(1)模块概述

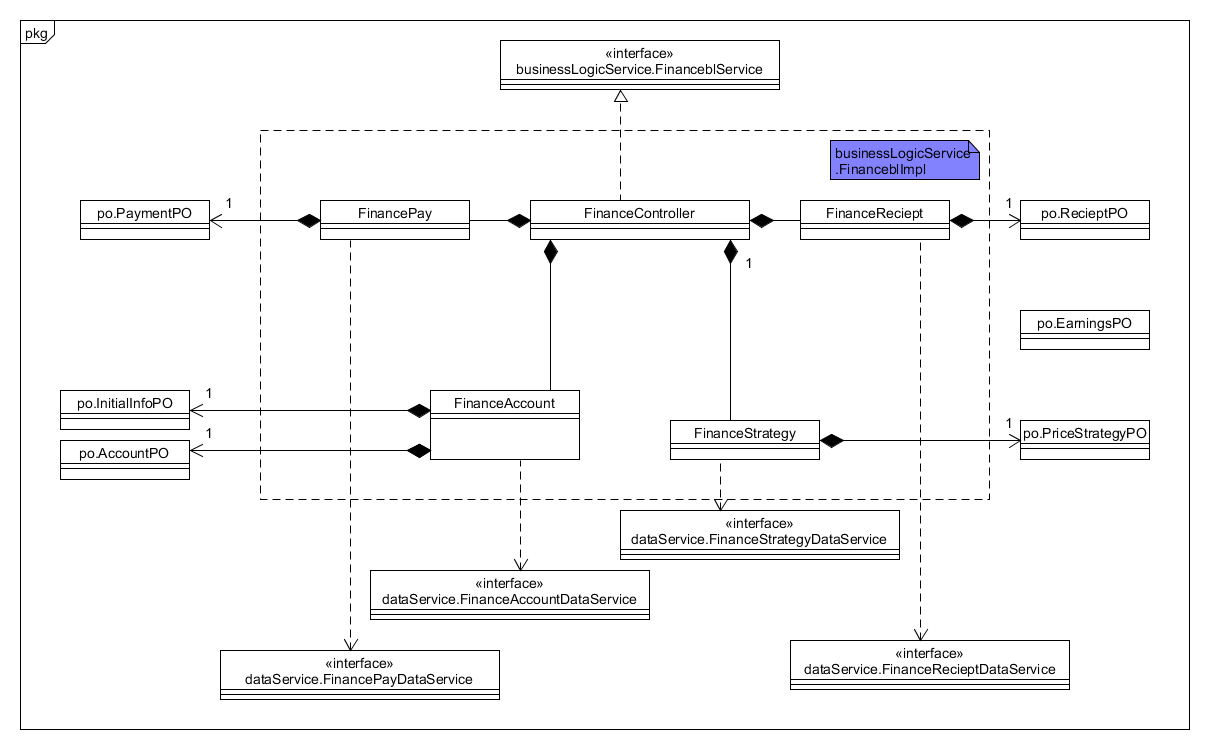
Financebl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Financebl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

根据体系结构的设计，系统分为展示层、业务逻辑层和数据层，并在展示层和业务逻辑层之间添加bussinssFinanceicServi ce.FinanceblService接口、在业务逻辑层和数据层之间添加dataService.FinanceDataService接口，从而增加了系统的灵活性。我们增加了FinanceController从而隔离了业务逻辑职责和逻辑控制职责，这样FinanceController会将业务逻辑处理委托给Financebl对象。FinancePO作为用户信息的持久化对象被添加到设计模型中去的。FinanceImpl用来进行对管理用户操作时产生的系统日志的记录。

Financebl模块的设计如图所示。



Financebl模块的各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| FinanceController | 负责实现付款、收款、账户、策略四大模块的协调与调度 |
| FinancePay | 负责实现付款单建立及付款所需要的服务 |
| FinanceAccount | 负责实现账户的添加，删除，修改，查找和添加期初建账信息所需要的服务 |
| FinanceStrategy | 负责实现价格策略、薪水策略的制定和修改所需要的服务 |
| FinanceReceipt | 负责实现收款单建立及收款所需要的服务 |

