项目概述

1. 选题原因

近年来，奶茶成为了现代餐饮的“流量担当”，各类连锁奶茶店如雨后春笋般涌现，各个奶茶品牌也陆续推出自己的线上点单平台，节省客户排队等待的时间，改进客户的用餐体验，降低店家点餐的人工成本，提高服务效率。为了更好地迎合当代市场需求，锻炼逆向工程能力，我选择了国际校区饮料点餐系统作为大作业程序设计的题目。

1. 题目分析以及功能模块的设计和划分
2. 用户信息管理模块

为了方便后续记录用户订单数据，保护用户隐私，扩展其他个性化服务，引入用户信息管理模块，提供账号注册与登录功能。这些功能均被封装在Account类中。

1. 点单模块

该模块负责展示店家提供的各类饮品，包含饮料的编号、饮料的名字、饮料的图片，饮料的类型、饮料价、饮料的介绍等信息。同时，用户可以在该模块进行饮品及其数量的选择，并定制其规格，温度，甜度等，将饮品加入购物车。

1. 购物车模块

该模块负责显示用户已点的饮品，包含已点饮品的名字，编号，图片，规格，温度，甜度，价格，数量，全部饮品的价格，服务费等信息，用户依然可以在此模块进行饮品的数量增减或删除，当用户在点单模块新点了饮品后，更新该模块的信息。同时用户需要在此模块选择桌号，人数，是否外带等信息。最后用户可以在此模块进行结算，完成结算后生成订单。

1. 订单模块

该模块负责展示用户的历史订单，包含订单编号，下单时间，外带还是堂食，消费总额等信息。在用户的订单完成结算后，刷新此模块的信息。

1. 开发环境

该系统的应用场景主要为奶茶店，适于使用移动端开发平台，因此选用与java语言适配的Android Studio平台进行开发。

操作系统：Windows 11

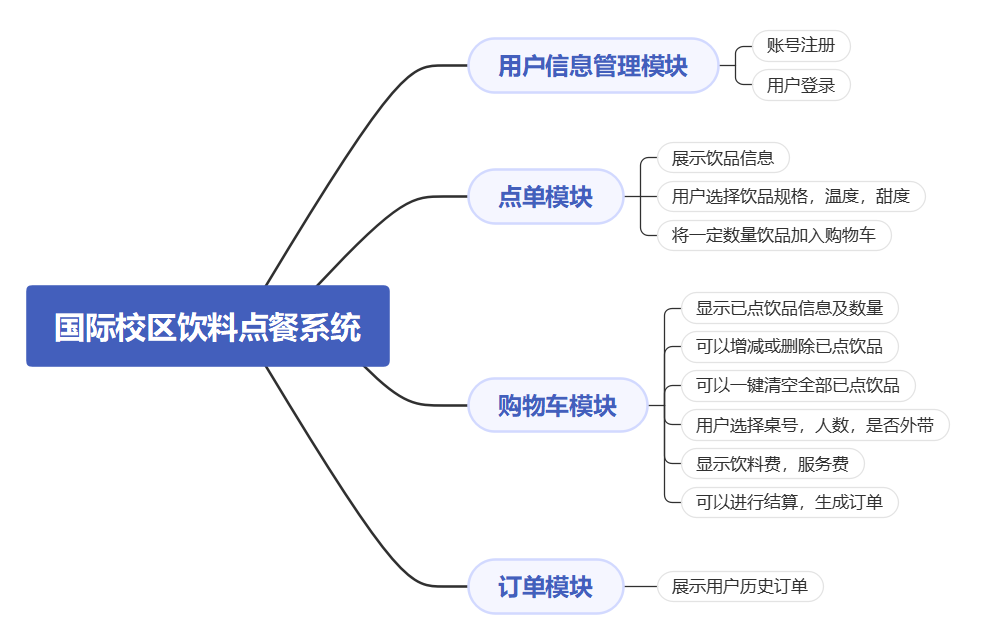
Java版本: openjdk 11.0.16.1 2022-08-12 LTS

OpenJDK Runtime Environment Microsoft-40648 (build 11.0.16.1+1-LTS)

OpenJDK 64-Bit Server VM Microsoft-40648 (build 11.0.16.1+1-LTS, mixed mode)

