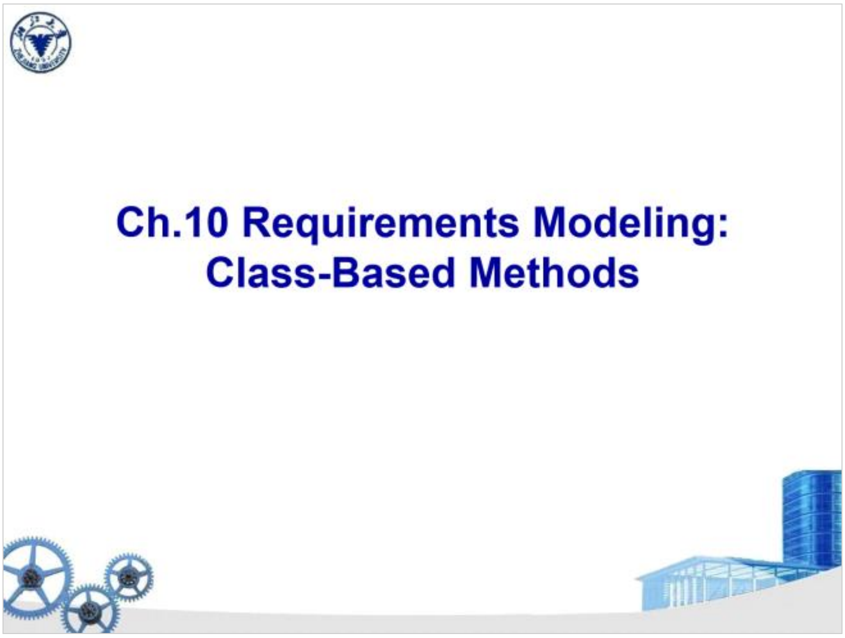
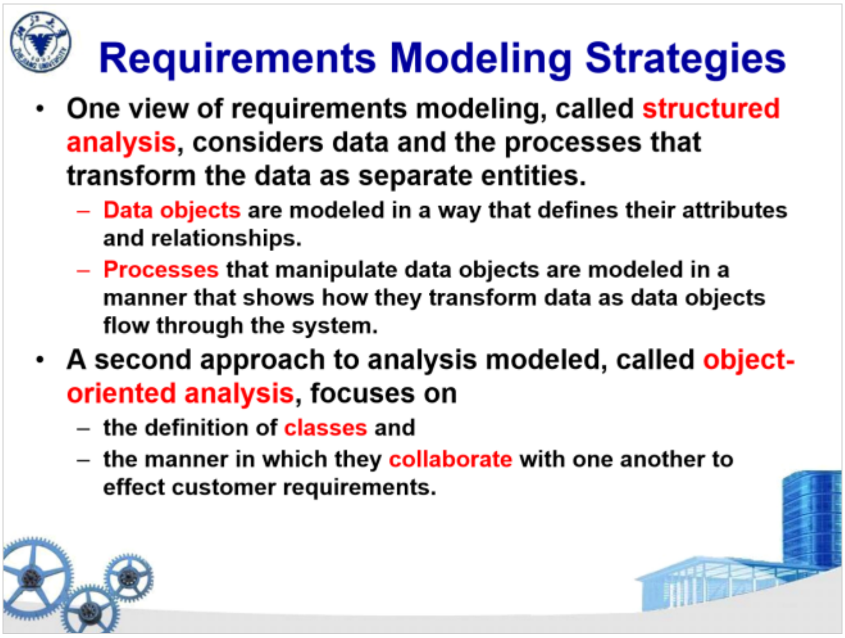
Lecture Note 10

2024年4月7日 10:26



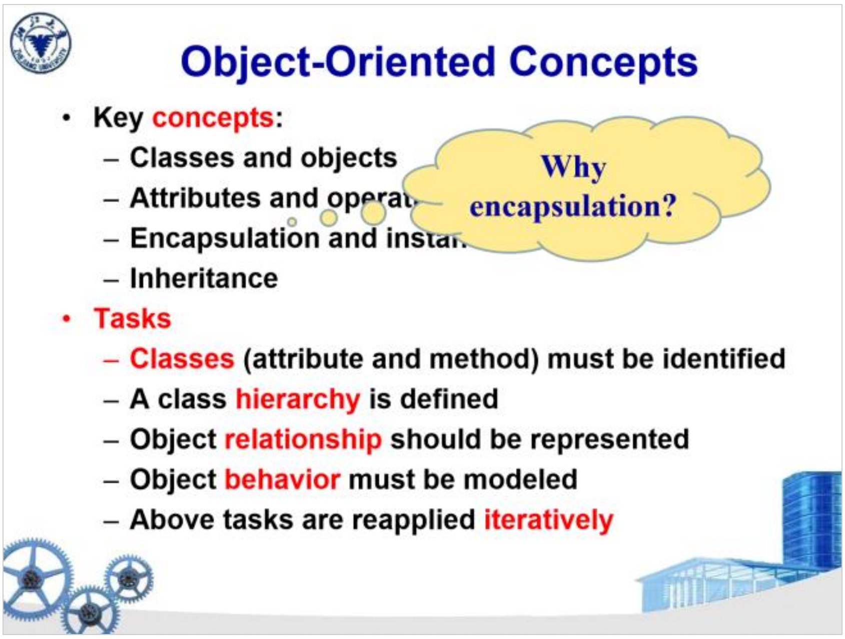
Ch.10 Requirements Modeling: Class-Based Methods



Requirements Modeling Strategies

- One view of requirements modeling, called structured analysis, considers data and the processes that transform the data as separate entities.
  - Data objects are modeled in a way that defines their attributes and relationships.
  - Processes that manipulate data objects are modeled in a manner that shows how they transform data as data objects flow through the system.
- A second approach to analysis modeled, called object-oriented analysis, focuses on
  - the definition of classes and
  - the manner in which they collaborate with one another to effect customer requirements.

A



Object-Oriented Concepts

- Key concepts:
  - Classes and objects
  - Attributes and operat
  - Encapsulation and instan
  - Inheritance
- Tasks
  - Classes (attribute and method) must be identified
  - A class hierarchy is defined
  - Object relationship should be represented
  - Object behavior must be modeled
  - Above tasks are reapplied iteratively

Why encapsulation?

