5100 Assignment3

1. 创建类

Airliner——Airliner Directory在最上层，作为Travel可以在第一个按钮直接创建Airliner；

Airplane——Airplane Directory是由Airliner创建的，所以要放在第二个按钮里面；

Flight——Flight Directory是每一架Airplane独有的，所以要顺承接在第二个按钮；

Customer——Customer Directory是单独存储用户的，单独在第三个按钮创建；

Customer Order——Customer Order Directory用来存储每一个Customer创建的订单；

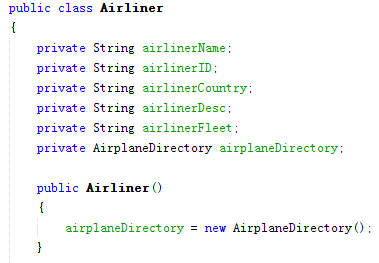
Seat用来存储用户订购的座位位置，通过To String方法返回String类型的位置；

Seat Set用二维数组存储座位编号以及剩余的座位数量。

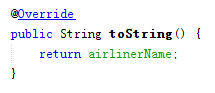
1. 类关系

**Airliner——Airliner Directory：**

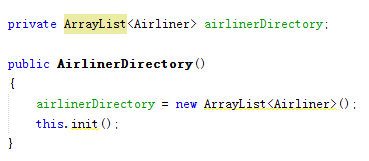
Airliner包含有Airplane Directory，以及自身所存有的变量，因为ID含有字母，因此不是integer类型，对每个变量做get和set，后几个类重复。



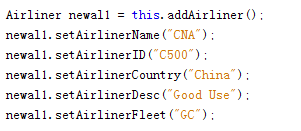
因为要在Table中显示飞机信息，所以第一个airlinerName变量要单独做To String函数；



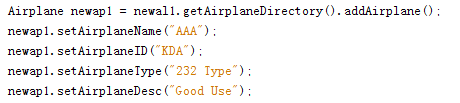
为了给TA演示航班信息，所以有预先写好的值，写入的方法是在最顶层的类进行初始化，因为Airliner Directory包含除了Customer类的所有信息，所以在这里写init()函数；



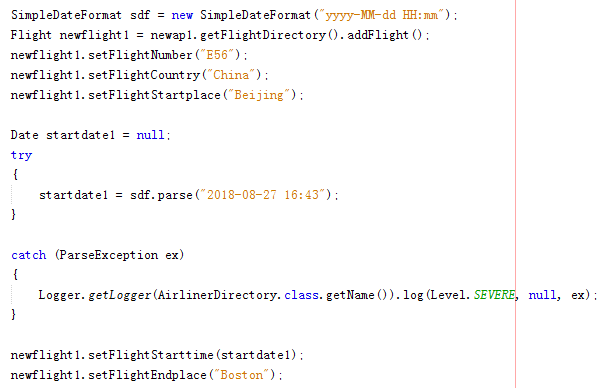
创建信息要一层一层的建，从airliner到airplane到flight，创建了上一层的实例化对象才能创建下一层；



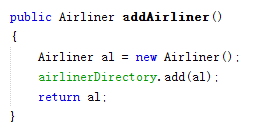
Airplane信息；



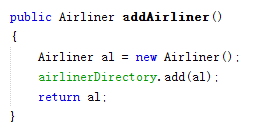
Flight信息。



除此之外，在每个Directory里面，还需要写add方法和delete方法，用于后续调用，创建删除ArrayList当中的对象；

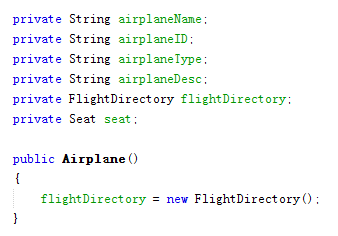


删除。



**Airplane——Airplane Directory：**

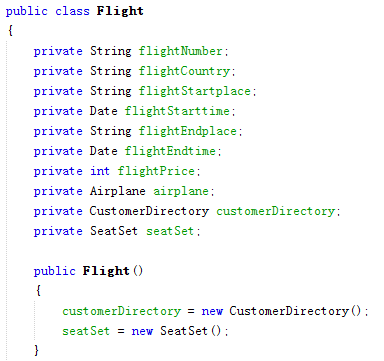
大体格式和上面的很像，不同的是Airplane类里套了一个Seat类，因为Seat是飞机的一个属性，所以需要有。



