**选择判断（选择题：2’X15=30，判断题：1’X10=10’）**

1. Package diagram

用来描述系统的逻辑[architecture]架构，组织元素

1. OO核心理念（抽象、封装、继承[inherit]、多态[polymorphism]）

对象：对象是要研究的任何事物

类：类是对象的**模板**，即类是对一组有相同**属性和操作**的对象的定义

抽象：区别于其他实体，最本质、内在的属性

1. UP过程

初始阶段[inception]：预见项目的范围、设想[vision]和业务案例

细化阶段[elaboration]：已精化的[defined]设想，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[realistic]评估[estimates]

构造阶段[construction]：迭代实现遗留下来的风险较低和比较容易的元素，准备部署

移交阶段[transition]：测试、部署

1. 科目与制品

九大科目[discipline]：一个主题域的一组活动，主要讨论：业务建模、需求、设计、实现

制品[artifacts]：科目活动过程中的产物

1. Actor（用例图，相当于用户）什么作用？

为了找到驱动用例的用户目标；To find user goals, which drive the use cases.

为了明确外部接口和协议；To clarify external interfaces and protocols.

为了确保所有必要的事物都被识别到并满足。To ensure that all necessary interests are identified and saisfied.

1. 领域[domain]模型、用例model

领域模型：对领域内的概念[conceptual]类或现实世界[real-situation]中的对象的可视化表示//即，不是软件类

可视化字典，没有操作的类图，范围：场景,

用例model：①所有用例的集合；②系统功能性[functionality]和环境的模型。

1. SD;SSD;用例实现

用例是文本形式的情节描述：摘要、非正式、详述[fully dressed]

其他的下边都有了

1. 通信图与SD区别

顺序图：强调对象之间消息传递的**时间**先后

通信图：强调有**特定结构**的对象群之间的消息传递

语法相同，语义不同：has the same semantic , but different representations

1. Controller

**Beyond UI**，**接收和协调**[coordinates]（控制、处理）**系统操作**的第一个对象

控制器一般把需要完成的工作**委派[delegation]**给其他对象

其他选项的option：

①外观（当系统操作没那么多时）

②用例控制器（当系统操作多到外观“臃肿”时，每个用例有自己的控制器）

优点：①增加了可复用和接口可插拔[pluggable]的潜力；②获得了推测用例状态的机会。

1. 活动图、交互图、领域模型（或称设计类图）

活动图：一个过程中的多个顺序活动和并行活动

交互图：描述对象之间通过消息的交互

领域类图：

1. 类的关系：一对一？多对多？零对多？

即多重性

1. 状态机图（chapter 29）

①In the context of一个用例，被系统操作者recognized and handled的**外部[**external**]系统事件**

②显示一个**对象**在其生命周期内的**行为**

1. 时序图

对于一个特定场景[scenario]，与系统相关的输入和输出

1. [三层架构](http://baike.baidu.com/item/%E4%B8%89%E5%B1%82%E6%9E%B6%E6%9E%84)(3-tier architecture) //课本并没有

通常将整个业务应用划分为：

界面层（User Interface layer）

业务逻辑层（Business Logic Layer）

数据访问层（Data access layer）

区分层次的目的即为了“[高内聚低耦合](http://baike.baidu.com/item/%E9%AB%98%E5%86%85%E8%81%9A%E4%BD%8E%E8%80%A6%E5%90%88)”的思想。

最后考试选择题问三层架构的缺点是啥[微笑]

