2017年UML考点整理

**自己手打，仅供参考**

1. 选择题（30’）
2. 判断or填空（20’）

选择15个小题、判断(选择)填空 10个小题，主要考查以下知识点：

1. 用例实现 Actor的作用

* 发现驱动用例的用户目标；为了明确外部接口和协议；为了确保确定并满足所有必要的重要实物

用例实现描述某个用例基于协作对象如何在设计模型中实现。

例题：

1. *Which of the following are not part of a use-case realization ?*

*A. Tagged Value B. Constraint C. Concurrent D. Stereotype*

1. *Which of the following is not part of a use-case relazation ?*

*A. Sequence diagrams B. Use case C. Communication D. Class diagram*

2、什么是时序图SD，什么是系统时序图SSD

3、什么是领域模型=领域类图、概念类图

A domain model is not a description of software objects; it is a visualization of the concepts or mental models of a real-world domain. Thus, it has also been called a **conceptual object model**.

**领域模型为领域内的概念类或现实世界中对象的可视化表示。领域模型也被称为概念模型、领域对象模型和分析对象模型。**

领域类图展示为：领域对象或概念类（具有符号、内涵、外延），概念类之间的关联，概念类的属性。

4、什么是用例模型

The UP defines the Use-Case Model within the Requirements discipline.Primarily,this is the set of all written use cases;it is a model of the system's functionality and environment.

Use case are text documents, not diagrams, and use-case modeling is primarily **an act of writing text**, not drawing diagrams