(3)模块内部类的接口规范

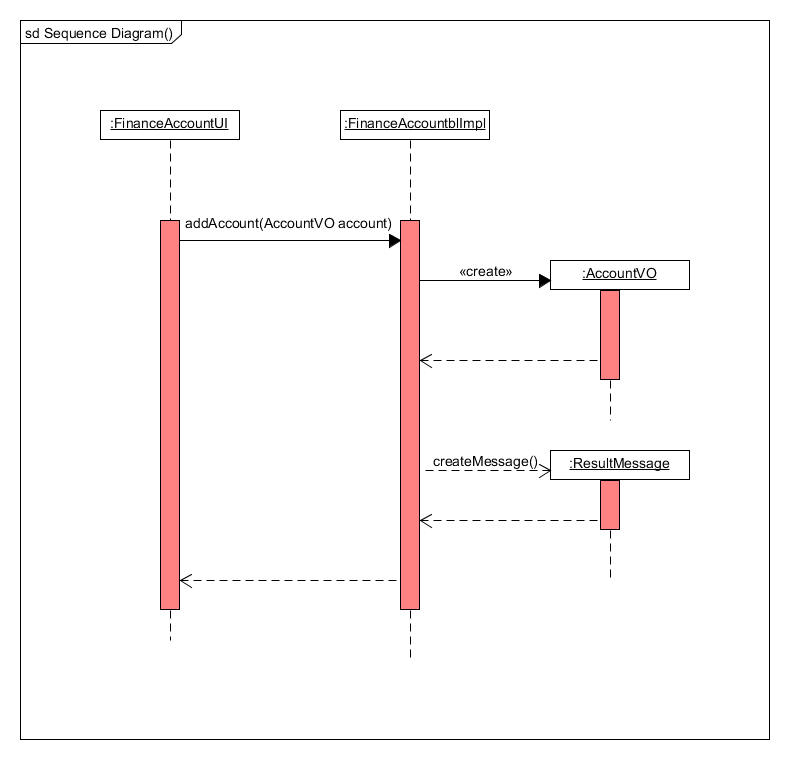
FinanceController和Financebl的接口规范如下表所示。

Finance的接口规范

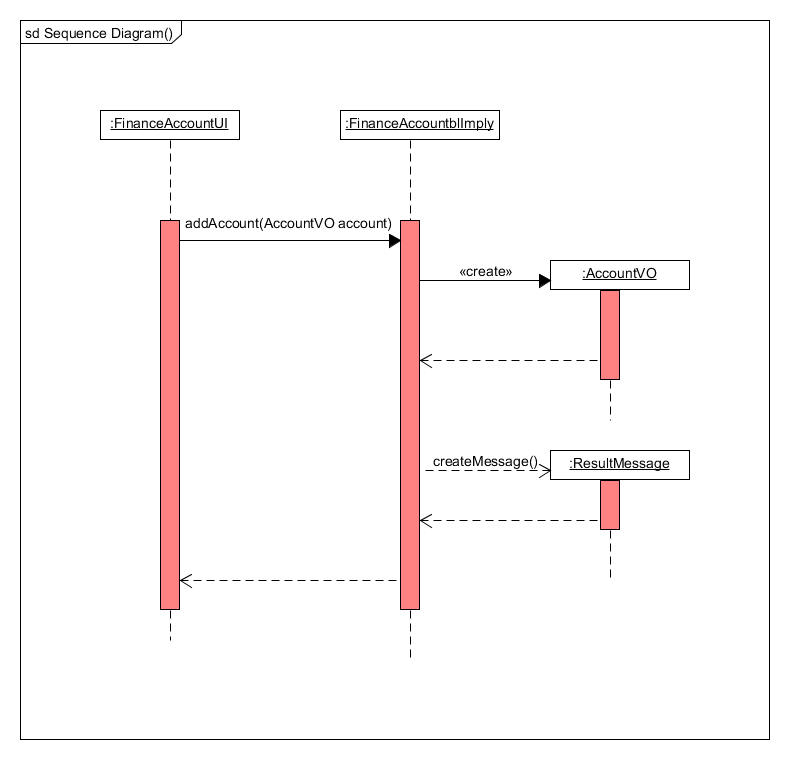
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Finance. checkUser | 语法 | public boolean isValid (UserVo user); |
| 前置条件 | User的工号不为null |
| 后置条件 | 如果该财务人员具有高级权限，返回true，否则返回false |
| Finance. createDebitnote | 语法 | public DebitVO createDebitNote(DebitVO debit) |
| 前置条件 | Debit符合规范 |
| 后置条件 | 返回给界面完整的付款单信息 |
| Finance. saveDebitnote | 语法 | public Boolean saveDebitnote(DebitVO debit) |
| 前置条件 | 无 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Finance. show Debitnote | 语法 | public ArrayList<debitnoteVO> showDebitnote (Calendar date, long Finance) |
| 前置条件 | Finance编号存在 |
| 后置条件 | 返回符合该时间段内所有收款单信息 |
| Finance.getDebitSum | 语法 | public long getDebitSum(Calendar date) |
| 前置条件 | debitnote 不为空 |
| 后置条件 | 返回该列表中收款单的收款总和 |
| Finance. show DebitnoteList | 语法 | public ArrayList<debitnoteVO> showDebitnoteList (Calendar date) |
| 前置条件 | 无 |
| 后置条件 | 返回该时间段各个营业厅的收款单 |
| Finance. showRent | 语法 | public RentVO showRent(int sum, int year) |
| 前置条件 | 无 |
| 后置条件 | 返回完整的租金付款单信息 |
| Finance. saveRent | 语法 | public boolean saveRent(RentVO rent) |
| 前置条件 | 已确认该付款单 |
| 后置条件 | 结束此次管理，持久化更新设计的领域对象的数据 |
| Finance. showFreight | 语法 | public FreightVO showFreight() |
| 前置条件 | 无 |
| 后置条件 | 返回当前的运费付款单 |
| Finance. changeFreight | 语法 | public boolean ­­changeFreight(int freight) |
| 前置条件 | 无 |
| 后置条件 | 如果修改成功，返回true，否则返回false |
| Finance. saveFreight | 语法 | public boolean saveFreight(FreightVO freight) |
| 前置条件 | 已确认运费付款单 |
| 后置条件 | 结束此次管理，持久化更新涉及的领域对象的数据 |
| Finance. showSalary | **语法** | public pesonVO showSalary(long person) |
| 前置条件 | 员工id符合输入规则 |
| 后置条件 | 返回该员工的工资信息 |
| Finance. showSalarySum | 语法 | public SalaryVO showSalarySum(int year, int month) |
| 前置条件 | 无 |
| 后面条件 | 返回该月的工资付款单信息 |
| Finance. showEarning | 语法 | public EarningVO showEarning() |
| 前置条件 | 无 |
| 后置条件 | 返回截止到当前日期的经营情况表 |
| Finance. exportEarning | 语法 | public Boolean saveEarning(EarningVO earnings) |
| 前置条件 | 无 |
| 后置条件 | 如果导出成功，返回true，否则返回false |
| Finance. addAccount | 语法 | public Boolean addAccount(AccountVO account) |
| 前置条件 | AccoutVO不为null |
| 后置条件 | 如果增加成功，返回true， 否则返回false |
| Finance. showAccount | 语法 | public AccountPO showAccount(long id) |
| 前置条件 | 输入的id符合规范 |
| 后置条件 | 返回查询的账户信息 |
| Finance. deleteAccount | 语法 | public Boolean deleteAccount(long id) |
| 前置条件 | 输入的id符合规范 |
| 后置条件 | 如果删除成功返回true，否则返回false |
| Finance. changeAccount | 语法 | public Boolean changeAccount(AccountVO account) |
| 前置条件 | Account不为null |
| 后置条件 | 如果修改成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| FinanceDataService  .addReceipt(ReceiptPo Receipt): boolean | 增加收款单的单一持久化对象 | |
| FinanceDataService  .findReceipt(Calendar date):List<ReceiptPo> | 根据输入的时间查找收款单的单一持久化对象，返回所有符合条件的收款单的列表 | |
| FinanceDataService  .deleteReceipt(long id):boolean | 根据输入的id查找收款单并删除该单一持久化对象 | |
| FinanceDataService  .updateReceipt(ReceiptPo receipt):boolean | 根据输入的收款单id查找收款单并用传入的收款单替换该单一持久化对象 | |
| FinanceDataService  .addPayment(PaymentPo payment):Boolean | 增加新的付款单单一持久化对象 | |
| FinanceDataService  .findPayment(Calendar date):List<PaymentPo> | 根据输入的date查找符合条件的付款单单一持久化对象 | |
| FinanceDataService  .deletePayment(long id):boolean | 根据输入的id查找到相应的单一持久化对象并删除 | |
| FinanceDataService  .updatePayment(PaymentPo Payment):boolean | 根据传入的PaymentPo的id查找到该单一持久化对象并替换 | |
| FinanceDataService  .addFreight(FreightPo freight):boolean | 增加新的freight单一持久化对象 | |
| FinanceDataService  .findFreight(Calendar date):List<FreightPo> | 根据输入的日期查找符合条件的单一持久化对象 | |
| FinanceDataService  .deleteFreight(long id):boolean | 按id查找相应的FreghtPo结果并删除 | |
| FinanceDataService  .updateFreight(FreightPo freight):boolean | 查找到相应的FreightPo并用传入的Freight替换 | |
| FinanceDataService  .findEarning(Calendar date):List<EarningPo> | 按date进行查找返回符合条件的成本收益表单一持久化对象 | |
| FinanceDataService  .deleteEarning(long id):boolean | 按id查找相应的持久化对象并删除 | |
| FinanceDataService  .updateEarning(EarningPo earning) | 查找到相应的EarningPo并用传入的Earning替换 | |
| FinanceDataService  .addAccount(AccountPo account):boolean | 增加新的银行账户单一持久化对象 | |
| FinanceDataService  .findAccount(long id):AccountPo | 按id进行查找返回相应的AccountPo结果 | |
| FinanceDataService  .deleteAccount(long id):boolean | 按id查找到相应的AccountPo单一持久化对象结果并删除 | |
| FinanceDataService  .updateAccount(AccountPo account) | 按id查找到AccountPo的单一持久化对象并替换 | |