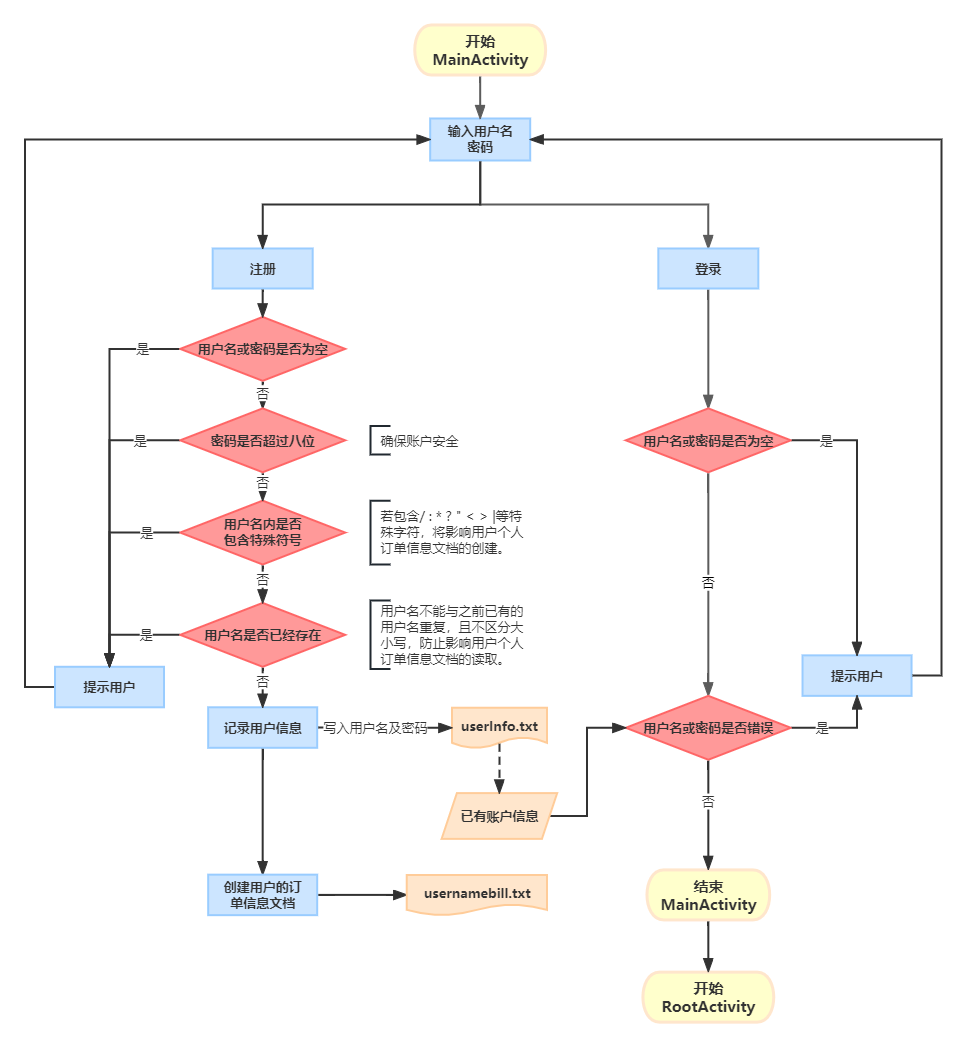
1. 程序概要设计

系统总功能模块图如下所示。其中用户信息管理模块在MainActivity中实现；点单模块，购物车模块，订单模块在Rootactivity中完成。



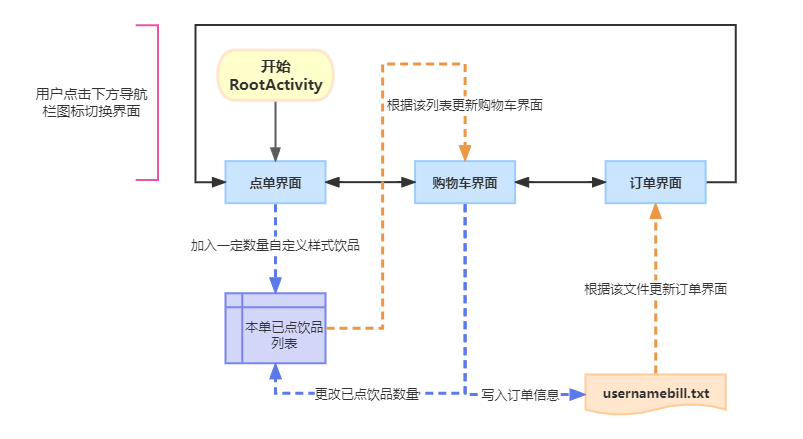
图表 1 系统总功能模块图

在MainActivity中包含用户名和密码两个输入框，注册和登录两个按钮。其运行逻辑以及具体细节如下图所示：



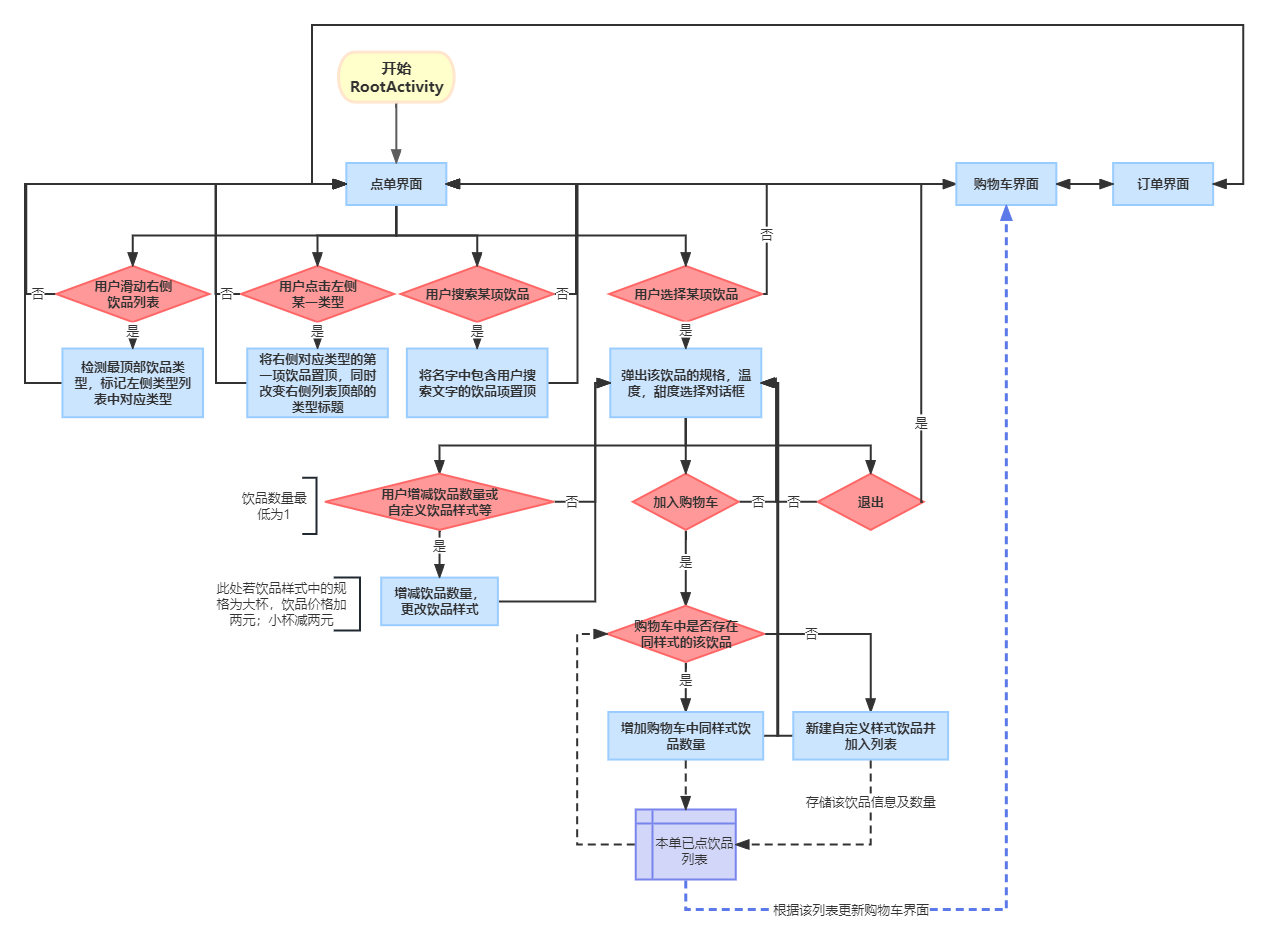
图表 2 系统总流程图（MainActivity部分）

RootActivity主要分为点单页面，购物车页面，订单页面；点单页面负责显示可选饮品，提供自定义饮品样式与将饮品加入购物车服务；购物车界面负责显示已点饮品，提供本单增减饮品，一键删除本单全部饮品，选择桌号、人数、是否外带，结算本单。订单界面负责显示历史订单，包含历史订单的编号，下单时间，堂食还是外带，消费总额的信息。RootActivity的数据传输以及文件读写均如以下流程图中所示。



