1. Explain OOA/D

- OOAD: 从对象的角度考虑问题领域和概念上的解决方案
- OOA: 在面向对象分析中，强调的是在问题领域内发现和描述对象（或概念）
- OOD: 在面向对象设计的过程中，强调的是定义软件对象以及他们如何协作以实现需求

2. OOA&D vs structured A&D

- OOA&D通过对象和概念分解。
- 结构化的A＆D按功能和过程分解。
- OOA和结构化分析之间的概念区别是：按概念类(对象)划分，而不是按功能划分

3. 迭代过程和敏捷模型

- 迭代开发是OOAD成为最佳实践的核心
    - 迭代和进化式开发（iterative and evolutionary development）：短时快速的开发步骤、反馈和改写来不断明确需求和设计。
- 敏捷建模是有效的应用UML的关键
    - 什么是敏捷建模?
        - 敏捷开发方法通常应用时间定量的迭代和进化式开发、使用自适应计划、提倡增量交付并包含其他提倡敏捷性（快速和灵活的响应变更）的价值和实践。
        - 采用敏捷方法并不意味着不进行任何建模，这是错误的理解
        - 建模和模型的目的主要是用于理解和沟通，而不是构建文档
        - 不要对所有或大多数软件设计建模或者应用UML
        - 尽可能使用最简单的工具

4. What is EBP?

- EBP即**基本业务过程**，是源于业务过程工程领域的术语，定义如下：

- 一个人于某时刻在一个地点所执行的任务，用以响应业务事件。该任务能够增加可量化的业务价值，并且以持久状态留下数据。如批准信用卡的信用额。

5. use case model vs feature list

- 对于用例：
  - 用例是文本情景，它们更简单且更为我们所熟悉。
  - 强调了用户的目标和观点：“谁使用系统？他们使用的典型场景是什么？他们的目的是什么？"
  - 发现和定义功能型需求的中心机制。
  - **能够根据需要对复杂程度和形式化程度进行增减调节。**
- 而功能列表：
  - 采用传统方法，对于面向用户的软件不太可取，它
    - 只提功能，不提流程
    - 如果流程写在其他地方，则冗余且难以并行维护。
- 在文档中加入简洁的高阶特性列表有助于概括系统的功能性

6. use case 是否是a good one

应用老板测试，EBP测试和规模测试进行分析

- 三种寻找有用的用例的**测试方法**：
  - **老板测试**
    - 你的老板问：“你整天都做了什么？”你回答“登录系统！”你的老板会高兴吗？
    - 如果不会，那么该用例不会通过老板测试，这意味着**该用例与达到可量化价值的结果没有太大关系。**
  - **EBP测试**
    - EBP即**基本业务过程**，是源于业务过程工程领域的术语，定义如下：
    - 一个人于某时刻在一个地点所执行的任务，用以响应业务事件。该任务能够增加可量化的业务价值，并且以持久状态留下数据。如批准信用卡的信用额。

    - **规模测试**
        - 用例通常包含多个步骤。
- **示例：应用上述测试**
    - 就供应者合同进行协商
        - 比EBP跟广泛，用时更长。更适合作为业务用例，而非系统用例。
    - 处理退货
        - 能够通过老板测试。看上去与EBP类似。规模合适。
    - 登录
        - 如果你一整天都在登录，老板不会满意。
    - 在游戏板上移动棋子
        - 单一步骤，不能通过规模测试。

7. create a domain model

- （以当前迭代中所要设计的需求为界）
    - a) 寻找概念类（参见后继准则）
    - b) 将其绘制为UML类图中的类
    - c) 添加关联和属性

8. 用例图 文字用例？？？

- 处理销售的详细用例：

用例UC1：处理销售

范围：NextGen POS应用

级别：用户目标

主要参与者：收银员

涉众及其关注点：

- 收银员：希望能够准确、快速地输入，而且没有支付错误，因为如果少收货款，将从其薪水中扣除。
- 售货员：希望自动更新销售提成。
- 顾客：希望以最小代价完成购买活动并得到快速服务。希望便捷、清晰地看到所输入的商品项目和价格。希望得到购买凭证，以便退货。
- 公司：希望准确地记录交易，满足顾客要求。希望确保记录了支付授权服务的支付票据。希望有一定的容错性，即使在某些服务器构件不可用时（如远程信用卡验证），也能够完成销售。希望能够自动、快速地更新账务和库存信息。
- 经理：希望能够快速执行超控操作，并易于更正收银员的不当操作。
- 政府税收代理：希望能从每笔交易中抽取税金。可能存在多级税务代理，比如国家级、州级和县级。
- 支付授权服务：希望接收到格式和协议正确的数字授权请求。希望准确计算对商店的应付款。