Class based model

面向service的理念（又进一步）：面向更开放性环境，企业之间如何协同

面向对象的理念

1、类和对象

2、属性和操作

3、封装和实例化

4、继承

主要任务：

1、搞清楚系统有哪些分析阶段的类（分析类）、设计类

2、类之间有层次关系（父类、子类之间的继承关系），将层次关系定义

3、类之间的关系（横向关系、纵向关系）

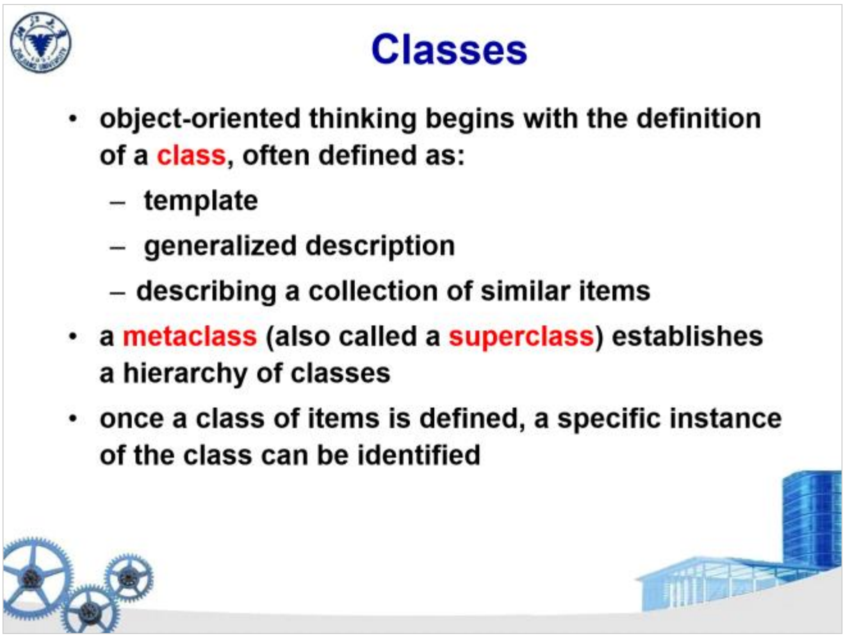
4、类之间的行为，如何进行交互？交互的行为？用状态图表示

5、

Classes

状态图表示

5、



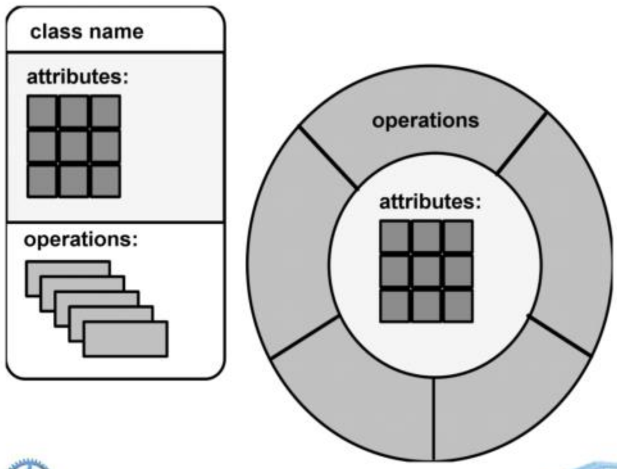
Classes

- object-oriented thinking begins with the definition of a class, often defined as:
  - template
  - generalized description
  - describing a collection of similar items
- a metaclass (also called a superclass) establishes a hierarchy of classes
- once a class of items is defined, a specific instance of the class can be identified

是对一类对象的抽象

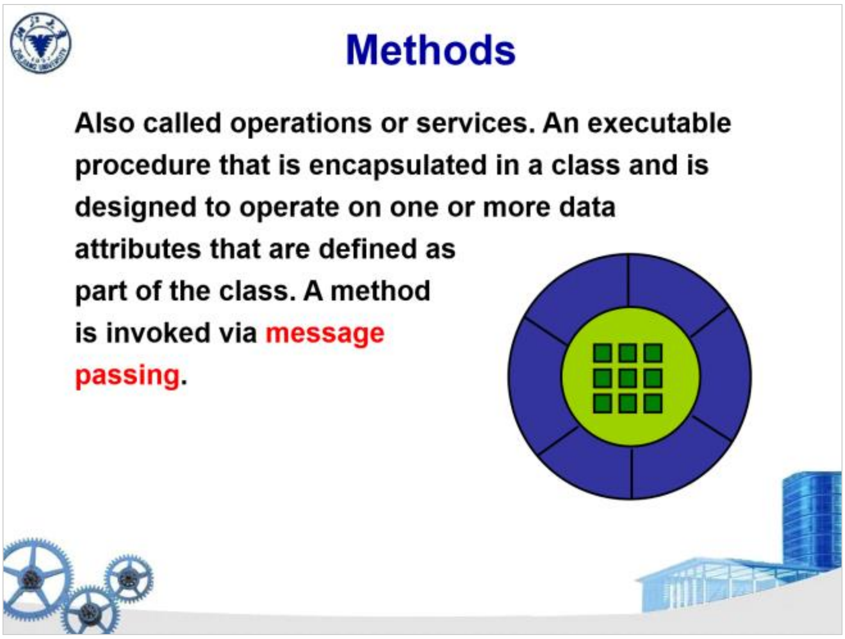
类之间有各种层次关系。基于一个superclass可以延伸出很多子类。继承/重载

Building a Class



一个类包含类名、属性（成员变量）、操作（成员函数）

一半类名都有规范的定义（eg. 第一个字母大写）



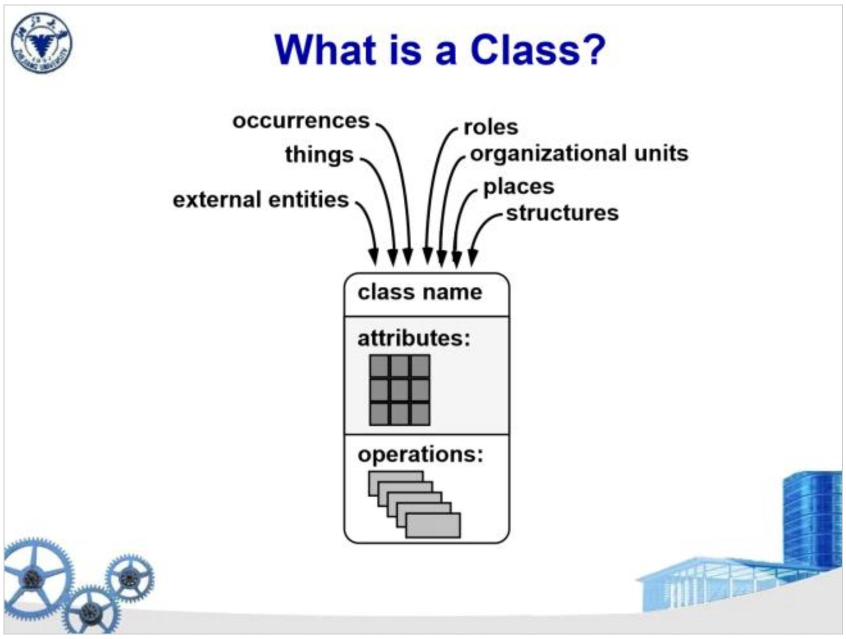
Methods

Also called operations or services. An executable procedure that is encapsulated in a class and is designed to operate on one or more data attributes that are defined as part of the class. A method is invoked via message passing.

类有很多方法组成。方法是用来描述类的能力。是对类的某些动作的抽象。每个方法就是一个成员函数

What is a Class?

什么可能是类：



What is a Class?

