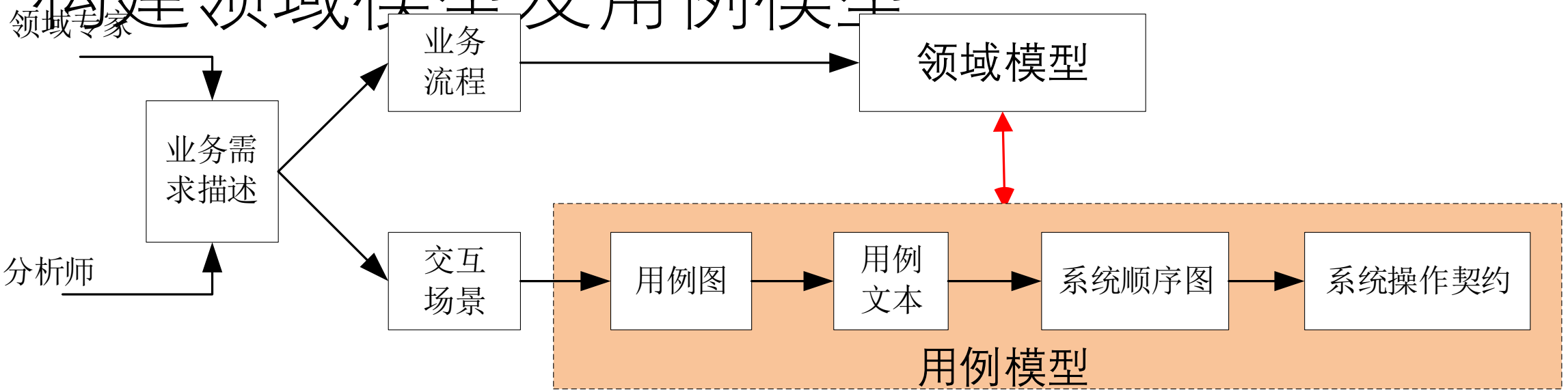
面向对象的需求分析建模

- 面向对象分析方法中的需求分析包含两个模型：领域模型和用例模型。
  - 领域模型表示了需求分析阶段“当前系统”逻辑模型的静态结构;
  - 用例模型是“目标系统”的逻辑模型，定义了“目标系统”做什么的需求。由以下四个部分组成:
    - 用例图
    - 用例说明
    - 系统顺序图（system sequence diagram）
    - 操作契约（operation contract）

构建领域模型及用例模型



业务需求描述：或称业务调研报告，由领域专家或需求分析师通过业务调研活动进行概括和总结；
业务流程：根据调研报告中的具体业务执行过程得到当前系统的物理模型，用于充分了解业务背景和活动；
交互场景：通过调研报告中确定的使用者角度描述角色与系统之间的交互过程，仅限于请求与服务之间的交互，而业务流程可表示服务是如何执行的过程；

领域模型

- 领域模型：针对某一特定领域内概念类或者对象的抽象可视化表示。
- 主要用于概括地描述业务背景及重要的业务流程，帮助软件开发人员在短时间内尽快了解业务。
  - 业务背景：可由用户需求说明书或者调研报告中具有代表业务概念或者业务对象的词汇获得，这些词汇可统称为“概念类”；并通过能够代表关系的词汇建立概念类之间的关系，表示成能够代表业务知识结构的类图；
  - 业务流程：一般由提交请求的角色及提供服务的对象所执行的活动（活动及任务节点）构成，活动的输出一般有数据对象和传给另一个活动的消息组成，建议使用UML的活动图进行描述。

领域模型与软件模型的区别

- 领域模型所关注的仅仅是客观世界中的事物并将其可视化，而非诸如Java或C#类的软件对象。
- 如果当前系统是非计算机系统，则以下元素不适用于领域模型

  软件制品，例如窗口、界面、数据库

  软件模型中具有职责或方法的对象

OO的关键思想：逻辑层（Logic Layer）中软件类的名称要源于领域模型的概念类和职责，减小人们的思维与软件模型之间的表示差异。

逻辑层的命名也取自于实际的业务逻辑，表明这些软件类替换了某些业务职责

构建领域模型方法：识别概念类

- 通过对用例描述中的名词或名词短语寻找和识别概念类；
- 需要注意的是名词可以是概念类，也可能是概念类中表示特征的属性；

  属性一般是可以赋值的，比如数字或者文本。

  如果该名词不能被赋值，那么就“有可能”是一个概念类。

  如果对一个名词是概念类还是属性举棋不定的时候，最好将其作为概念类处理。

  需要注意的是：不存在名词到类的映射机制，因为自然语言具有二义性

  这种方法的弱点是自然语言的不精确性，建议初学者使用

创建领域模型的步骤

- 理解领域模型对理解系统需求至关重要，领域模型的创建步骤如下：
  - 第1步，找出当前需求中的候选概念类；
  - 第2步，在领域模型中描述这些概念类。用问题域中的词汇对概念类进行命名，将与当前需求无关的概念类排除在外。
  - 第3步，在概念类之间添加必要的关联来记录那些需要保存记忆的关系，概念之间的关系用关联、继承、组合/聚合来表示。
  - 第4步，构建UML类图
  - 第5步，在概念类中添加用来实现需求的必要属性。