前置条件：收银员必须经过确认和认证。

成功保证（或后置条件）：存储销售信息。准确计算税金。更新账务和库存信息。记录提成。生成票据。记录支付授权的批准。

主成功场景（或基本流程）：

1. 顾客携带所购商品或服务到收银台通过POS机付款。
2. 收银员开始一次新的销售交易。
3. 收银员输入商品条码。
4. 系统逐条记录出售的商品，并显示该商品的描述、价格和累计额。价格通过一组价格规则来计算。

收银员重复3～4步，直到输入结束。

5. 系统显示总额和所计算的税金。
6. 收银员告知顾客总额，并请顾客付款。
7. 顾客付款，系统处理支付。
8. 系统记录完整的销售信息，并将销售和支付信息发送到外部的账务系统（进行账务处理和提成）和库存系统（更新库存）。
9. 系统打印票据。
10. 顾客携带商品和票据离开（如果有）。

扩展（或替代流程）：

*a. 经理在任意时刻要求进行超控操作：

1. 系统进入经理授权模式。
2. 经理或收银员执行某一经理模式的操作。例如，变更现金结余，恢复其他登录者中断的销售交易，取消销售交易等。
3. 系统回复到收银员授权模式。

*b. 系统在任意时刻失败：

为了支持恢复和更正账务处理，要保证所有交易的敏感状态和事件都能够从场景的任何一步中完全恢复。

1. 收银员重启系统，登录，请求恢复上次状态。
2. 系统重建上次状态。

2a. 系统在恢复过程中检测到异常：

1. 系统向收银员提示错误，记录此错误，并进入一个初始状态。
2. 收银员开始一次新的销售交易。

1a. 客户或经理需要恢复一个中断的销售交易。

1. 收银员执行恢复操作，并且输入ID以提取对应的销售交易。
2. 系统显示被恢复的销售交易状态及其小计。

2a. 未发现对应的销售交易。

1. 系统向收银员提示错误。
2. 收银员可能会开始一个新销售交易，并重新输入所有商品。

3. 收银员继续该次销售交易（可能要输入更多的商品或处理支付）。

2-4a. 顾客告诉收银员其免税状况（例如，年长者、本国人等）。

1. 收银员进行核实，并输入免税状况编码。
2. 系统记录该状况编码（在计算税金时使用）。

3a. 无效商品ID（在系统中未发现）：

1. 系统提示错误并拒绝输入该ID。
2. 收银员响应该错误。

2a. 商品ID可读（例如，数字型的UPC（通用产品代码））：

1. 收银员手工输入商品ID。
2. 系统显示商品项目的描述和价格。

2a. 无效商品ID：系统提示错误。收银员尝试其他方式。

2b. 系统内不存在该商品ID，但是该商品附有价签：

1. 收银员请求经理执行超控操作。
2. 经理执行相应的超控操作。
3. 收银员选择手工输入价格，输入价签上的价格，并且请求对该价目进行标准计税。（因为没有产品信息，计税引擎无法确定如何计税。）