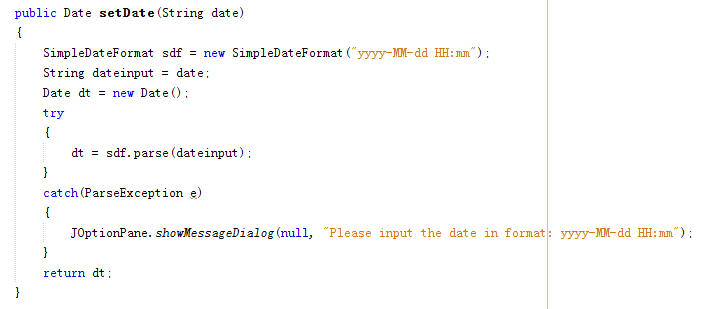
Airplane Directory和Airliner Directory很像，相同的就不用记录了，只是没了初始化。

**Flight——Flight Directory：**

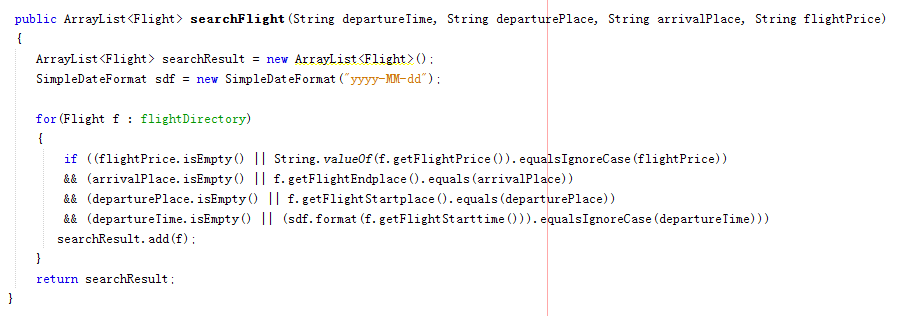
Flight类就很不一样了，一是因为它多了Date类型以及Integer类型的验证，二是因为它还与Seat Set，Customer建立了联系；包含Airplane类是因为Flight里面要能打印出飞机的一些参数；包含Customer Directory类是因为航班信息要被所有乘客选择；包含Seat Set类是因为在订票信息中有座位数组被传输。



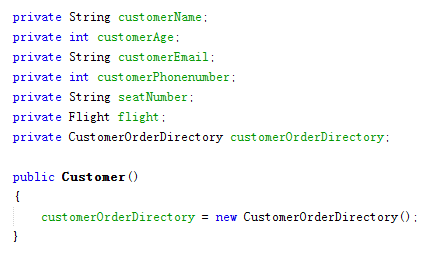
刚开始在Flight类里面还写了一个Date类型数据的转换方法，通过调用这个方法，在类中就可以将String类型的用户输入转换为Date类型了，但是由于最后的Try-catch并没有抓住日期转换所抛出的异常，所以这个方法被直接写入了创建的时候。



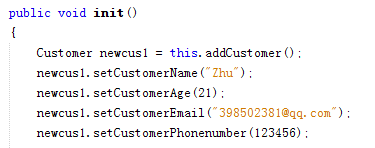
这里面比较牛逼的方法还有一个是模糊搜索，当用户需要搜索他们想要找的航班的时候，可以选择输入起始地点，起始时间，结束地点以及票价这四项中的任何一项，或者也可以选择不输，都是可以获得用户需要的目标值的，这个方法不用嵌套N多个if-else循环，只要判断每个Text field是否为空就可以了，而每一个都用与连起来，但是每个条件在内部都是用或链接，这样就可以轻松选择到底根据什么信息来筛选了；在用户的输入界面，我定了4个String类型的变量，用来接收Text field所获得的参数，然后再把String传到这个方法里面；我创建了一个新的ArrayList为searchResult，目的是用来存储符合要求的航班信息，因为不止一个，所以要用ArrayList来存；注意，如果比较的内容并不是一个完整的String的类型，比如Date或者是Integer类型，就不要直接用equals方法，而是使用equalsIgnoreCase方法，原因大概就是这样就可以跨越类型判断了吧。



