Rose实验报告

Rational rose是一个面向对象的可视化建模软件。语言是UML

支持多种语言，支持c++、java、vb等语言，此次实验中使用java。

Rose支持四种视图：

1）用例视图：use case view

2）逻辑视图：logical view

3）组件视图：component view

4）配置视图：deployment view

1、用例视图

用例视图可以帮助你 理解和使用系统，这个视图展示了系统和用例是怎么相互作用的。它包括系统中所有的角色，用例，use case视图是系统中与现实无关的视图，关注系统功能的高层形状，而不关心系统的具体实现方法。

这个视图默认生成一个名为main的Use Case图表。在这个视图，用户可以凭借图形直观的了解到用例之间的作用关系。

2、逻辑视图：

我认为逻辑视图和用例视图不同，它更加关注系统如何实现项目的功能，呈现项目的详细图形，各个组件间如何关联。和它的名字一样，注重的是项目的逻辑结构，在这个视图中要标识系统组件，检查系统的信息和功能，检测组件之间 的关系。这里，重复使用是一个主要目的，通过认真指定类的信息和行为，组合类，以及检查类和包之间的关系，就可以确定重复使用的类和 包，完成多个项目后，就可以将新包和新类加进重复使用库中，以后的 项目就可以组装现有的类和包。

我们可以在这个视图下建立Class digram（类图）、Statechart digram（状态图）Sequence Diagram（顺序图）、Collaboration Diagram（协作图）、Activity Diagram（活动图）。

这个视图也有默认生成的main ，效果一致。

3、组件视图：

组件视图以代码的形式显示不同模块间的关系。我们通过组件视图可以知道哪些组件是代码库，哪些组件是运行组件。

Component只包括Component（组件图）。

Component视图的main主要是用户用来控制代码的。

4、部署视图：

部署视图视图显示的进程和设备及其相互间的实际连接，它关注系统的实际部署，比如部署可能分成两层，界面的部署放在一台机器上，而业务和数据库逻辑的部署放在另外一台机器上。

部署视图只包括Deployment视图。

除了四种视图，Rose还介绍了九种图：

用例图：Use Case Diagram

类图：Class Diagram

对象图：Object Diagram

状态图：State Diagram

顺序图：Sequence Diagram

协作图：Collaboration Diagram

组件图：Component Diagram

部署图：Deployment Diagram

活动图：Activity Diagram

空心的三角表示继承关系（类继承）

有一个菱形（空心）表示聚合（aggregation）

菱形为实心的，它代表了一种更为坚固的关系——组合（composition）（聚合类型为复合）。

类之间的关系：

继承、关联、聚合、依赖

(1) 类之间的关系在工具栏中显示。

(2) 对于关联关系来说，双击关联关系，就可以在弹出的对话框中对关联的名称和角色进行编辑。

(3) 编辑关联关系的多重性：右单击所要编辑的关联的一端，从弹出的菜单中选择Multiplicity，然后选择所要的基数

关系：关联、限定关联、泛化、依赖、继承、实现

本次熟悉了Rose的各类图后，还实践了正向工程和反向工程。

正向工程：

我们基于tank大战的思维，利用类和用例等工具将目标项目以uml图的方式先存储在一个Rose的项目里，添加属性和方法，然后通过generate code ，可以生成自己想要的代码，

老师为了让我们熟悉rose，先做了一遍正向工程，然后做了一遍反向工程。但是由于每个人画的图不一样，结构不一样，那么最后生成的代码也不一样。那么在做反向工程的时候就会有偏差，为了统一进度，老师事先准备了一份代码，这样就保证最后的结果一致同时可以检验对错（由于版本等一些问题，代码需要修改才可使用）。  
反向工程：

反向就是和正向相反，从代码生成我们想要的类，我们选择的是java语言。首先我们需要编辑类路径Edit CLASSPATH，把我们准备的代码和Android包导入进去。然后选择Java/J2EE下的[Reverse Engneer，找到所有的.java文件 ，全部add，然后select all，reverse，全部反向。想查看结果的话，打开main然后在query一栏下点击add classes就可以。

安装完可能无法启动，需要在环境变量里面自己配置一下，就是把common改成Common