



面向对象的软件构造实践



HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

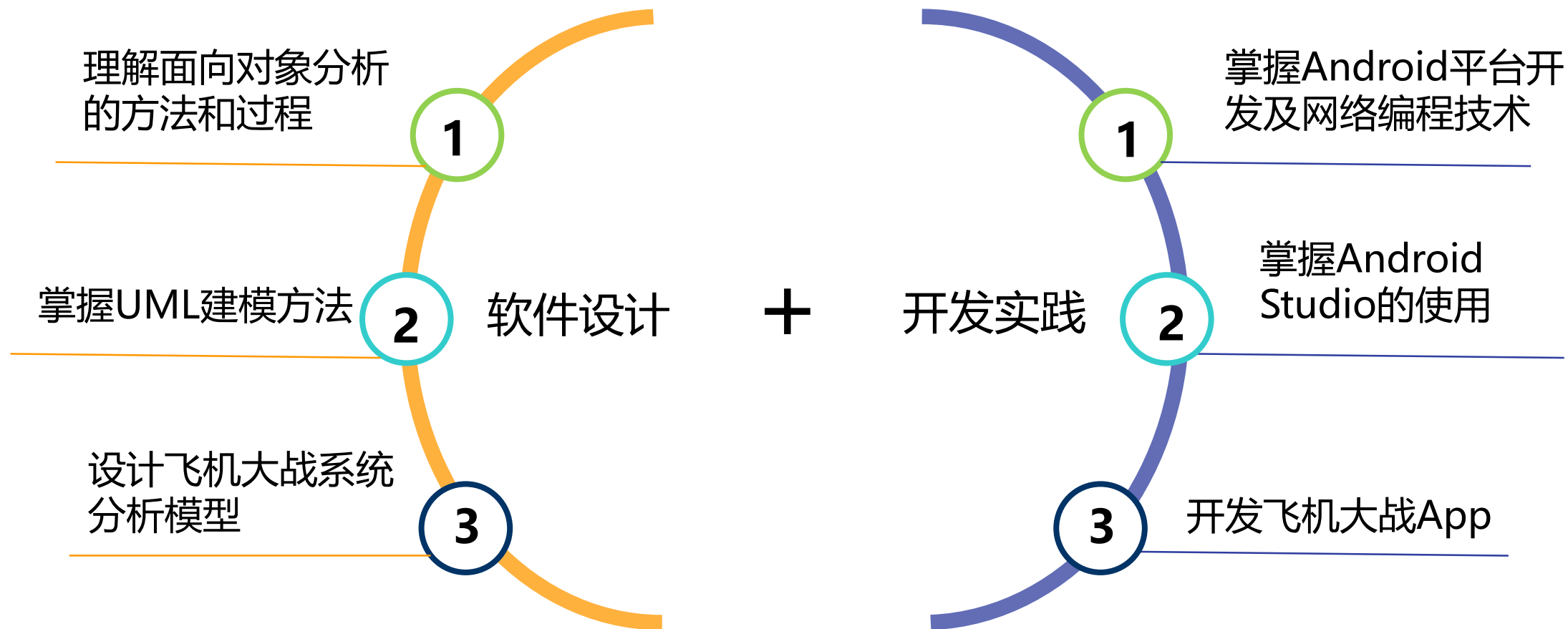
面向对象 的软件构造实践



一种思维方式

一门使用面向对象的思维方式，去进行软件构造的实践课程。





面向对象的软件构造导论：



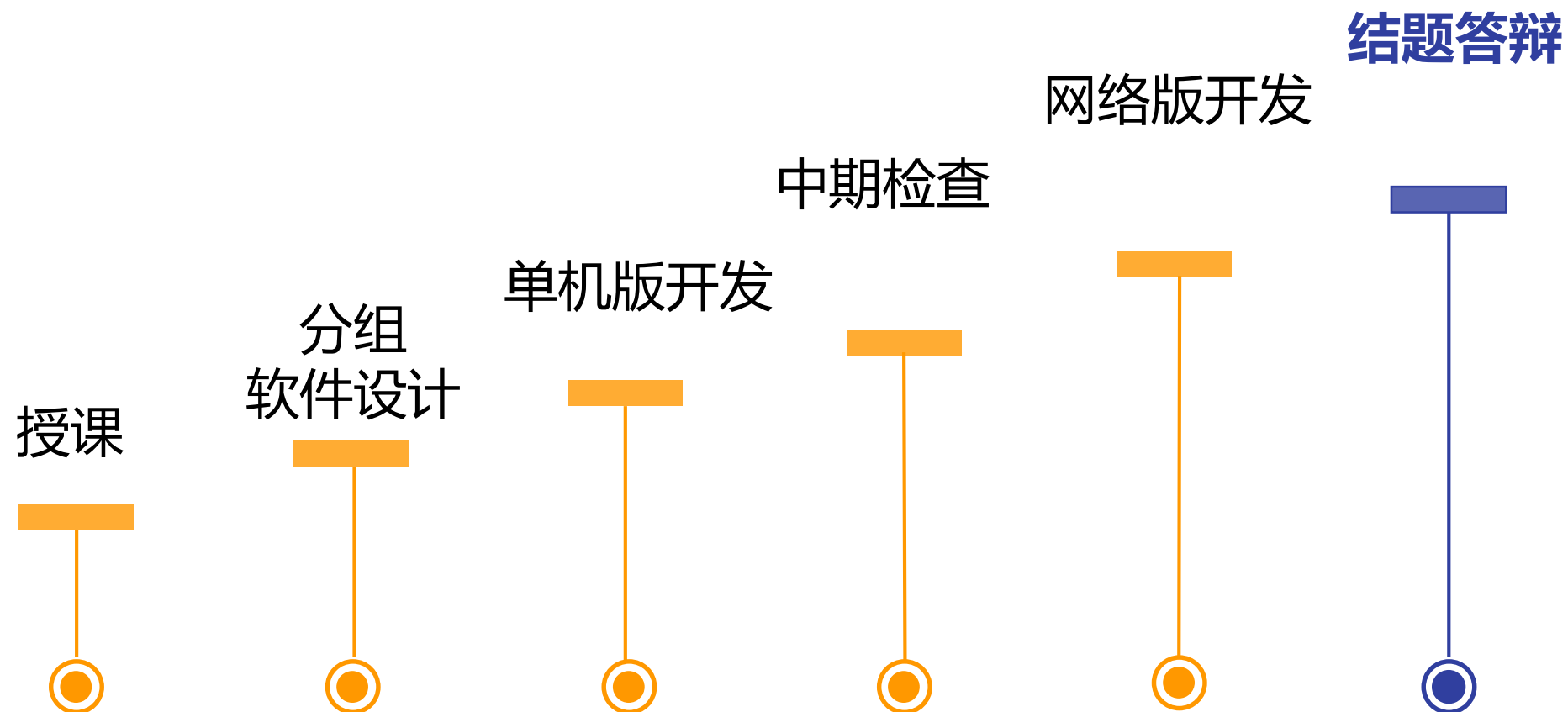
- Java语言
- 面向对象思想
- 设计模式
- 多线程、泛型
- Swing、网络编程
- PC客户端飞机大战



面向对象的软件构造实践：



- Android App飞机大战
- Android系统平台架构
- Android开发技术基础
- Android应用开发实践
- 调试、打包、发布等



考察点	分数	评分标准
开题报告	10分	<ul style="list-style-type: none">• 设计方案是否清晰合理• 任务分工、进度规划是否合理
中期检查	20分	<ul style="list-style-type: none">• 代码是否规范• 实现和设计是否相符• 功能是否完善• 进度是否合理• 任务分工、进度规划是否合理，工作量是否饱和
验收检查	50分	<ul style="list-style-type: none">• 是否完成了全部的系统功能• 编码规范性• 软件产品的用户友好性• 系统性能
结题报告	20分	<ul style="list-style-type: none">• 面向对象分析过程是否详尽完整• 核心算法描述是否清晰• 系统核心功能是否实现• 调试分析是否到位



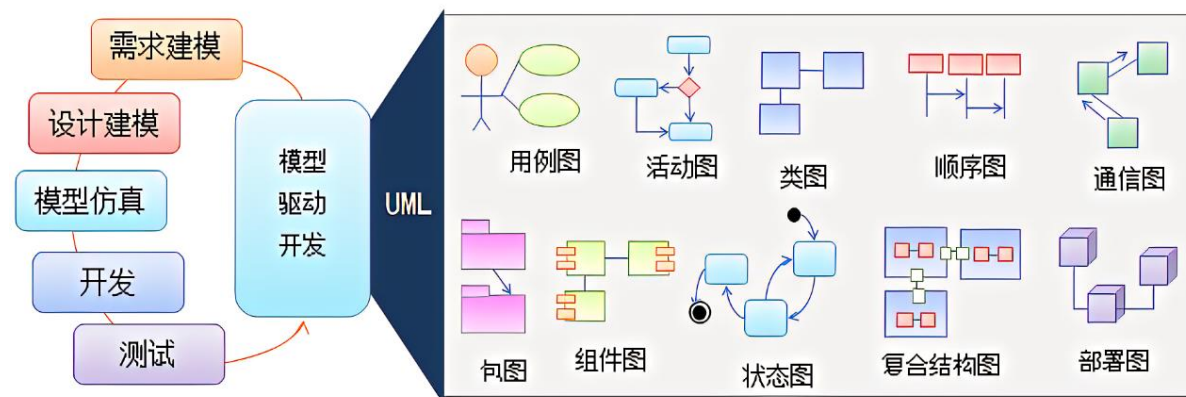


1

面向对象分析模型



HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ



- 模型就是对现实的简化。
- 建立模型的过程，称为建模。
- 软件模型提供了系统的蓝图。
- 软件系统的模型用**建模语言**来表达，包括语义信息和表示法。



捕获和**精确表达**项目的需求



分解复杂系统



完成**系统设计**



分离需求与具体实现细节

统一标准

统一了各种方法对不同类型的系统、不同开发阶段以及不同内部概念的不同观点，从而消除了各种建模语言之间不必要的差异。

独立于开发语言

UML作为一种建模语言，使开发人员专注于建立产品的模型和结构，而不是选用什么程序语言和算法实现。

可视化表示

提供了语义和可视化的表示法，语义保证了模型和模型元素应用的一致性，可视化的表示法有利于建模技巧的使用。

支持面向对象

UML支持面向对象技术的主要概念，提供了一批基本的模型元素的表示图形和方法，能简洁明了地表达面向对象的各種概念。

需求分析

设计阶段

开发阶段

测试阶段

通过建立**用例图**等模型来描述系统的使用者对系统的功能要求。



需求分析

设计阶段

开发阶段

测试阶段

通过类和对象等主要概念及其关系建立**静态模型**，
对类、用例等概念之间的协作设计**动态模型**，为
开发工作提供详尽的规格说明。



需求分析

设计阶段

开发阶段

测试阶段

将设计的模型转化为编程语言的
实际代码，
指导编码工作。



需求分析

设计阶段

开发阶段

测试阶段

用UML图作为测试依据，用类图指导单元测试，用用例图指导系统测试等。



- 事物：模型中关键元素的抽象体现**
- 关系：事物和事物间联系的方式**
- 图：相关的事物及其关系的聚合表现**



□事物

- 结构事物：作为UML模型的静态部分，用于描述概念元素或物理元素。
 - 例：类、接口、用例、组件、节点等。
- 行为事物：是UML模型的动态部分，用于描述UML模型中的动态元素。
 - 例：状态机、活动等。
- 分组事物：是UML模型的组织部分，用来组织系统设计的事物。
 - 例：包。
- 注释事物：是UML模型的解释部分，用来描述、说明和标注模型的元素。
 - 例：注解。

