1.按照如下的 UML 类图, 创建相应的类, 提供必要的结构

Account -balance : double +Account(init_balance : double) +getBalance() : double +deposit(amt : double) +withdraw(amt : double)

在提款方法 withdraw()中,需要判断用户余额是否能够满足提款数额的要求,如果不能,应给出提示。deposit()方法表示存款。

2. 按照如下的 UML 类图, 创建相应的类, 提供必要的结构

Customer	
- firstName : String - lastName : String - account : Account].
+ Customer(f:String, 1:String) + getFirstName():String + getLastName(): String + getAccount(): Account + setAccount(acct:Account)	

3. 按照如下的 UML 类图, 创建相应的类, 提供必要的结构

Bank
-customers:Customer[]
-numberOfCustomer:int
<u>+Bank ()</u>
+addCustomer(f:String, 1:String)
+getNumOfCustomers():int
+getCustomer(index:int):Customer

- ➤ addCustomer 方法必须依照参数(姓,名)构造一个新的 Customer 对象,然后把 它放到 customer 数组中。还必须把 numberOfCustomer 属性的值加 1。
- ▶ getNumOfCustomers 方法返回 numberofCustomers 属性值。

- ▶ getCustomer 方法返回与给出的 index 参数相关的客户。
- 4. 创建 BankTest 类,进行测试。

