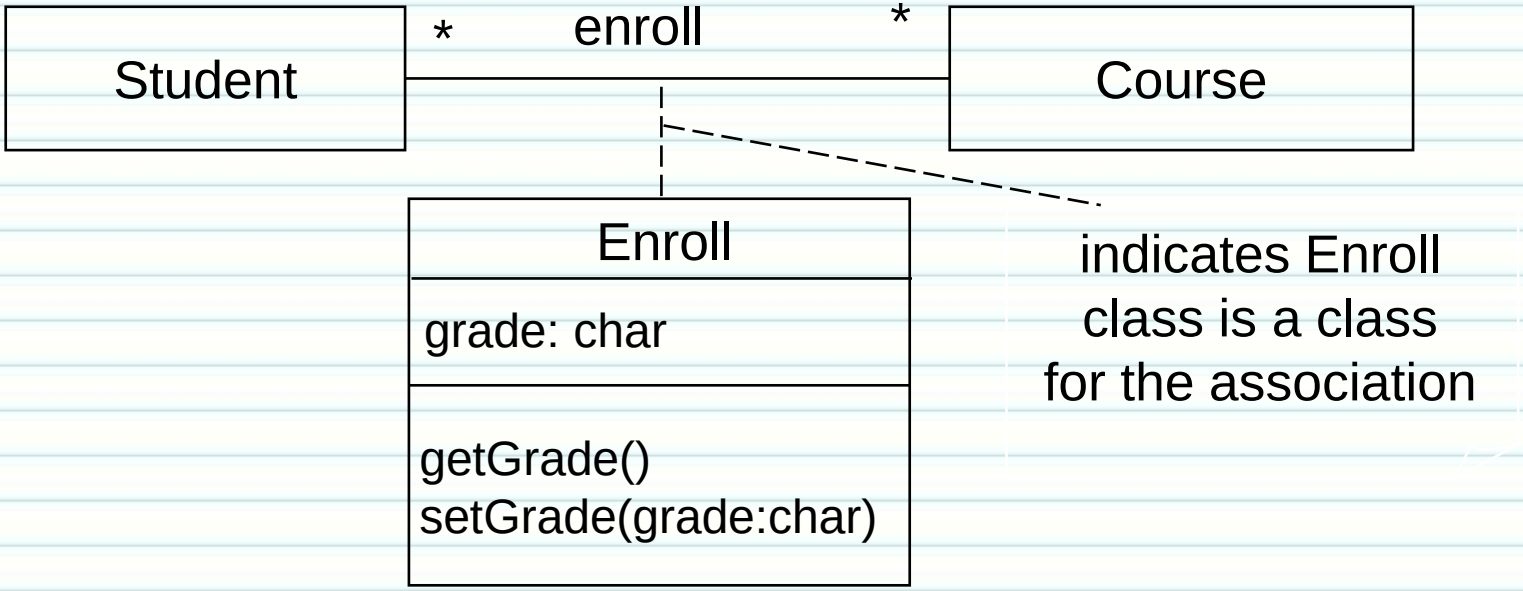
Students enroll in courses and receive grades.



Association Class

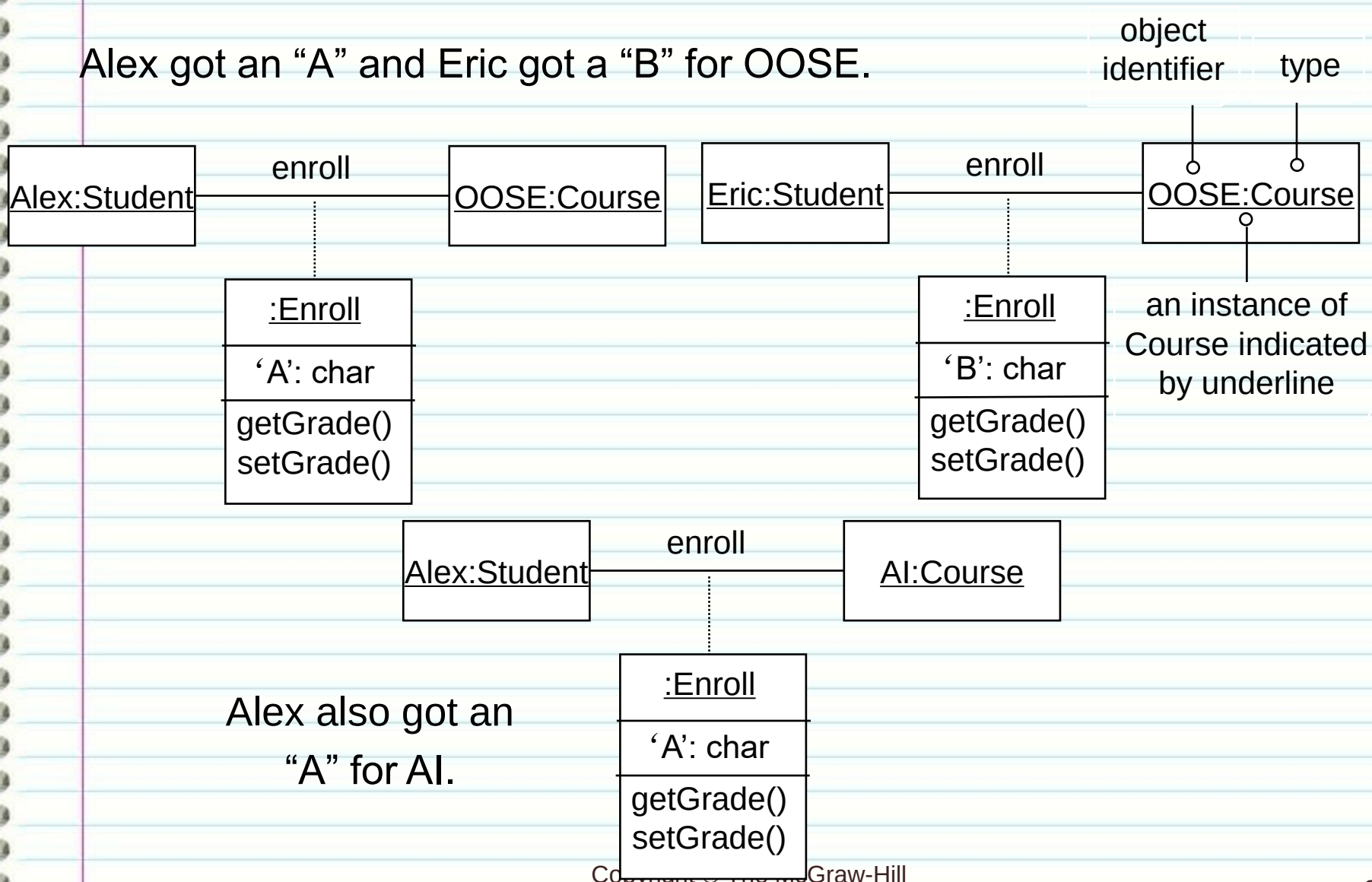
An association class defines properties and operations for an association between two classes.