□关系

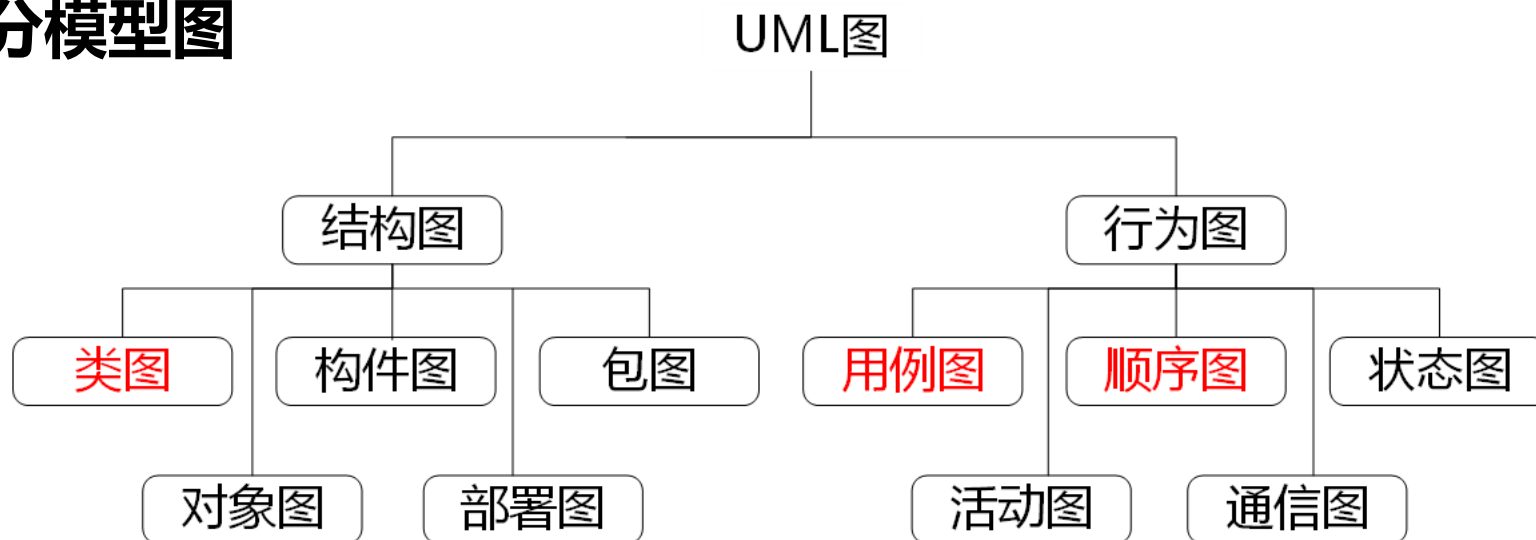
- 关系是模型元素之间具体化的语义连接，负责联系UML的各类事物，构造出结构良好的UML模型。
- 四种关系：
 - 关联关系：描述模型元素之间的**一般性**或者**长期性**的关系。
 - 依赖关系：描述模型元素之间的**偶然**或者**临时**的关系。
 - 泛化关系：描述模型元素之间存在的**特殊**和**一般**的归纳关系。
 - 实现关系：描述**规格说明**和**实现了**说明的元素之间的关系。

□图

UML图根据基本功能和作用，可分为：结构图与行为图。

- **结构图**：捕获事物与事物之间的**静态**关系，用来描述系统的**静态结构**模型。
- **行为图**：捕获事物的**交互**过程，用来描述系统的**动态行为**模型。

UML 2中的部分模型图



类图：类图描述系统中类的静态结构。

对象图：类图的实例化。

构件图：描述构件的组织关系和相互关系。

部署图：描述节点之间的关系以及构件和节点之间的部署关系。

包图：描述模型元素分组以及分组之间依赖的图。

用例图：从用户角度描述系统功能并指出各功能的操作者。

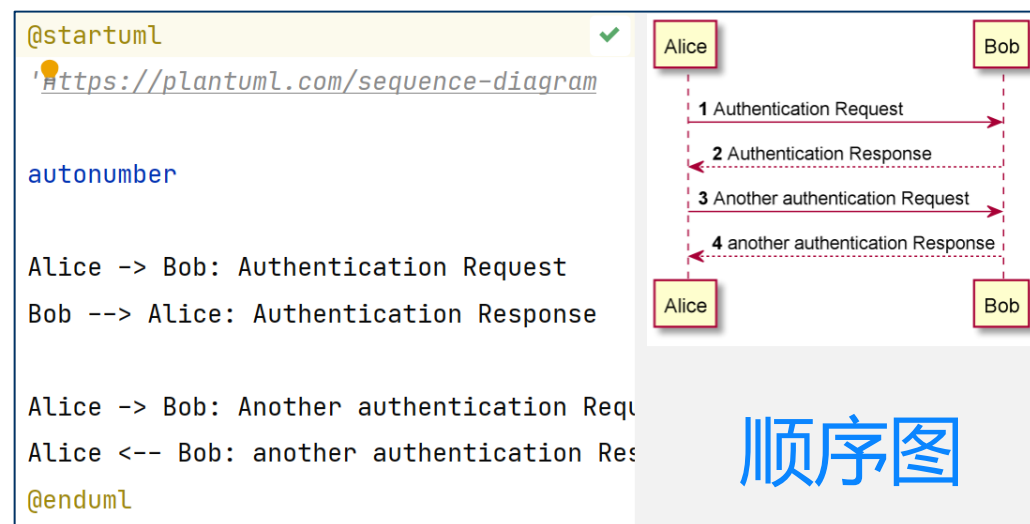
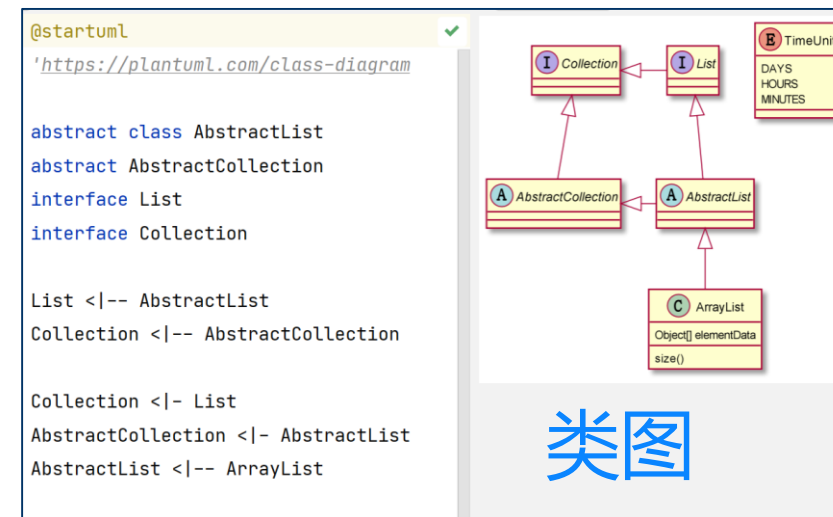
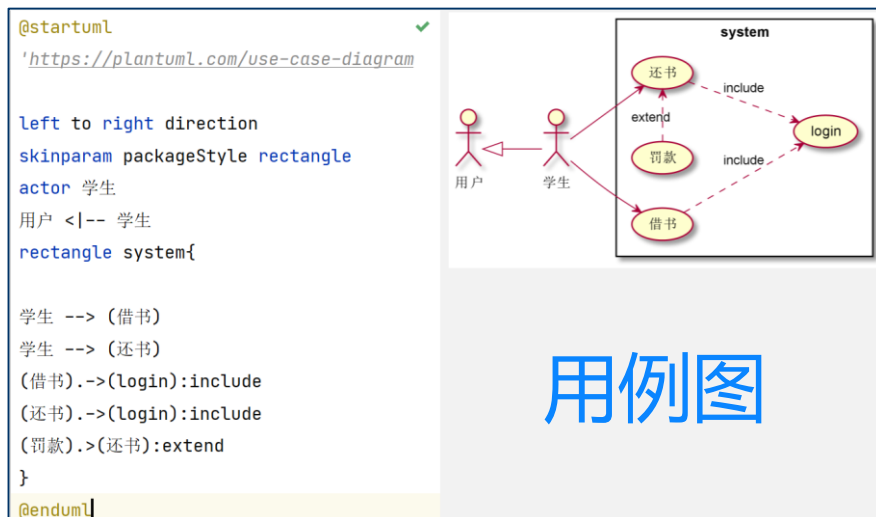
顺序图：描述对象之间发送和接收消息的图，强调时间顺序。

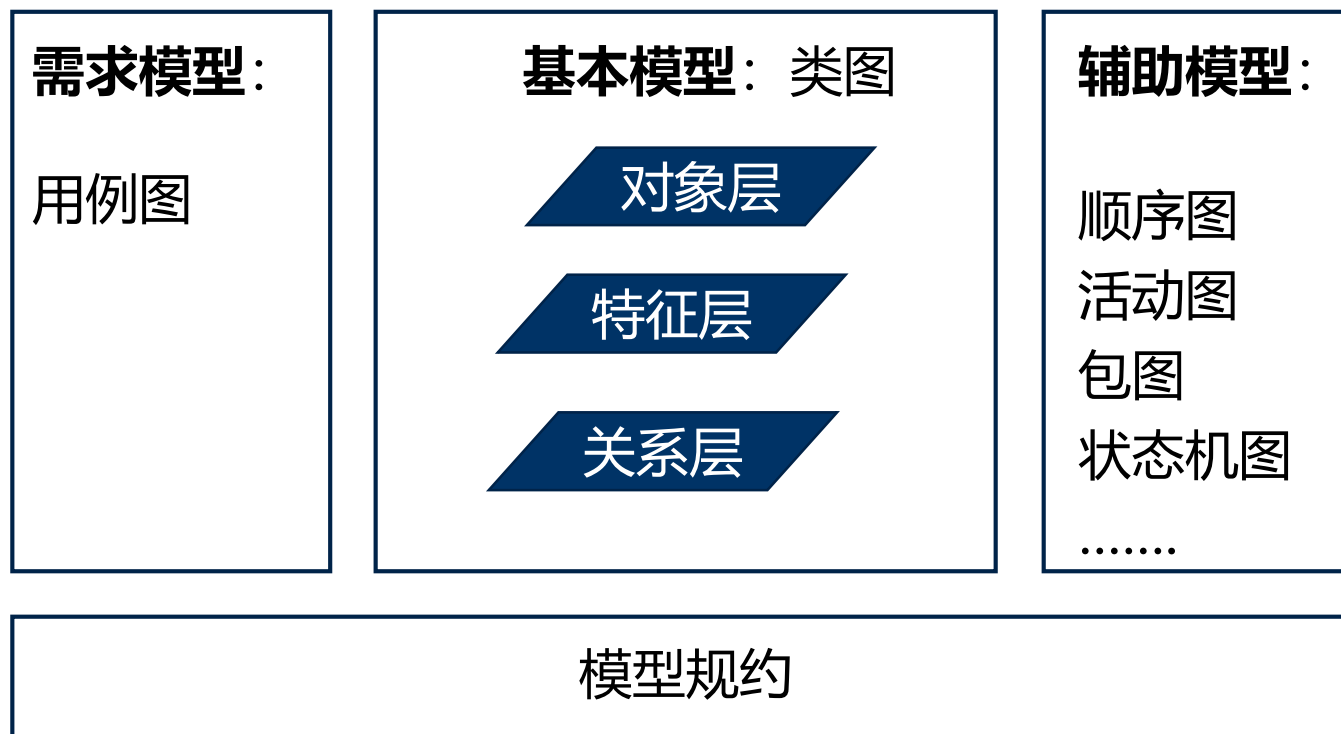
活动图：描述活动、活动的执行顺序以及活动输入输出。

状态图：描述对象在其生命周期内的状态及其变化。

通信图：描述对象交互关系，强调对象之间的关联。

PlantUML







2

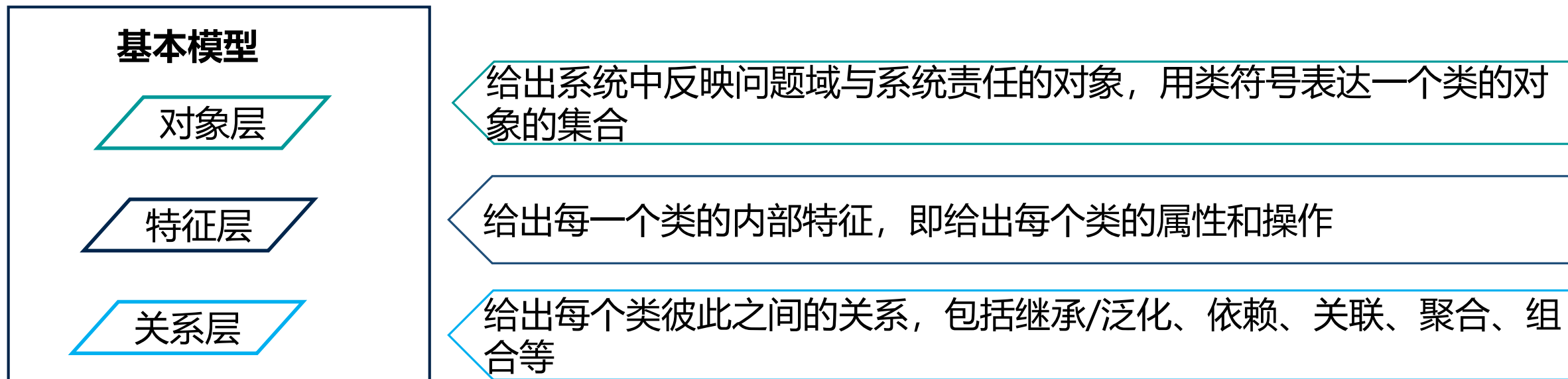
基本模型



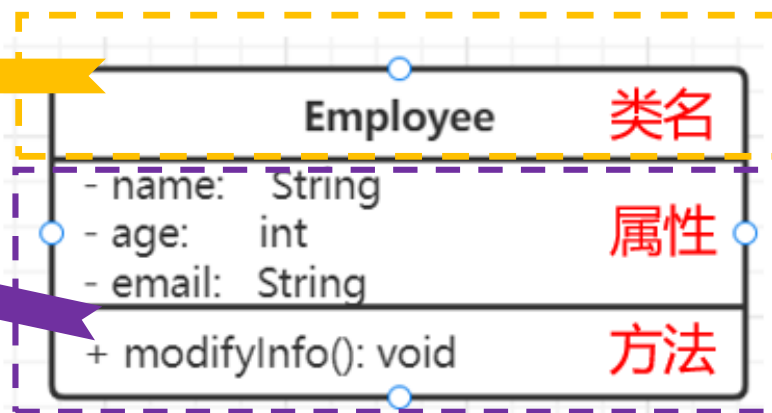
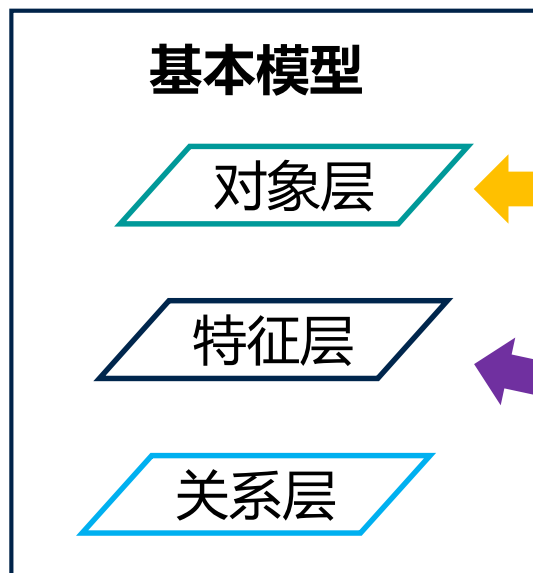
HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