(4)业务逻辑层的动态模型

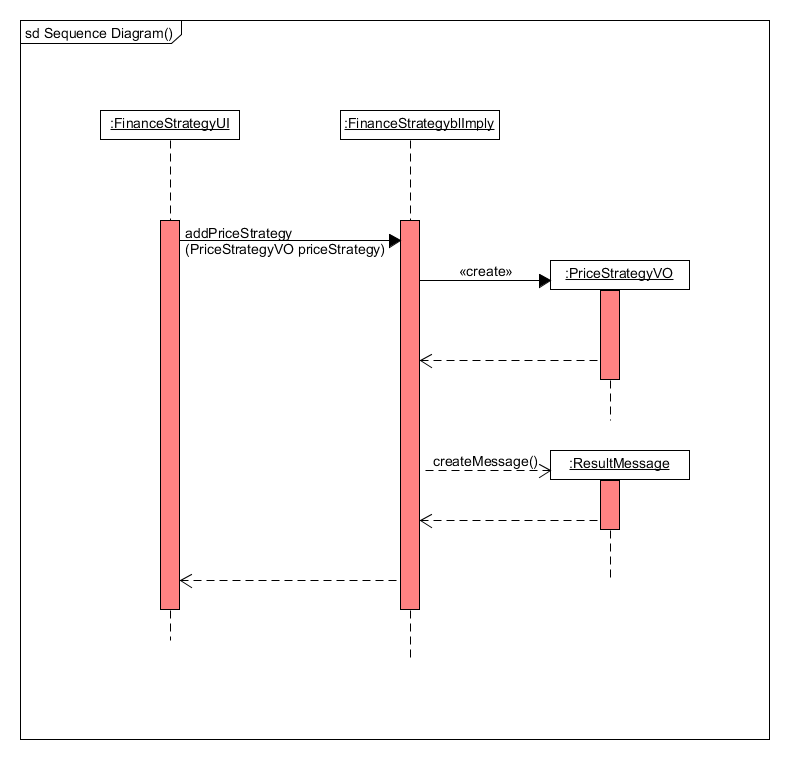
下图表示了物流管理系统中，当用户选择添加账户时，Financebl处理的相关对象的协作。



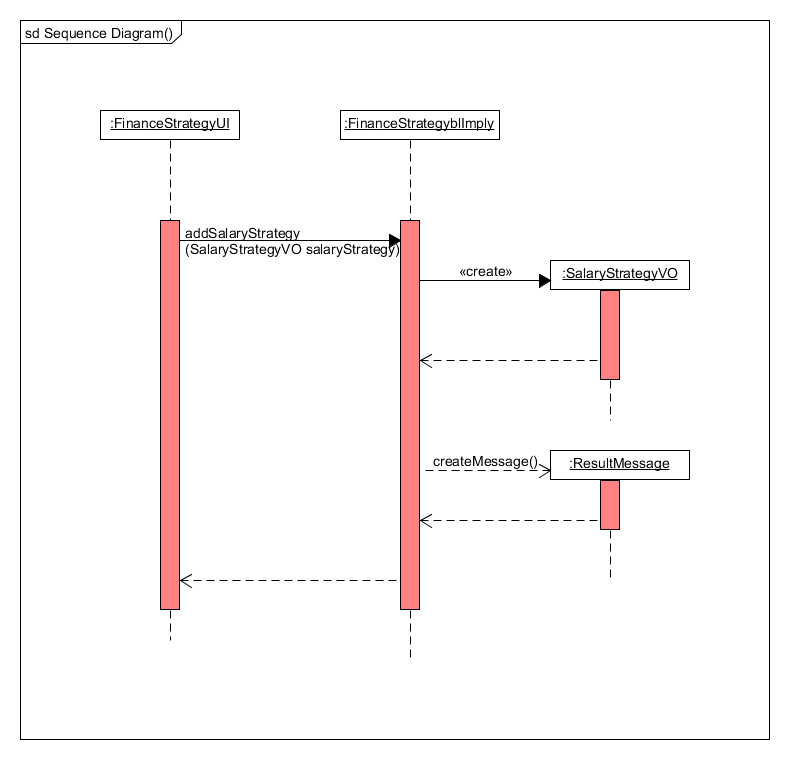
下图表示了物流管理系统中，当用户添加期初建账信息时，Financebl处理的相关对象的协作。