识别名词短语

- 医院的挂号流程：（取自调研报告）
  - 医院一般由挂号处、门诊科室、收费处和取药处四个部门构成；
  - 挂号处的挂号人员接受病人的就诊请求，并根据门诊科室各医生病人的排队情况，分配合适的医生，记录并打印挂号凭据，收取挂号费完成挂号请求。
  - 挂号过程中，如果病人没有病历，则可以创建病历。
  - 医生的分配规则有三种：排队医生人数最少，挂号费用最少以及指定医生三种方式进行。
- 抽取名词并区分概念类和属性
  - 概念类：医院、挂号处、门诊科室、收费处、取药处、挂号员、病人、医生、挂号凭据、病历、
  - 属性：挂号费、队列人数，可被赋值的名词
  - 不确定：排队情况 →队列、分配规则，这些名词先考虑为概念类

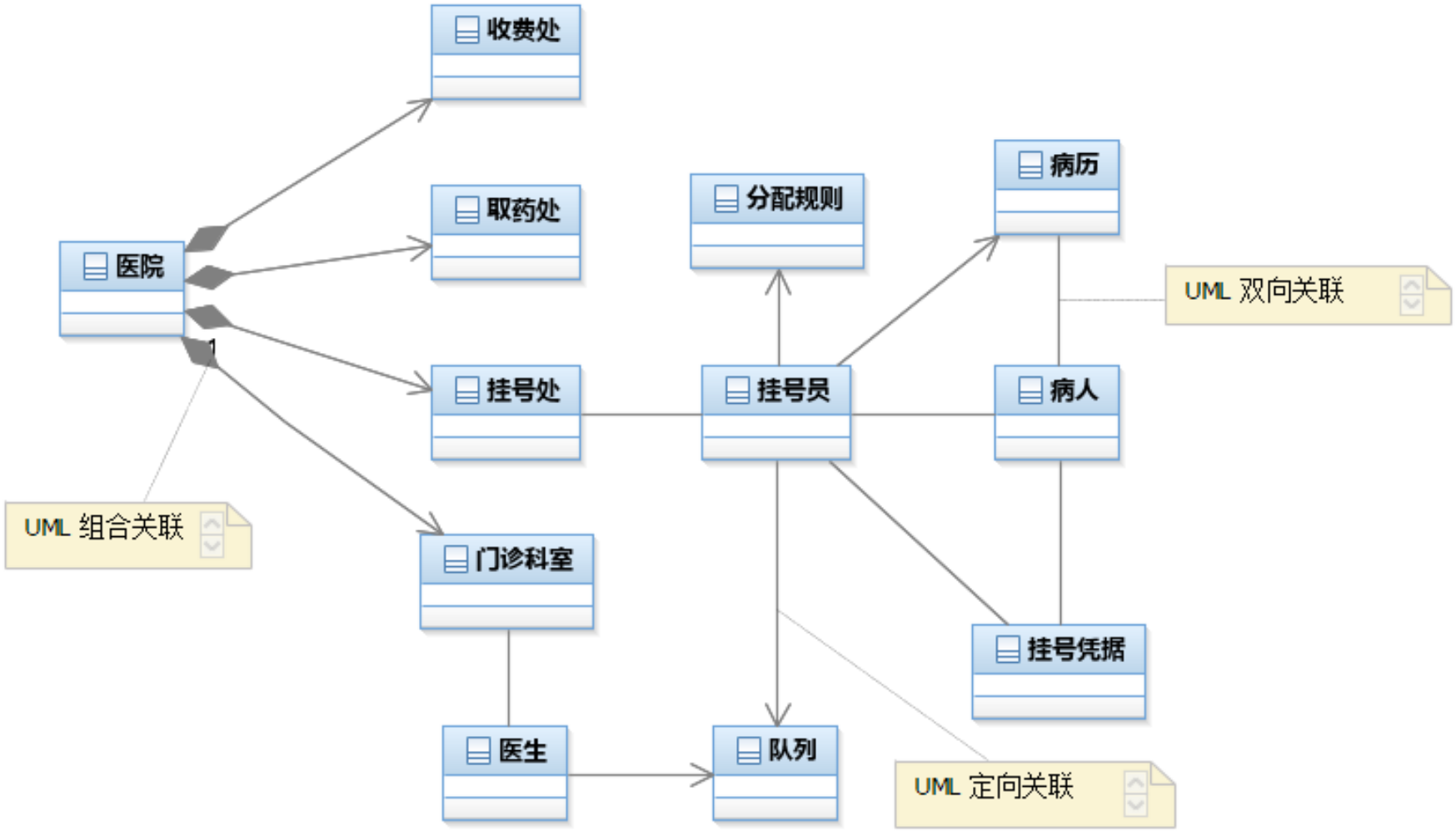
添加关联

- 领域模型中的关联可分为两种：
  - “需要知道”型关联：需要将概念之间的关系信息保持一段时间的关联。领域模型中需要着重考虑。
  - “只需理解”型关联：有助于增强对领域中关键概念的理解的关联。
- 寻找关联时要遵循下述指导原则：
  - 将注意力集中在需要知道型关联。
  - 识别概念类比识别关联更重要，因此领域模型创建过程中应该更加注重概念类的识别。
  - 太多的关联不仅不能有效地表示领域模型，反而容易使领域模型变得混乱。
  - 避免显示冗余或导出关联。

建立概念类之间的关系

- 医院由四个部门构成: 为了知道医院的组织结构;
- 挂号人员工作于挂号处: 为了知道挂号员的工作岗位;
- 医生工作于门诊科室: 为了知道医生的工作岗位;
- 病人请求挂号人员进行挂号: 为了知道挂号员的工作职责;
- 挂号人员查看医生的就诊队列: 为了知道医生的问诊状态;
- 挂号人员给医生分配病人: 为了知道医生有哪些病人;
- 挂号人员给病人创建病历: 为了知道病人的基本信息;
- 挂号人员给病人创建挂号凭据: 为了知道病人的挂号信息;