□ 概念

基本模型是系统的静态模型，描述系统的结构特征。

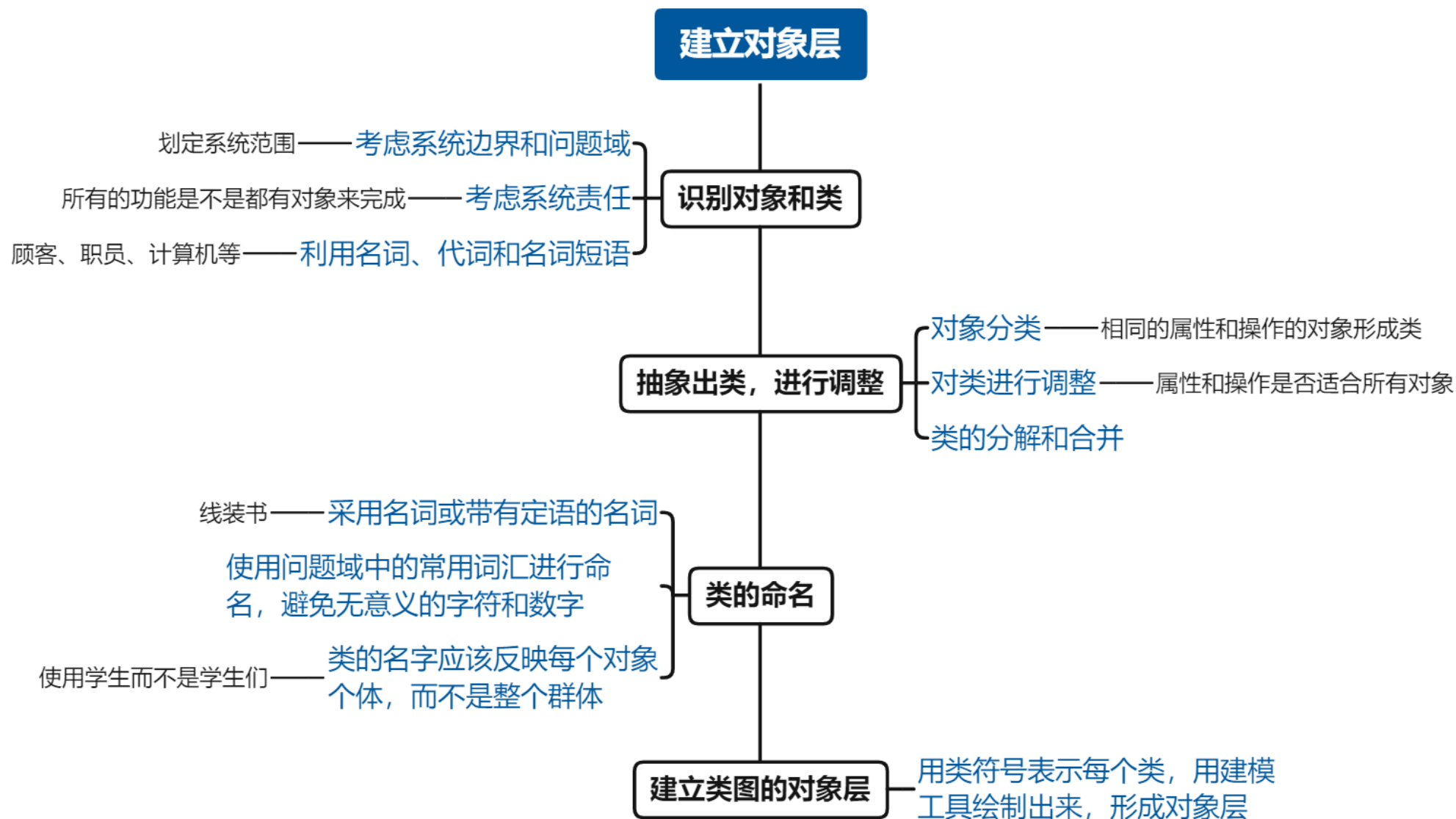


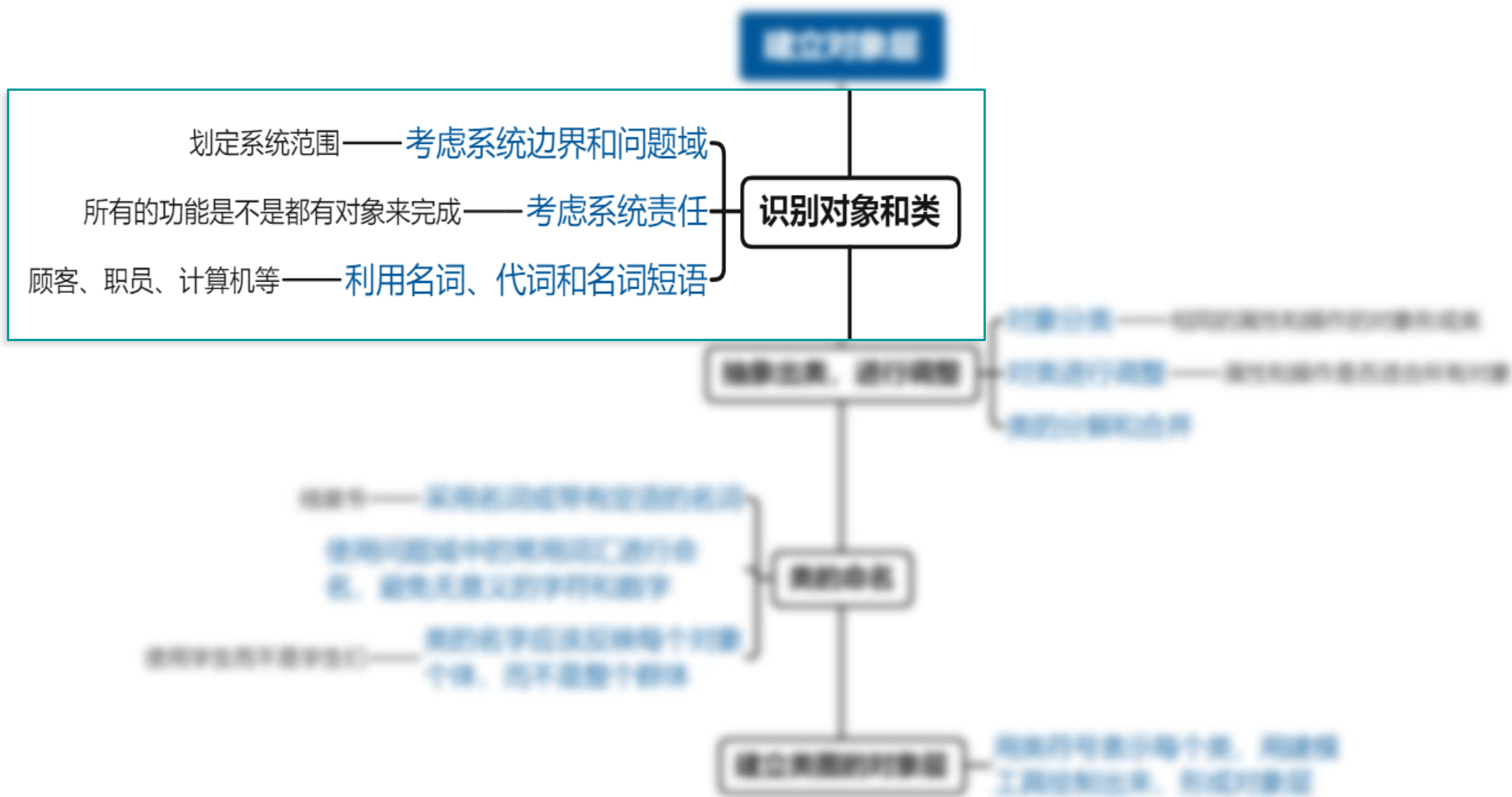
类图 (Class Diagram) : 用来显示系统中的类、接口以及它们之间的静态结构和关系的一种静态模型。