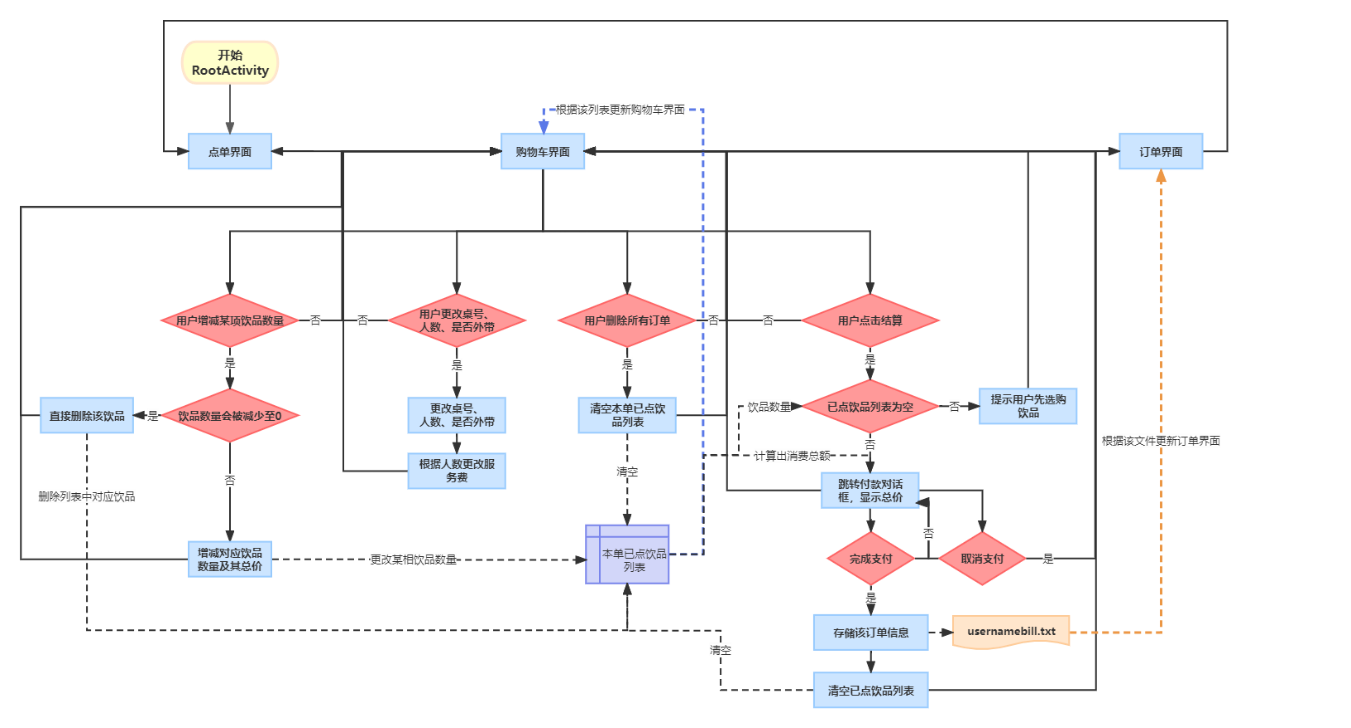
图表 3 系统总流程图（RootActivity部分）

点单界面的细节流程图如下。



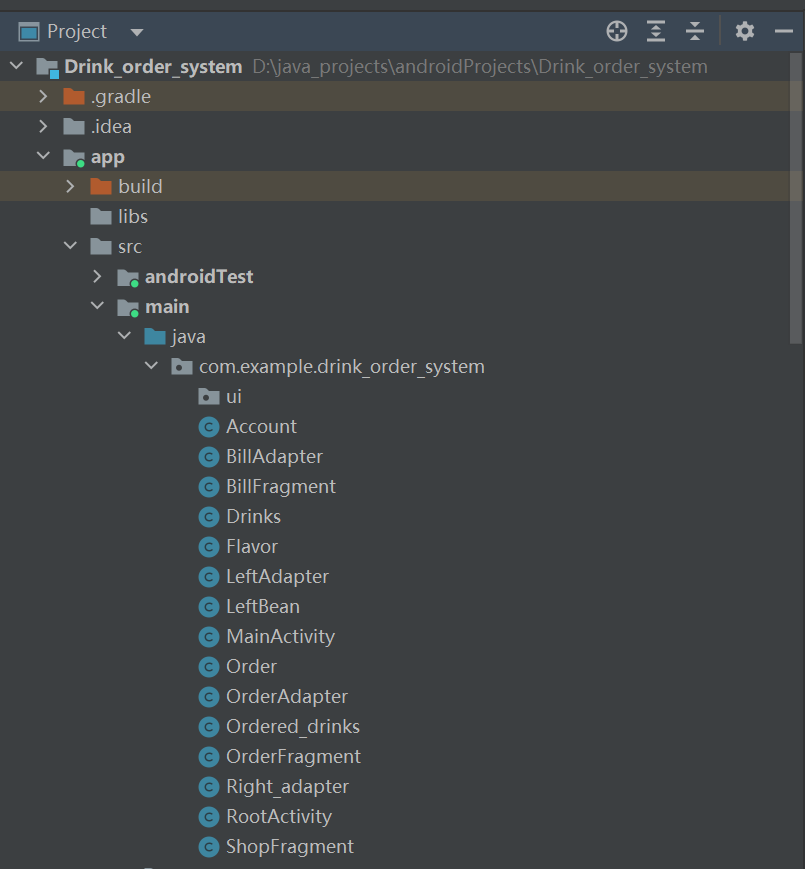
图表 4 点单界面流程图

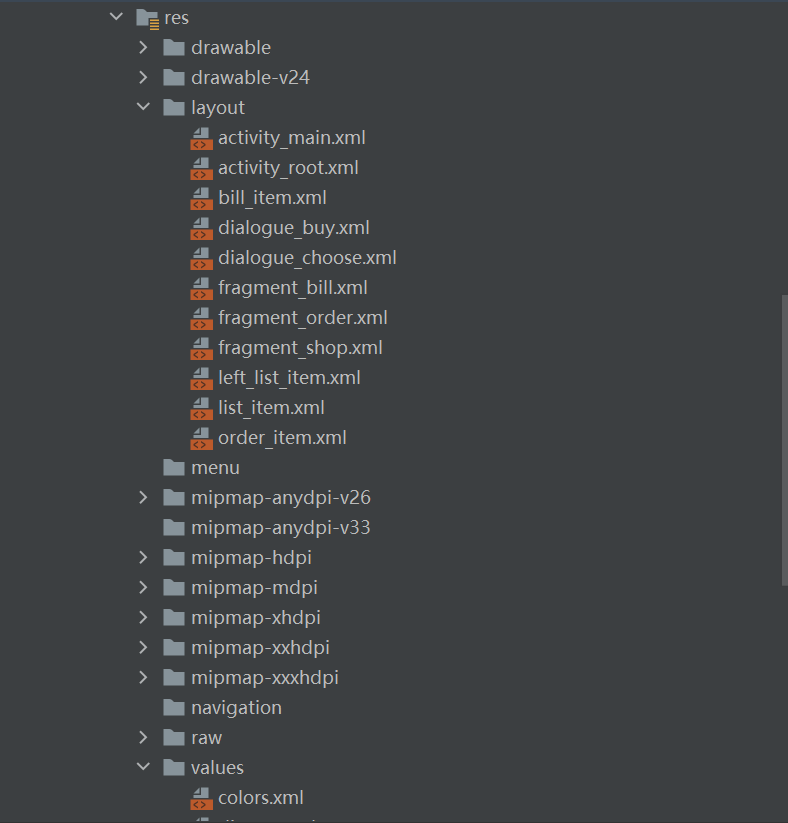
购物车界面的细节流程图如下所示。

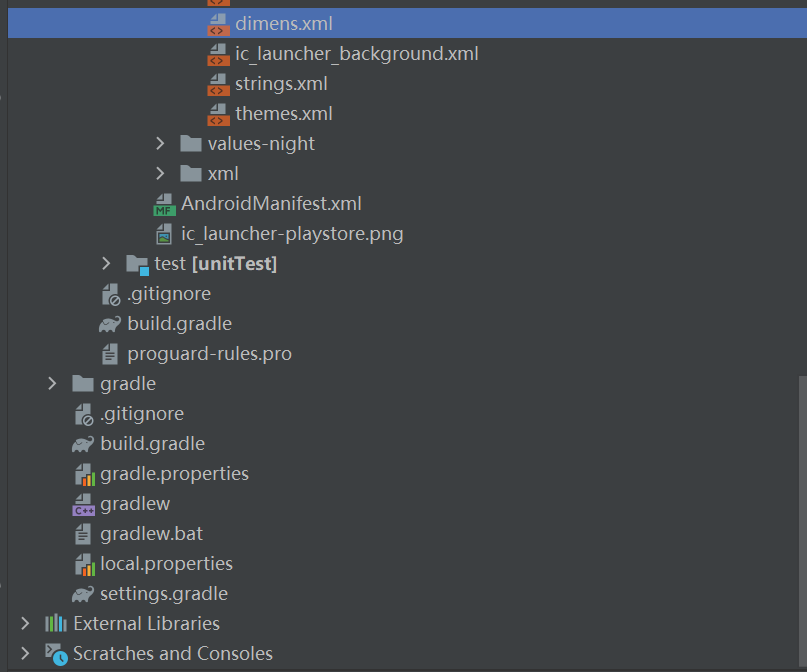


图表 5 购物车界面流程图

文件结构设计如下图所示，基本遵照Android Stduio的文件结构目录，在com.example.drink\_order\_system文件夹中放置所有java文件，在res/drawable文件夹中放置界面图标及饮品图像资源，在res/layout文件夹中放置所有xml界面布局文件，在res/values/colors.xml中定义界面颜色，在res/values/strings.xml中定义界面中常用的字符串。







图表 6 文件结构设计

|  |  |
| --- | --- |
| 图表 7 Account类图 | 图表 8 MainActivity类图 |
| 图表 9 Drink类图 | 图表 10 Flavor类图    图表 11 Ordered\_drinks类图 |
| 图表 12 OrderFragment类图 | 图表 13 Right\_adapter类图 |
| 图表 14 LeftAdapter类图 | 图表 15 LeftBean类图 |
| 图表 16 ShopFragment类图 | 图表 17 OrderAdapter类图    图表 18 BillFragement类图 |
| 图表 19 BillAdapter类图 |  |