2c. 收银员通过执行寻找产品帮助以获取正确的商品ID及其价格。

2d. 另外，收银员可以向其他员工询问商品ID或价格，然后手工输入ID或价格（参见以上内容）。

3b. 当有多个商品项目属于同一类别的时候（如5个汉堡包），不必记录每个商品项目的唯一标识：

1. 收银员可以输入类别的标识和商品的数量。

3c. 需要手工输入类别和价格（例如，花卉或纸牌及其价格）：

1. 收银员手工输入特定的类别代码及其价格。

3-6a. 顾客要求收银员从所购商品中去掉一项：

所去除商品的价格必须小于收银员权限，否则需要经理执行超控操作。

1. 收银员输入商品ID并将其删除。
2. 系统删除该项目并显示更新后的累计额。

2a. 商品价格超过了收银员权限：

1. 系统提示错误，并建议经理超控。
2. 收银员请求经理超控，完成超控后，重做该操作。

3-6b. 顾客要求收银员取消销售交易：

1. 收银员在系统中取消销售交易。

3-6c. 收银员延迟销售交易：

1. 系统记录销售交易信息，使其能够在任何POS登录中恢复操作。
2. 系统显示用来恢复销售交易的“延迟票据”，其中包含商品项目和销售交易ID。

4a. 系统定义的商品价格不是顾客预期的价格（顾客对此产生抱怨并要求减价）：

1. 收银员请求经理批准。
2. 经理执行超控操作。

3. 收银员手工输入超控后的价格。
4. 系统显示新价格。

5a. 系统检测到与外部税务计算系统服务的通信故障：
1. 系统在POS机节点上重启此服务，并继续操作。
1a. 系统检测到该服务无法重启。
1. 系统提示错误。
2. 收银员手工计算和输入税金，或者取消该销售交易。

5b. 顾客声称他们符合打折条件（例如，是雇员或重要顾客）：
1. 收银员提出打折请求。
2. 收银员输入顾客ID。
3. 系统按照打折规则显示折扣总计。

5c. 顾客要求兑现账户积分，用于此次销售交易：
1. 收银员提交积分请求。
2. 收银员输入顾客ID。
3. 系统应用积分直到价格为0，同时扣除结余积分。

6a. 顾客要求现金付款，但所携现金不足：
1. 顾客要求使用其他支付方式。
1a. 顾客要求取消此次销售交易，收银员在系统上取消该销售交易。

7a. 现金支付：
1. 收银员输入收取的现金额。
2. 系统显示找零金额，并弹出现金抽屉。
3. 收银员放入收取的现金，并给顾客找零。
4. 系统记录该现金支付。

7b. 信用卡支付：
1. 顾客输入信用卡账户信息。
2. 系统显示其支付信息以备验证。
3. 收银员确认。
3a. 收银员取消付款步骤。
1. 系统回复到“商品输入“模式。
4. 系统向外部支付授权服务系统发送支付授权请求，并请求批准该支付。
4a. 系统检测到与外部系统协作时的故障：
1. 系统向收银员提示错误。
2. 收银员请求顾客更换支付方式。
5. 系统收到批准支付的应答并提示收银员，同时弹出现金抽屉（以便放入签名后的信用卡支付票据）。
5a. 系统收到拒绝支付的应答：
1. 系统向收银员提示支付被拒绝。
2. 收银员请求顾客更换支付方式。

5b. 应答超时。

1. 系统提示收银员应答超时。

2. 收银员重试，或者请求顾客更换支付方式。

6. 系统记录信用卡支付信息，其中包括支付批准。

7. 系统显示信用卡支付的签名输入机制。

8. 收银员请求顾客签署信用卡支付。顾客输入签名。

9. 如果在纸质票据上签名，则收银员将该票据放入现金抽屉并关闭抽屉。

7c. 支票支付……

7d. 记账支付……

7e. 收银员取消支付步骤：

1. 系统回到“商品输入”模式。

7f. 顾客出示优惠券：

1. 在处理支付之前，收银员记录每张优惠券，系统扣除相应金额。系统记录已使用的优惠券以备账务处理之用。

1a. 输入的优惠券不适用于所购商品：

1. 系统向收银员提示错误。

9a. 存在产品回扣：

1. 系统对每个具有回扣的商品给出回扣表单和票据。

9b. 顾客索要赠品票据（不显示价格）：

1. 收银员请求赠品票据，系统给出赠品票据。

9c. 打印票据。

1. 如果系统能够检测到错误，给出提示。

2. 收银员更换纸张。

3. 收银员请求打印其他票据。

特殊需求：

- 使用大尺寸平面显示器触摸屏UI。文本信息可见距离为1米。
- 90%的信用卡授权响应时间小于30秒。
- 由于某些原因，我们希望在访问远程服务（如库存系统）失败的情况下具有比较强的恢复功能。
- 支持文本显示的语言国际化。
- 在步骤3和步骤7中能够加入可插拔的业务规则。
- ……