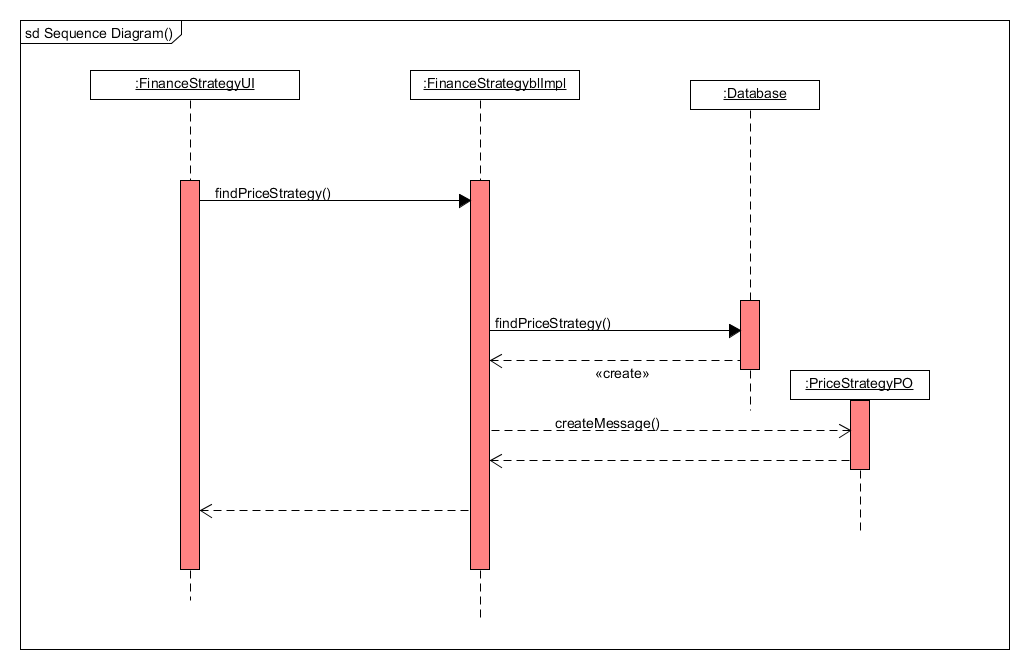
下图表示了物流管理系统中，当用户选择添加价格策略时，Financebl处理的相关对象的协作。



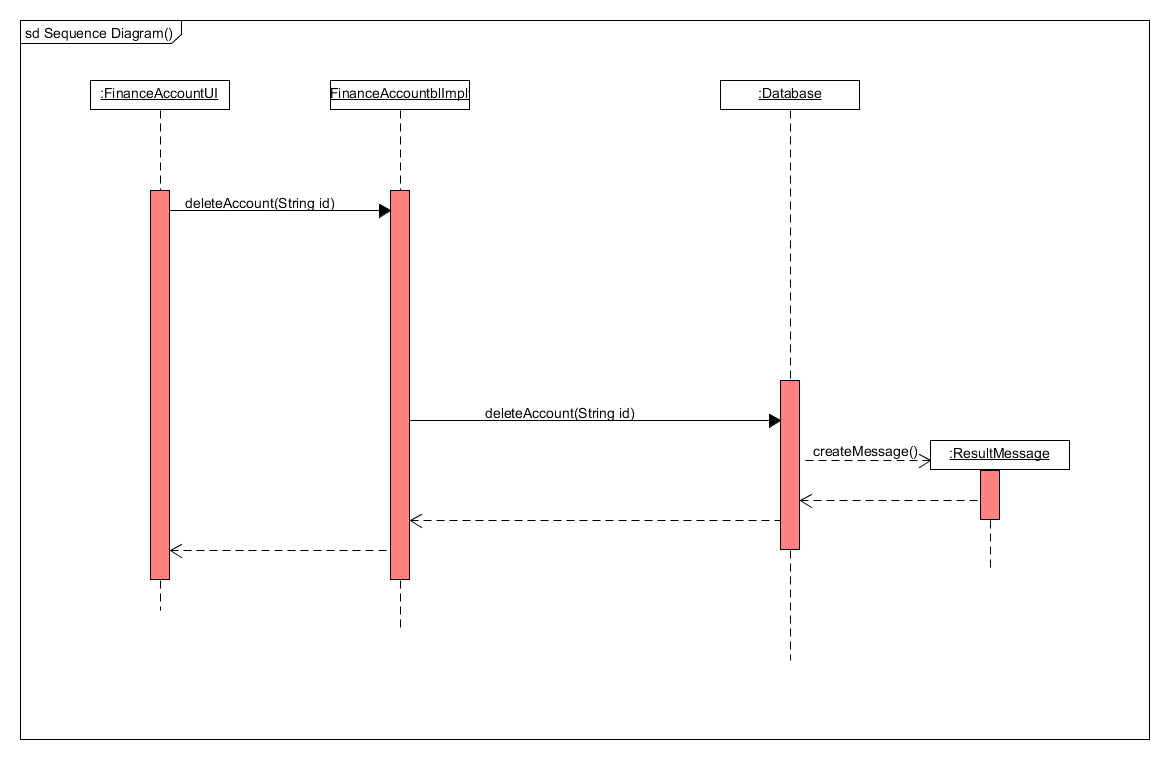
下图表示了物流管理系统中，当用户选择添加薪水策略时，Financebl处理的相关对象的协作。