1. 程序详细设计
2. MainActivity界面的注册登录功能

MainActivity界面中的注册登录功能全部通过Account类实现。MainActivity中的ET\_username负责读入用户名，ET\_password负责读入密码。BT\_signUp\_onClick()函数监听注册按钮，若用户名与密码均不为空，且用户名不包含/ : \* ? " < > |等特殊字符，密码大于等于八位（保障账户安全），就创建相应Account对象，调用saveAccount()函数，saveAccount()函数遍历userInfo.txt文中的用户名信息，若不存在该用户名，就进行账户创建，将用户名及密码存入userInfo.txt文件，并创建以用户名+“bill.txt”为名的用户个人订单信息记录文件，返回true；若该用户名已存在（不区分大小写，防止用户个人订单文件名重复），返回false，提示用户更改用户名。BT\_longIn\_onClick()函数监听登录按钮，若用户名与密码均不为空，创建Account对象，调用exist()函数，exist（）函数遍历userInfo.txt文中的用户名及密码信息，若存在用户名与密码均匹配的，返回true，成功实现登录，跳转至RootActivity界面；若不存在该用户名与相应密码，返回false，登录失败，提示用户用户名或密码错误。



图表 20 登录注册界面

1. RootActivity的三个不同界面切换

RootActivity中共包含三个不同的Fragment，分别对应OrderFragement类，ShopFragment类，BillFragment类，分别实现点单界面，购物车界面以及订单界面。用户可以通过点击界面下方导航栏内对应图标实现三个界面的互相切换。

界面切换的具体逻辑通过FragmentManager创建FragmentTransaction来完成。FragmentManager 管理 fragment 返回堆栈。在运行时，FragmentManager 可以执行添加或移除 fragment 等返回堆栈操作来响应用户互动。每一组更改作为一个单元（称为 FragmentTransaction）一起提交。

具体的实现流程如下，当RootActivity被创建时，默认的界面是点单界面，它在RootActivity的onCreate（）函数被调用时首先通过defaultFragment()函数被实例化，通过一个FragmentTransaction加载并展示到界面中。同时RootActivity的onCreate（）函数被调用时，在该函数中对底部导航栏的RadioGroup设置监听，一旦检测到按钮的状态变化，调用onCheckedChanged()函数。该函数先隐藏所有不为空的fragment，再检测该fragment是否有被实例化过，若没有就实例化对应fragment，再将该fragment通过一个FragmentTransaction加载并展示到界面中；若对应fragment有被实例化过就直接展示到界面中。此时界面中仅展示了底部导航栏被选中的图标对应的fragment，其余两个fragmnet被隐藏了起来。

|  |  |  |
| --- | --- | --- |
| 图表 21点单界面 | 图表 22 购物车界面 | 图表 23 订单界面 |

1. Drinks类的具体设计

Drinks类用于实例化在点单界面中所有可供选择的饮品。包含饮料的编号、饮料的名字、饮料的类型、饮料价、饮料的介绍，饮料对应的图片，可选饮品列表等成员变量。包含三个构造函数：

① Drinks(String name, String type, float price, String introduction, int ImageResId)可以将所有传入形参赋值给对应实参，同时将实例化后的Drinks对象加入Drinks类中的静态ArrayList<Drinks> all\_drinks中，将该Drinks对象在该ArrayList中的index赋值给该Drinks的number变量，作为饮品编号。

② Drinks(String name, float price, String introduction, int ImageResId)比第一个构造函数少传入一个type形参，所以将其对应实参type设为null，其余操作与第一个构造函数一致。

③ Drinks(int i)传入的形参为饮品编号，可以根据该饮品编号直接通过all\_drinks.get(i)获取对应的Drinks对象，直接将该对象的所有变量赋值给当前对象对应变量，此时不用再将当前对象加入all\_drinks列表中。

1. Flavor类的具体设计

Flavor类用于存储与饮品样式有关的信息，包含size（饮品规格），sugar（饮品甜度），temperature（饮品温度）等成员变量。

① 重写了toString()方法，返回size+"，"+sugar+"，"+temperature拼接出的新字符串。

② 重写了equals()方法，直接返回两个对象toString()后的比较结果。

1. Ordered\_Drinks类的具体设计

Ordered\_Drinks类用于存储用户自定义样式后的饮品及其数量。使用组合的设计思维，包含Drink，Flavor的成员对象，被点饮品数量以及一个静态ArrayList<Ordered\_drinks> ordered\_array的成员变量。

① Ordered\_drinks(Drinks drink, Flavor flavor, int drink\_number)构造函数将所有传入形参赋值给对应实参后，遍历ArrayList<Ordered\_drinks> ordered\_array，检查是否存在Drink与Flavor均相同的Ordered\_Drinks对象，若存在则将原列表中的Ordered\_Drinks对象增加对应数量。若不存在，则直接将当下Ordered\_Drinks对象加入列表。

②getOrdered\_array()返回Ordered\_array。

③ clearOrdered\_array()清空Ordered\_array。

④ getDrinkCost()返回Ordered\_array中所有已点饮品的总价。在遍历计算时，若该被点饮品的规格是大杯，则在原本的价格上加2元；若被点饮品的规格是小杯，则保持原本的价格不变；若被点饮品的规格是小杯，则在原本的价格上减2元。

⑤ addDrink(int i)将Ordered\_array.get(i)得到的Ordered\_Drinks对象数量加一。

⑥ subtractDrink(int i) 将Ordered\_array.get(i)得到的Ordered\_Drinks对象数量减一，减后其数量变为0，则直接删除该Ordered\_Drinks对象。

1. LeftBean类的具体设计

LeftBean用于存储点单界面的左侧类别列表中每项类别及对应的右侧列表中该类别的第一项饮品的位置。包含title类别名，rightPosition右侧列表对应位置，isSelect是否被选中这三个成员变量。

① isSelect()函数返回isSelect。

② setSelect(boolean select)可以设置isSelect的状态。

1. 点单界面的饮品及类别展示与双列表联动效果

如下图所示，在点单界面的左侧是饮品的类别列表，右侧是饮品列表，饮品列表的上方是最顶部饮品的类别标题。饮品列表中每个类别的第一个饮品上方都存在对应的类别标题。当左侧的类别列表中的某项类别被选中后，右侧的饮品列表就滑动至对应类别的饮品区域，将该类别中的第一项饮品置顶，并更新类别标题。当右侧的饮品列表被滑动后，左侧的类别列表始终将饮品列表中显示出的最顶部的饮品对应的类别设为选中状态，将其背景调为灰色，并更新右侧类别标题。实现了类别列表与饮品列表之间的联动效果。