**Customer——Customer Directory：**



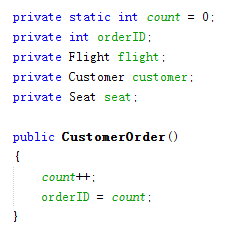
Customer类和上面几个比起来几乎都是单独存在的，它只需要包含Flight类，因为用户是拥有调取航班信息的权利的，但是我并没有在代码里实现这个功能，感觉在Customer界面显示具体航班信息的时候就需要了；当然，Customer类还需要包含有Customer Order Directory类，因为一个用户可以买多张票，而每个人的订票信息都应该在查询界面显示出来，所以要包含那个类。



同理，Customer类也要进行初始化，因为Customer Directory是最大的包含Customer信息的类，所以在这里进行初始化，在code review的时候，TA说不要对电话号码设定为Integer类型，因为11位数会超出范围，用String做验证就好了。

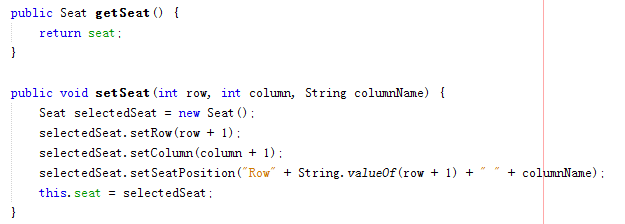
**Customer Order——Customer Order Directory：**

Customer Order类最特殊的是，它的Order ID应该做成自增的，也就是Customer创建一次Order，都会使它的ID自动加1，实现这个功能需要用的static和一个Integer类型的变量count进行自增。



在Customer Order类中，还需要引入其他的类，比如Flight类，因为Order信息可以说是Flight信息的另一个翻版，几乎Flight里面的数据这里面都需要，所以肯定是需要的，之所以引入Customer类，感觉这样两个类就互相包括了，这是因为Order界面我在刚开始是想显示Customer name之类的具体信息，后来没这么做，但还是保留了这个类；最后那个Seat类是因为Seat类存了作为信息，通过引入这个类，我们就能直接读取到拼接好的座位号了，这样更像是出票的过程。

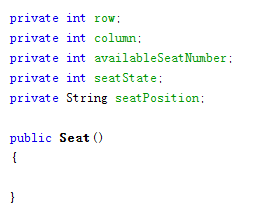
在这些get和set函数中，最与众不同的是对seat的set方法，因为每一次调用这个方法，代码都要实现一个字符串拼接，这样set后得到的就是个String类型的座位了。



先调用Seat类的构造函数，生成一个新的座位，因为row和column都是从Combo Box里面读出来的，是index值，所以还要对其加1才可以使用，所以对新生成的座位做index+1的操作，然后再调用setSeatPosition函数把row和column拼接起来，前者是把Integer类型转换为String类型，后者是直接get到String类型的名称。Customer Order Directory类同理，只有add和delete方法，没什么区别。

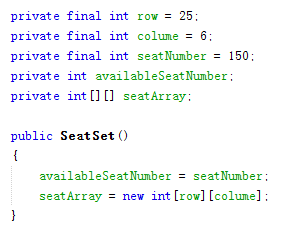
**Seat：**

在Seat类中不用包含其他类里面的参数，其作用只是为了返回一个座位的位置，用To String函数来返回seatPosition值；其中包含一个空的座位的构造函数。



**Seat Set：**

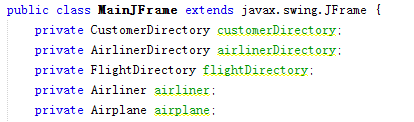
在类中，用二维数组seatArray来存储座位坐标，第一纬度存row，第二纬度存column，row被设定为25，column被设定为6，初始的座位数量存为seatNumber，也就是固定的150个，再新建一个变量为availableSeatNumber，在刚开始的初始值设定为和seatNumber相同，都为150，但随着每次提交座位订单，availableSeatNumber的数量都会减1。



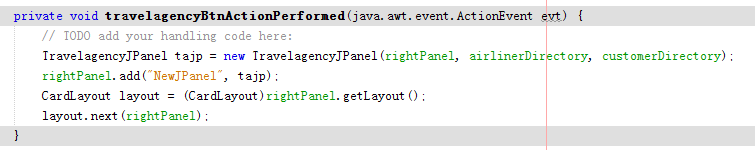
在构造函数中就进行了初始化，对availableSeatNumber和二维数组都进行了初始化。