什么可能是类:

1、外部实体（eg. 教务管理系统需要去出版社购买教材。出版社就是外部实体）

2、事物（eg. 教材）

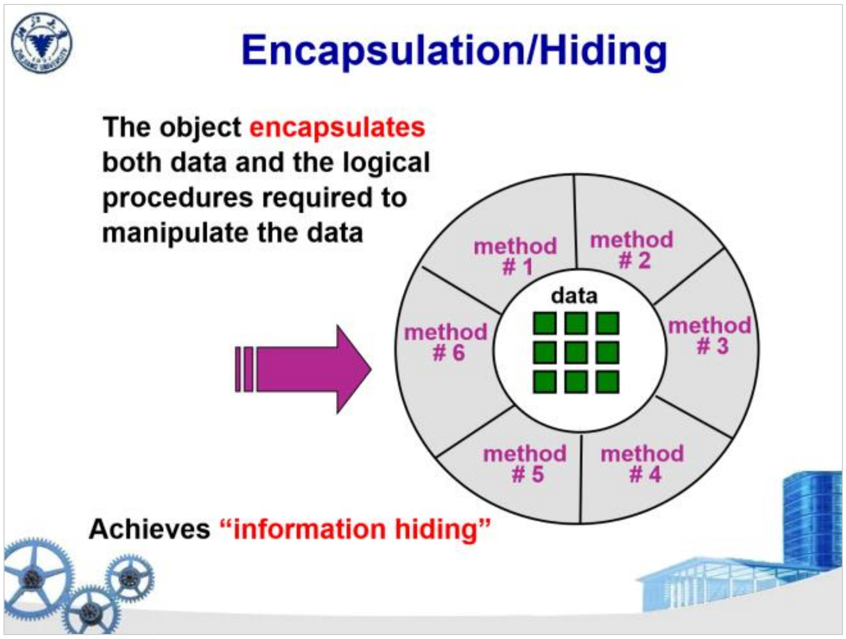
3、事件（eg. 开课）

4、角色（eg. 学生、老师、管理者）

5、组织单元（eg. 班级、院系、学校）

6、场所（eg. 教室）

7、结构/数据结构（eg. 开课的相关信息）



Encapsulation/Hiding

The object encapsulates both data and the logical procedures required to manipulate the data

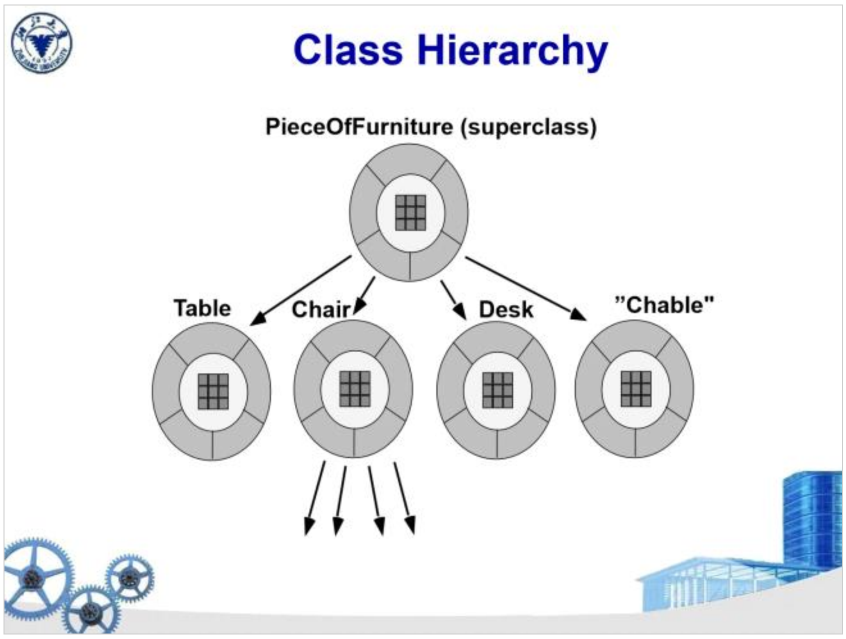
Achieves “information hiding”