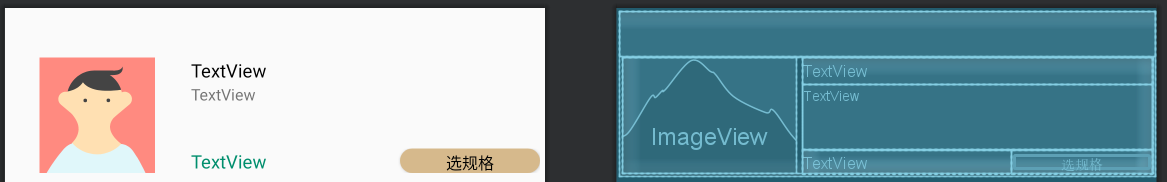


图表 24 OrderFragment界面设计

此处的类别列表与饮品列表均使用RecyclerView作为外部框架（即容器），通过继承自RecyclerView.Adapter的自定义Adapter创建列表项目（即内容物）。

RecyclerView 可以高效地显示大量数据。可以提供给定数据并定义每个列表项的外观，而 RecyclerView 库会根据需要动态创建元素。RecyclerView 会回收这些单个的元素。当列表项滚动出屏幕时，RecyclerView 不会销毁其视图。相反，RecyclerView 会对屏幕上滚动的新列表项重用该视图。这种重用可以显著提高性能，改善应用响应能力并降低功耗。在RecyclerView中，列表中的每个独立元素都由一个 ViewHolder 对象进行定义。创建 ViewHolder 时，它并没有任何关联的数据。创建 ViewHolder 后，RecyclerView会将其绑定到其数据。可以通过扩展 RecyclerView.ViewHolder 来定义 ViewHolder。RecyclerView 会请求这些视图，并通过在 Adapter中调用对应方法，将视图绑定到其数据。可以通过扩展 RecyclerView.Adapter 来定义 Adapter。

①右侧饮品列表对应RecyclerView类的right\_listView，Right\_adapter与RightViewHolder。



图表 25 RightViewHolder的界面设计

RightViewHolder的界面中最上方对应该项饮品的类别标题，最左侧对应该饮品图片，右侧的TextView从上至下依次对应饮品名字及编号，饮品介绍，饮品价格。其数据在OrderFragment创建时，通过initData（）函数导入，该函数实例化了不同Drinks对象，并将其加入OrderFragment的drinks\_array中。在实例化时，每开始创建一个新的类别的饮品，就使用Drinks(String name, String type, float price, String introduction, int ImageResId)的构造函数，该构造函数中传入了饮品的类别。在此饮品之后再创建同类别的其他饮品，调用Drinks(String name, float price, String introduction, int ImageResId)的构造函数，此构造函数不再传入饮品类别，直接将其类别设为null。在Right\_adapter实例化时传入该drinks\_array，将drinks\_array中的每一个Drinks对象对应的数据与RightViewHolder进行绑定。在RightViewHolder的bindBean（）函数中进行数据绑定时，若此时对应的Drinks对象的类型变量为空，则将对应显示饮品类别的TextView设为null；若不为空，则正常通过setText()绑定类型数据。这样就能够实现每个类别的第一个饮品显示对应类别标题，以下饮品不再重复显示类别。

在OrderFragment创建时对right\_listView添加对列表滚动的监听。若列表发生滚动，调用onScrolled()的函数,在该函数中获得此时饮品列表显示出的位于最上方的饮品在列表中的位置。再调用leftAdapter.setCurrentPosition(firstItemPosition)函数，遍历leftAdapter的成员变量ArrayList<LeftBean> mList，找到该饮品对应类别的LeftBean对象，将其isSelect设置为true，其他类别的LeftBean对象的isSelect设置为false。然后调用notifyDataSetChanged()，刷新leftAdapter。最后将该饮品的对应类别名称赋值给OrderFragment中饮品列表的上方的类别标题。

② 左侧类别列表对应RecyclerView类的left\_listView，LeftAdapter和LeftViewHolder。



图表 26 LeftViewHolder的界面设计

LeftViewHolder的界面中的TextView用于绑定饮品类别的数据。其数据同一在OrderFragment创建时，通过initData（）函数导入，在将所有Drinks实例加入drinks\_array中后，遍历drinks\_array中的Drinks对象，当该对象的成员变量type不为null中，创建对应的LeftBean对象，传入此Drinks对象的类别及其在drinks\_array中的位置，并将此LeftBean对象加入OrderFragment的成员变量ArrayList<LeftBean> titles\_array中。

在将左侧类别列表的数据绑定到LeftViewHolder 的界面中时，即调用onBindViewHolder()时，该函数先是获取当前的LeftBean对象，将其传入LeftViewHolder的bindBean（）函数进行数据绑定，将LeftViewHolder中的TextView显示的文字设置为该LeftBean对象对应的饮品类别。若该LeftBean对象isSelect为true，则将holder背景设为灰色，实现被选中的效果；若该LeftBean对象isSelect为false，则将holder背景设为白色，实现未被选中的效果。

为了实现左侧的类别列表中的某项类别被选中后，右侧的饮品列表就将该类别中的第一项饮品置顶，需要在LeftAdapter中回调它的点击事件到OrderFragment当中，所以先在LeftAdapter中定义了一个接口如下所示

interface OnItemClickListener {  
 void onItemClicked(int rightPosition);  
}

再定义函数如下所示，将一个OnItemClickListener绑定至LeftAdapter的成员变量。

public void setOnItemClickListener(OnItemClickListener onItemClickListener) {  
 this.onItemClickListener = onItemClickListener;  
}