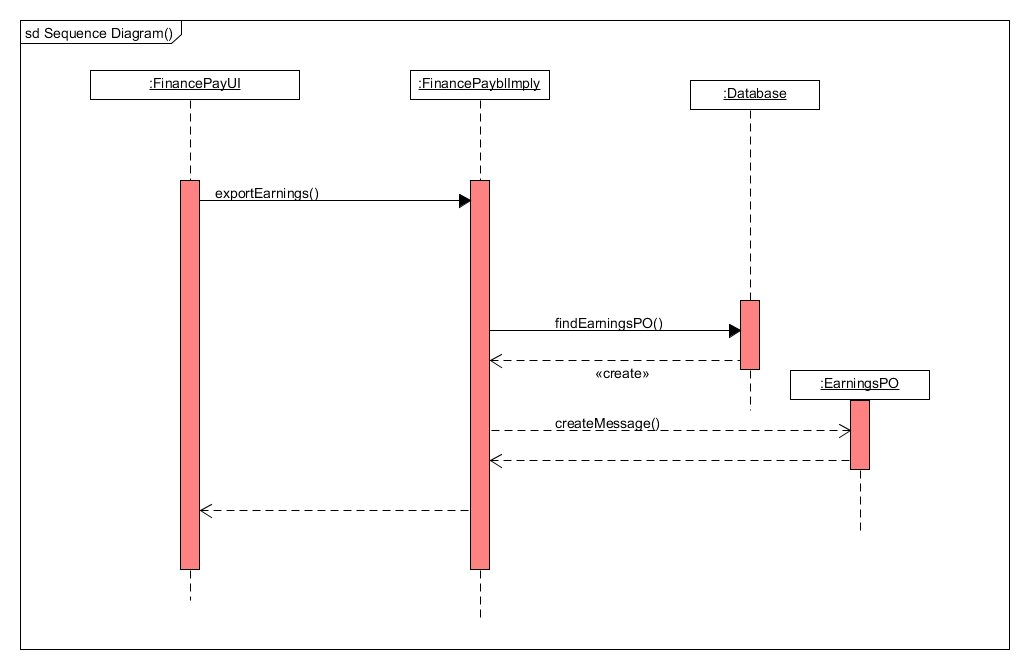
下图表示了物流管理系统中，当用户选择创建收款单时，Financebl处理的相关对象的协作。



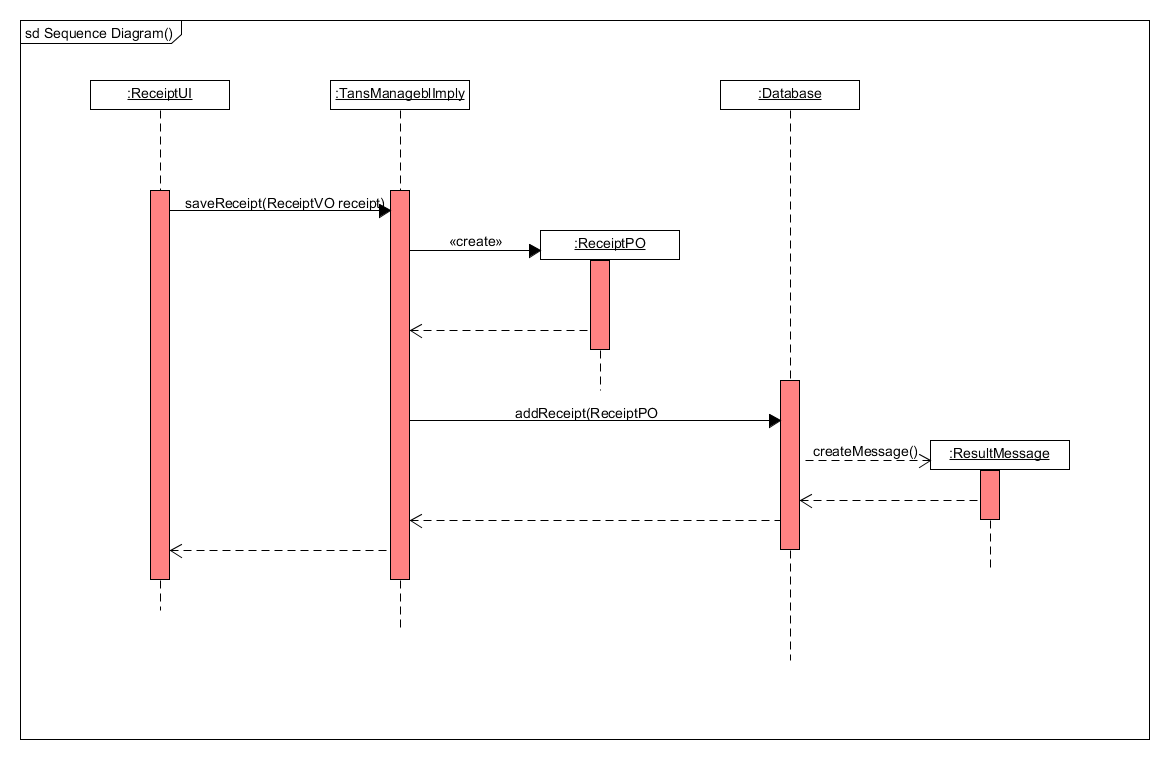
下图表示了物流管理系统中，当用户选择删除账户时，Financebl处理的相关对象的协作。