1. Main Interface

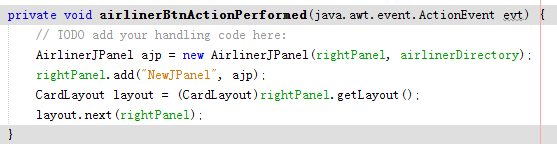
主界面有三个按钮，分别是创建Airliner，创建Airplane + Flight，还有创建Customer，所以需要实例化的类有如下。



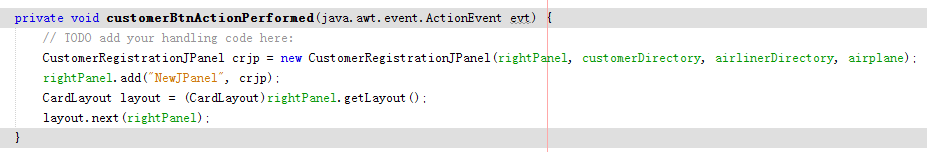
对于Travel Agency按钮来说，它需要有rightPanel，这个是都需要的，还要有Airliner Directory，因为要创建一系列，还要有Customer Directory，因为里面有个按钮是查看所有用了我这个Travel Agency的客户的订单信息。通常做按钮跳转都是这么写的，下一个Panel的名字，加一个自己起的实例化的名字，等于new关键词加上那个Panel的名字，对于里面传的值，都是要在下一个界面显示出来或者被存储的参量。因为整个layout被设定了为了Card Layout，分为左右两个Panel，因此每次只要对右侧的Panel也就是right Panel做操作就可以了，调用add方法，第一个引号里面写想要创建Panel的名称，第二个直接写刚才实例化好的Panel名字。后面的两行也是固定的写法，先是实例化Card Layout并调用getLayout方法；然后再调用next方法，告诉这个layout，我要改变的是右边的部分。



同理，第二个按钮因为不涉及Customer类的调用，因此只需要rightPanel和Airliner Directory。



第三个按钮要创建的比较多，但我也忘了为啥要传airplane了，感觉是不需要的，后面看到再说吧。



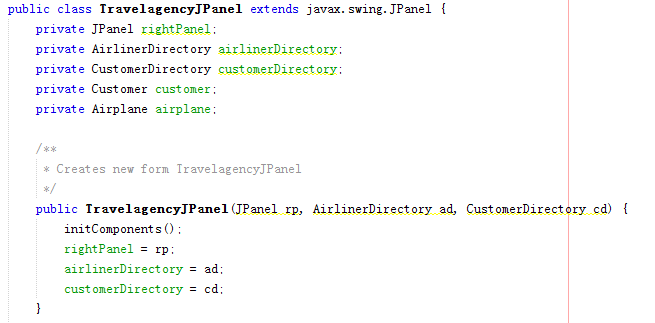
1. Travel Agency Interface

**TravelagencyJPanel：**

我先写的类是TravelagencyJPanel，在这个Panel中，用户可以选择创建一个新的Airliner，点击这个Button会跳转到管理界面；也可以选择搜寻航班，这样会跳转到后面已经写好的搜寻界面；最后一个功能是查看所有已经订票的客户的用户信息，正是因为要查看Customer信息，所以才要引入Customer Directory类。

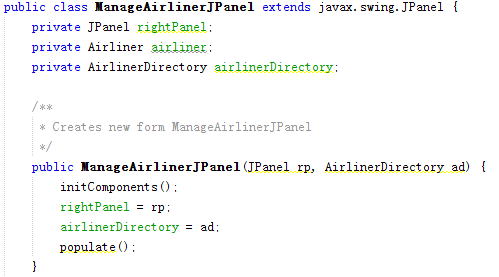
但是对于第二个Button去搜寻航班信息来说，那个界面虽说是后来写的，但是因为要具体到每个飞机和航班的信息，所以我是把单个的Airplane也传进去了，其实我感觉可以通过调用大类给点出来，但我后来就没试过。

对于每个Button需要传的参量，除了搜寻界面多个Airplane，其他的都差不多。



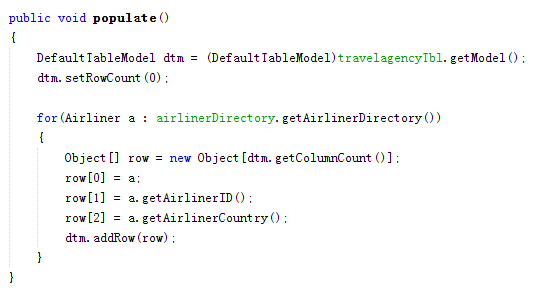
**ManageAirlinerJPanel：**

管理所有航空公司信息的界面称为ManageAirlinerJPanel，在这个界面当中，先创建一个Table，用来显示所有创建的信息，Table就是直接拖进来的，将默认的行数置为0，列数为3，分别用来显示航空公司的名字，ID以及航空公司所属的国家。

