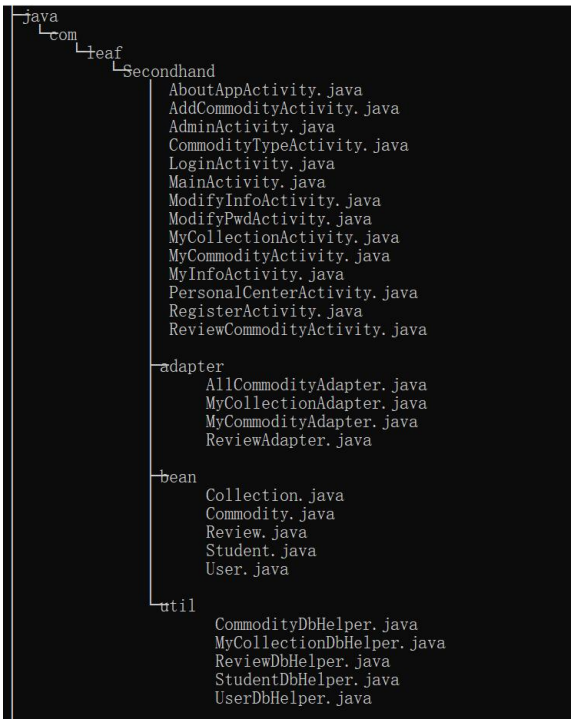
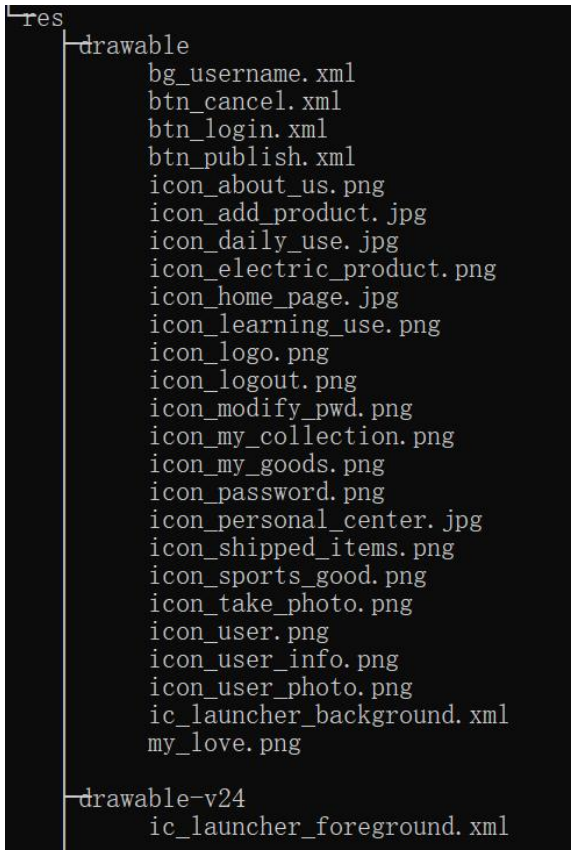
安卓 Pj 结题报告

一.代码结构树形图展示





二.安卓 Pj 功能介绍

2.1 用户的注册与登录

本 pj 主要实现的功能是一个校园二手交易平台，用户在下载完 APP 之后，需要从“新用户注册”入口进去***注册账户***，注册账户需要用户***输入学号与密码***。当用户注册完毕之后，便可以登录进入 APP 的功能界面。

2.2 校园二手交易平台的首页与用户浏览商品

该 APP 的首页如图所示：在首页会显示欢迎（学号名），并在其下方展示学习用品，电子用品，生活用品以及体育用品这四大类，用户可以点击这***四大类物品的图标***并在其中看到各类商品的发布。同时，在这四个图标下面会展示当前发布的***所有商品***。当用户点击某个商品的时候，便会进入商品的***详情页***，可以看到所有当前商品的***名称，价格，分类，联系方式***以及发布者对其的***描述***。用户可以点击界面右上角的红心按钮，从而将商品添加至“***我的收藏***”，以便用户进行回顾。同时，用户可以在添加评论一栏中***输入评论***并点击“提交评论”，该商品就

