面向对象分析与设计 A 卷答案

一、 简答题（每题 10 分，共 30 分）

1、 简述层次体系结构的设计原则及其在课程设计中的体现。

a. 把系统按照一定的逻辑关系划分成一个由上至下的层级结构，上层代码只能调用其直接下层代码。（3 分）

b. 在课程设计中分为 Controller 层，Service 层，Dao 层和 Mapper 层（3 分）

c. Controller 层负责输入和输出数据的检查和转换，Service 层负责业务，Dao 层负责数据存取，Mapper 负责对象模型与关系模型的转换（4 分）

2、 简述用 UML 时序图描述系统顺序图和设计动态模型时的差别。

a. 系统顺序图描述的是人和系统之间，以及系统和集成系统之间的交互（1 分）。采用 UML 时序图描述系统顺序图时，lifeline 描述的是人、待开发的系统和需集成的系统（2 分），Message 描述的是三者之间的交互（2 分）。

b. 设计动态模型描述的是对象之间的协作（1 分）。采用 UML 时序图描述设计动态模型时，lifeline 描述的是对象（2 分），message 描述的是对象之间的方法的调用关系（2 分）。

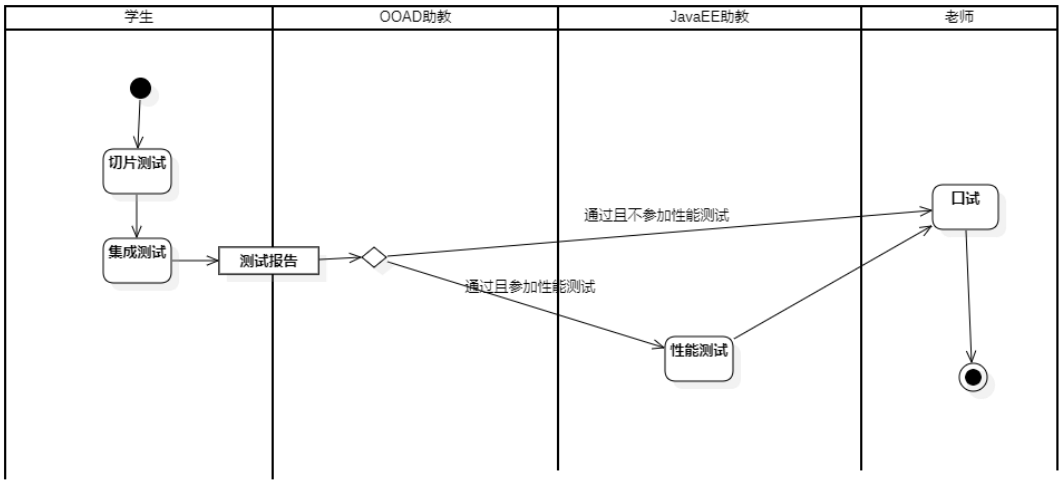
3、 简述活动图和状态机图的差别以及各自的用途。

a. 活动图中基本的符号是动作节点（Action）、判断结点、分支、合并节点、开始和结束节点等（3 分），用于描述业务流程和代码的逻辑关系（2 分）。

b. 状态机图中的基本符号是状态、状态的迁移、初始状态符号和终结状态符号（3 分），可用于描述对象的状态以及状态的迁移（2 分）。

二、 识图画图题（每题 15 分，共 30 分）

1、请用活动图描述课程设计的期末检查流程。



a. 切片测试，集成测试，性能测试和口试四个动作及顺序描述正确（5 分）

b. 学生、助教和老师用不同泳道描述（5 分）

c. 判断节点、起始和终止节点以及其他（5 分）

2、图 1 是某小组描述支付待支付的订单的时序图，请标出图中画图错误的部分并简述理由。

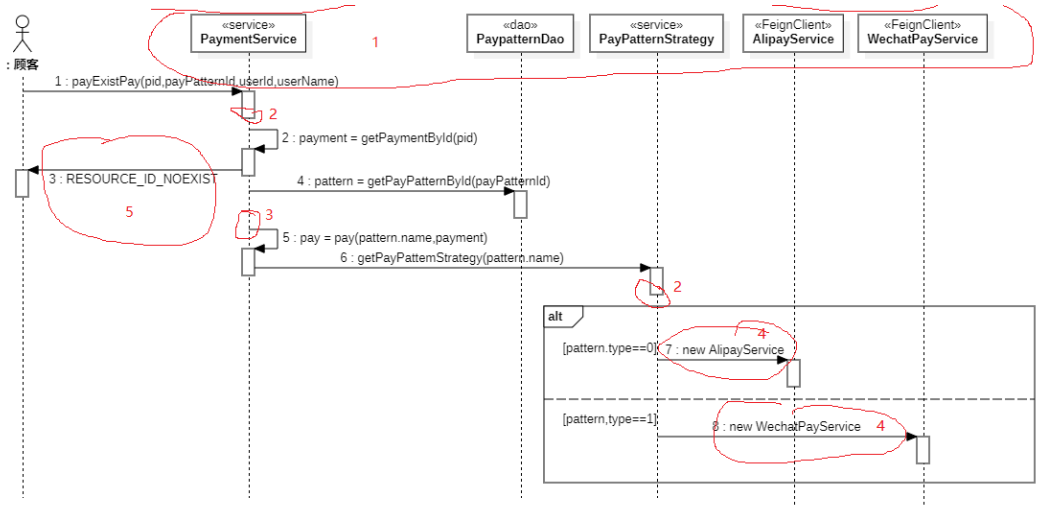


图 1：支付订单时序图

1. 所有的对象均未标对象名称（3 分）
2. Lifeline box 未延伸到方法的结束，（3 分）
3. 消息从 lifeline 上直接引出，而不是从 lifeline box（3 分）
4. 构造对象应该用约定的 create 方法（3 分）
5. 返回消息需要用虚线（3 分）

三、分析题（每题 20 分，共 40 分）：

1、图 2-4 是商品优惠规则的设计类图和时序图，该设计的意图是试图保护优惠规则变化对于代码的影响，请用 GRASP 方法分析该设计是怎样达成设计意图。

a. 虚拟的方法-----将商品的优惠活动分成优惠折扣和优惠条件两个部分，通过两个部分的组合来形成优惠活动，从而实现不修改代码增加新的优惠活动—开闭原则（3 分）