医院挂号处的领域模型

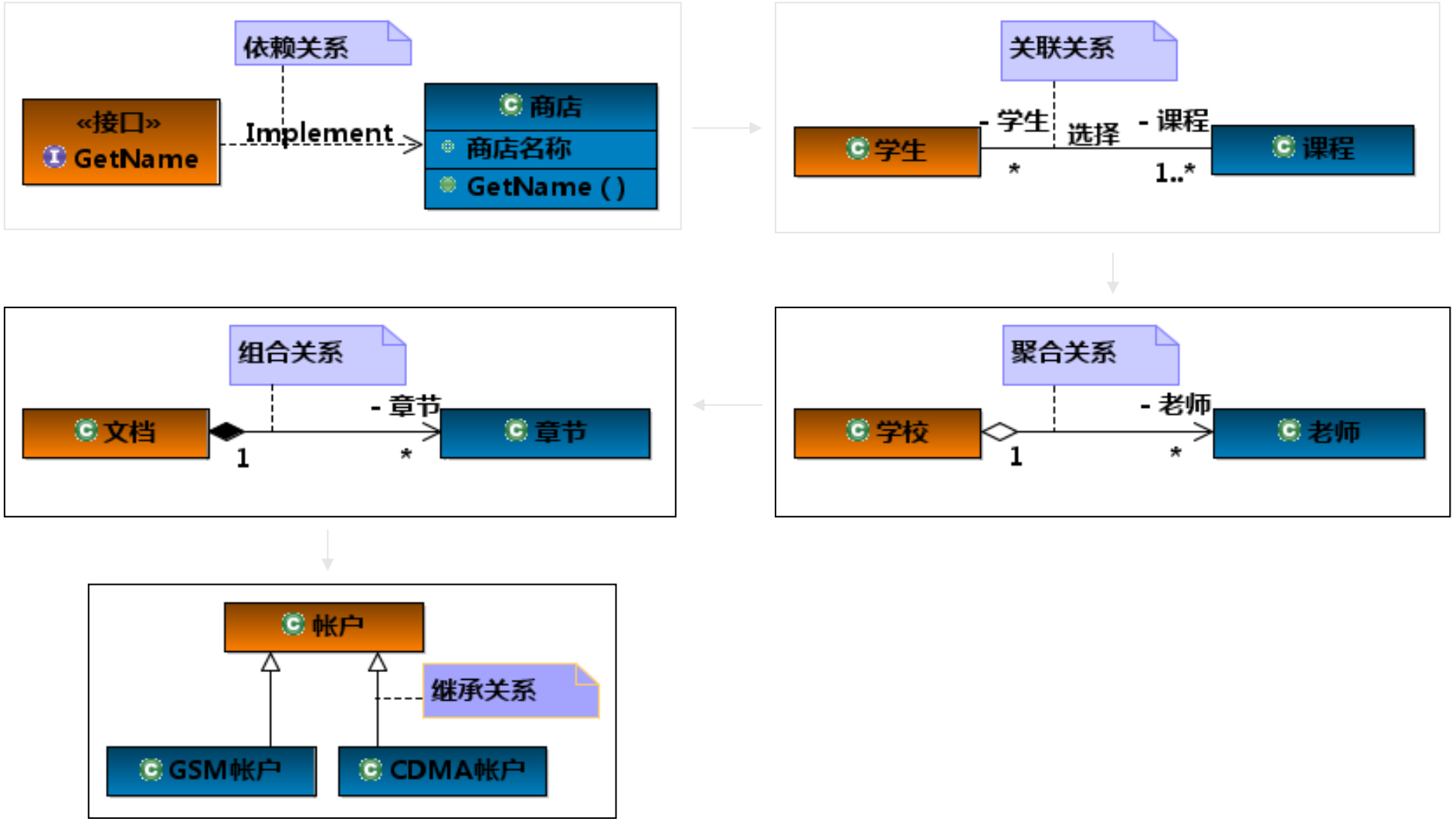


UML 类图的组成

- UML类图用于描述类以及类之间的关系。
- 类包含三个部分：
  - 类名：表示问题域中的概念，含义清晰准确
  - 属性：**可见性 属性名：类型名= 初始值 {性质串}**
  - 操作：**可见性 操作名（参数表）：返回值类型 {性质串}**
- 类的关系有：
  - 关联：普通关联、导航关联、递归关联
  - 组合与聚合
  - 依赖和继承

类的关系

- 以下按照由松散到紧密地的关系进行说明

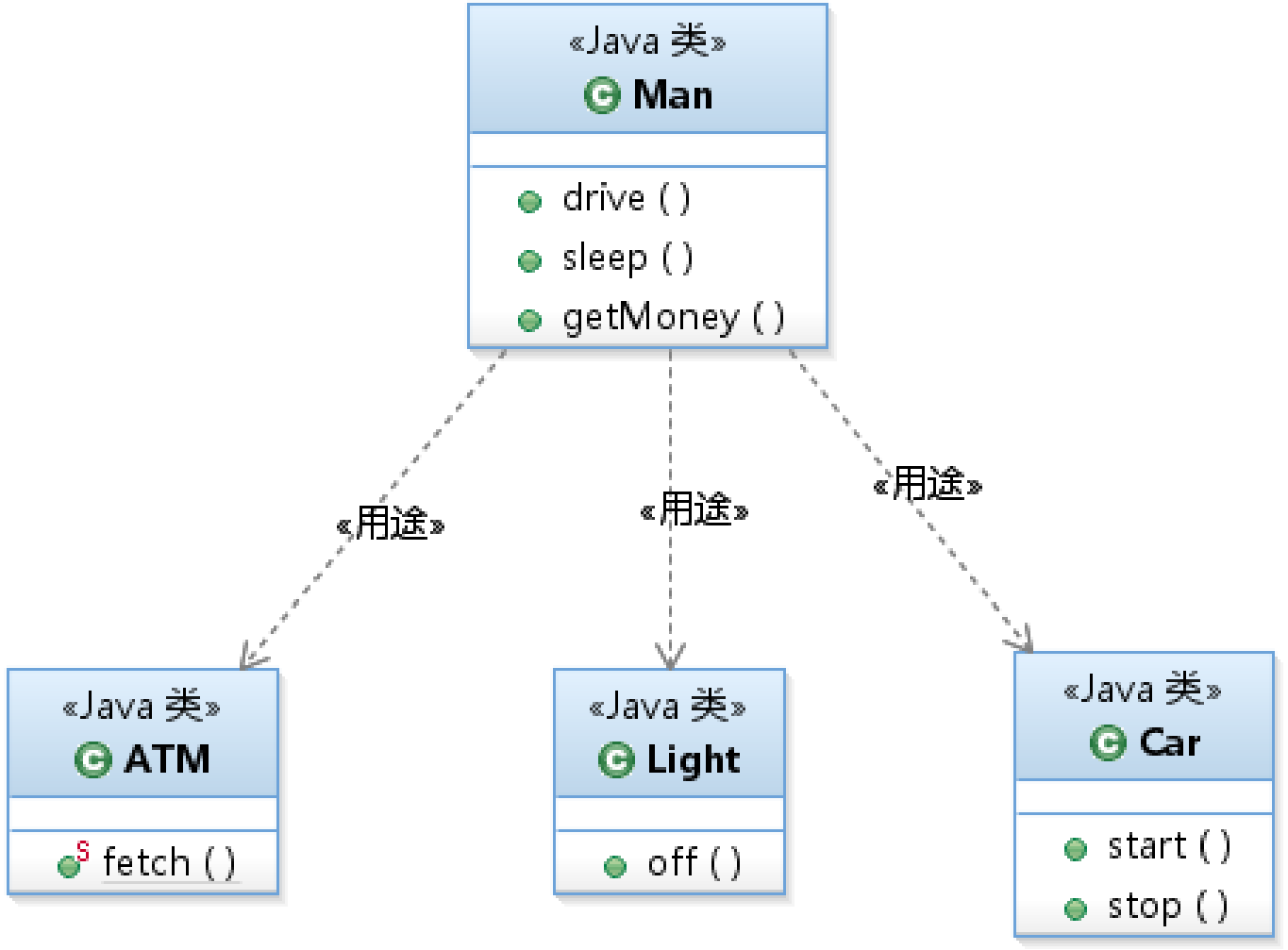


类的依赖关系-Java代码

```
public class Man
{
	public void drive(Car car)
	{
		car.start();
	}

	public void sleep()
	{
		Light light = new Light();
		light.off();
	}

	public int getMoney(int amountOfNeed)
	{
		return ATM.fetch(amount OfNeed);
	}
}
```

```
public class Car
{
	public void start()  { }
	public void stop()  { }
}

public class Light
{
	public void off() {}
}

public class ATM
{
	public static int fetch(int amountOfMoney)
	{
	  return amountOfMoney;
	}
}
```