用封装的思想对信息进行抽象

对象之间的关联方式很复杂。因此可以把它模块化，将模块相关联的东西放在一个对象里，也就是封装，坐信息的隐藏。

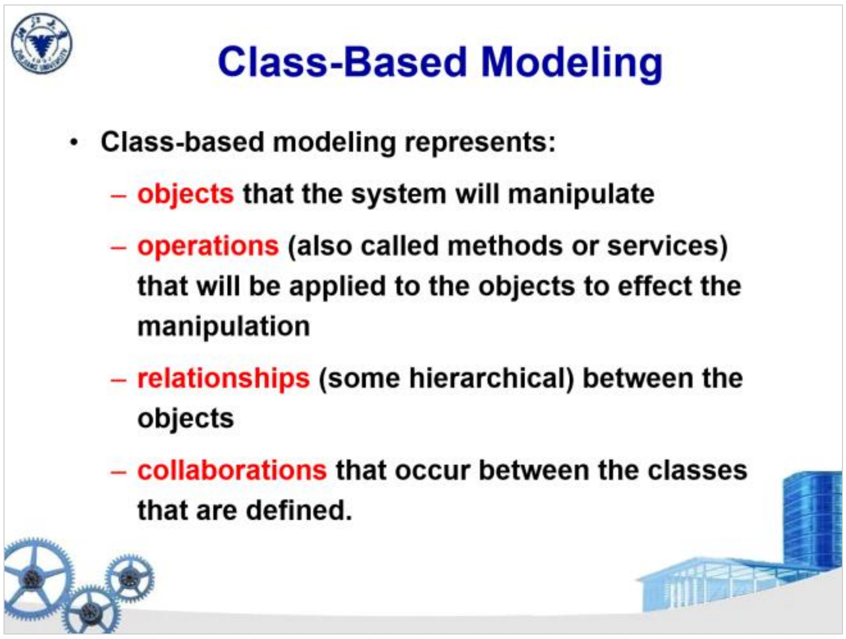
交互是通过接口进行交互。

更改一个程序时，只需要对对象内部进行更改就好，不需要对整个接口更改。可以让影响最小。



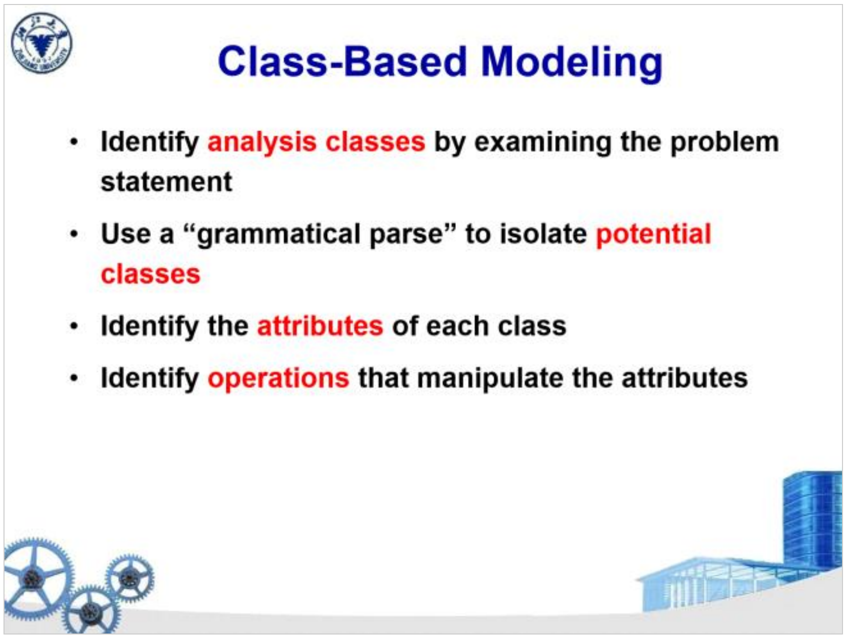
Class Hierarchy

Class-Based Modeling



Class-Based Modeling

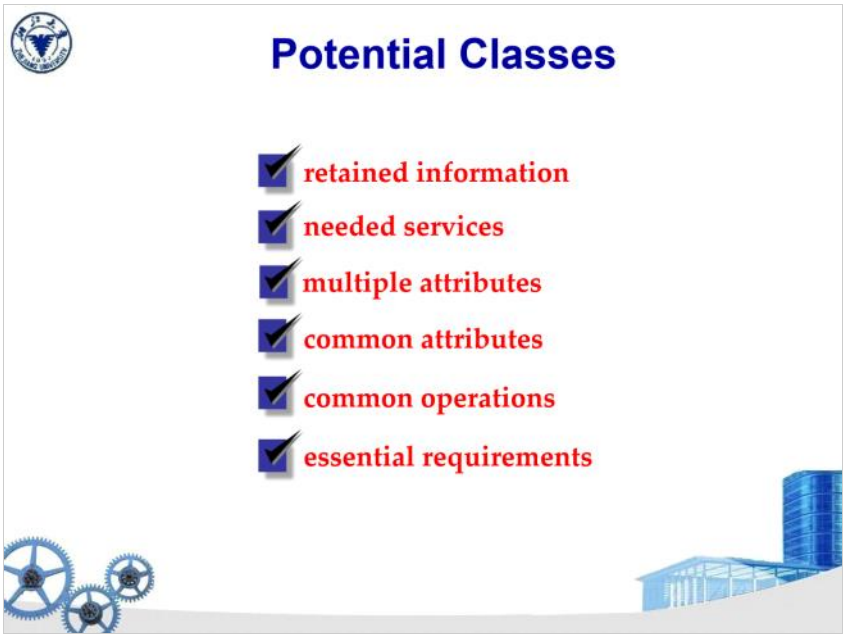
- Class-based modeling represents:
  - objects that the system will manipulate
  - operations (also called methods or services) that will be applied to the objects to effect the manipulation
  - relationships (some hierarchical) between the objects
  - collaborations that occur between the classes that are defined.



Class-Based Modeling

- Identify analysis classes by examining the problem statement
- Use a “grammatical parse” to isolate potential classes
- Identify the attributes of each class
- Identify operations that manipulate the attributes

识别整个问题域相关的分析类
如何识别？
使用语法分析的方法：输入是用户的描述，输出是分析类模型
通过语法分析识别出潜在的类（eg. 用户要在家安装安防系统），找到名词和动词，名词可能就是潜在类。动词可能变成某个类的成员函数。
名词有两种可能：有可能是某个类，也可能是某个属性。因此要对名词进行聚类

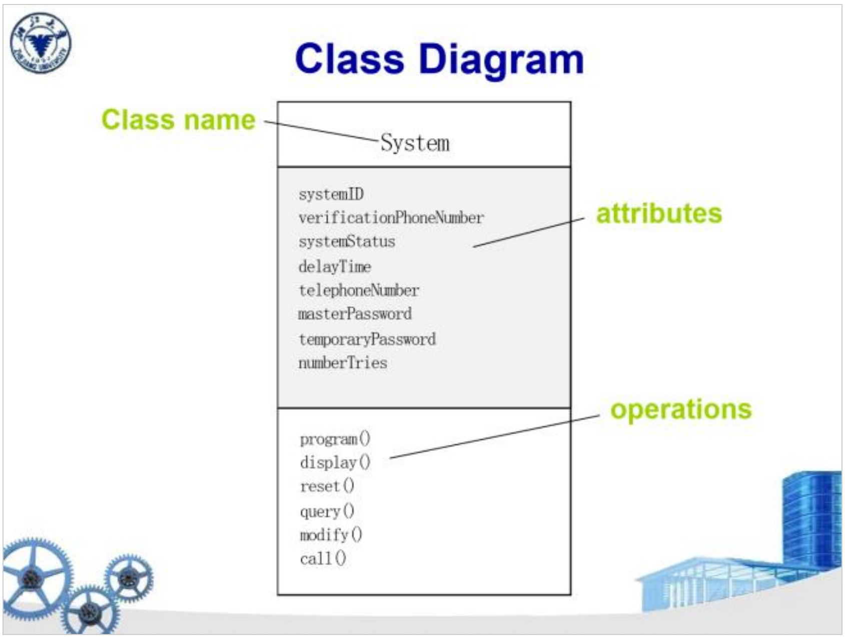


Potential Classes

- ✓ retained information
- ✓ needed services
- ✓ multiple attributes
- ✓ common attributes
- ✓ common operations
- ✓ essential requirements

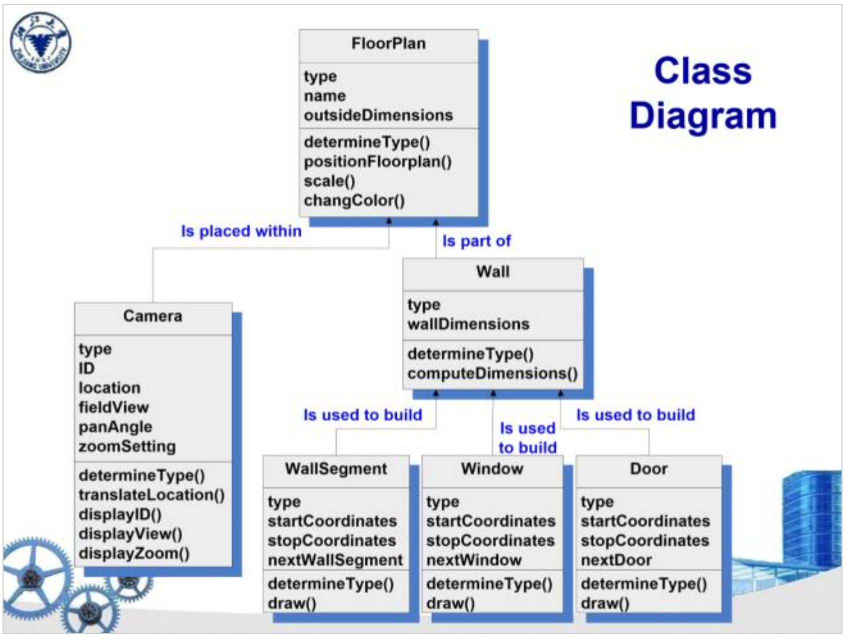
潜在类的属性：
1、各种保留信息
2、动态行为，不仅仅是一个简单的数据结构
3、多个属性
4、公共属性
5、公共操作
6、必要需求