b. 多态和 Liskov 替换原则----在 BaseCouponDiscount 和 BaseCouponLimitation 两边都采用多态的方式实现不同的优惠折扣和优惠条件，所有的优惠折扣和优惠活动都符合 Liskov 原则，可以任意替换而不影响父类的行为（3 分）
c. ComplexCouponLimitation 同样符合多态和 Liskov 替换原则，使得复合优惠条件在行为上与单个优惠条件一致，可以实现和各类优惠折扣组合形成优惠活动。（2 分）
d. 信息专家—按照信息的归属将折扣和条件的代码分别归属不同的类（2 分）
e. 间接---通过 BaseCouponDiscount 来使用 BaseCouponLimitation，原因是优惠条件的变化性更大，所有通过间接的方式隐藏在 BaseCouponDiscount 后面，使得其变化不会影响其他部分的代码---开闭原则（3 分）
f. 创建者—由 CouponDiscount 创建 CouponLimitation，原因是 BaseCouponLimitation 只被 BaseCouponDiscount 使用（3 分）
g. 高内聚和低耦合---所有不同的折扣和条件的代码都限定在自己特定的类里，利用 Liskov 原则和多态与其他部分低耦合。（2 分）
h. PV 变化—所有的值以 JSON 方式存储，体现了 PV 的数据驱动方式（2 分）

2、图 5-8 是某小组设计的支付模块的设计类图和时序图，着重解决支付渠道的变动对于系

统的影响，请从 GRASP 方法和面向对象设计的基本原则的角度分析该设计是怎样达成设计意图。

a. 多态和 Liskov 替换原则----在 BasePaymentPattern 中采用多态的方式将不同的支付渠道 API 统一成一个接口，所有的 API 都符合 Liskov 原则，可以相互替换而不影响在 PayService 中的调用（5 分）
b. 间接—PayService 没有直接调用 WechatService 和 AlipayService，而是分别通过 WechatPattern 和 AliPayPattern 来间接使用接口，这样的设计在未来 Wechat 或者 AliPay 接口发生改变时，有可能不会影响 PayService（5 分）
c. 创建者—由 PayPatternLoader 获得 WechatPattern 和 AliPayPattern 对象，WechatPattern 和 AliPayPattern 对象均为 Spring 容器的 Bean 对象，在 PayPatternLoader 的 getPatternByName 利用 Spring 容器的 ApplicationContext 从 Spring 容器中找到 WechatPattern 和 AliPayPattern 对象（5 分）。
d. 高内聚和低耦合---不同的支付渠道的代码都限定在 WechatPattern 和 AliPayPattern 类中里，利用 Liskov 原则和多态与其他部分低耦合。当增加一种新的支付渠道时，只需要在 BasePaymentPattern 下增加一个新的实现（5 分）