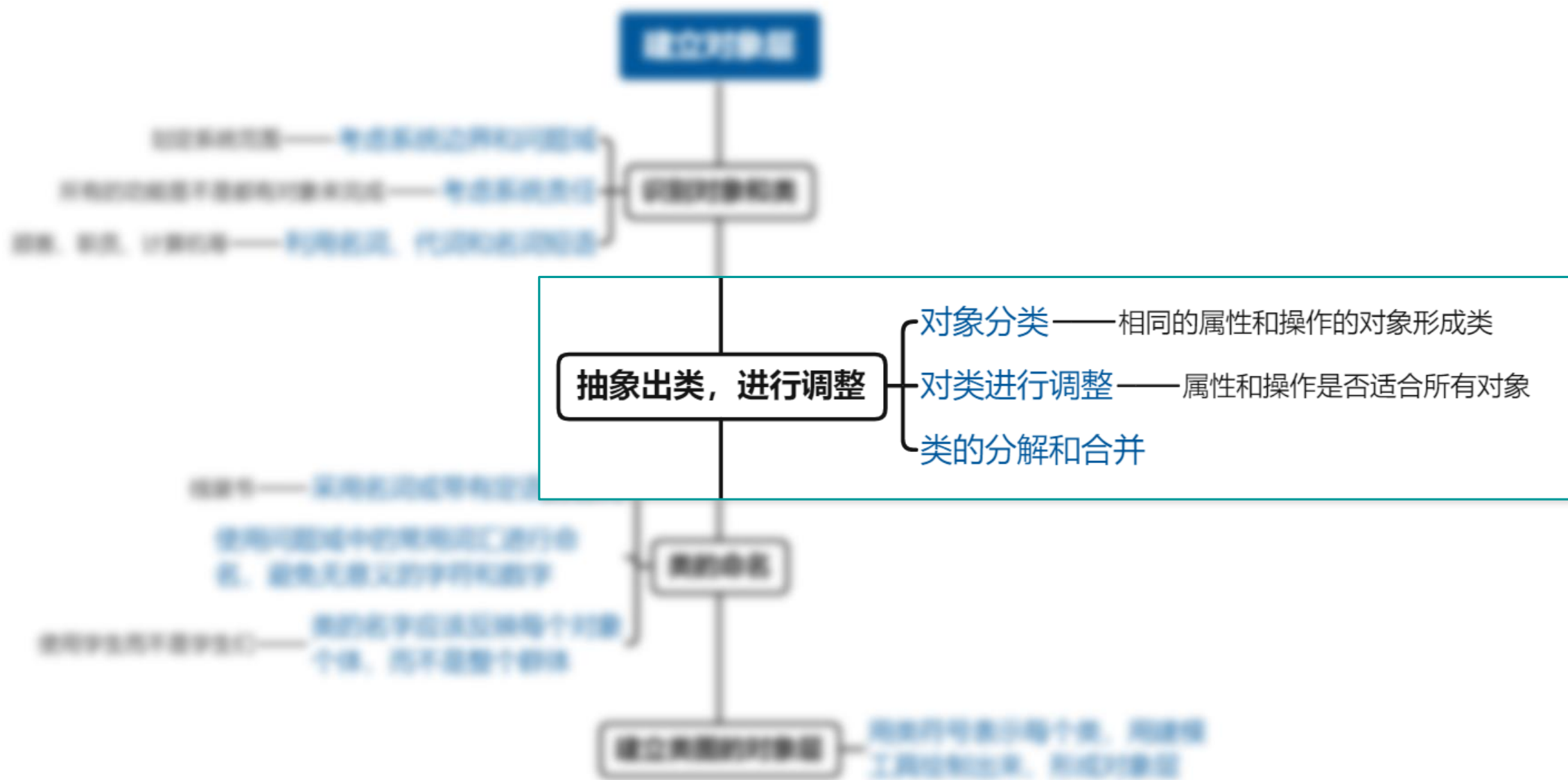


```
public class Employee {
    private String name;
    private int age;
    private String email;

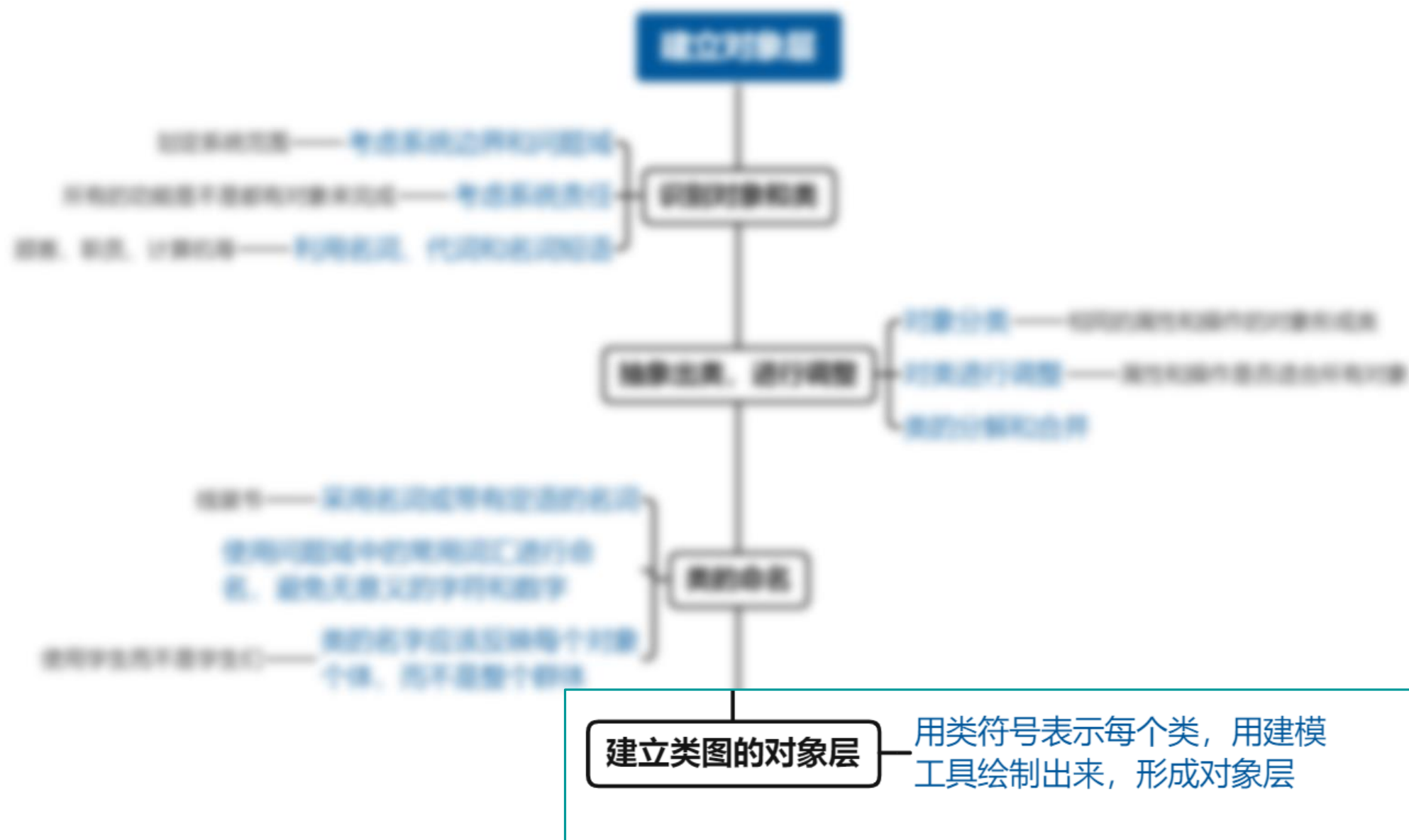
    public void modifyInfo() {
        .....
    }
}
```