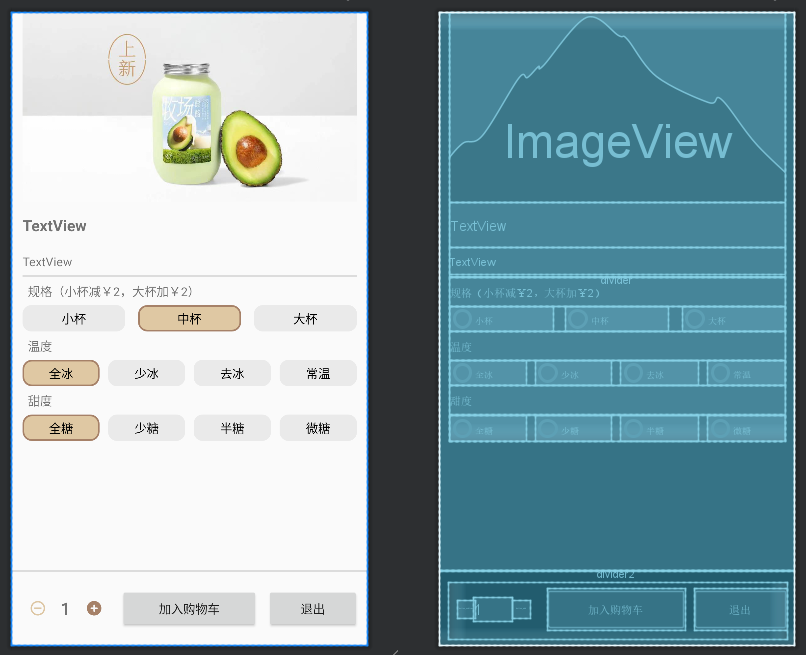
再在OrderFragment的onCreateView()中调用该函数，并传入实现了onItemClicked（）方法的OnItemClickListener对象，重写的onItemClicked（）方法实现了使右侧饮品列表的滑动至被传入的位置。

leftAdapter.setOnItemClickListener(new LeftAdapter.OnItemClickListener() {  
 @Override  
 public void onItemClicked(int rightPosition) {  
 if (right\_llM != null) {  
 right\_llM.scrollToPositionWithOffset(rightPosition, 0);  
 }  
 }  
});

最后在LeftAdapter的onBindViewHolder()中对左侧类别列表的每一项设置点击监听。若被点击，调用onClick（）函数，在该函数中调用在OrderFragment中被重写的onItemClickListener.onItemClicked（）方法，传入此时被点击的LeftBean对象对应的右侧列表饮品位置，实现右侧的饮品列表将左侧列表被点击的类别中的第一项饮品置顶的效果。同时，将被选中的LeftBean对象的isSelect设置为true，其余LeftBean对象的isSelect设置为False，再通过notifyDataSetChanged()刷新LeftAdapter。

1. 点单界面的自定义饮品样式并加入购物车功能

此处依然时在Right\_adapter中回调选规格按钮的点击事件到OrderFragment当中，获取被点击的选规格按钮相对应的饮品信息，加载到自定义饮品样式对话框中并展示。以下自定义饮品样式的对话框首先在OrderFragment的onCreateView()函数中通过AlertDialog.Builder创建。同时对对话框内的各类按钮设置点击监听。



图表 27 自定义饮品样式的对话框界面

① 对其中的增减按钮进行监听。若被点击，增加时若饮品数量等于100则不能再增减，减少时若饮品数量等于1则不能再减少。

② 对加入购物车按钮进行监听。若被点击，获取饮品的编号以及用户选择的规格，温度，甜度信息，通过该饮品编号使用Drinks(int i)直接得到完整信息的该饮品，创建对应Flavor对象。然后传入Drinks对象，Flavor对象，饮品数量，创建对应Ordered\_drinks对象，更新Ordered\_drinks类中的ordered\_array。最后关闭该对话框。

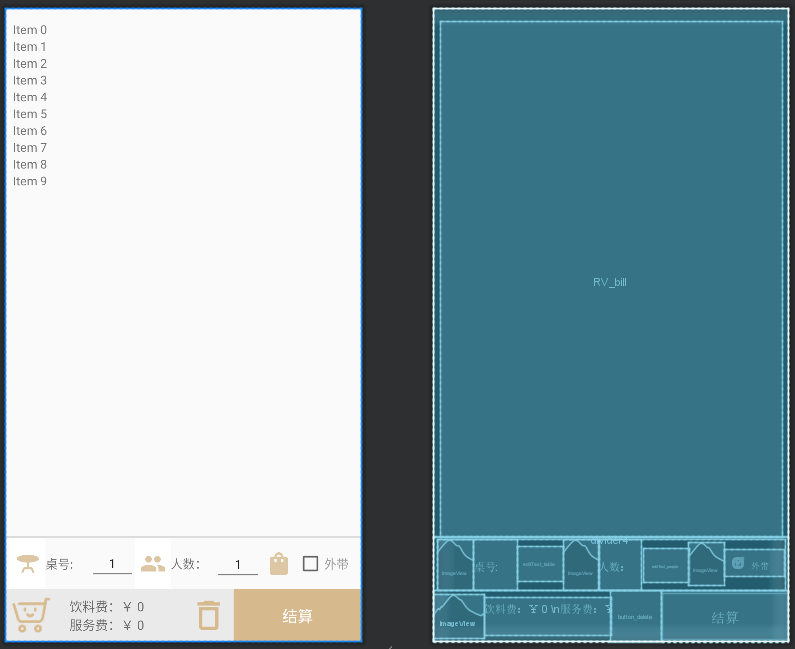
③ 对退出按钮进行监听。若被点击，直接关闭该对话框。

1. 点单界面的饮品搜索功能

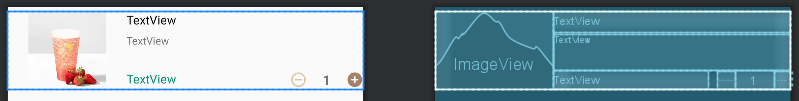
用户在搜索栏输入想要搜索的文本，若可选饮品列表中有饮品名字包含该文本，就将列表移至本饮品项目置顶；若没有饮品名字包含该文本，保持列表当前状态不动。

1. 购物车界面的数据载入与更新方式

同样使用RecyclerView作为外部框架，通过OrderAdapter创建列表项目。



图表 28 购物车界面设计



图表 29 购物车中已点饮品项目界面设计

通过Ordered\_drinks.getOrdered\_array()获得Ordered\_array，将其传入Adapter进行已点饮料列表的界面与数据绑定，更新饮料总价。重写ShopFragment的onHiddenChanged()方法，每当该ShopFragment的显示/隐藏状态发生变化时调用该方法，在该方法中调用refresh()函数，重新遍历Ordered\_array，绑定数据，更新饮料总价。实现在点单界面点单后，返回购物车界面时数据刷新的效果。