类的依赖关系-UML类图

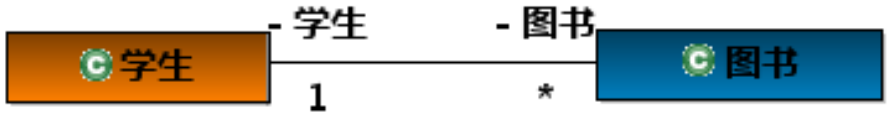


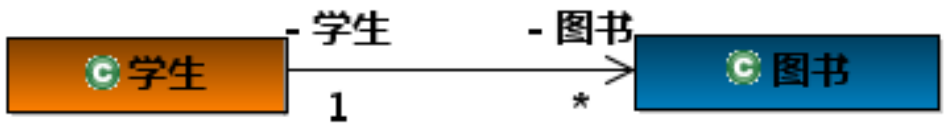
类的依赖关系

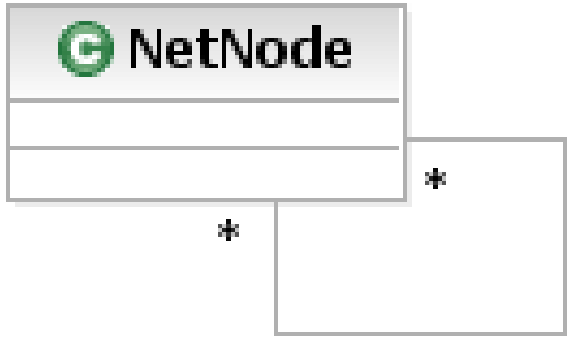
- 类A依赖于类B的三种情况
  - 类A 把 类B 的实例作为方法里的参数使用；
  - 类A 的某个方法里使用了类B 的实例作为局部变量；
  - 类A 调用了 类B的静态方法

```
public class Man {
    public void drive(Car car)
    //类A(Man) 把 类B(Car) 的实例作为方法里的参数使用
    {
        car.start();
    }

    public void sleep()
    //类A(Man) 的某个方法(sleep方法) 里使用了类B 的实例作为局部变量
    {
        Light light = new Light();
        light.off();
    }
    public int getMoney(int amountOfNeed)
    //类A(Man) 调用了 类B(ATM)的静态方法 static int fetch(int amountOfMoney)
    {
        return ATM.fetch(amountOfNeed);
    }
}
```

类的关联关系





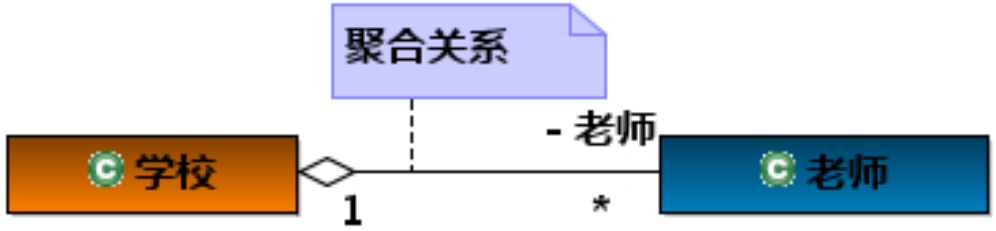


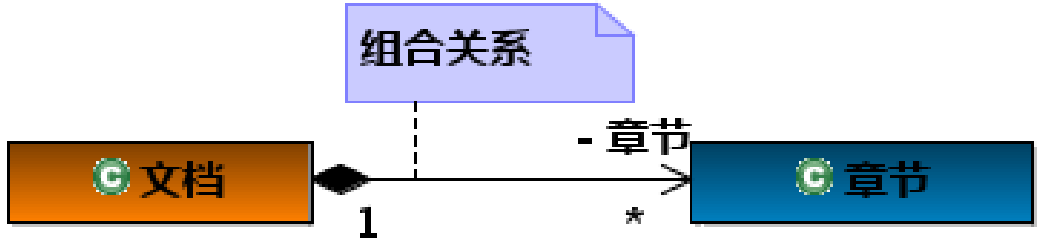
关联关系与Java


```
Public class student {
  private Book [ ] book;
  … // other attributes &
methods
}

Public class Book {
  private Student student;
  … // other attributes &
methods
}
```

```
Public class student {
  private Book [ ] book;
  … // other attributes &
methods
}

Public class Book {
  //  private Student student;
  … // other attributes &
methods
}
```