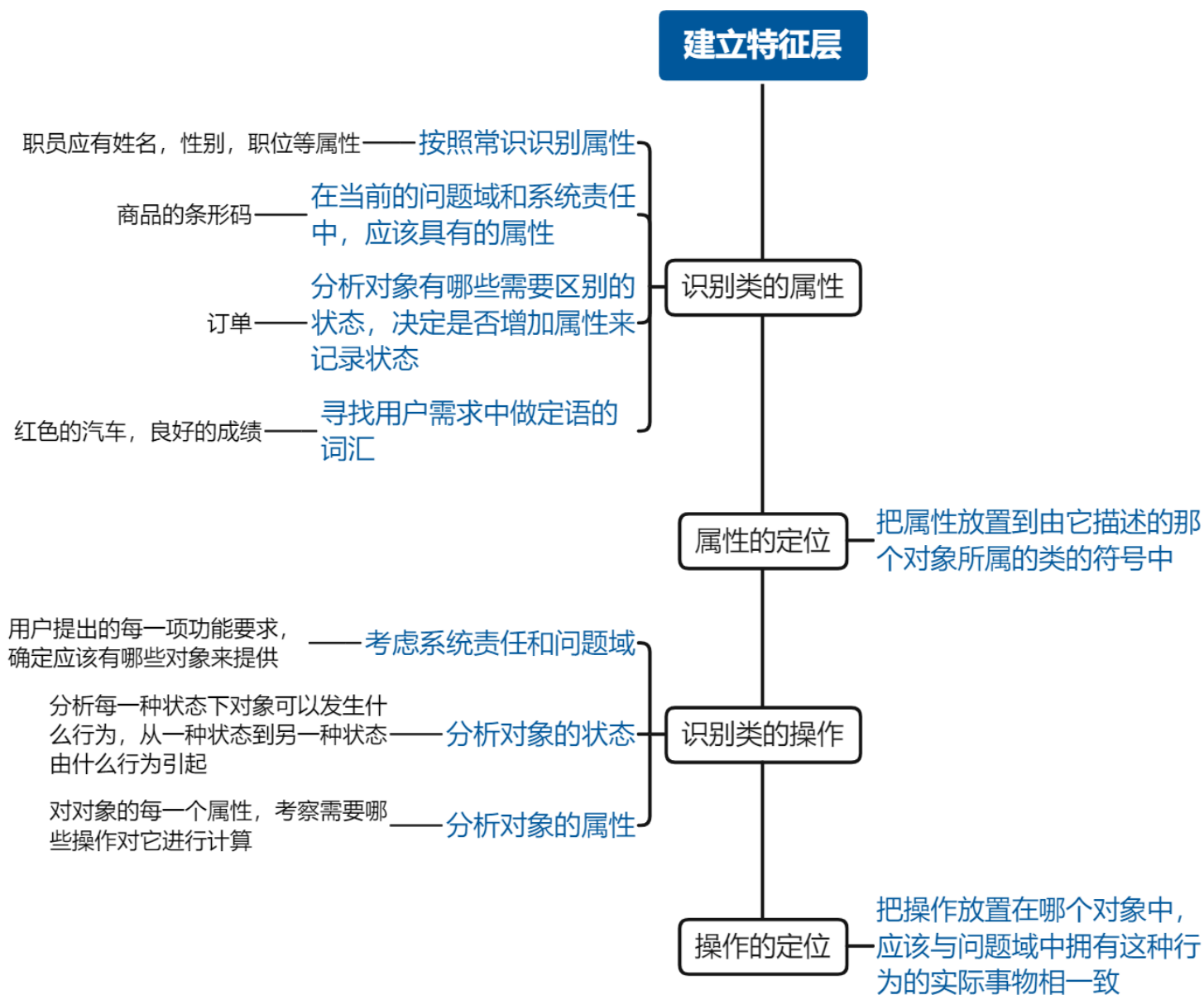



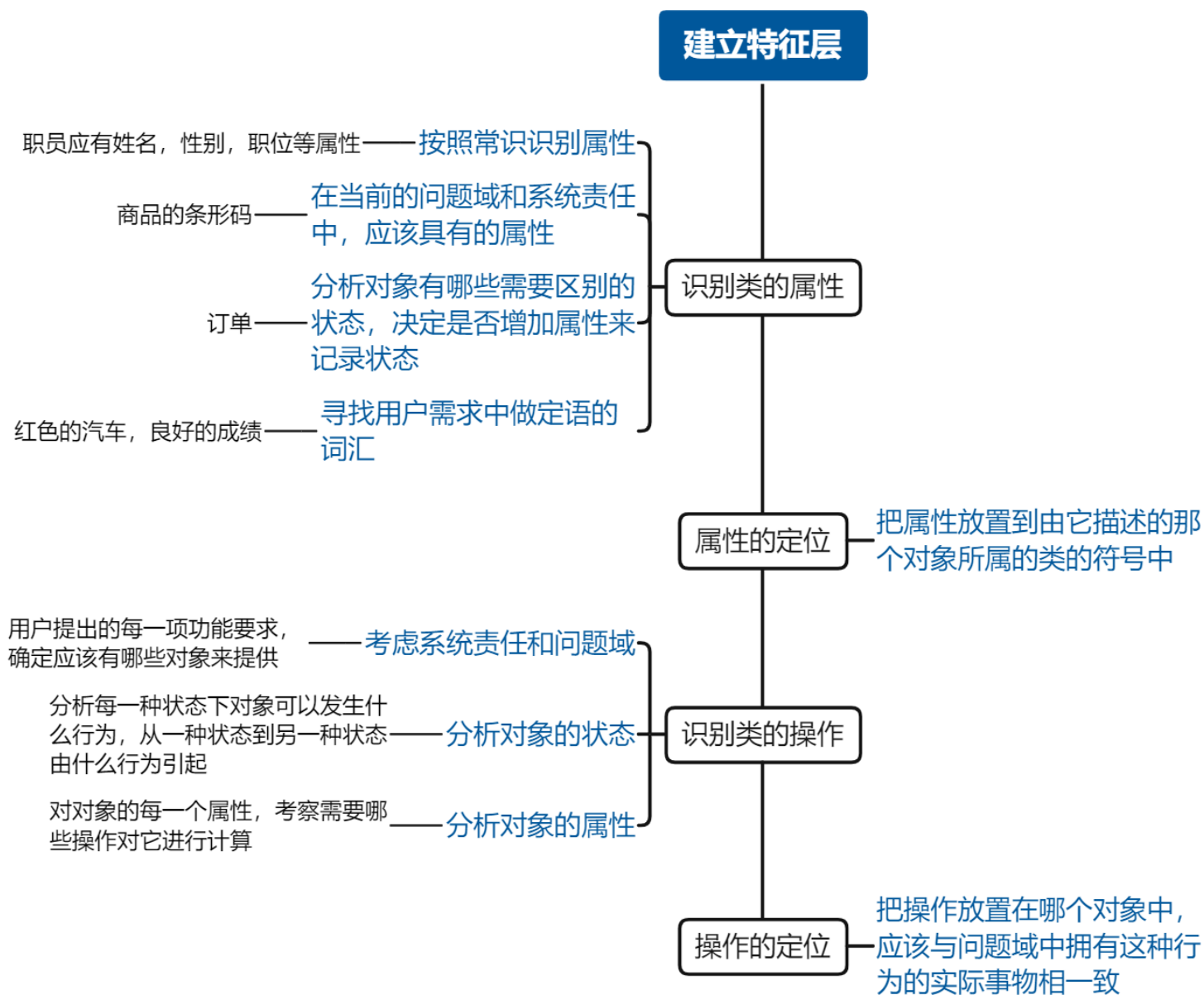


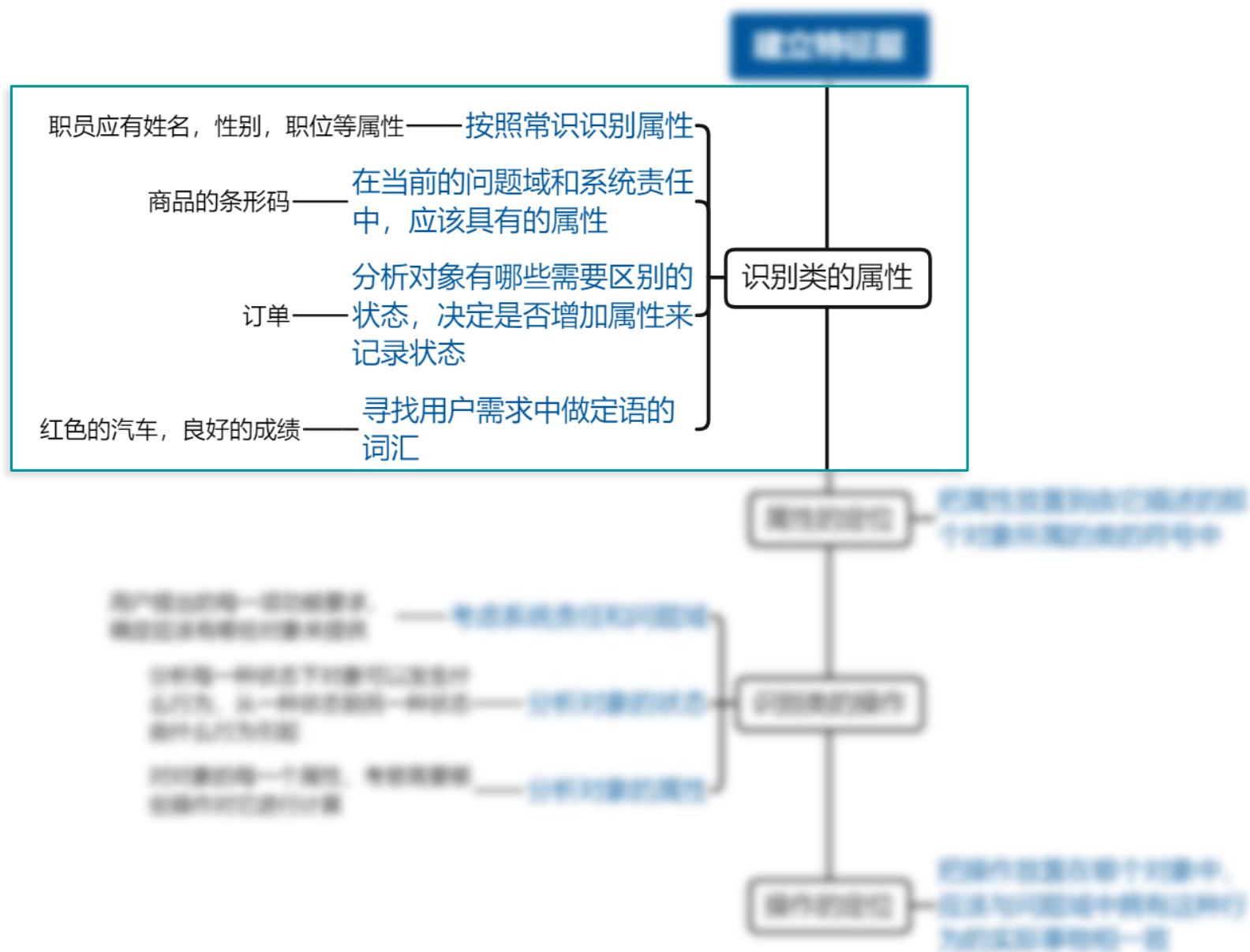




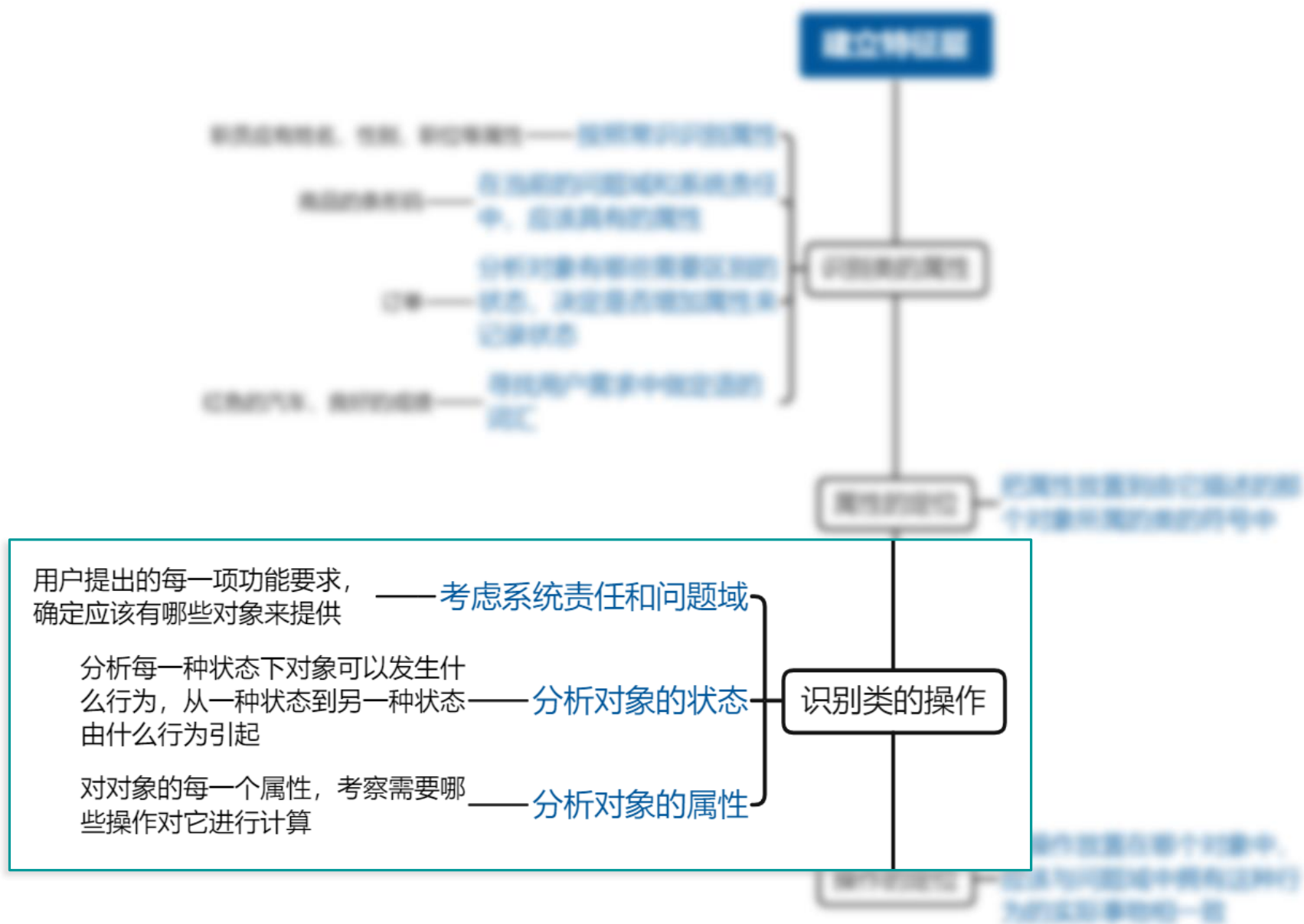










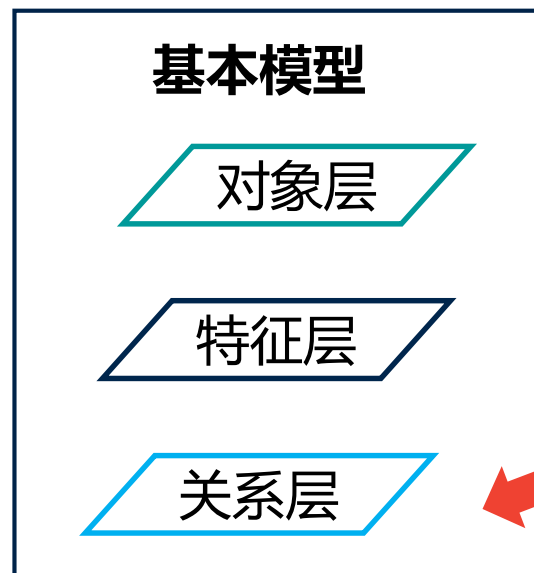




操作的定位

把操作放置在哪个对象中，应该与问题域中拥有这种行为的实际事物相一致

类与类之间的关系: 泛化、实现、依赖和关联。
其中关联又分为**一般关联**和**聚合关系**, **组合关系**。



01 ↑

泛化关系

继承关系, 子类继承父类的所有行为和属性。如: 老虎和动物

02 ↑

实现关系

类与接口的关系, 表示类是接口所有特征和行为的实现者。如: 鸟和飞行。

03 ↓

依赖关系

一种使用关系, 一个类的实现需要其他类的协助。如: 驾驶员和汽车。

04 ↓

一般关联

对象之间的一种引用关系, 用于表示一类对象与另一类对象之间的联系。如: 顾客和商品。

05 ↓

聚合关系

整体与部分的关系, 且部分可以离开整体而单独存在。如: 汽车和轮胎。

06 ↓

组合关系

整体与部分的关系, 但部分不能离开整体而单独存在。如: 公司和部门。








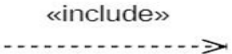
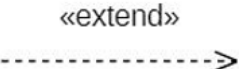
3

需求模型



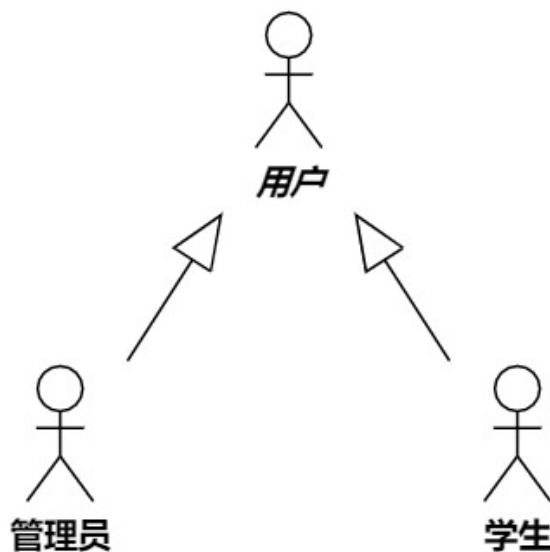
HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

- 用例图是描述了系统中相关的用户和系统对不同用户提供的**功能和服务**。
- 用例图相当于从**用户的视角**来描述和建模整个系统，分析系统的功能与行为。
- 用例图中的主要元素包括**参与者**、**用例**以及元素之间的**关系**。
- 当系统十分复杂时，可以使用一个用例图表示系统的一个方面（子系统），使用多个用例图共同来表示系统的用例视图。

基本要素		含义	UML表示
参与者	描述了一个或一组与系统产生交互的 外部用户 或 外部事物 。		 参与者
用例	描述系统的一项功能的一组动作序列，这样的序列表示参与者与系统间的交互，系统执行该序列要为参与者 产生结果 。		 用例
关系	参与者与用例之间的关联关系		
	参与者之间的泛化关系		
	用例之间的关系	泛化	
		包含	
		扩展	

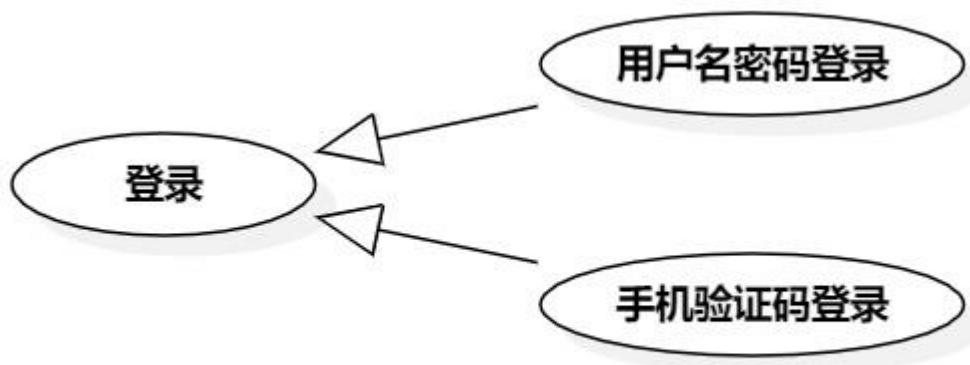
□ 参与者之间的泛化

- 当系统中的多个参与者有共同的行为特征，可以**抽象**出更一般的参与者，通过泛化来描述多个参与者之间的共同行为，不同的参与者以独特的方式使用系统。
- 与类相似，父参与者可以是**抽象的**（抽象元素表示为斜体），即不能创建一个父参与者的直接实例。



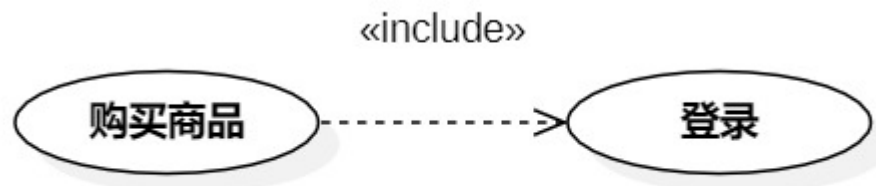
□ 用例之间的泛化

- 与参与者的泛化关系相似，用例的泛化关系将特殊的用例与一般化的用例联系起来。子用例继承了父用例的属性、操作和行为序列，并且可以增加属于自己的附加属性和操作。
- 父用例同样可以定义为抽象用例。



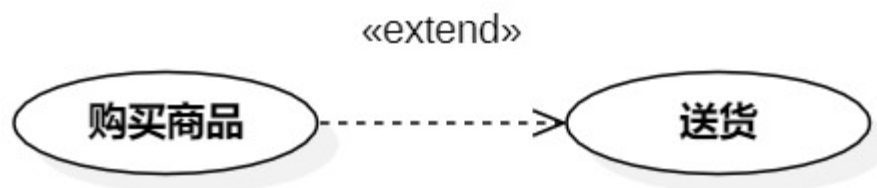
□ 用例之间的包含

- 包含指的是一个用例（基用例）可以包含其他用例（包含用例）具有的行为，其中包含用例中定义的行为将被插入基用例定义的行为中。
- 包含的两个基本约束：
 - 基用例可以看到包含用例，并需要依赖于包含用例的执行结果，但是它
对包含用例的内部结构没有了解；
 - 基用例一定会要求包含用例执行。