Class Diagram



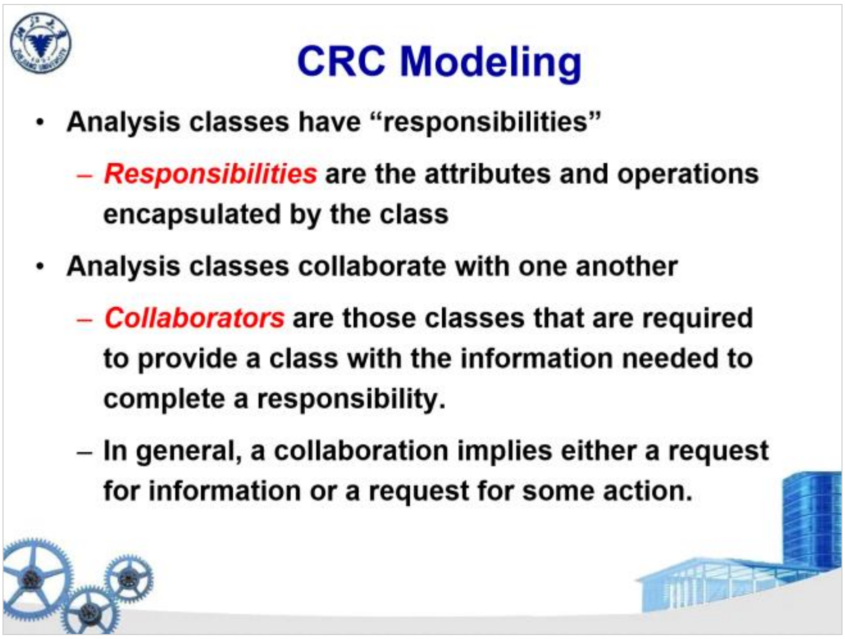
Class Diagram

类一般可以使用类图进行表示
可以使用UML
用工具做，分析模型可以转换为design model，这些过程可以做到自动化



Class Diagram

类之间的层次关系除了继承之外还可以有其它的层次关系。
安防系统
首先要有个平面布局图，就是个类FloorPlan，允许别人放置camera。墙体组成了平面布局
墙上面会有墙段、窗户、门

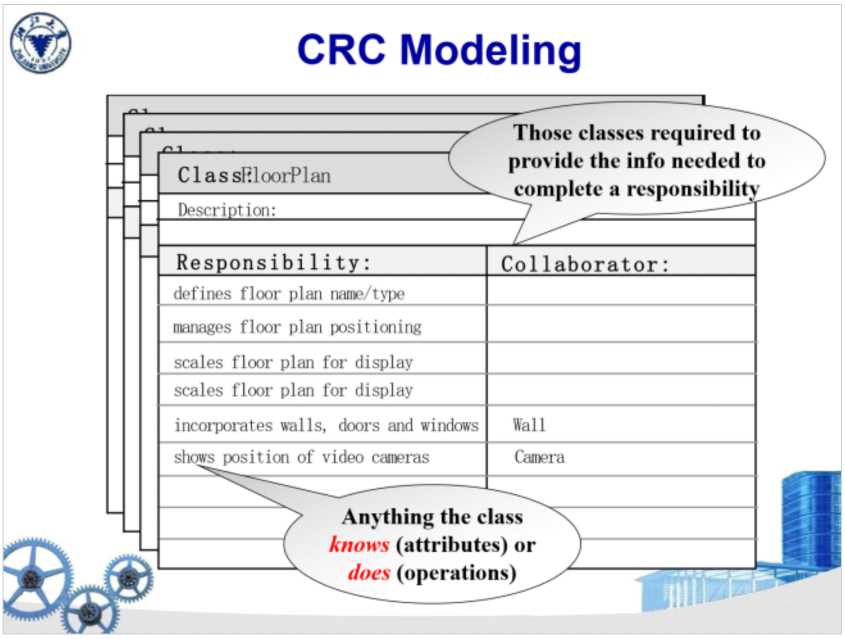


CRC Modeling

- Analysis classes have "responsibilities"
  - *Responsibilities* are the attributes and operations encapsulated by the class
- Analysis classes collaborate with one another
  - *Collaborators* are those classes that are required to provide a class with the information needed to complete a responsibility.
  - In general, a collaboration implies either a request for information or a request for some action.

类之间还有相互的协同关系，可以用CRC model进行表示。
类、责任、协调者

CRC Modeling



CRC Modeling

Class: FloorPlan

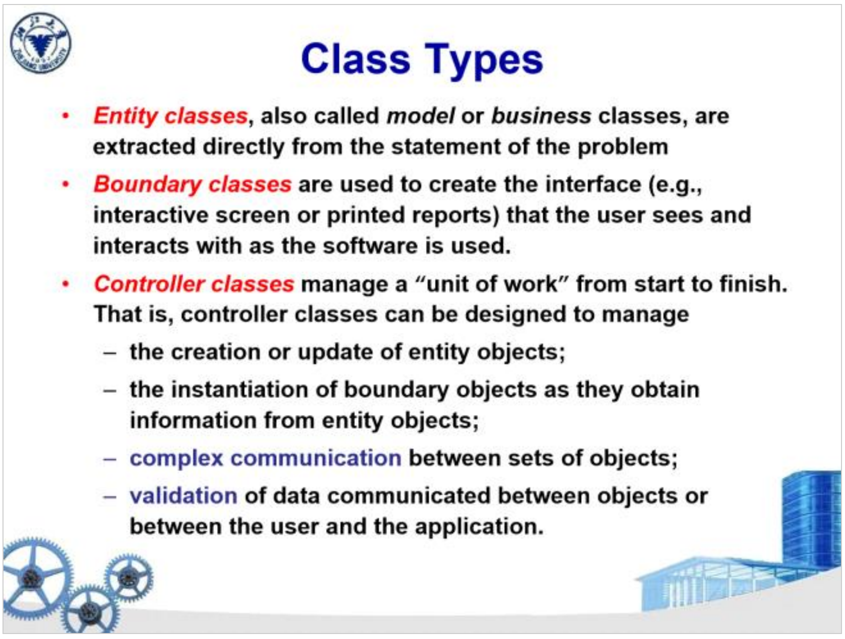
Description:

Responsibility:	Collaborator:
defines floor plan name/type	
manages floor plan positioning	
scales floor plan for display	
scales floor plan for display	
incorporates walls, doors and windows	Wall
shows position of video cameras	Camera

Those classes required to provide the info needed to complete a responsibility

Anything the class *knows* (attributes) or *does* (operations)