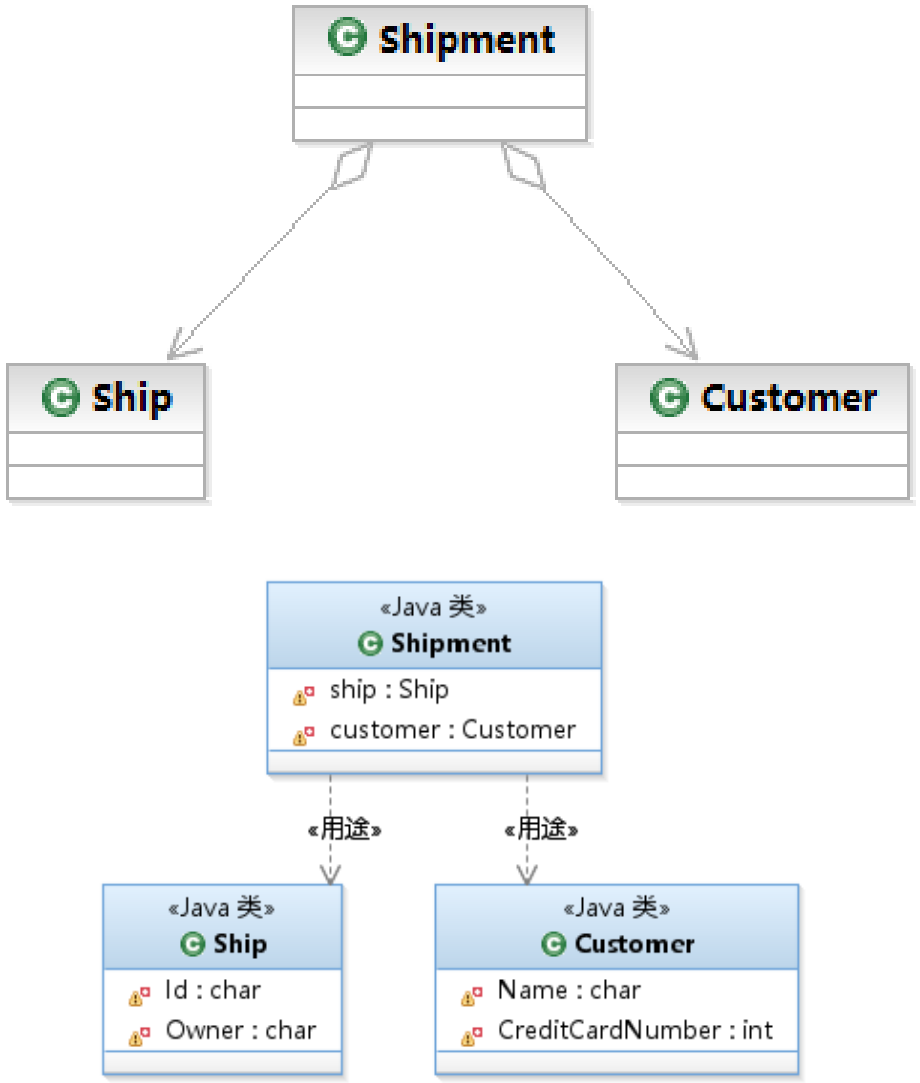
类的聚合与组合





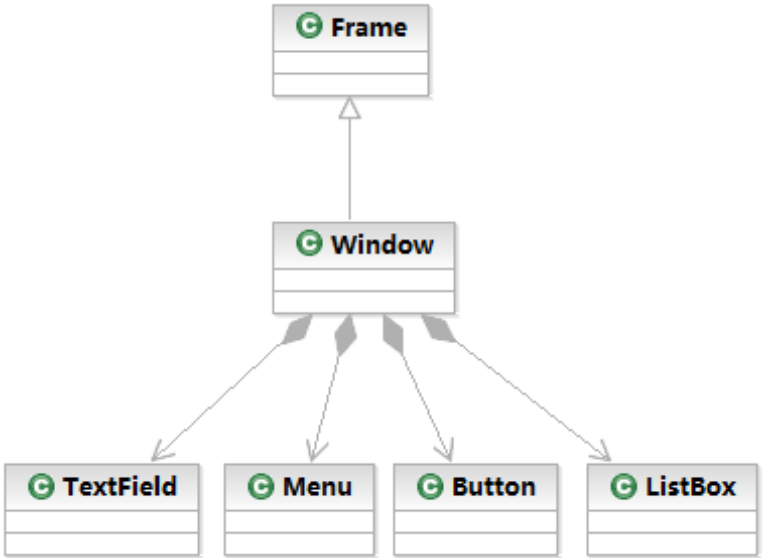
聚合关系与Java

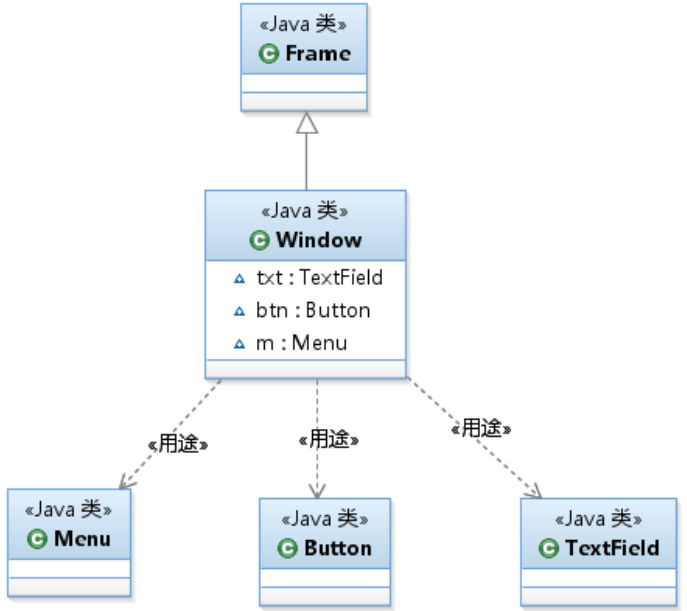
```
public class Shipment
{
        private Ship ship;
        private Customer[] customer;
}

public class Ship
{
        private String id;
        private String owner;
}

public class Customer
{
        private String Name;