□ 用例之间的扩展

- 扩展指的是一个用例（扩展用例）对另一个用例（基用例）行为的增强。
 - 在这一关系中，扩展用例包含了一个或多个片段，每个片段都可以插入到基用例中的一个单独的位置上，而基用例对于扩展的存在是毫不知情的。
- 使用扩展用例我们就可以在不改变基用例的同时，根据需要自由地向用例中添加行为。



识别参与者

识别用例

绘制用例图

细化用例模型

用例描述

系统的功能的使用者

从系统获得信息者

向系统提供信息者

系统需要访问的外部硬件设备

与该系统进行交互的其他系统

特殊参与者：系统时钟



识别参与者

识别用例

绘制用例图

细化用例模型

用例描述

□ 用例的特征

- 用例是动宾短语
- 用例是相对独立的
- 用例是由参与者启动的
- 用例要有可观测的执行结果

参与者的主要任务是什么？

参与者需要系统的什么信息？

参与者可以为系统提供什么信息？

参与者是否会将外部的某些事件通知该系统？

系统是否会将内部的某些事件通知该参与者？



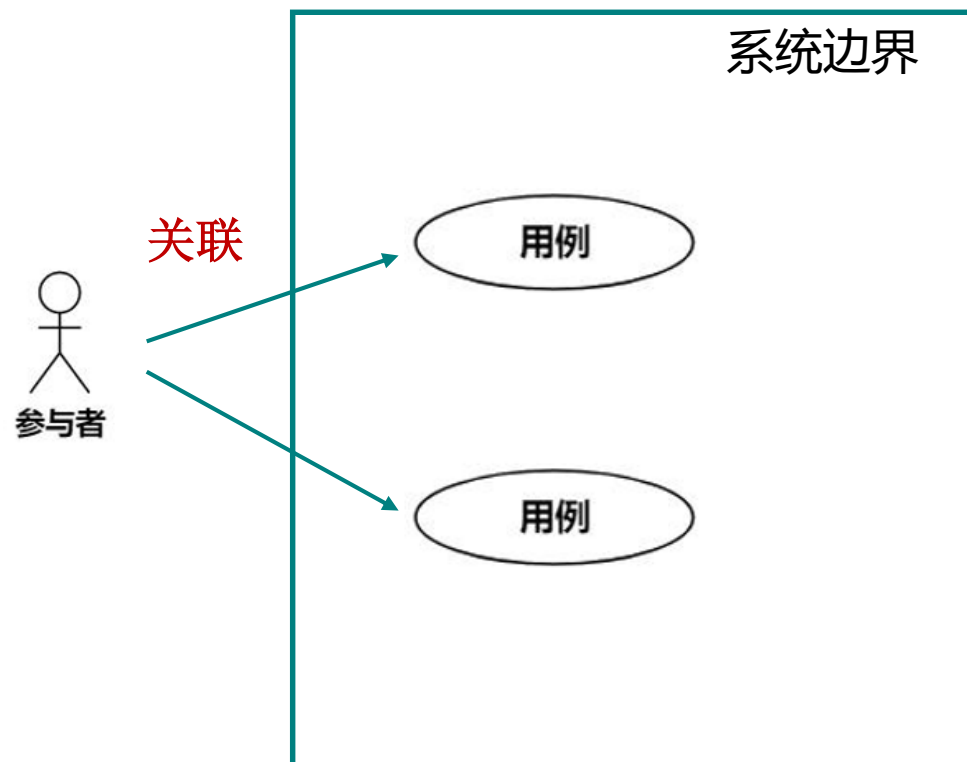
识别参与者

识别用例

绘制用例图

细化用例模型

用例描述



识别参与者

识别用例

绘制用例图

细化用例模型

用例描述

参与者之间的关系

- 泛化

用例之间的关系

- 泛化
- 扩展
- 包含



- 识别参与者
- 识别用例
- 绘制用例图
- 细化用例模型
- 用例描述**

用例描述（又叫用例规约）实际上是关于参与者与系统如何交互的规格说明，使用文字描述那些不能反映在图形上的信息。

用例描述	内容
简要描述	对用例的角色、目的的简要描述
前置条件	执行用例之前系统必须要处于的状态，或者要满足的条件
基本事件流	描述该用例的基本流程，指每个流程都正常运作时发生的事情
扩展事件流	表明用例处理过程中的一些分支或者异常情况。一般是从基本事件流的某个步骤中分离出来的备选步骤。
后置条件	用例一旦执行后系统所处的状态





辅助模型



HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

- 类图描述了类为了完成责任需要哪些操作，还描述了类之间的关系，但是在类图中没有描述对象的行为以及对象间如何交互。
- 描述清楚对象的行为以及对象之间的**交互**，有助于进一步发现与定义对象的操作，更有助于确定对象之间的关系。
- 辅助模型是系统的动态模型，描述系统结构元素的动态特性及行为。
 - 顺序图
 - 状态图
 - 活动图
 - 通信图



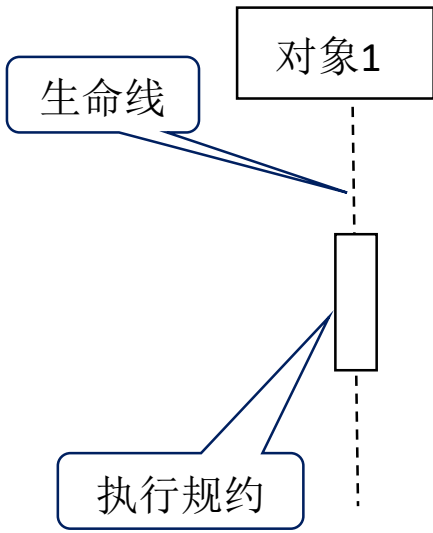
□ 基本概念

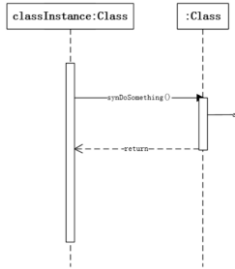
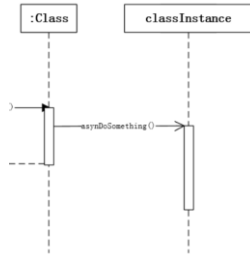
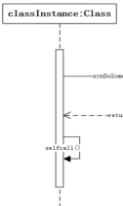
- 顺序图时**按时间顺序显示对象交互**的图。它显示了参与交互的对象和所交换信息的先后顺序，用来表示用例中的行为。
- 在UML中，顺序图将交互关系表示为一张**二维图**。纵向代表**时间维度**，时间向下延伸，按时间一次列出各个对象所发出和接收的消息，水平方向代表**对象的维度**，排列着参与交互的各个独立的对象。

□ 组成元素

- **对象**
- **生命线**
- **执行规约**
- **消息**



基本要素	含义	UML表示
对象	<p>顺序图中的对象可以是系统的参与者或者任何有效的系统对象。</p> <p>若对象置于顶部，表示在交互初对象就已经存在，若不在顶部，表示对象是在交互过程中被创建的。用矩形表示。</p>	 <p>The diagram illustrates the UML notation for objects and their execution periods. It shows a rectangular box labeled '对象1' (Object 1) at the top. A vertical dashed line, labeled '生命线' (Lifeline), extends downwards from the box. On this dashed line, there is a narrow vertical rectangle labeled '执行规约' (Execution Specification). Callout boxes with leader lines point to the dashed line and the narrow rectangle, identifying them as the lifeline and execution specification respectively.</p>
对象生命线	<p>表示对象在一段时间内的存在。</p> <p>用位于对象符号下的垂直虚线表示。</p>	
执行规约	<p>一个对象执行一个操作的时期。</p> <p>当一个对象处于一个执行规约的范围内，说明该对象的操作正在执行，否则该对象不做什么事情，但它是存在的，等待消息触发它。用窄长的矩形表示。</p>	

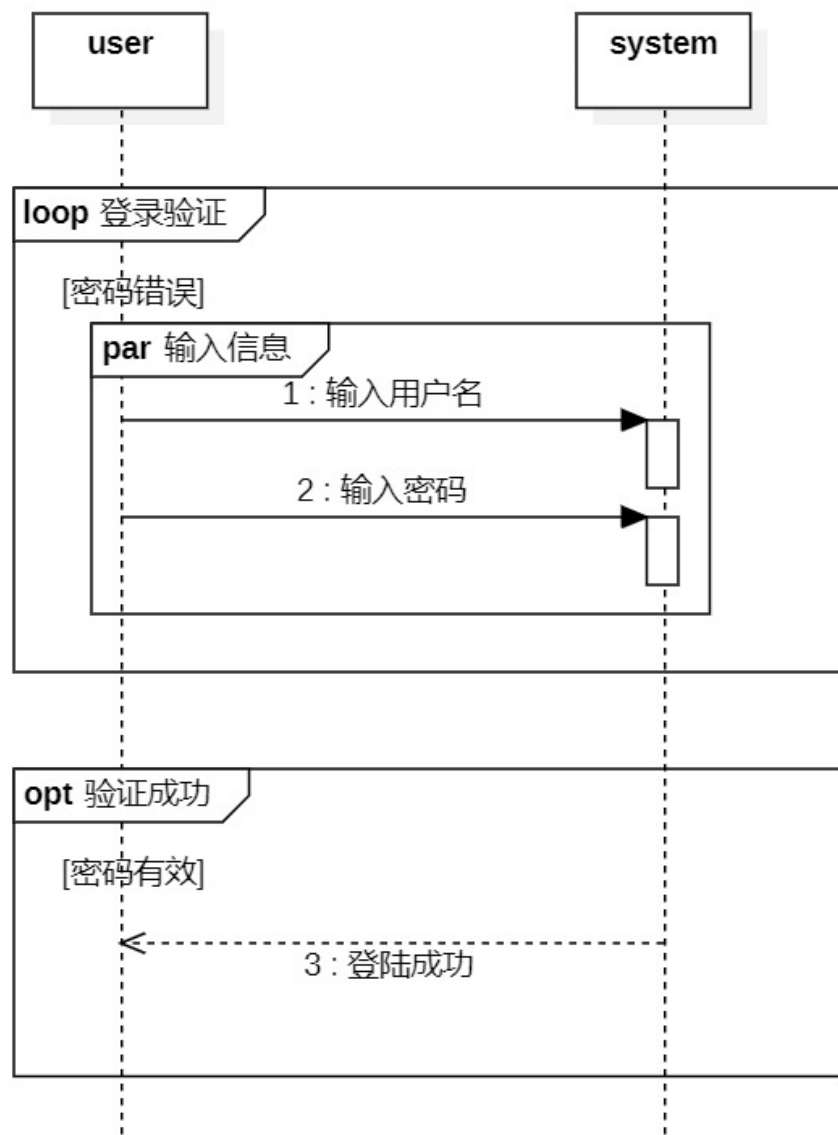
基本要素	含义		UML表示
消息	代表对象之间的通信，用于传输将发生的活动所需要的信息。表示为从一条对象生命线到另一条对象生命线的一条带有箭头的水平实现，从源对象指向目标对象。	同步消息： 发出消息的对象，会等待消息的对象的响应，有了响应后再继续。	
		返回消息： 表示从过程调用返回。	
		异步消息： 发消息的对象，发出消息后继续做自己的事情，不等待接收消息的对象的响应。	
		自身消息： 对象发给自己的消息，也就是执行自己的职责。	

□ 组合片段

UML2中，顺序图提供了“片段”机制，可以通过顺序图来表达更加复杂的动作序列。

- **可选片段OPT**：表示一种单条件分支，类似于if。
- **条件片段ALT**：表示多条件分支，类似于if -else if-else，或者switch。
- **并行片段PAR**：表示片段内有多个并行子片段，这里的并行不一定是物理上的同时执行，也指两个动作没有协作关系且可按任意次序发生的情况。
- **循环片段LOOP**：表示循环。
- **交互片段REF**：表示对其他顺序图的引用。

