1. 设计思路

要求中出现了两个主体：用户、股票，故可将用户、股票分别抽象为对象。

二者间的关系：<用户>操作<股票>。

<用户界面>(interface)操作<用户实例>

为便于管理大量用户与股票实例（要求能够注册用户，创建新股票），构造<股票池>作为<用户对象>操作<股票实例>的中间层，构造<用户池>作为<用户界面>操作<用户实例>的中间层。<股票池>以股票代码为索引，<用户池>以用户名为索引。

股票（池）

用户（池）

用户界面（interface）

1. 分析对象结构（非常不完整）
2. 股票

|  |  |
| --- | --- |
| String | 股票代码 |
| String | 股票名称 |
| Array(Float) | 历史价格（包括当前价格） |
| Int | 发行数量 |

包含以下接口：

|  |  |
| --- | --- |
| Float getCurrentPrice() | 返回股票当前价格 |
| Float getHistoryPrice(int) | 返回n天前的价格（包括当前价格） |
| Float getRise() | 返回当天的涨幅 |
| Float getMax() | 返回历史最高 |
| Float getMin() | 返回历史最低 |
| String toString() | 返回实例的所有信息  //格式:”String,String,Float,…;” |

1. 用户

|  |  |
| --- | --- |
| String | 用户名 |
| String | 密码 |
| Bool | 是否管理员 |

包含以下接口：

|  |  |
| --- | --- |
| Bool checkPwd(string) | 检查输入的密码是否正确 |
| Bool editPwd(string) | 修改密码 |
| Vector<Stock> query(string,string) | 查询所有满足条件的股票(股票代码，股票名称)(股票代码暴力模糊查询)(实际上可以用一个名叫字典树的数据结构，思想非常简单，可以研究研究) |
| Bool addStock(string,string,float,int) | 添加股票(代码，名称，价格，数量) |
| Bool editStock(string,string,string,int) | 修改股票代码，名称，发行数量 |
| Bool suspend(string) | 将相应代码的股票停止交易 |
| Bool recover(string) | 将相应代码的股票恢复交易 |
|  |  |
| String toString() | 返回实例的所有信息  //格式:”String,String,Int;” |

管理员当然不能控制股票走势，但模拟程序中股票当然不能自己走

所以创建一个super user控制股票走势

同样，通过控制股票池达到目的。

它还有一个重要功能就是注册与登陆用户

Super user本身不存储任何信息，仅有各种操作