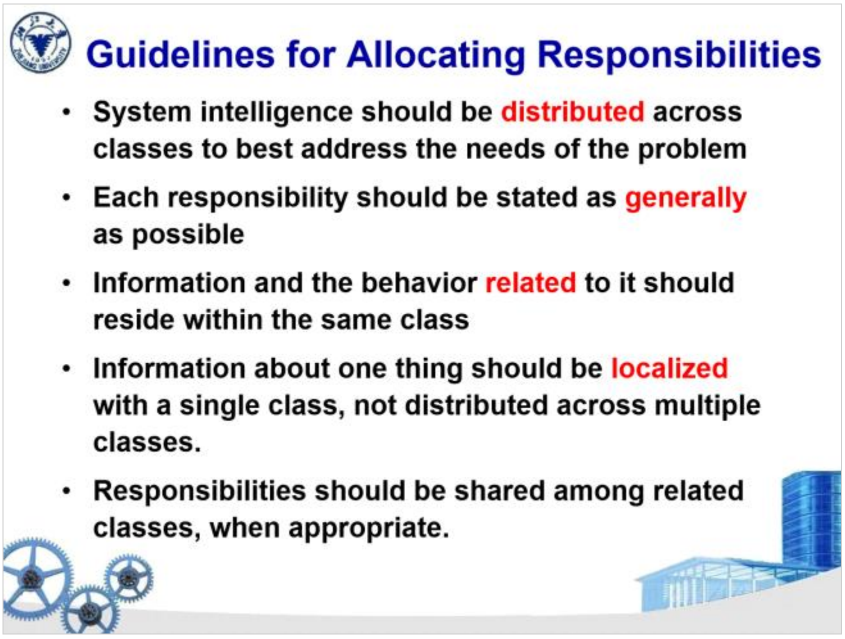
CRC中每个类是一张卡片。要包含三个要素
1、class（eg. 平面布局图），对类进行简单描述
2、responsibility，责任是类的属性和操作，是成员变量和成员函数
3、需要协同类来辅助定义协同关系



Class Types

- ***Entity classes***, also called *model* or *business* classes, are extracted directly from the statement of the problem
- ***Boundary classes*** are used to create the interface (e.g., interactive screen or printed reports) that the user sees and interacts with as the software is used.
- ***Controller classes*** manage a "unit of work" from start to finish. That is, controller classes can be designed to manage
  - the creation or update of entity objects;
  - the instantiation of boundary objects as they obtain information from entity objects;
  - complex communication between sets of objects;
  - validation of data communicated between objects or between the user and the application.

实体类：用来描述整个商业逻辑的类
边界类：用来做interface的类。可能不一定显性地表述出来。边界类可以和用户进行交互
控制类：将实体类，边界类放在一起
MVC：M就是实体类，V就是边界类，C就是控制类
控制类可以管理实体类的生命周期，初始化边界类。
做各个类之间的通讯。对象之间传递信息，信息需要做验证



Guidelines for Allocating Responsibilities

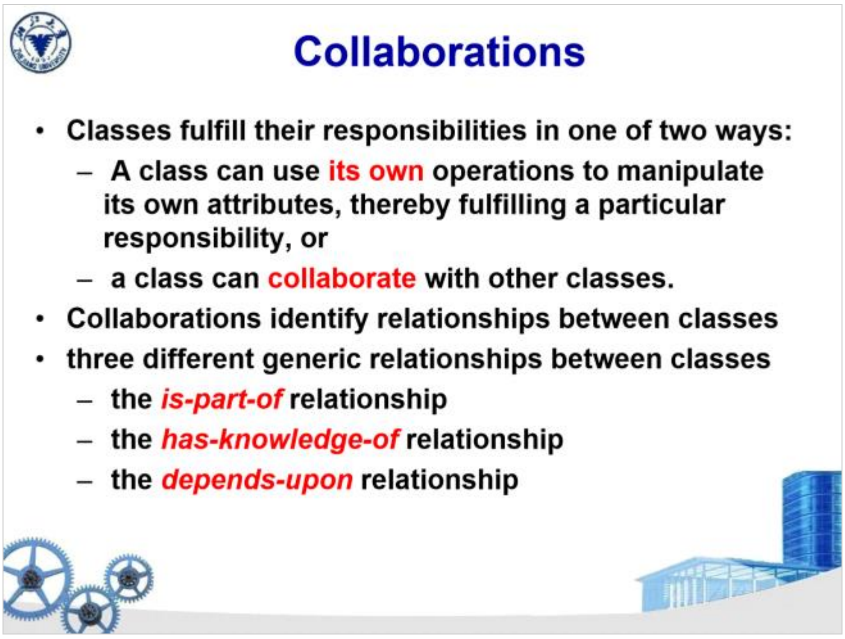
- System intelligence should be distributed across classes to best address the needs of the problem
- Each responsibility should be stated as generally as possible
- Information and the behavior related to it should reside within the same class
- Information about one thing should be localized with a single class, not distributed across multiple classes.
- Responsibilities should be shared among related classes, when appropriate.

类要分散，不能只有几个类。那样就不是面向对象的了。OO希望可以分散、模块化。
信息需要具有普适性。

信息和相关行为要放在一起。这样更改时影响范围就限制在一小块。
同一个事情的相关信息要本地化（“内聚”），即要放在一个类里。如果放在多个类就需要交互，就很麻烦。
各个类公共协同完成一件事情

Collaborations

类的协同



Collaborations

- Classes fulfill their responsibilities in one of two ways:
  - A class can use its own operations to manipulate its own attributes, thereby fulfilling a particular responsibility, or
  - a class can collaborate with other classes.
- Collaborations identify relationships between classes
- three different generic relationships between classes
  - the *is-part-of* relationship
  - the *has-knowledge-of* relationship
  - the *depends-upon* relationship