技术与数据变元表：

*a. 经理超控需要刷卡（由读卡器读取超控卡）或在键盘上输入授权码。

3a. 商品ID可以用条码扫描器（如果有条形码）或键盘输入。

3b. 商品ID可以使用UPC（通用产品代码）、EAN（欧洲物品编码）、JAN（日本物品编码）或SKU（库存单位）等任何一种编码方式。

7a. 信用卡账户信息可以用读卡器或键盘输入。

7b. 记录在纸质票据上的信用卡支付签名。但我们预测，两年内会有许多顾客将希望使用数字签名。

发生频率：可能会不断地发生。

未决问题：

- 税法如何变化？
- 研究远程服务的恢复问题。
- 针对不同的业务需要怎样进行定制？
- 收银员是否必须在从系统注销后带走他们的现金抽屉？
- 顾客是否可以直接使用读卡器，还是必须由收银员完成？

此用例是示范性的，而并追求详尽彻底（尽管此例基于真实POS系统的需求——OO设计并使用Java开发）。然而在这里，此例已经足够详细和复杂，足以让我们体会到详述用例能够记录的大量需求细节。此例将能够成为解决众多用例问题的模型。

9. domain model，以及用到什么技术（确定名词短语。。。）

- **领域模型是对领域内的概念类或现实世界中对象的可视化表示。**领域模型也称为概念模型、领域对象模型和分析对象模型。它：
  - 说明问题领域中有意义的概念类
  - 是设计软件对象的灵感来源
- 应用UML表示法，领域模型被描述为一组没有定义操作（方法的特征标记）的类图，它提供了概念透视图，可以展示：
  - 领域对象或概念类
  - 概念类之间的关联
  - 概念类的属性
- **领域模型是OO分析中最重要和经典的模型。**
- UP对领域模型的定义是，可以在业务建模科目中创建的制品之一

- 如何找到概念类（Conceptual Classes）？
  - 三种方法：
    - i. 重用和修改现有的模型（首要、最佳且简单的方法，若条件许可通常从这一步开始）
    - ii. 使用分类列表

        - iii. 确定名词短语：语言分析 —— 对领域文本性描述中识别名词/名词短语，作为候选的概念类或属性
            - 语言分析：在对领域的文本性描述中识别名词和名词短语，将其作为候选的概念类或属性。

10. OOA, OOD, OOP的关联?

- **面向对象分析的结果可以作为开始面向对象设计的模型，面向对象的设计结果可以作为蓝图，利用面向对象编程方法最终实现一个系统。**
- **OOA**过程中，强调的是在问题领域内发现和描述对象（或概念）。
    - 如，在航班信息系统里包含飞机、航班和飞行员等概念。
- **OOD**过程中，强调的是定义软件对象以及它们如何协作以实现需求。
    - 如，软件对象飞机可以有tailNumber属性和getFlightHistory方法
- 最后，在**OOP**过程中，会实现设计出来的对象。
    - 如，Java中的Plane类。

期末里考过的期中的内容

1. RUP相关内容

- 什么是RUP?
    - 软件开发过程描述了构造、部署以及维护软件的方式。统一过程已经成为一种流行的构造面向对象系统的迭代软件开发过程。特别是Rational Unified Process（RUP）是对统一过程的详细精化，并且已经被广泛采纳。
- RUP包含六项最佳实践：
    - 迭代式软件开发
    - 需求管理
    - 基于构建的架构应用
    - 建立可视化的软件模型
    - 软件质量验证

- 软件变更控制

2. FURPS

- **FURPS+ 模型**（在UP中，需求按照FURPS+模型进行分类）
  - **功能性 Functional**：特性、功能、安全性
  - **可用性 Usability**：人性化因素、帮助、文档
  - **可靠性 Reliability**：故障频率、可恢复性、可预测性
  - **性能 Performance**：响应时间、吞吐量、准确性、有效性、资源利用率
  - **可支持性 Supportability**：适应性、可维护性、国际化、可配置性
  - **+**：一些辅助性的和次要的因素如下：
    - **实现 Implementation**：资源限制、语言和工具、硬件等
    - **接口 Interface**：强加于外部系统接口之上的约束
    - **操作 Operation**：对其操作设置的系统管理
    - **包装 Packaging**：例如物理的包装盒
    - **授权 Legal**：许可证或其他方式
  - **使用FURPS+分类方案作为需求范围的检查列表是有效的**
- **质量属性Quality attribute** 或 **质量需求Quality requirement**：
  - 包括：**可用性、可靠性、性能和可支持性。**