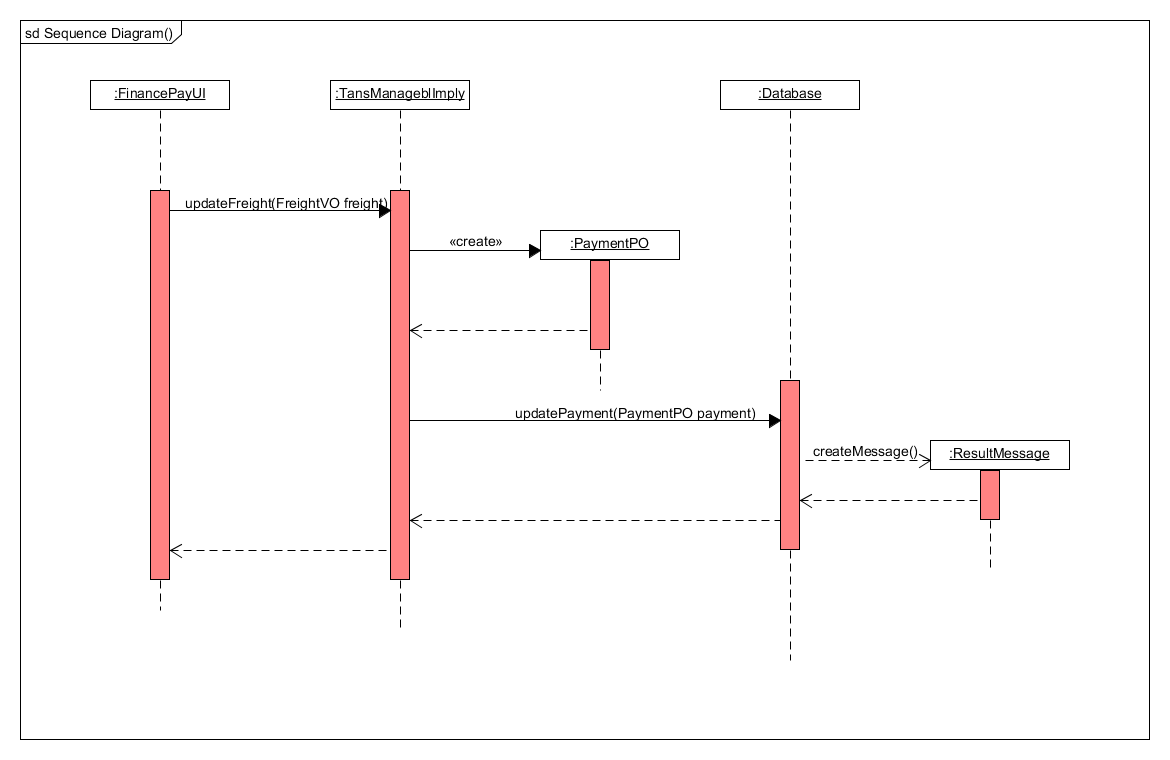
下图表示了物流管理系统中，当用户选择导出成本收益表时，financebl处理的相关对象的协作。



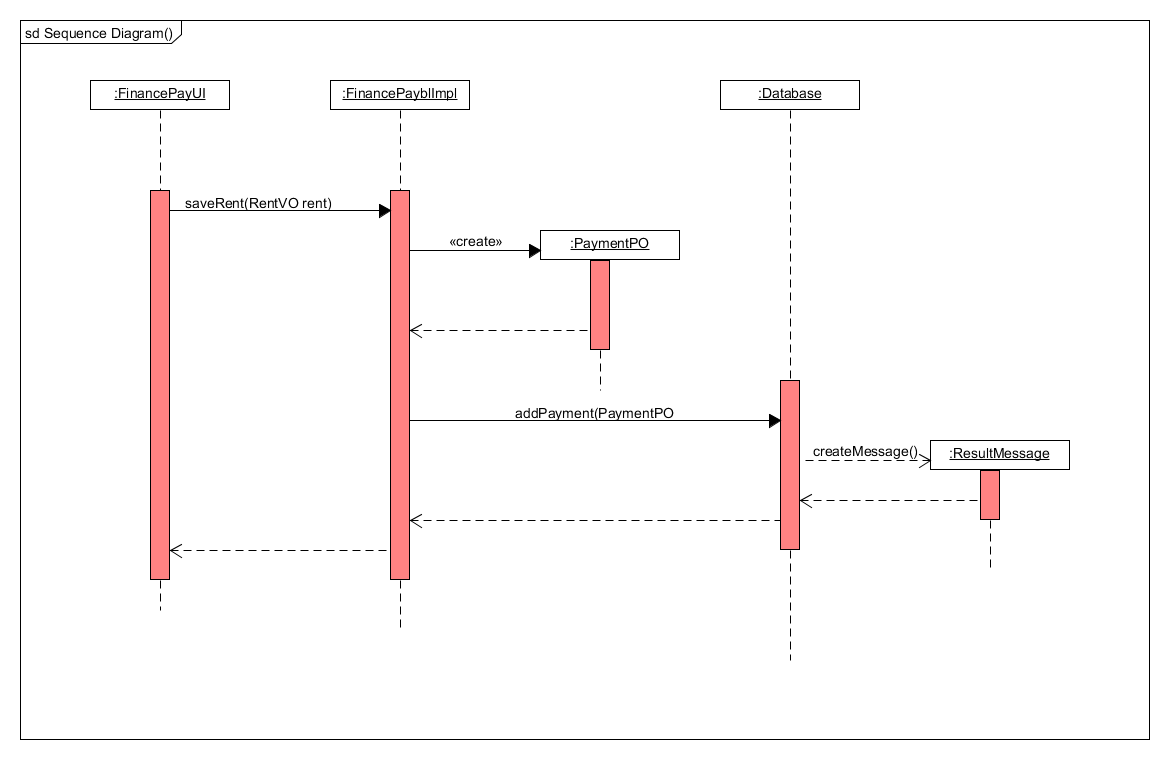
下图表示了物流管理系统中，当用户选择合计收款单时，Financebl处理的相关对象的协作。



