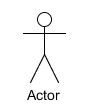
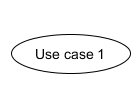
用例图（Use case）

参与者，通过使用系统服务实现其目标的那些人或者事物



用例，外部可见的系统功能，对系统提供的服务进行描述。用椭圆表示。用例是动词或者动名词。可以从每一个界面的主要功能来析取用例。



关系：

用例图中涉及的关系有：关联、泛化、包含、扩展。

如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 关系 | 涉及对象 | 符号 | 箭头方向 |
| 关联 | 参与者与用例 |  | 无箭头 |
| 泛化 | 参与者与参与者  用例与用例 |  | 指向父用例或者父参与者 |
| 包含 | 用例与用例 |  | 指向分解出来的功能 |
| 扩展 | 用例与用例 |  | 指向基础用例 |

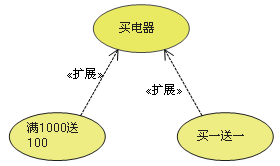
泛化



包含



扩展



扣分点：参与者，用例的词性（动名词），关联是实线还是虚线，关联是否有箭头

活动图（Activity）：每一个用例有一个活动图，活动图尽量简单

开始节点：（只有一个）



结束节点：（可以有多个）



同步条：（必须成对出现），表示并行执行



选择：（需要写明分支的判断条件）



活动：（圆边矩形，必须与状态图进行区分，动名词，可以从一个用例的执行流程中分析出动词）



扣分点：同步条必须成对出现，判定需列明条件，活动的框一定是圆边矩形

状态图（statement）：每一个用例有一个状态图，显示一个对象从创建到消亡的整个生命周期

状态：表示对象具有的一个状况，条件（名词或者说是动名词格式）



转移：事件名 [监护条件]/动作名（如果前面出现“/”,说明是系统的动作，不加是使用者进行的操作）



开始/结束状态：

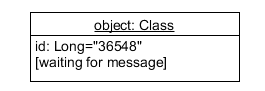


扣分点：状态是圆角矩形，状态不能是一个动作（动名词格式），而是一个状况（名词或者名词+动词的格式）必须要有状态发生变化的条件

领域模型（domain model）：一组没有定义操作（方法的特征标记）的类图，也称为概念类图

步骤：（1）寻找概念类

概念类：思想，事物或对象（也就是说找名词）



描述类：描述其他事物的信息，如Flight和Airport之间最好添加一个FlightDescription这个描述类。

（2）将其绘制为UML类图的类

（3）添加关联和属性

关联：名称需要首字母大写，一般以类名-动词短语-类名的格式来命名。但在领域模型中，避免加入太多关联，是否需要记录关联，要基于现实世界的需要，就是那些“需要记住”的关联关系。

多重性：类A有多少个实例可以和类B的一个实例关联

0或更多



1或更多



1~40



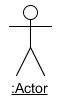
属性：对象的逻辑数据值。当需求(用例)建议或暗示需要记住信息时，引入属性

（可以只有属性名称，不需要类型，可见性）

扣分点：老师没讲这个图，pml的扣分点是通常不超过10个概念类，至少有一个描述类，不能出现XX列表

系统顺序图（SSD）：对于用例的一个特定场景（一般是一个用例），外部参与者产生的事件，其顺序和系统之类的事件。该图强调的是从参与者到系统的跨越系统边界的事件。（也就是说系统被视为黑盒）

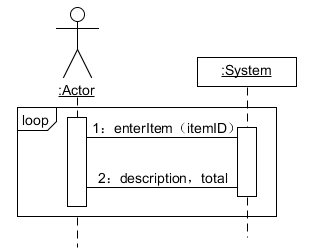
参与者：需要加‘：’和下划线



系统：



循环：



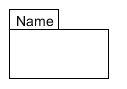
扣分点：对象必须有“：”和下划线，消息需要有编号，并且消息为动作，必须要有控制焦点（以上为顺序图的扣分点），因为我们没讲系统顺序图，盗用pml的扣分点，不超过5个消息，尽可能简单