□ 组合片段例子



选择场景

识别角色

确定消息序列

结构化控制

确定要用顺序图描述的**交互场景**，详细的
审阅有关材料，如用例图和类图等。

选择场景

识别角色

确定消息序列

结构化控制

识别对象在交互中扮演的角色，在顺序图的上部列出所选定的一组对象，并为每个对象设置生命线，通常把发起交互的对象放在左边。



选择场景

识别角色

确定消息序列

结构化控制

决定消息将怎样或以什么样的序列在对象之间传递，通过首先发出的对象，看它需要哪些对象为它提供操作，它向哪些对象提供操作。



选择场景

识别角色

确定消息序列

结构化控制

根据消息序列之间的关系，选择适合的结构化控制操作符来表达更加复杂的动作序列。



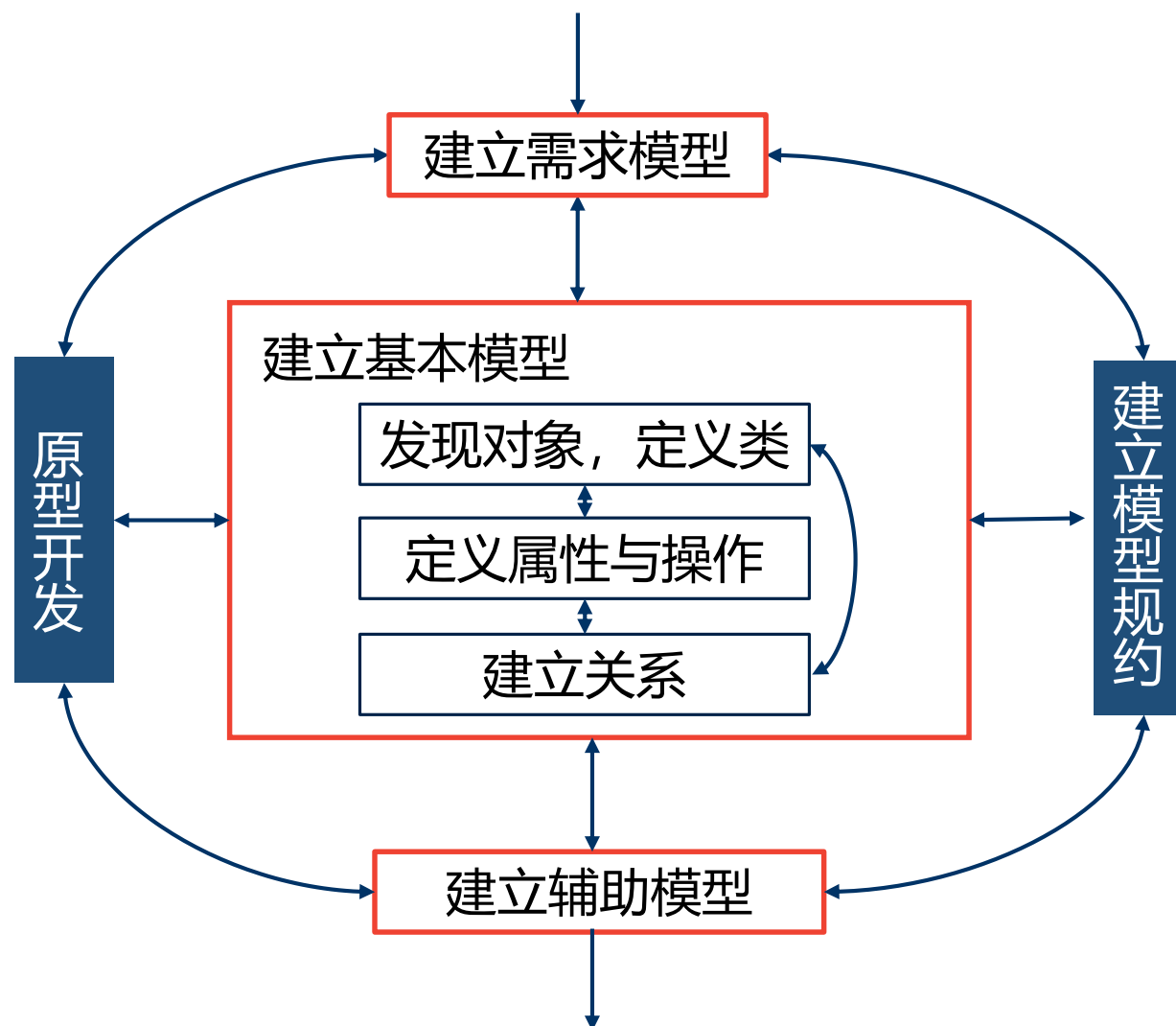


5

应用场景



HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ



机票预订系统

- 未登录的用户只能查询航班信息
- 已登录的用户可以网上购买机票，查看行程，也可以退订机票
- 系统管理员可以安排系统中的航班信息
- 该系统与外部的一个信用评价系统有交互，当用户一个月之内退订两次以上的机票时，需要降低该用户在信用评价系统中的信用等级，当信用等级过低时，则不允许该用户再次购买机票