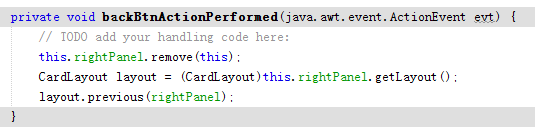


Populate方法是用来每次加载和刷新Table的函数；DefaultTableModel是Table内置的一个函数，通过调用这个方法，创建一个实例化的dtm对象，就相当于创建了每一行可以写入的一个对象。

通过应用for循环，遍历airliner Directory里面的所有airliner对象，并把它们都打印出来，每一个row都是一个Object类的实例化，并把得到的值赋给这个row对象。切记，对于这个数组的第一个对象row[0]，一定要把整个实例化的airliner对象，比如说是a，都传进去才可以，而后面的row[1]和row[2]就可以通过对象a给点出来，且每次执行之后都要调用addRow方法补充下一行。



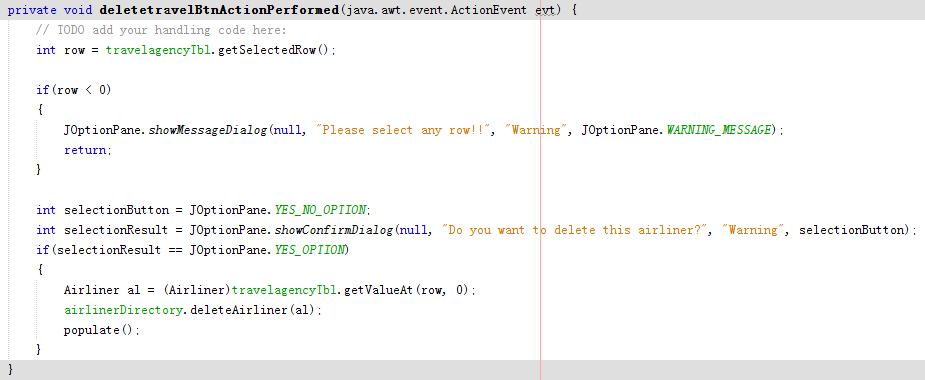
在管理界面还需要4个不同功能的Button，一个是Back Button，这个和向后跳的比较类似，只是把next改成了previous。对于Create界面的跳转，和正常的下一页跳转是一样的。



对于View界面的跳转，因为要先选中一行，然后把选中的内容最为对象，传入到下一个界面，所以代码有所不同。



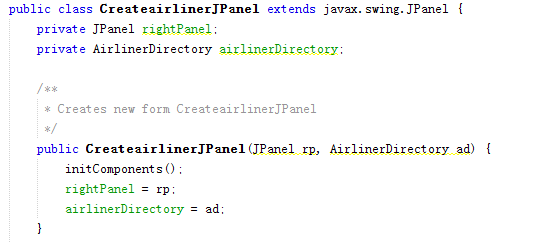
创建一个Integer类型的变量selectedRow，用于存储从Table中被选中的行数。做一个if-else判断，用于判定是否有行被选中了，没有被选中，selectedRow自然为0，所以可以使用弹出提示。如果选中了，则执行里面的代码，创建一个新的airliner对象，其内容来自于Table调用的getValueAt方法所指向的那一行的对象，剩下4行和create的跳转相同，只是要把刚获得的airliner也一起传进去。



同理，想要调用Delete方法的时候也要选中一行，如果没有选中，也调用查看selectedRow是否小于0的方法，在这里我还添加了一个YES或者NO的判定，用来显示一个弹窗，向用户确认是否确定要删掉。如果确定要删掉，则执行YES里面的代码，同样的是把实例化的airliner对象创建出来，并且调用已经写好的数组delete方法，最后再调用populate方法来刷新页面。

**CreateairlinerJPanel：**

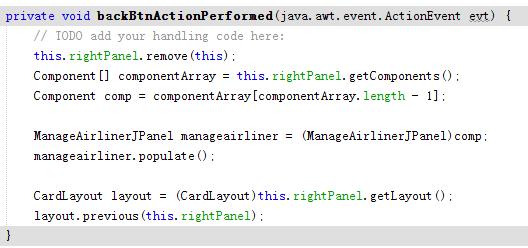
因为创建的内容只和airliner有关，所以只要引入Airliner Directory就可以了。