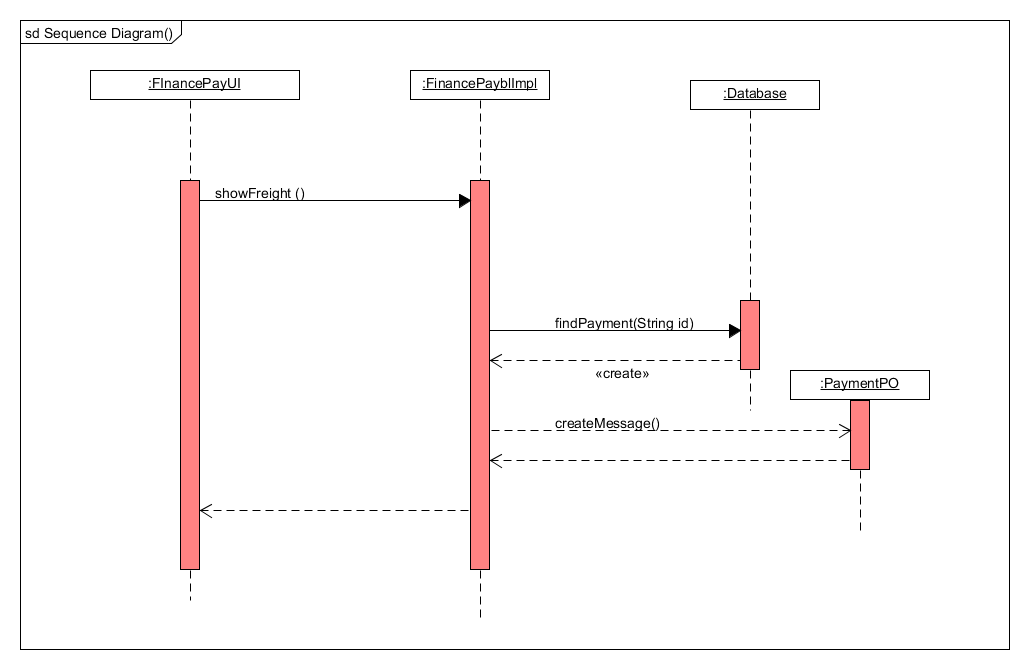
下图表示了物流管理系统中，当用户选择修改账户时，Financebl处理的相关对象的协作。



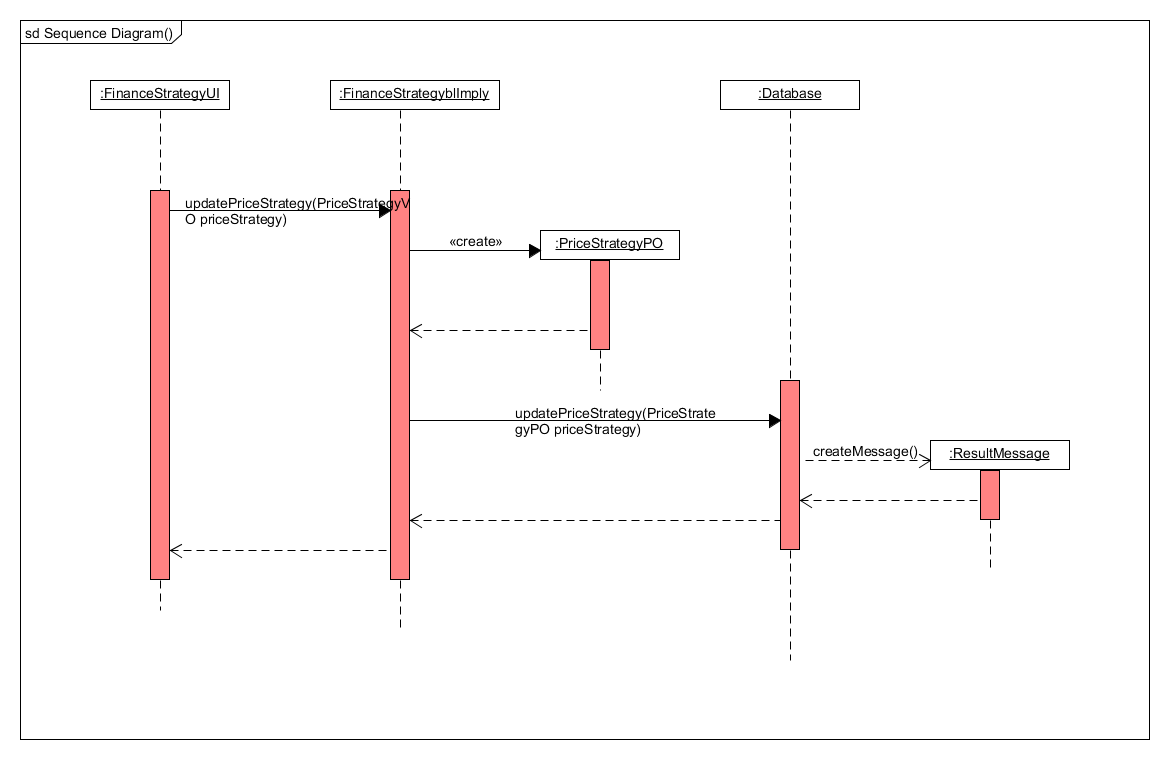
下图表示了物流管理系统中，当用户选择修改租金时，Financebl处理的相关对象的协作。



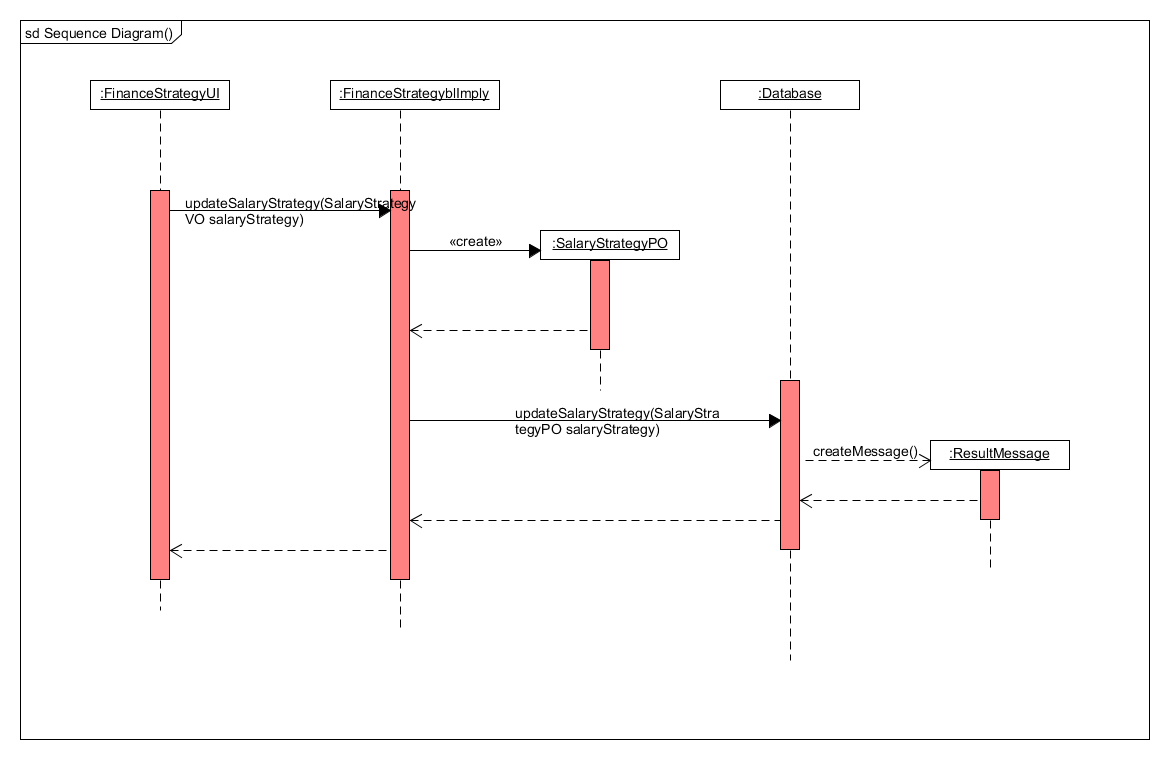
下图表示了物流管理系统中，当用户选择查看成本收益表时，Financebl处理的相关对象的协作。