        private String creditCardNumber;
}
```



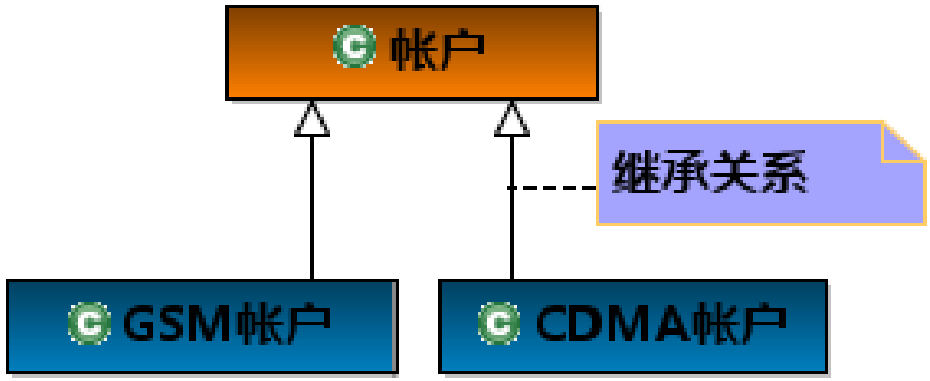
组合关系与Java

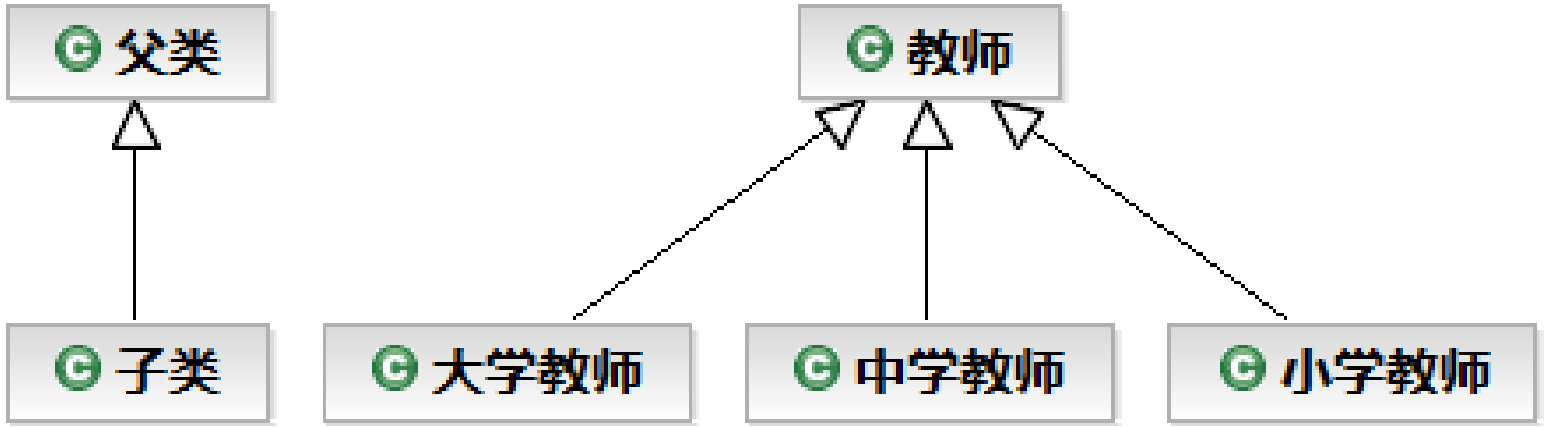
```
public class Window extends Frame
{
    TextField txt = new TextField();
    Button btn = new Button();
    MenuBar mbr = new MenuBar();
    MenuItem item = new MenuItem();
    Menu m = new Menu();
.....
}
```