**简答题（5’X3=15）**

1. 用例的驱动设计及实现是什么（chapter6）

用例的驱动设计：功能需求首先记录在用例中，用例是迭代计划的重要部分。

用例实现：某个用例 基于[in terms of]协作[collaborating]对象 如何在 设计模型 中实现

1. SSD系统时序图（chapter10），SD时序图（chapter15）的区别

顺序图：表示对象之间消息的传递

SSD：显示system&actors间的交互作用。

1. P245图18-13 Payment，GRASP

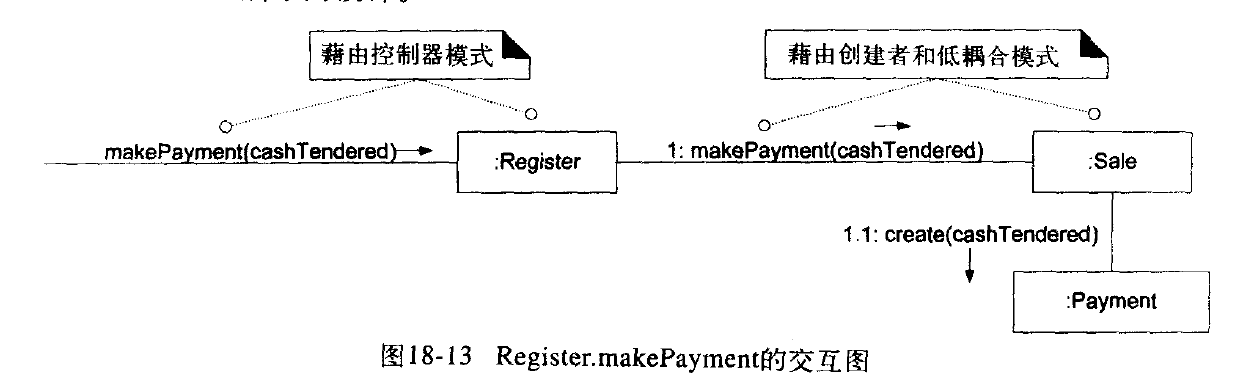
Q：谁来创建payment实例？

创建者模式：Register（记录payment）、Sale（频繁地使用payment）

信息专家模式：Register（拥有其初始化数据：现金收到的数额）

准则：当存在多个option时，应更深入地研究option的内聚、耦合、未来可能存在的进化压力

结果：使用Sale来创建payment，Register的工作就会减轻，同时，Register不需要知道payment是否存在，降低了Register的耦合



1. P310防止变异PV，列举是干什么的，概念，措施，如何实现，用来干什么？

目的：the variations or instability in these elements does not influence other elements

使得元素内部的变化/不稳定不会影响其他元素

定义：Identify points of predicted variation or instability; assign responsibilities to create a stable interface around them.

识别预见到的变化或不稳定之处，分配职责，在这些变化之外创建稳定的接口

隐藏结构的设计

1. GRASP和GOF的区别

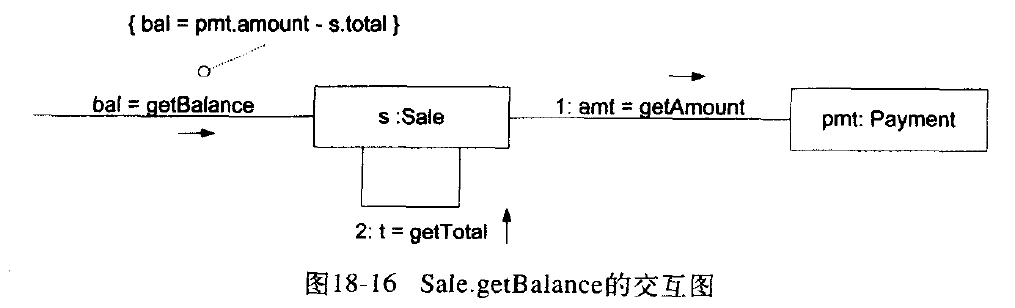
①GRASP:核心思想是职责分配（Responsibility Assignment）

GoF：在**具体**实现中解决对象的**耦合**问题。它在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。

②GRASP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

**分析建模题15’（2~3题）**

1. 画类图，类之间的关系（依赖……），画清楚
2. 基于GRASP规则做一个SALE,REGISTER分析改进 P247例子



Q：谁负责获知支付余额？

信息专家：Sale、payment

选payment：需要对sale的可见性，增加了耦合

选Sale：需要对payment的可见性，但因为创建，所以已经有可见性了，所以选择Sale

**综合题30’（四主题备选，考5个图\*6’）**

1.图书管理

2.在线租车系统

3.智能牙刷

4.智能温度计

（画用例图、活动图、时序图、类图、状态机图）

P352 状态机图 P25 UP的四个阶段 P148 系统架构图用包图实现

例如：租车（租，计费，还）

活动图——如何计价格。