类的协同

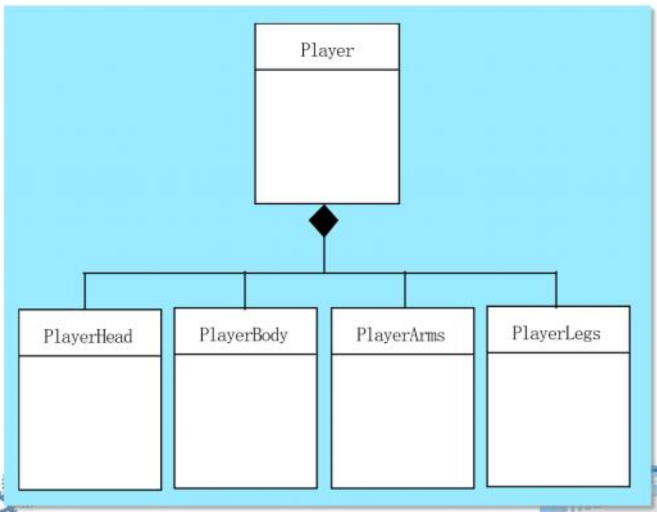
类可以调用自己的成员函数完成各种动作，也可以和其它类协同，就形成了协同关系：

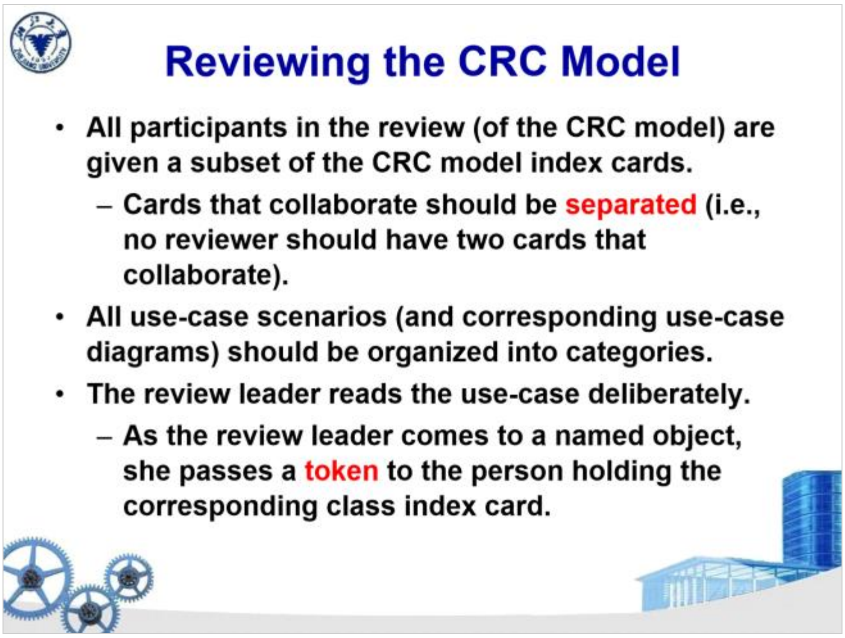
1、is-part-of：这个类是另一个类的一部分

2、has-knowledge-of：两个类之间传递信息

3、depends-upon：依赖关系

Composite Aggregate Class





Reviewing the CRC Model

- All participants in the review (of the CRC model) are given a subset of the CRC model index cards.
  - Cards that collaborate should be separated (i.e., no reviewer should have two cards that collaborate).
- All use-case scenarios (and corresponding use-case diagrams) should be organized into categories.
- The review leader reads the use-case deliberately.
  - As the review leader comes to a named object, she passes a token to the person holding the corresponding class index card.

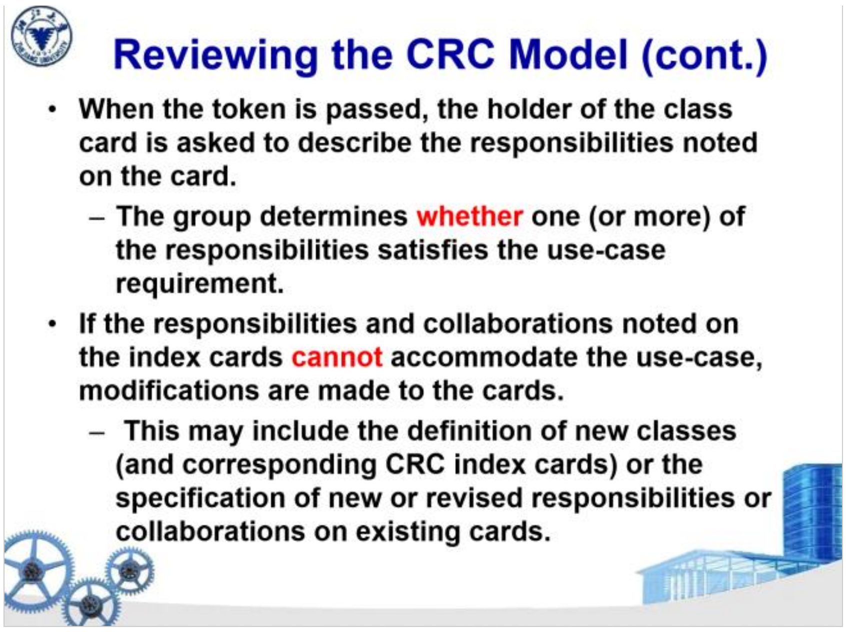
需要对CRC model进行审查

这样就需要成立一个review组。

首先分配任务。每个人会拿到一些卡片。如果两个类之间存在协同关系，就分给两个人。

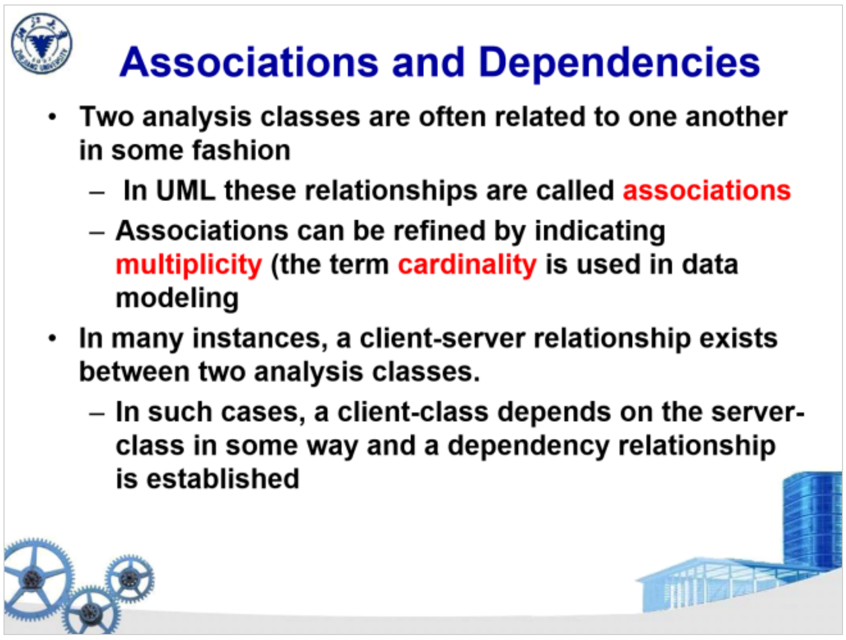
将所有的用例找到。拿出一个场景，看场景的描述，如果发现一个类，就把令牌交给掌握这个类的评审员。评审员会看描述里的需求，看看这个类可不可以完成需求。如果可以完成就ok。如果需要涉及第三方，则会将令牌传递给下一个人。

Reviewing the CRC Model (cont.)



Reviewing the CRC Model (cont.)

- When the token is passed, the holder of the class card is asked to describe the responsibilities noted on the card.
  - The group determines whether one (or more) of the responsibilities satisfies the use-case requirement.
- If the responsibilities and collaborations noted on the index cards cannot accommodate the use-case, modifications are made to the cards.
  - This may include the definition of new classes (and corresponding CRC index cards) or the specification of new or revised responsibilities or collaborations on existing cards.