包图（MVC架构 view【也就是UI，边界类】model【实体类，从领域模型中选取】 control【控制类】）

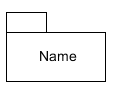
依赖：箭头指向被依赖的包（包的图都是虚线）



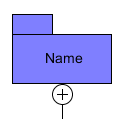
层：



类，用例等：



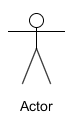
嵌入包：



扣分点：包与包之间一定是虚线

顺序图：在时间上对象交互的安排

角色（Actor）系统角色，可以是人、及其甚至其他的系统或者子系统。



对象：（名称前面要加“：”，还有下划线）（对象一般是前面包图提及的与用例相关的边界类，控制类，实体类）



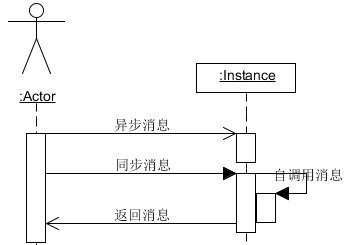
生命线：表示对象存在的时间



控制焦点：表示对象执行一个动作的期间



消息：（消息需要有序号，且消息为动作）



同步消息：消息的发送者把控制传递给消息的接收者，然后停止活动，等待消息的接收者放弃或者返回控制。用来表示同步的意义。

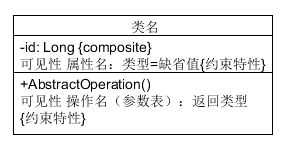
异步消息：消息发送者通过消息把信号传递给消息的接收者，然后继续自己的活动，不等待接受者返回消息或者控制。异步消息的接收者和发送者是并发工作的。

自调用信息：表示方法的自身调用以及一个对象内的一个方法调用另外一个方法。

扣分点：对象必须有“：”和下划线，消息需要有编号，并且消息为动作，必须要有控制焦点，对象来源于包图

设计类图

类：（和顺序图对应，分为实体类（领域模型的类），边界类（界面），控制类）

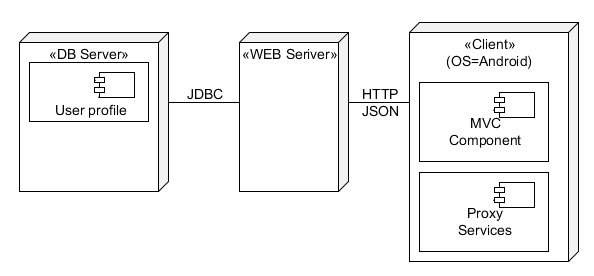


|  |  |  |  |
| --- | --- | --- | --- |
| 关系 | 表示关系 | 符号 | 箭头方向 |
| 关联 | 对象之间具有永久关系 |  | 可以用箭头表示关联的方向。没有箭头表示双向关联 |
| 多重性 | 多重性：类A有多少个实例可以和类B的一个实例关联 |  | 无箭头 |
| 依赖 | 表示一个类依赖于另一个类的定义，其中一个类的变化将影响另外一个类（use） |  | 指向被依赖的类 |
| 泛化 | 类与类的继承关系。子类继承超类的所有特（属性、操作、关系），并具有独自的特性  场景：父与子、植物与树等 |  | 指向父用例或者父参与者 |
| 实现 | 类与接口的实现关系 |  | 指向接口 |
| 聚集 | 当某一个类成为另一类的一部分时，可使用聚集的关系，表示类之间的关系是整体与部分的关系（has） |  | 部分指向整体 |
| 组合 | 对象A包含对象B，对象B离开对象A没有实际意义。是一种更强的关联关系。人包含手，手离开人的躯体就失去了它应有的作用。（contain） |  | 部分指向整体 |

组合>聚合>关联>依赖

扣分点：类名需要和领域模型，顺序图的一样；每个类至少1属性，1操作；注意继承，聚合这些关系，实线和虚线的区别，类之间的关联关系是几对几的关系；重要的是与前面的图保持一致性

部署图（整个系统就一个）可通过修改这个基础模型来做



扣分点：节点之间没有箭头，节点是软件还是物理设备需要标记，需要写清连接的介质（如网线那些），协议（HTTP等）