1 识别参与者



用户



管理员



信用评价系统

2 识别用例

用户

登录

查看行程

注册

查询航班

购买机票

退订机票

管理员

登录

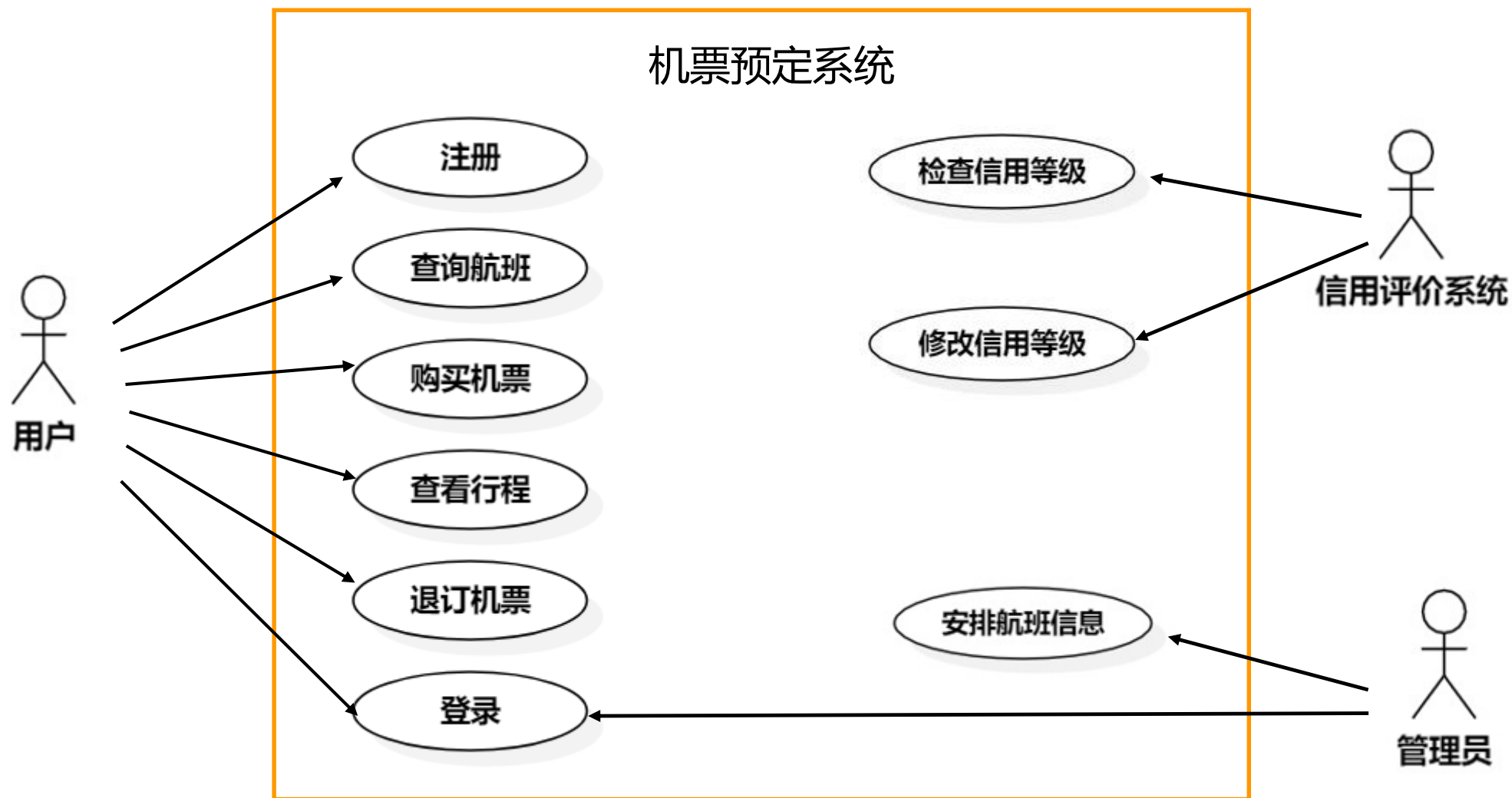
安排航班信息

信用评价系统

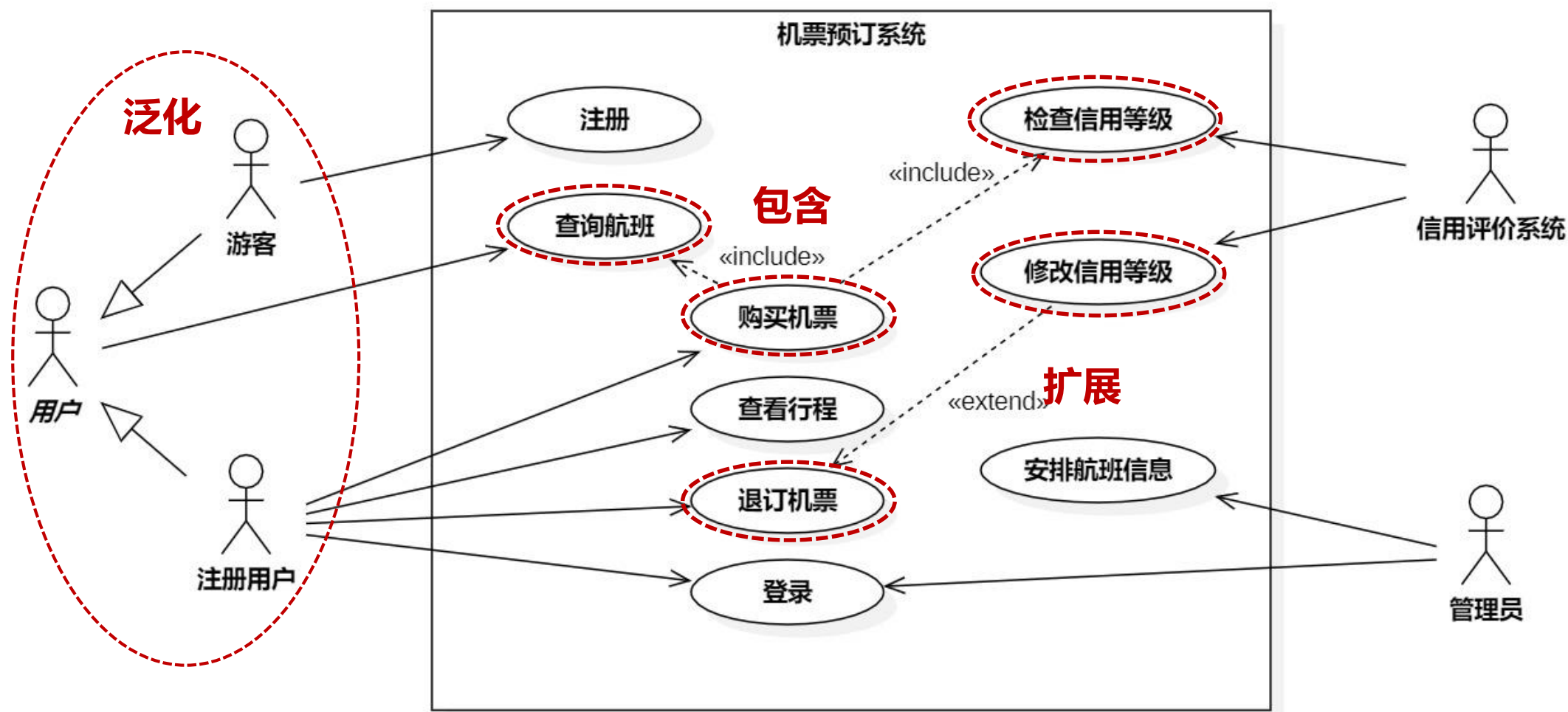
检查信用等级

修改信用等级

3 绘制用例图



4 细化用例模型

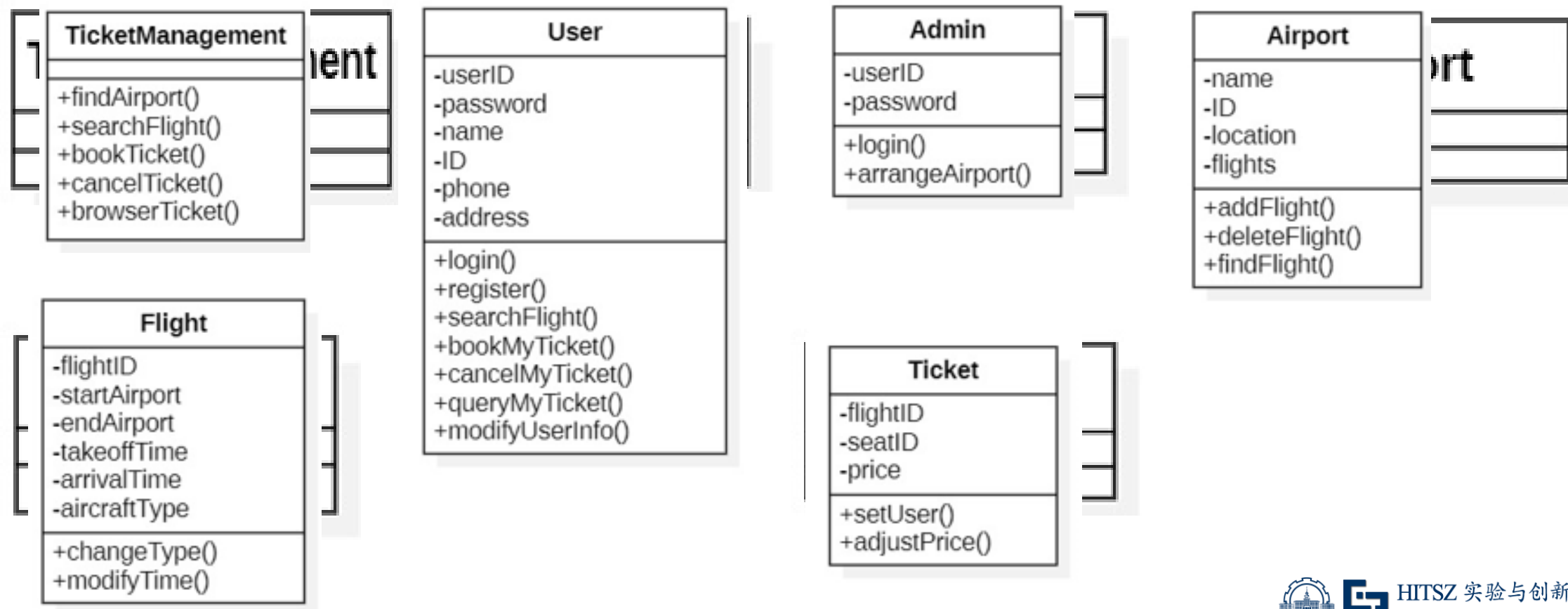


5

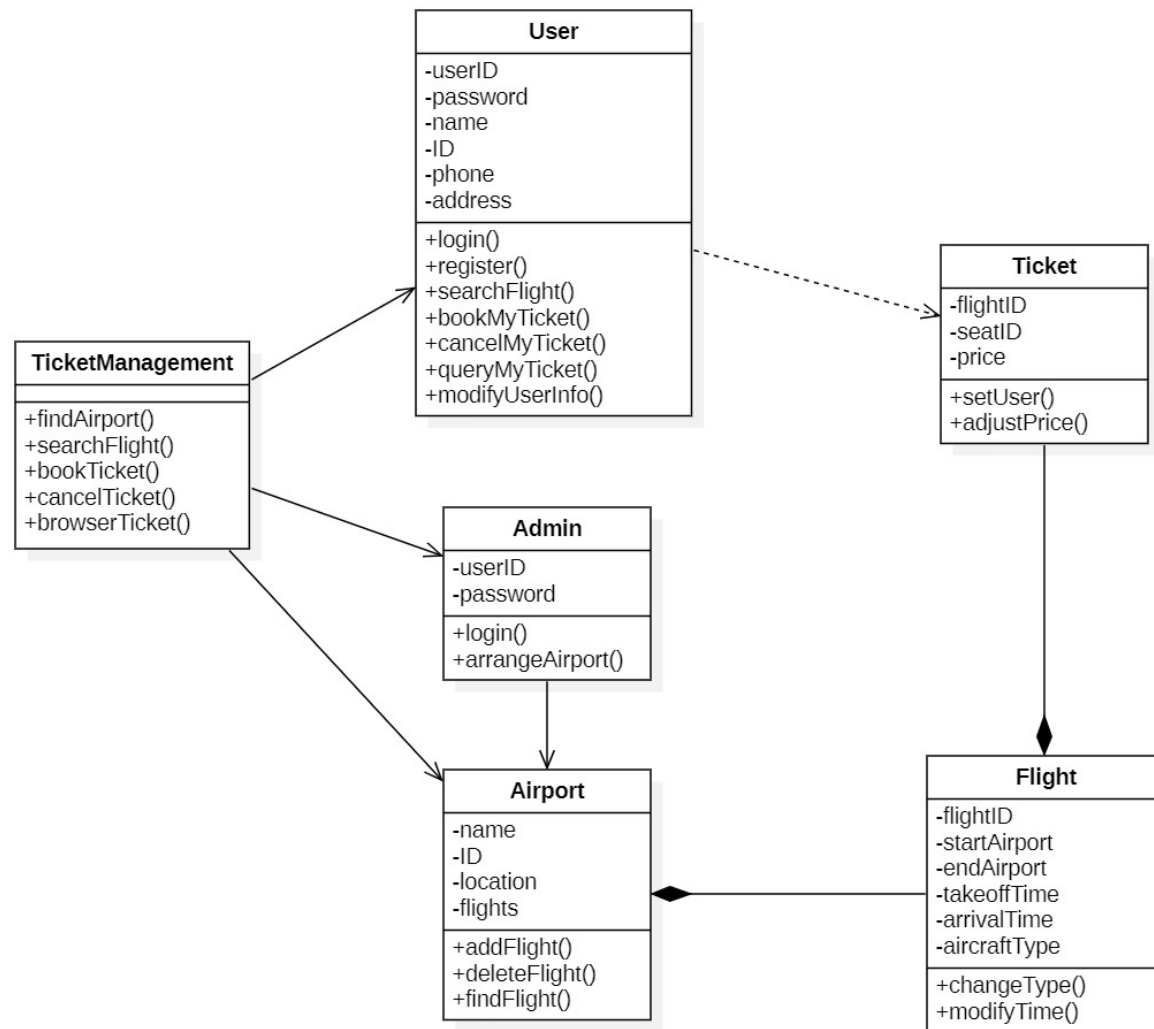
用例描述

用例名：订购机票
用例描述：描述系统用户订购机票的行为
参与者：用户，信用评价等级
前置条件：用户已经注册
基本事件流： <div>1、输入起飞和到达机场，出行日期默认为当天，可以修改</div> <div>2、查询符合条件的航班信息</div> <div>3、用户选定航班和舱位后点击“预订”</div> <div>4、查询用户信用等级</div> <div>5、查询舱位剩余机票</div> <div>6、返回订购成功信息</div>
扩展事件流： <div>1、用户信用等级不足的情况下返回订购失败，终止用例</div> <div>2、所选舱位不足返回订购失败，提醒用户重新选择舱位或者更改航班</div> <div>3、扣款失败时返回订购失败</div>
后置条件： <div>为用户生成票号，用户可以查询到预订信息</div>

- 1 确定类元素，建立对象层
- 2 确定类的属性和操作，建立特征层

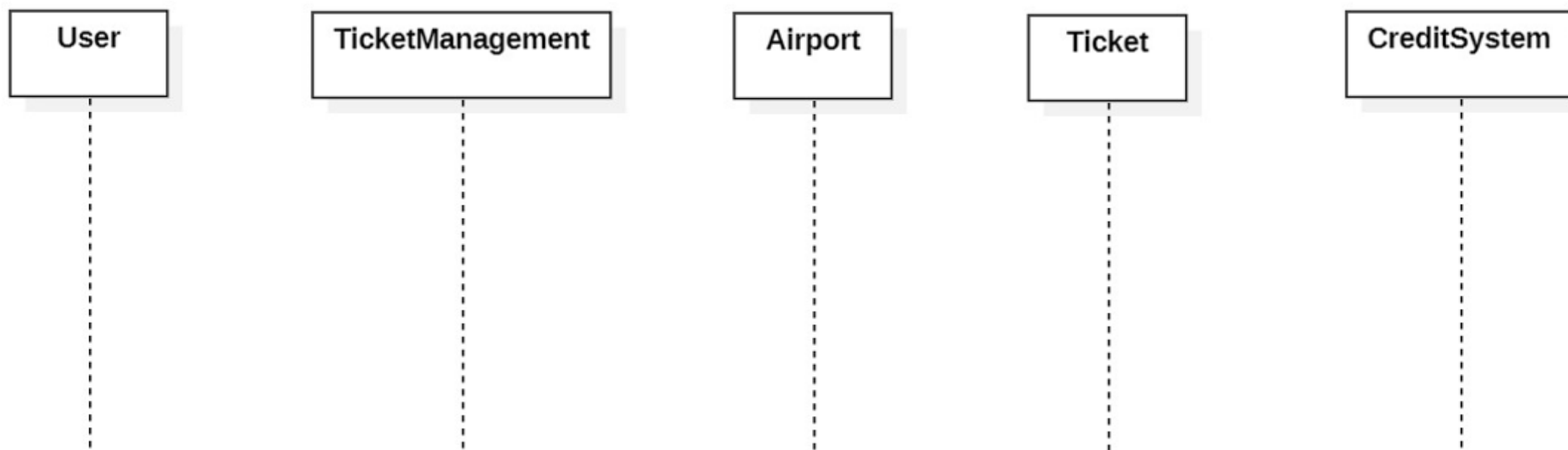


3 确定类之间的关系，建立关系层



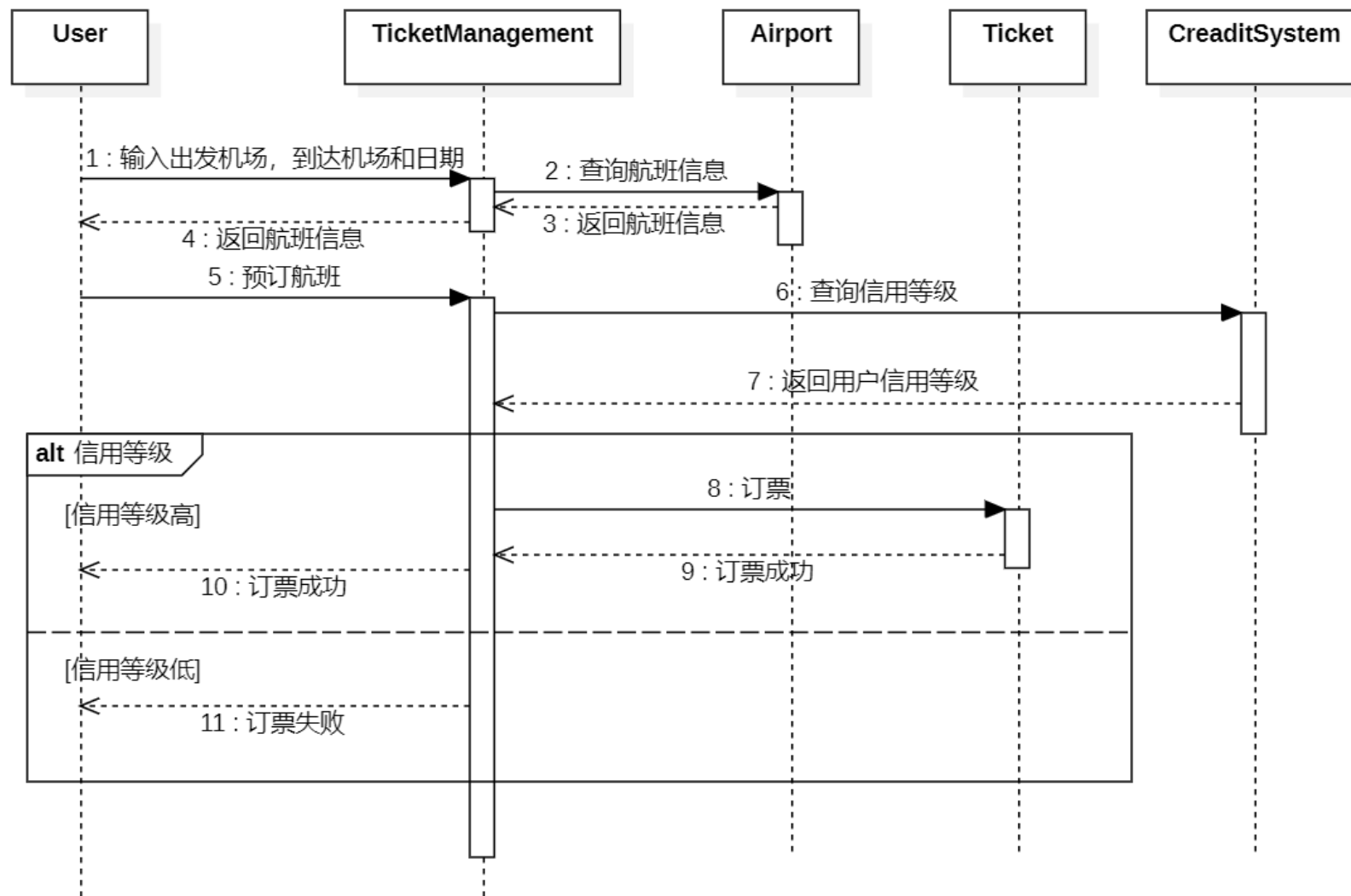
- 1 选择场景，审阅模型
- 2 识别角色，画出对象

已注册用户订购机票顺序图



3 确定消息序列

4 按需使用结构化控制



使用面向对象分析为系统建模

1 建立需求模型

用例图和用例描述

2 建立基本模型

- 建立对象层
- 建立特征层
- 建立关系层

类图

3 建立辅助模型

顺序图等



群名称:面向对象的软件构造实践

群 号:132038969