3. What solved in OOA?

- 在OOA的过程中，在问题领域内发现和描述了对象（或概念）
  - OOA识别了：
    - 问题领域中的概念
    - 这些概念之间的关系
    - 这些概念的属性

4. OOA的制品是什么?

- 制品是对所有工作产品的统称，如代码、web图形、数据库模式、文本文档、图、模型等。
- OOA的制品
    - 领域模型：现实世界中对象的概念透视图
    - 用例模型：描述功能需求，一组使用系统的典型场景
    - 补充性规格说明：描述其他需求，主要是非功能行需求
    - 词汇表：关键领域术语和数据字典
- OOD的制品
    - 静态模型： 类图、包图
    - 动态模型： 顺序图、通信图

5. operation contract

- 什么是**操作契约（contract）**？
    - 操作契约在系统操作执行之后，根据域模型中对象的状态更改来描述详细的系统行为
    - 该契约的主要输入为SSD中确定的系统操作、领域模型、领域专家的见解
    - 操作契约使用前置和后置条件的形式，描述领域模型里对象的详细变化，并作为系统操作的结果。
    - 操作契约可以视为UP用例模型的一部分，因为它对用例指出的系统操作的效用提供了更详细的分析。
- **契约有哪些部分?**
    - 操作：操作的名称和参数
    - 交叉引用：会发生此操作的用例
    - 前置条件：执行操作之前，对系统或领域模型对象状态的重要假设。应该告诉读者
    - 后置条件：最重要的部分。完成操作后，领域模型对象的状态
- 什么是**系统操作**?
    - 系统操作时是作为黑盒构件的系统在其公共接口中提供的操作，可以在绘制SSD草图时确定。

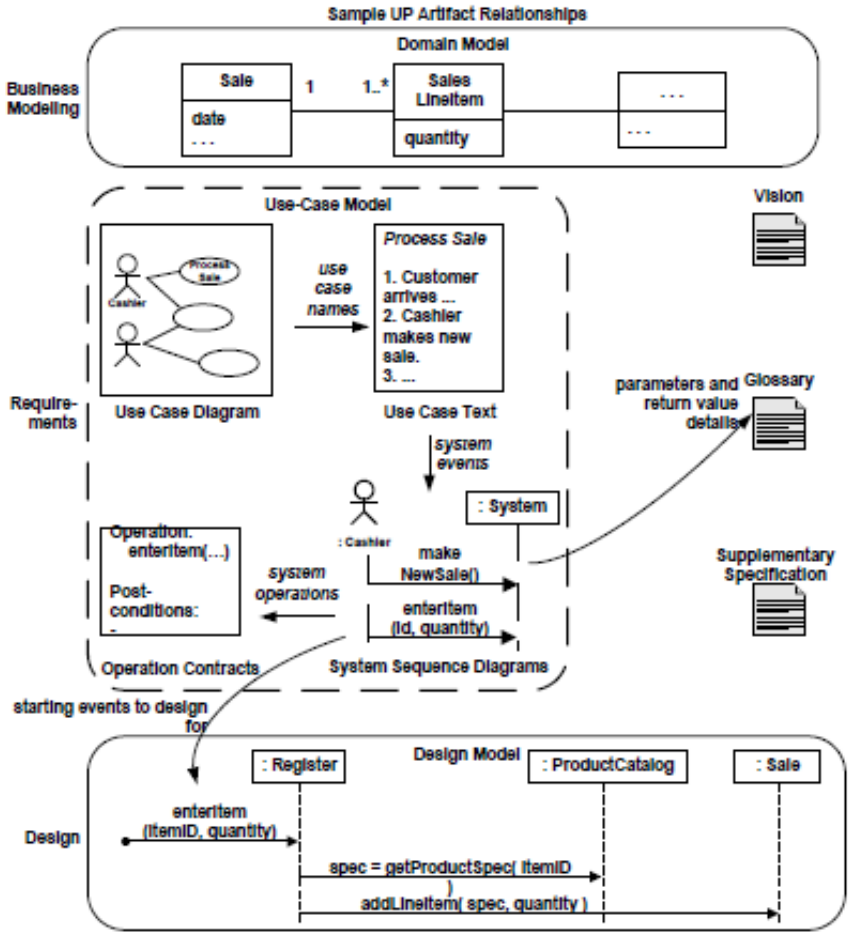
    - 更精确地讲，SSD展示了**系统事件**（涉及系统的事件或I/O消息）。输入的系统事件意味着系统具有用来处理该事件的系统操作