1. 通讯图和时序图

协作图和顺序图都表示出了对象间的交互作用，但是它们侧重点不同。

顺序图清楚地表示了交互作用中的时间顺序(强调时间)，但没有明确表示对象间的关系。

协作图清楚地表示了对象间的关系(强调空间)，但时间顺序必须从顺序号获得。

协作图和顺序图可以相互转化。

6、Controller

控制器是UI层之上的第一个对象，它负责接收和处理系统操作信息。

对于同意用例场景上的所有系统事件使用相同的控制器类。

控制器应当把需要完成的工作委派给其他的对象。控制器只是协调或控制这些活动，本身并不完成大量工作。

当把职责分配给外观控制器会导致低内聚或高耦合的设计时，通常是当外观控制器的职责过多而变的“臃肿”时，就需要考虑使用用例控制器。

GRASP控制器是领域层的一部分，它控制或协调工作请求的处理。

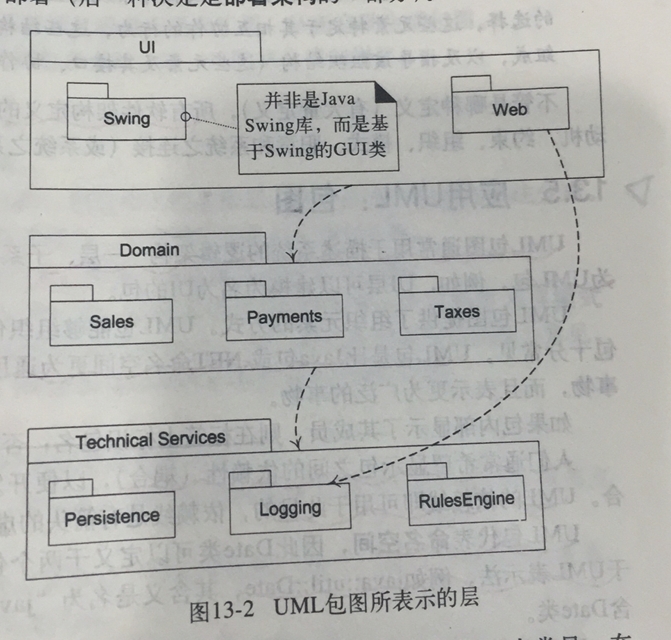
控制器优点：增加了可复用和接口可拔插的潜力。获得了推测用例状态的机会。

7、活动图、交互图

8、三层架构图 Layer 特征

较高层可以调用较低层的服务。在严格的分层架构（网络协议栈）中，层只能调用与其相邻的下层的服务。在宽松的分层架构（信息系统）中，较高层可以调用其下任何层的服务。

OO系统通常包含的层有：用户界面，应用逻辑和领域对象，技术服务。



9、P25 UP（统一过程）重点考 4个阶段，四个单词要记住。

初始阶段(Inception)：**大体上的构想**、业务案例、范围和模糊**评估**。

细化阶段(Elaboration)：已精化的构想、**核心架构**的迭代实现、**高风险**的解决、确定**大多数**需求和范围以及进行更为实际的评估。

构造阶段(Construction)：对遗留下来的风险较低的和比较简单的元素进行迭代实现，准备部署。

移交阶段(Transition)：进行beta测试和部署。

1. 信用卡服务是什么

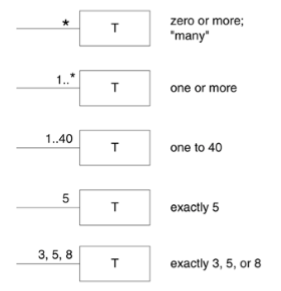
存在冗余→包含关系

用例在其他用例中重复使用

用例非常复杂并且冗长，将其分解为子单元便于理解。

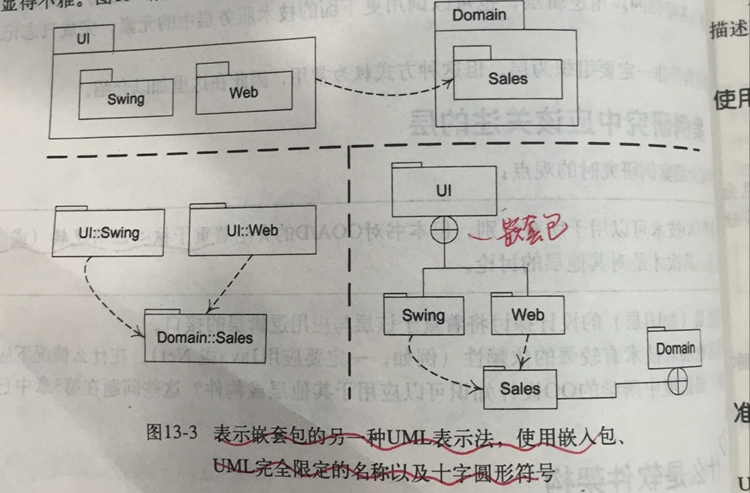
11、状态机图 P352

12、类的关系，一对一、一对多、多对多的规则



1. 系统架构图→包图

UML包图显示了包之间的依赖性，UML的依赖线（有箭头的虚线指向被依赖的包）用来表示系统内大型事物之间的耦合。



1. 简答题x3
2. SSD和SD的区别（一个分析阶段一个设计阶段等）

虽然两者在画法上有很多类似之处，系统顺序图又是从顺序图变化而来，但是两者在细节上有不少区别。

系统顺序图SSD是用例的可视化表述，是用例模型的一部分。而顺序图SD是对象方法的可视化表述。

系统顺序图SSD的研究对象是Actor参与者以及System系统，而顺序图SD的研究对象是Object对象。

系统顺序图SSD中传递的消息可以是参与者与系统之间的任意操作，可以是一个具体的网络请求的方法调用，也可以是抽象的人类行为。而顺序图SD中的消息传递是具体的方法调用（所以所指对象必须包含消息表示的方法）。

系统顺序图SSD是用例的延伸，用于帮助分析用例中参与者与系统的交互行为，大部分SSD在细化阶段创建。顺序图SD是类的延伸，用于帮助设计某个类中的具体方法。

1. Use case Driven Design、Use case Realization是什么（第六章）

用例驱动设计：系统用例开发为系统定义的用例作为整个开发过程的基础。

用例实现：描述了某个基于协作对象的用例如何在设计模型中实现。

3、GROSF中Protected Variation（防止变异）概念、用什么方式实现，回答它的应用可以用例子——数据驱动设计）

PV是通过对对象、子系统和系统进行设计来使其内部的变化或不稳定性不会对其他元素产生不良影响。

识别预计变化或不稳定之处，分配职责用以在这些变化之外创建稳定接口。数据封装、接口、多态、间接性和标准都是源于PV的。

应用：数据驱动设计。数据驱动设计，包括读取来自外部的代码、值、类文件路径、类名等，用以在运行时以某种方式改变或“参数化系统”