1. 购物车界面的增减饮品功能

在OrderAdapter中回调增减按钮的点击事件到ShopFragment当中，分别调用Ordered\_drinks中的addDrink(int i)与subtractDrink(int i)。

1. 购物车界面的一键删除全部饮品功能

在OrderAdapter中回调垃圾桶图标按钮（删除按钮）的点击事件到ShopFragment当中，调用Ordered\_drinks中clearOrdered\_array()函数。

1. 购物车界面的桌号，人数及外带选择功能

① 在OrderAdapter中回调桌号EditText输入文字改变事件到ShopFragment当中，使桌号不为空，且范围限制在1—30之内。

② 在OrderAdapter中回调人数EditText输入文字改变事件到ShopFragment当中，使人数不为空，且范围限制在1及以上。最后根据更改后的人数更新服务费。

③ 用户可以勾选外带CheckBox。

1. 购物车界面的结算功能

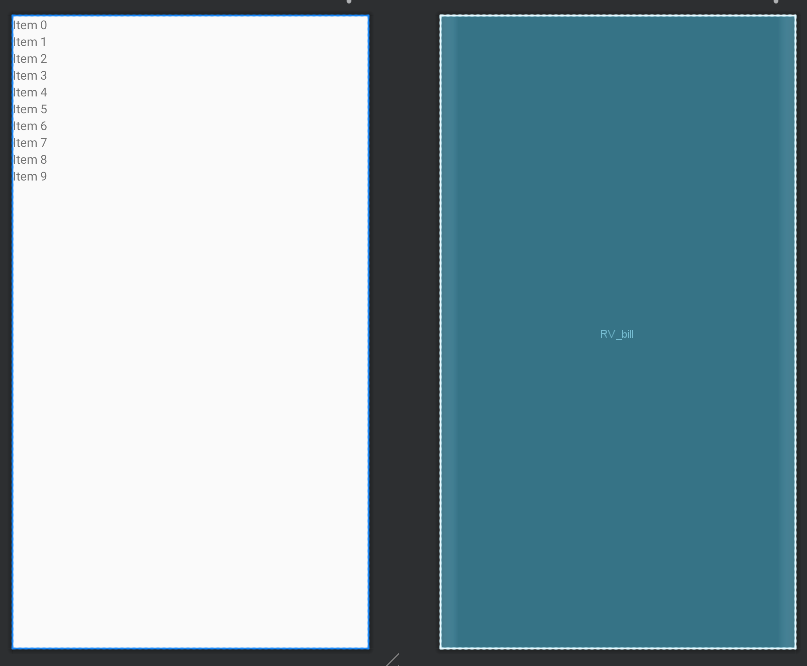
在OrderAdapter中回调结算按钮点击事件到ShopFragment当中，若此时Ordered\_array为空，则提示用户先选取饮品下单；若不为空，获取饮料总价与服务费，计算消费总额，在弹出的结算对话框中显示。用户点击完成支付，创建当前用户的Account对象，调用saveBill()，将此订单信息写入该用户对应文件中，然后调用Ordered\_drinks中的clearOrdered\_array()函数与ShopFragment的refresh函数，关闭该对话框。用户点击取消支付，关闭该对话框，之前点的饮品列表依然存在。



图表 30 结算对话框界面设计

1. 订单界面的数据载入与更新方式

同样使用RecyclerView作为外部框架，通过BillAdapter创建列表项目。通过读取当前用户对应的个人订单信息记录文件，绑定数据到界面。同样与购物车界面一样，重写BillFragment的onHiddenChanged()方法，每当该BillFragment的显示/隐藏状态发生变化时调用该方法，在该方法中调用refresh()函数，重新读取用户对应的个人订单信息记录文件，绑定数据到界面。



图表 31 BillFragment界面设计



图表 32 订单项目界面设计

其中订单编号的前12位由年月日时分秒组成，后五位按用户的历史下单时间，按序生成，第几次下单就是几。总价是这一订单饮料费和服务费的总和

1. 程序测试

|  |  |
| --- | --- |
| 图表 33 当登录或注册时用户名或密码为空弹出提示 | 图表 34 注册时用户名中包含特殊字符弹出提示 |
| 图表 35 注册时密码少于八位弹出提示 | 图表 36 注册时用户名已存在弹出提示 |
| 图表 37 双向列表联动效果 | 图表 38 双向列表联动效果 |
| 图表 39 饮品搜索功能 | 图表 40 当购物车为空时提示用户先选购饮品 |
| 图表 41 当同一饮品的用户定义样式不同时，作为三个项目分别展示，且其价格随规格大小而变化 | 图表 42 人数发生改变时服务费相应改变，人数只能输入大于等于1的数；在购物车界面点击某饮品增加按钮 |
| 图表 43 清空购物车；可选择外带与否 | 图表 44 桌号只能输入1-30范围内的数 |
| 图表 45 点击结算弹出支付对话框 | 图表 46 显示历史订单信息 |

1. 小结

优点：界面美观，功能完整，流程合理。较好得复刻了市面上现有饮料点餐程序的主要功能，用户使用友好。饮品可以自定义规格，温度，甜度。饮品的增添删改十分便捷。类的功能划分较为明确，代码的可拓展性与复用性强。

缺点：部分功能因为时间原因不能完全复刻市面上现有饮料点餐程序，如历史订单界面可以将每一单具体的饮品列表通过点击进行显示。饮品的自定义样式部分也可以进一步拓展，如选择小料等。可以进一步拓展定位不同门店的功能。

困难：实现双列表的联动效果

解决问题；通过对市面上已有饮料点餐程序的界面研究，模块细分，逻辑分析，将复杂的效果拆分成几个核心的实现步骤，完成了双列表的联动效果。

总结：掌握了java语言的语法，结构，运行流程；学会了基本的安卓开发方法；深刻理解了程序设计思路。

1. 参考目录

Android 开发者指南