- **契约在何时有效?**
    - 用例是项目需求的主要知识库
    - 契约只在用例中所需状态变化的细节和复杂性难以处理时才需要
- **如何创建和编写契约?**
    - 从SSD中确定系统操作
    - 如果系统操作复杂，其结果可能不明显，或在用例中不清楚，则可以为其构造契约
    - 使用以下几种类别来描述后置条件：
        - 创建和删除实例
        - 修改属性
        - 形成和清楚关联

6. Please give the benefit of drawing system sequence diagram—SSD

- 绘制系统时序图的好处：
    - SSD使事件变得更加具体和明确；
- 绘制的动机
    - SSD表示的是，对于用例的一个特定场景，外部参与者产生的时间，其顺序和系统之内的事件。
    - 要对系统行为进行分析，就要准确地知道什么是系统事件。
    - 系统行为描述的是系统做什么，这种描述的一部分就是SSD。

7. Explain the diagram（20分）（在书上找找这个图的完整版）



版本一：

①业务建模阶段---建立领域模型：

sale表示了销售交易过程，其中包括日期date属性

SalesLineItem表示具体的购买商品条目，其中包括此商品的数量quantity属性

Sale与SalesLineItem存在一对多的关联关系，即一次销售过程有许多购买条目。

②需求分析阶段---建立用例模型：

1) 建立用例图：系统与环境通过外框分开，参与者cashier有processSale的用例行为。

2) 建立用例文本：包括一个processSale用例，其中描写了一系列的（主成功或者拓展）场景，如Customer arrives等

3) 建立系统顺序图：描述了某场景的系统事件，外部参与者cashier向系统发出不带参数的makenewsale()请求和一个带有参数的enterItem(id,quantity)请求。

4) 建立操作契约：描述了系统顺序图中某系统操作enteritem（）的详细操作契约，包括名称，交叉引用，前置条件，后置条件等。

5) Vision设想：范围、目标、参与者、特性

6) Glossary词汇表：术语、属性、验证

7)补充规格说明：非功能性需求、质量属性

③设计阶段：

图中只有系统时序图：外部向register对象发送enteritem（）请求希望输入销售商品条目，register收到后，　　调用productCatalog的getProductDescription（）方法并获得返回值用来确定商品的详细信息，接着调用sale　　类的addlineitem让其处理此次输入的商品条目并加入销售清单。这样，一次商品添加过程完成。

版本二：

用例图会为用例文本提供用例名称

用例文本会为SSD提供系统事件

SSD会为操作契约提供系统操作

用例文本会为操作契约提供后置条件的想法

用例文本会为领域模型提供概念（对象，属性和关联）

用例模型给vision提供（scope,goal,参与者和特性）：用例文本里的内容

用例文本给Glossary提供术语，属性和验证

领域模型给Glossary提供某些领域模型术语的细化

SSD会给Glossary提供参数和返回值的细节

领域模型-》操作契约：经历状态变化的对象，属性和关联

用例模型会给补充性规格说明提供非功能性需求和质量属性

领域模型里的概念类会给设计模型里软件类的名称带来灵感

Glossary->设计模型：Item细节，格式和验证

补充性规格说明->设计模型：非功能性需求和领域规则

用例文本->设计模型：软件必需满足的功能性需求

SSD->设计模型：用于设计的启动事件（系统操作成为领域层交互图控制器对象的起始消息，所以是设计协作对象的起点）

操作契约->设计模型：用于设计的启动事件，以及软件必需满足的详细后置条件

附：补充性规格说明->包图：非功能性需求会影响系统的逻辑体系结构，设计模型包括包图，DCD和交互图

版本三：

用例模型：用例名称：处理销顾客携带所购商品到达收银台，收银员使用POS系统记录每件商品，系统连续显示累计总额，并逐行显示细目。顾客输入支付信息，系统对支付信息进行验证和记录。系统更新库存信息。顾客从系统得到购物小票，然后携带商品离开。