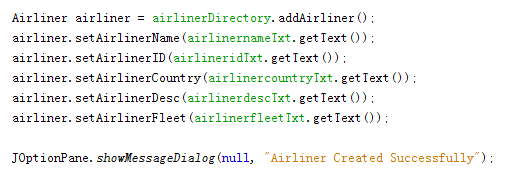
这里的Back Button比较不同，因为这里不能只是简单的跳转回上一页，还需要将创建的值传回去。首先调用remove(this)将当前的layout层移除，我感觉我这里好像写的是有点问题，因为如果值没有传对，会乱跳。后两行感觉是创建了一个component对象，并将其对应成一个array，长度减1。

除此之外，在指定跳转Panel的时候，还要重新调用被跳转Panel的populate函数，因为如果不调用populate方法的话，那个Table就不会更新了。

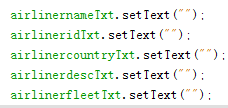
最后还是一样，继续用previous方法，告诉界面我是往回跳了一层。



至于输入东西并创建的过程，和前面的Assignment几乎是一样的，只需要用set方法，把Text field读到的用户String输入一一对应的传到变量里面就可以了。切记，创建airliner对象一定要在判断是否输入为空之后，有其他的转型判断也一样，永远要在判断之后，不然判断就算跳出了，还是会有新的对象生成。

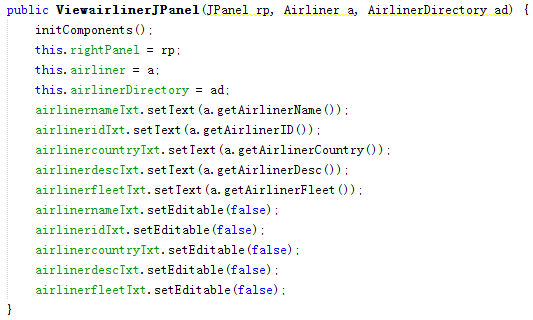


最后还可以将Text field再置为空，这样的目的是比较方便后续的继续输入，感觉比较智能吧。

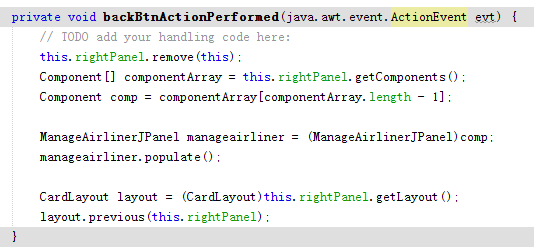


**ViewairlinerJPanel：**

View和Create差不多，几乎是对称的，刚才是通过set把内容放进了变量，这里就要调用get方法把变量提取出来，并显示在框里面。也可以通过使用setEditable方法，把框的修改先改成false，这样就在不update的时候不能更新了。而在点击UPDATE之后，就要把setEditable方法置为true，这样就可以直接使用了。在点击SAVE之后，再用setEditable方法把输入框变为false，其余代码和create相同，同样还要验证是否输入为空。



对于Back Button也是一样的，因为涉及后退时传回修改的内容，同样别忘了调用populate方法，这样才能刷新Table里面的内容。

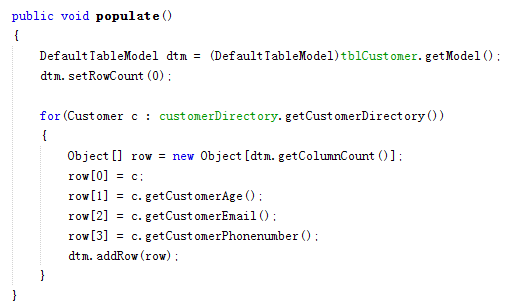


**ManageCustomerJPanel：**

最后一个ManageCustomerJPanel类我做的不是很多，里面就放了一个Table用来显示customer的所有信息，写法和manage飞机或者航班的都一样。

里面同样要包含一个populate方法，用来刷新表格，这里的for循环需要遍历所有的Customer Directory，第一行也就是row[0]也要把整个customer对象c传进来，这样就不会发生空指针的问题了。

因为这里的back不存在传值的问题，所以直接用三行的back就可以了。



1. Airliner Manager Interface