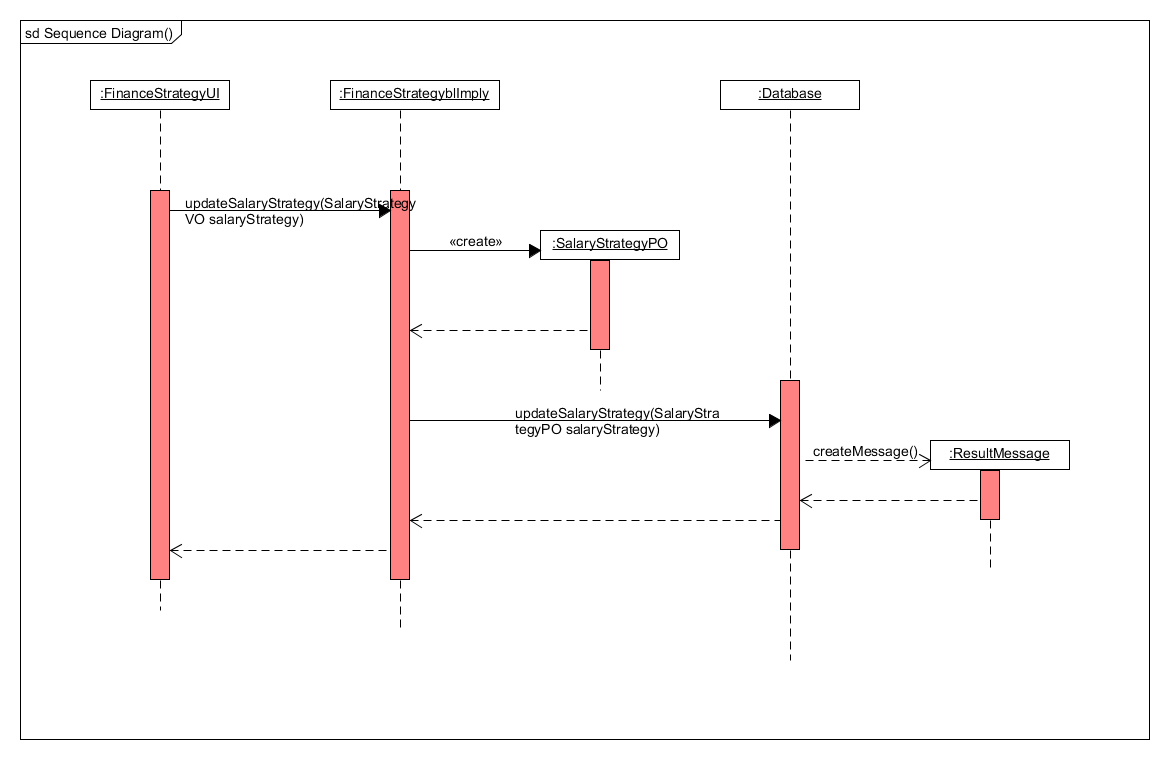
下图表示了物流管理系统中，当用户选择修改运费时，Financebl处理的相关对象的协作。



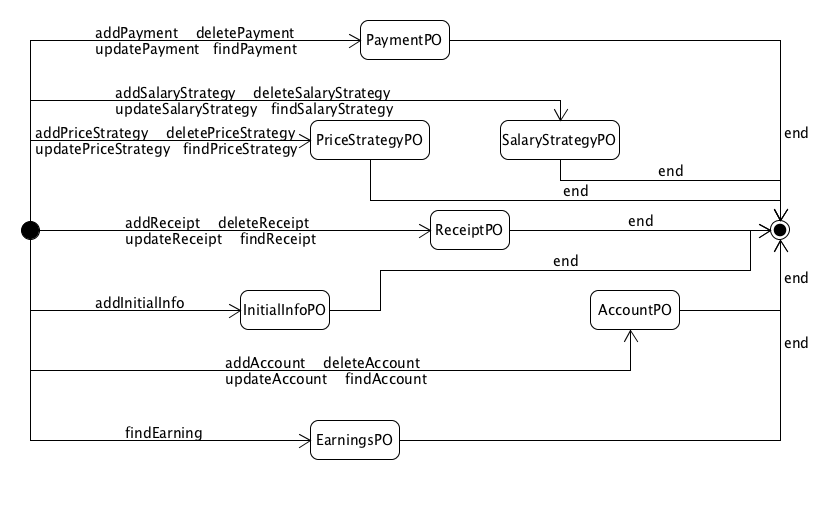
下图表示了物流管理系统中，当用户选择修改价格策略时，Financebl处理的相关对象的协作。



下图表示了物流管理系统中，当用户选择修改薪水策略时，Financebl处理的相关对象的协作。



如下图所示的状态图描述了Financebl对象的生存期内的状态序列、引起转移的事件以及因状态转移而伴随的动作。



(5)业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。