4.给出一个用原型开发但并没有节省大量开发时间的例子

高精运算的软件，各个运算逻辑紧密耦合，难以使用原型开发进行分块原型化，整体原型化又会使原型过程过于冗长

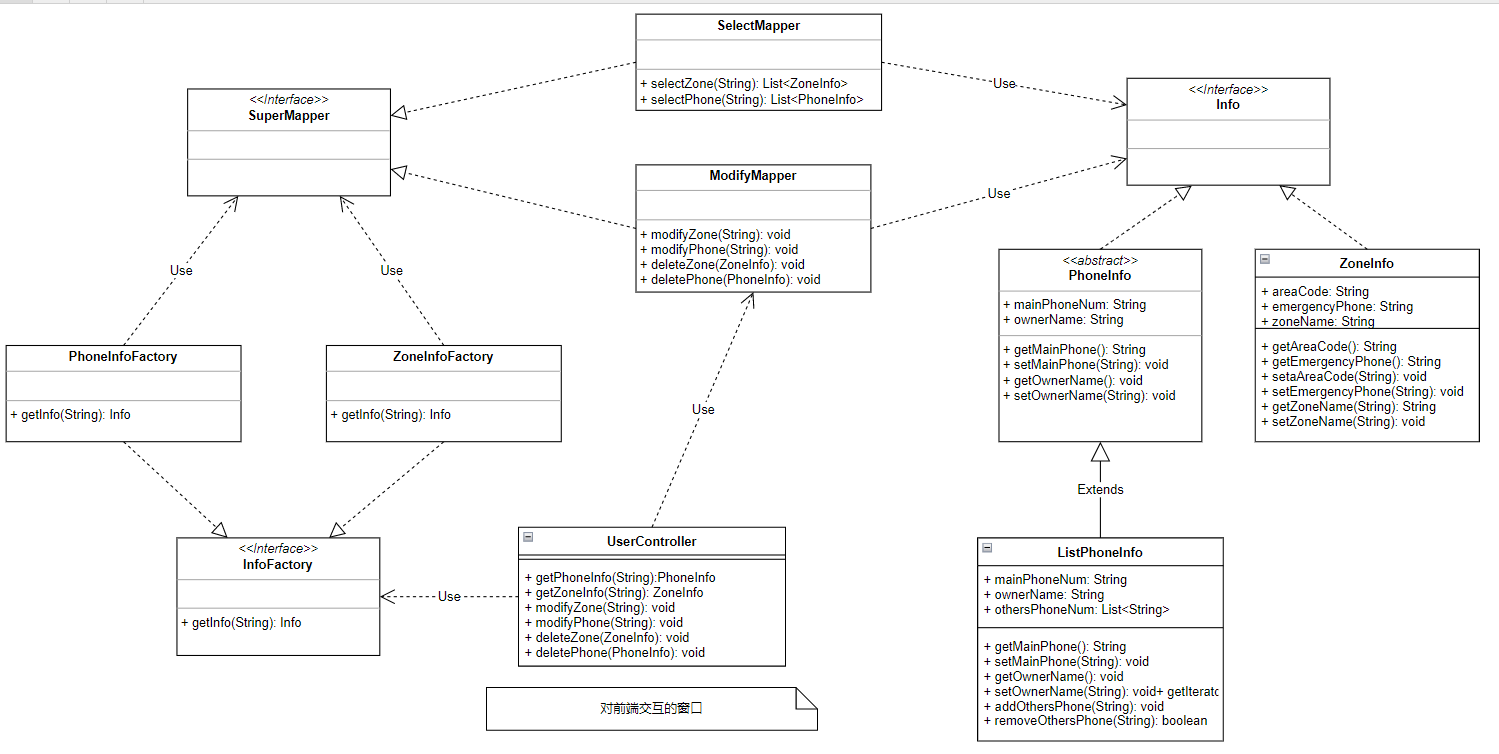
5.列举出最适合原型设计的系统特征

- 系统可以被分为比较小的模块，分开攻坚

- 系统有不确定之处，比如有多种方案需要权衡，或有未确定的功能需要测试

12续

分块介绍类的设计，和最开始的类图已经有不同了



DAO：

<<Interface>>SuperMapper：没有实际内容，所有DAO层对象的接口，我认为各个Mapper间可能会在将来有需要实现的共性，所以预先设计了

SelectMapper implements SuperMapper：用于实现各种查询方法，不允许有任何修改操作

方法：

+ selectZone(String): List<ZoneInfo>

选择区域信息

+ selectPhone(String): List<PhoneInfo>

选择电话信息

ModifyMapper implements SuperMapper：用于实现各种修改方法，不允许有任何查询操作

方法：

+ modifyZone(ZoneInfo): void

修改区域信息，如果不存在则新增，存在则修改

+ modifyPhone(PhoneInfo): void

修改电话信息，逻辑同上

+ deleteZone(ZoneInfo): void

删除区域信息

+ deletePhone(PhoneInfo): void

删除电话信息

POJO：

统一使用lombok，具有get、set等方法

<<Interface>>Info：没有实际内容，所有POJO对象的接口，理由同SuperMapper

<<abstract>>PhoneInfo implements Info：抽象类，作为后续电话信息类的父类(当然，后续可以有结构更复杂的存储类)

属性：

+ mainPhoneNum: String

主电话号码

+ ownerName: String

电话的主人

ListPhoneInfo extends PhoneInfo：具有一个List，存储主电话主人所具有的其他非主要号码

属性：

+ mainPhoneNum: String

+ ownerName: String

+ othersPhoneNum: List<String>

存储更多电话的List

方法：

+ addOthersPhone(String): void

+ removeOthersPhone(String): Boolean

ZoneInfo implements Info：存储这个城市的区号、急救电话、名称

属性：

+ areaCode: String

区号

+ emergencyPhone: String

急救电话

+ zoneName: String

城市名

工厂：

<<Interface>> InfoFactory：获取Info对象的所有工厂的接口

方法：

+ getInfo(String): Info

规定的获取信息的方法

PhoneInfoFactory implements InfoFactory：获取电话信息的工厂，依赖于Mapper实现getInfo方法

ZoneInfoFactory implements InfoFactory：获取区域信息的工厂，依赖于Mapper实现getInfo方法

Controller层：

使用springboot

UserContoller：提供给用户，用于获取信息。get方法依赖于Factory获取信息并返回，但剩下的修改方法目前还直接依赖于mapper

方法：

+ getPhoneInfo(String):PhoneInfo

+ getZoneInfo(String): ZoneInfo

+ modifyZone(String): void

+ modifyPhone(String): void

+ deleteZone(ZoneInfo): void

+ deletePhone(PhoneInfo): void