领域模型：（展示了：领域对象或概念类；概念类之间的关联；概念类的属性）表明了Sale和Sales LineItem领域的重要概念类，Sale和Sales LineItem之间具有关联并加以有意义的标记，Sale具有date属性，Sales LineItem具有quality属性信息，并且他们之间是1对1的关系。

设计领域：职责、角色、写作驱动了设计。

Register通过调用ProductCatalog中的getProductDescription函数，从ProductCatalog处获取了商品的ID，通过调用addLineItem函数，将产品的ID以及产品的量发送给Sale。Register自身创建了makeNewSale与enterItem函数，收银员可以利用enterItem函数将信息传递给Register。

补充型规格说明：用力之外的内容，主要用于非功能需求。

词汇表：以最简单的形式定义重要的术语。

此用例文本使用了词汇表中定义的重要术语。关联到系统事件，从而触发系统操作。

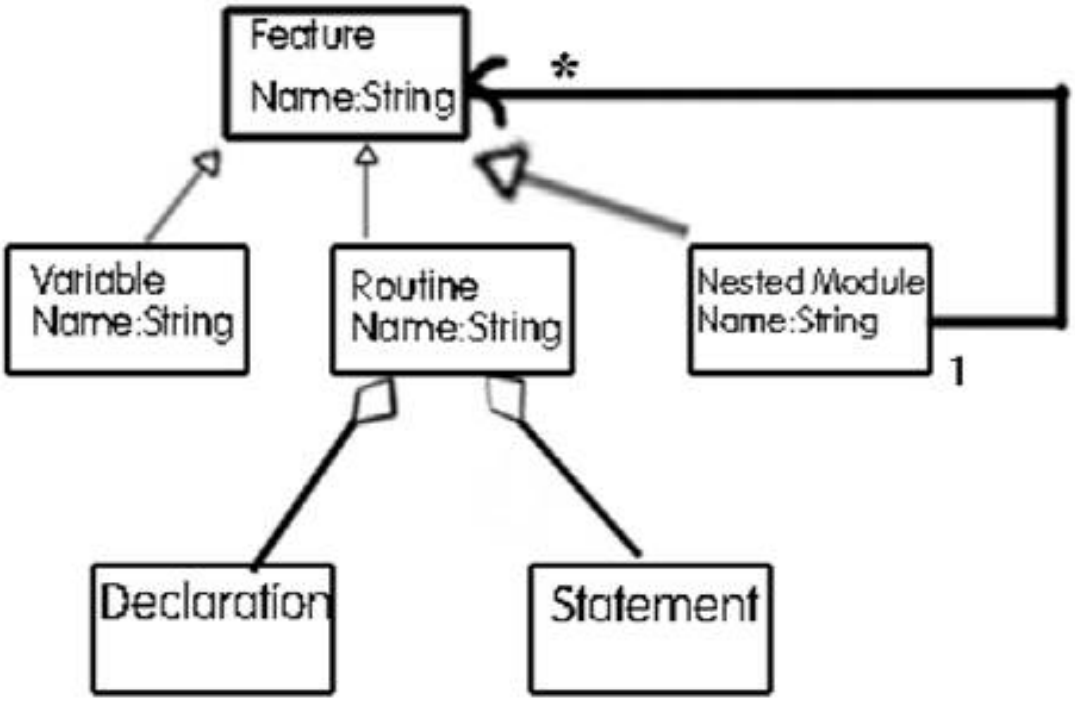
8. 画class diagram，一个实例

根据下面的描述画出类图（我只是尽量回忆啊= =），下面的描述是一种编程语言。

一个module中可以有很多个feature，feature可以包括variable, routine, 和嵌套的module。一个routine中包含一个declaration和一个statement

一个module有很多feature，每个feature可能是variable,也可能是routine,还有可能是nested module。一个routine由declaration part和statement part组成。画出类图。

Give you a module maked up by some named features, a feature may be a variable/routine/nested module, and the routines include two parts ,they are declaration partand statement part.


Feature
Name:String
*
Variable
Name:String
Routine
Name:String
Nested Module
Name:String
1
Declaration
Statement