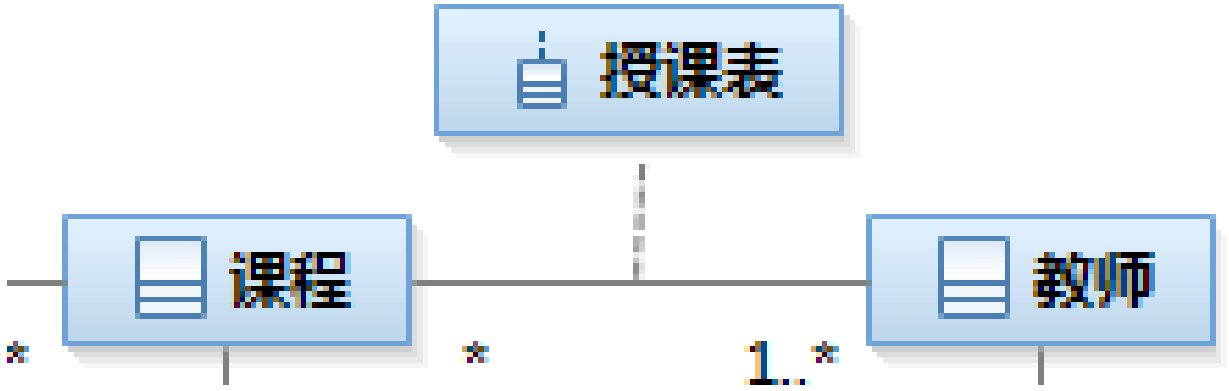
类的继承





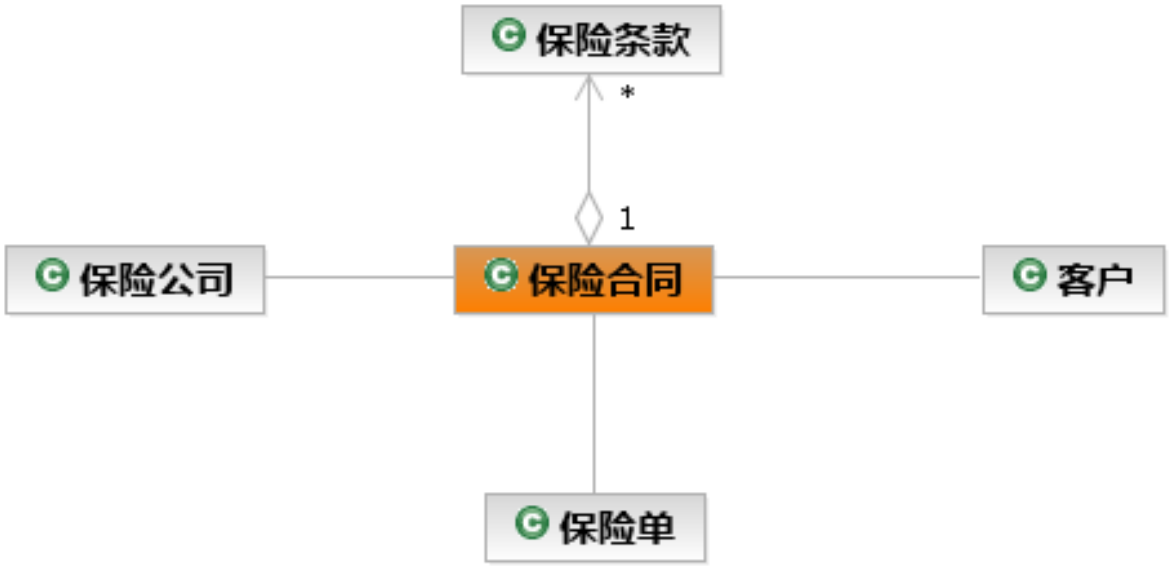
关联类

- 在关联建模中，存在一些情况下，需要包括其它类，因为它包含了关于关联的有价值的信息。
- 对于这种情况，使用关联类来绑定这些基本关联。关联类和一般类一样表示。不同的是，主类和关联类之间用一条相交的点线连接。



保险业务 案例

- 保险公司使用保险合同来代表保险业务，这些合同与客户有关。客户可没有或者具有多个保险合同，这些合同至少与一个保险公司有关。
  - 保险合同位于一家保险公司和一个或多个客户之间，它建立保险公司和客户之间的保险关系
  - 保险合同使用保险单表示，也就是合同的书面表示,一个保险单表示一份保险合同。



例子：油画

- 一幅油画由许多图形组成，图形可以由直线、圆、多边形和各种线型混合而成的组合图等。