Students enroll in courses and receive grades.



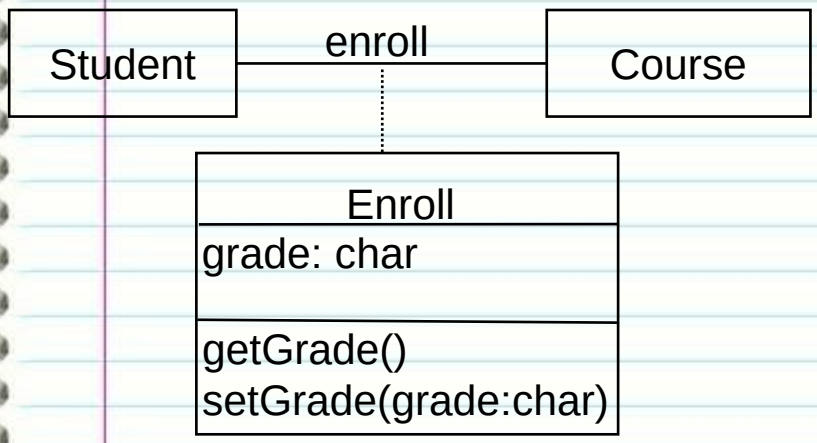
Understand Association Class

Alex got an “A” and Eric got a “B” for OOSE.



Alex also got an “A” for AI.

Understand Association Class




```
Student *alex=new Student( ... );
Course *oose=new Course ( ... );
...
Enroll *e=new Enroll(alex, oose);
e->setGrade('A');
```

Implementation

```
class Student { ... }
class Course {...}
class Enroll {
private:
   char grade;
   Student* student;
   Course* course;
public:
   Enroll (Student* s, Course* c);
   char getGrade();
   void setGrade(char grade);
}
Enroll::Enroll(Student* s, Course* c) {
   student=s; course=c;
}
```

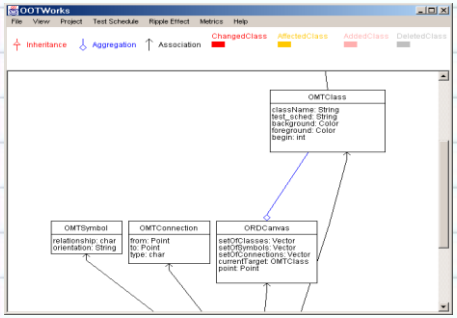
Tip for Domain Modeling

Do not do brainstorming and drawing at the same time; the result could be very poor

1) Team brainstorming: List the concepts and then classify them on a whiteboard

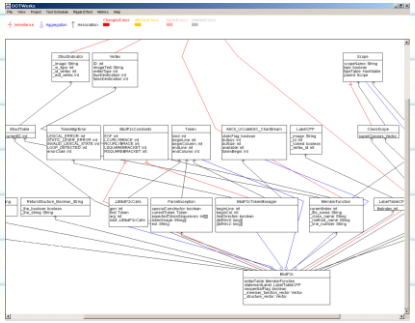
2) Take a picture(s) of the whiteboard using a digital camera

3) Email the digital images to team members



4) Have a member or two to convert the result to a UML class diagram

5) Email the UML class diagram to all members to review



6) Modify the diagram to reflect corrections and comments

Applying Agile Principles

1. *Work closely with the customer and users to understand their application and application domain.*

2. *Perform domain modeling only if it is needed. Keep it simple and expand it incrementally.*

3. *Domain modeling may be performed simultaneously with actor-system interaction modeling, object interaction modeling, object state modeling, and activity modeling.*