1. GRASP和GOF两个模式的区别

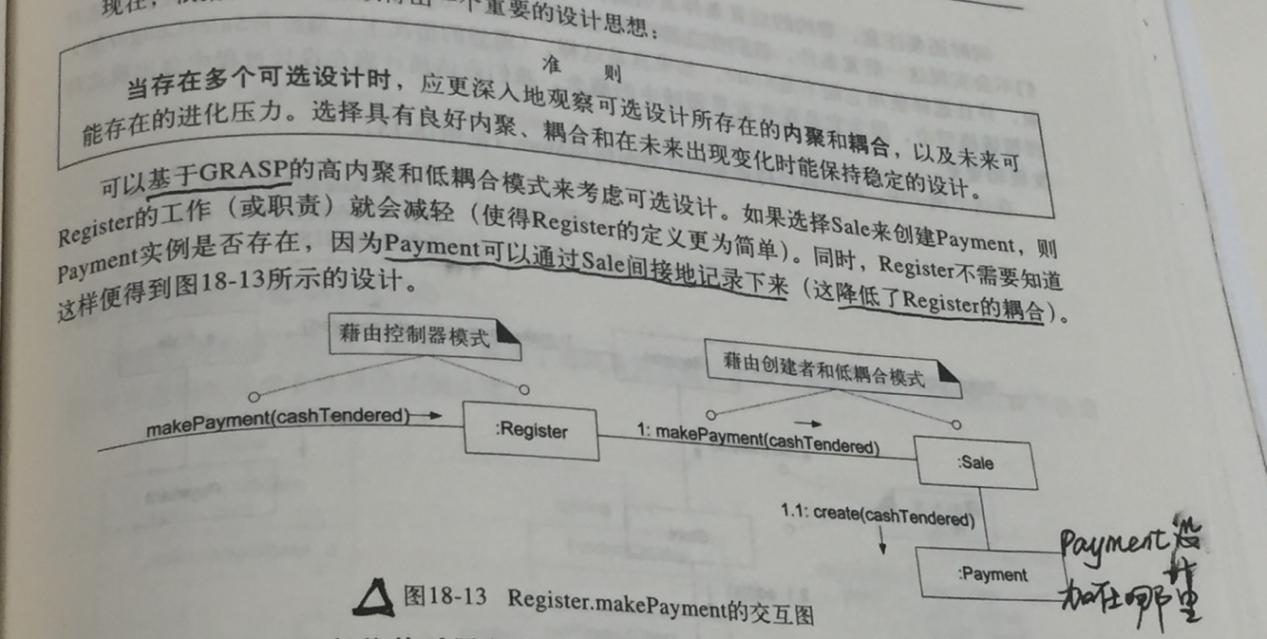
不同点：GRASP只是在对象分析的初期运用，而GoF的运用贯穿整个软件设计的始末，作用时间更长。GRASP核心思想是职责分配，只解决了整体设计中的耦合问题，GoF是在具体实现中去解决对象的耦合问题的。它是在GRASP分析的整体框架下，对一些具体的对象及其方法进行重新组织，解决对象耦合问题的。

1. 分析建模题

画一个具体应用的类图，类图的关系要描述正确，如依赖关系，类的属性和方法等。

【P245 Payment、sale那段话】用GROSP分析 高内聚低耦合的应用，降低耦合程度。

可以基于GRASP高内聚和低耦合模式来考虑可选设计。如果选择Sale来创建Payment，则Register的工作就会减轻。同时Register不需要知道Payment实例是否存在，因为Payment可以通过Sale间接地记录下来，这降低了Register的耦合。



1. 综合设计题（30’）

4个小题（用例图、活动图、时序图、状态机图(不必描述所有可能的事件)）

以下三种中考一个：

在线租车系统（book、还、计费）

智能牙刷或温度计（对应app，把信息传到app上）

图书管理系统

**P.S. 建模题没有标准答案，所以一定要画图！画图就有分拿！**

补充：每种图的细节