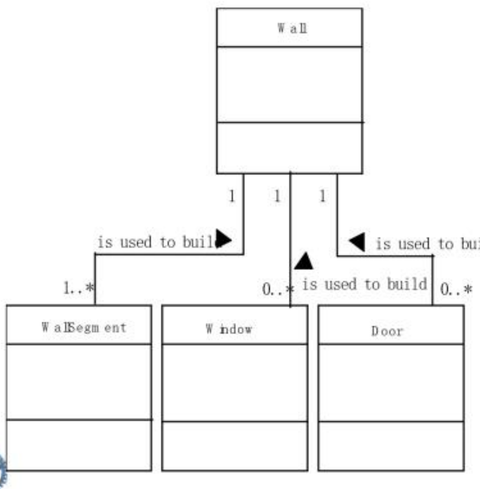


Associations and Dependencies

- Two analysis classes are often related to one another in some fashion
  - In UML these relationships are called associations
  - Associations can be refined by indicating multiplicity (the term cardinality is used in data modeling
- In many instances, a client-server relationship exists between two analysis classes.
  - In such cases, a client-class depends on the server-class in some way and a dependency relationship is established

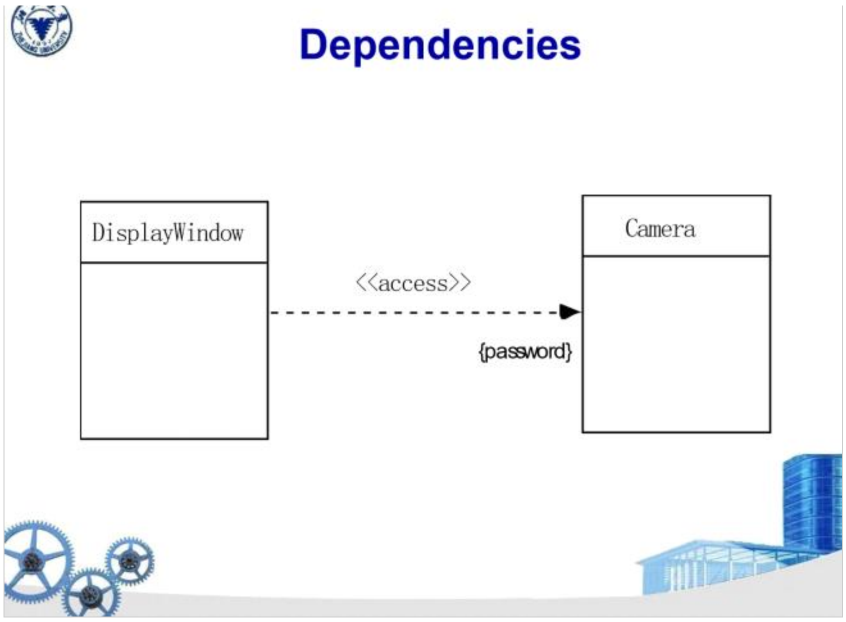
类之间有关联关系和依赖关系。
关联关系具有多样性，例如一对一、一对多

Multiplicity

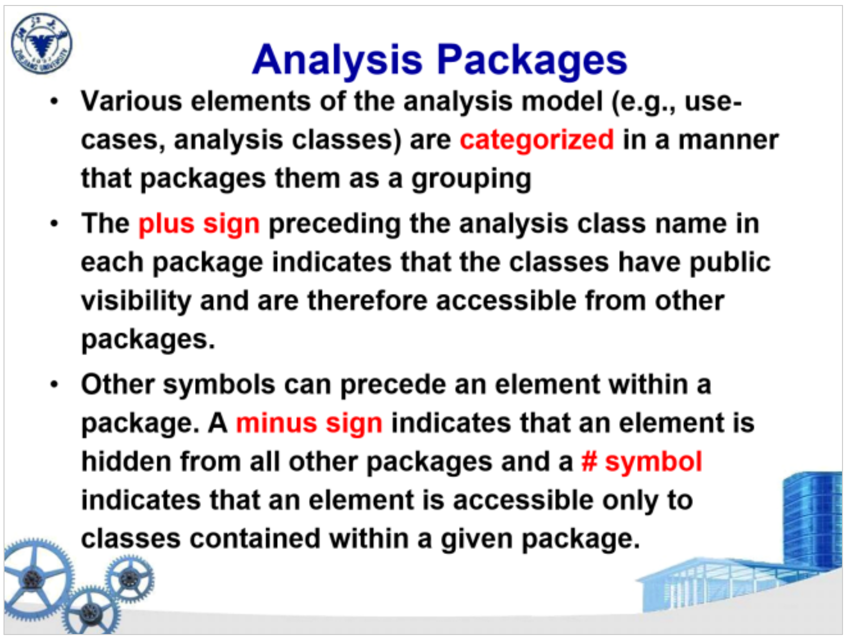


墙和窗户的对应关系是一对0或一对多。因此用多样性描述类之间的关联关系。

Dependencies



Dependencies

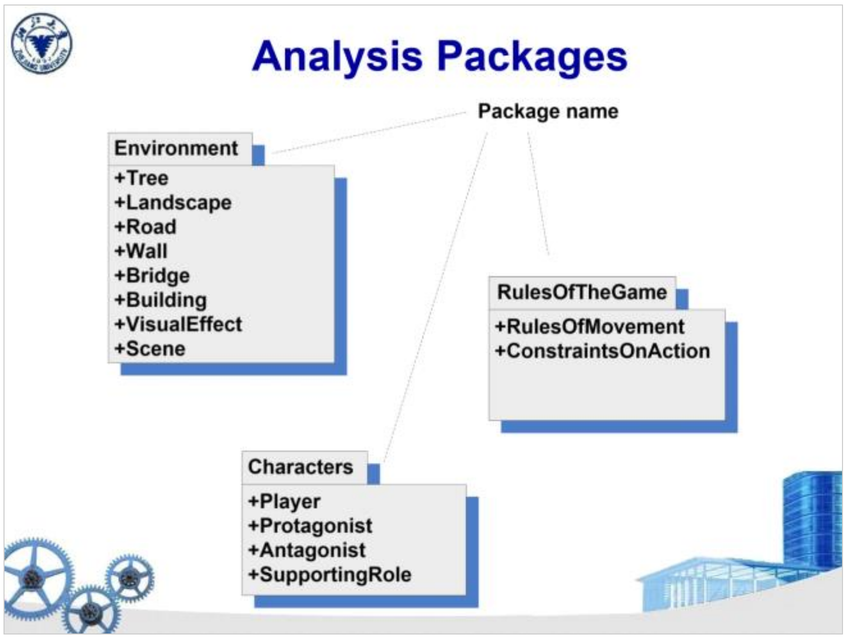


Analysis Packages

- Various elements of the analysis model (e.g., use-cases, analysis classes) are categorized in a manner that packages them as a grouping
- The plus sign preceding the analysis class name in each package indicates that the classes have public visibility and are therefore accessible from other packages.
- Other symbols can precede an element within a package. A minus sign indicates that an element is hidden from all other packages and a # symbol indicates that an element is accessible only to classes contained within a given package.

包

如果类都是一个list，那么系统就会很庞大。package可以对类进行划分。不同类可以组成一个模块。package是很多类的集合。按照功能或者语义组成不同模块。



Analysis Packages

+号表示公共类，除了包内部可见，其它包的类也可以直接调用。即所有包都可见。-号表示仅有这个包内部的类可用。其它包不可见。还有#号，表示这个包只有对某些类开放。