会展示出评论发布者的***学号，发布时间以及评论内容***，并且评论是***对所有用户可见***的。

2.3 用户发布商品

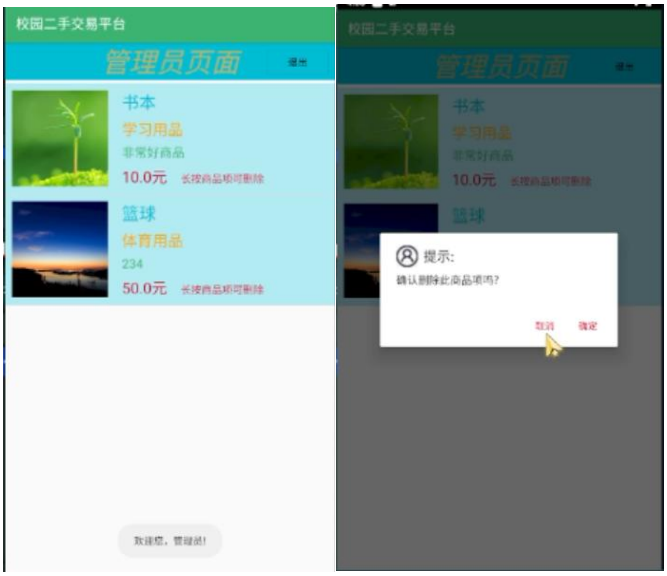
用户可以通过 APP 底栏的“***发布闲置***”进行商品的发布操作。当用户点击之后，会要求用户上传***照片，名称，选择商品的类别***（也就是四大类商品），***价格，联系方式以及描述***。如果用户有任意一个部分没有填写就点击“发布”按钮的话，那么会显示一个通知告诉用户哪里没有填写。当用户发布完商品后，便可以在首页看到自己发布的商品了。

2.4 用户个人信息的维护

用户点击首页的“个人中心”的话便可以看到“个人信息”，“我的发布”，“我的收藏”，“商品发货”（待实现），“修改密码”，“关于系统”，“退出登录”这些按钮。其中，“个人信息”里面支持用户添加并修改***用户姓名，用户专业，联系方式，QQ 号，宿舍地址***这些信息。这些个人信息我的想法是会支持以后的“商品发货”的功能，例如当卖家的商品被买家购买之后，便可以在“商品发货”这里收到提示，告诉卖家购买者的详细信息，以便于进行商品的发货等，而我目前实现的是仅支持买家通过商品详情页上通过卖家的联系方式进行联系。但由于时间关系、课业繁忙等原因没有实现。在“***我的发布***”中可以看到当前用户***发布的所有商品***，在“***我的收藏***”中可以看到用户***收藏的所有商品***。在“***修改密码***”中支持用户***修改当前账户的密码***，并且如果用户两次输入的新密码不一致的话也会提示用户两次的输入不一致。在“关于系统”这里可以看到关于 APP 的***简介以及开发人员***，用户点击“***退出登录***”之后便会重新回到登录界面。

2.5 管理员的功能

当管理员输入对应的“管理员学号”以及“密码”之后，便会来到管理员的页面，在此页面可以看到***所有用户发布的商品，长按可以进行删除操作，***从而可以删除一些违禁物品的发布。



三.代码细节介绍

3.1 整体项目框架

本项目主要使用的安卓四大组件是 ***activity***，并且使用 ***intent 和 bundle*** 在不同 activity 之间传递数据。同时，为所有商品（也就是首页）、用户收藏商品、用户发布的商品以及评论做了相应的 ***adapter*** 去适配前端页面展示（例如 ListView 等）。

3.2 Adapter 的实现细节

接下来我以 AllCommodityAdapter 为例详细解释我的 Adapter 是如何构造的，首先附上 IDE 自动生成的类图：

首先其构造方法是接收一个上下文参数，用于初始化适配器；接着在 setData 中***设置商品列表数据***，并通过 notifyDataSetChanged()***通知数据变化***。

重写的 ***BaseAdapter*** 方法返回数据项的***个数***、***指定位置的数据项以及 ID***。其次定义了 ***ViewHolder*** 静态类，包含了每一个 Item 的所有元素例如***名字，种类，价格***等信息，将图片信息解码为 ***Bitmap*** 对象，从而将商品元素与对应的视图元素***绑定***，并在 getView 方法中使用了 ViewHolder 模式，通过 ***convertView 来重复***

使用视图，提高性能。如果该位置的视图已经存在，直接从 location 中获取，否则通过 layoutInflater.inflate()方法创建新的视图。Clear 方法是***清空***适配器的数据并且通知数据的变化。

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder = null;
    if(location.get(position) == null){
        convertView = layoutInflater.inflate(R.layout.layout_all_commodity,
        Commodity commodity = (Commodity) getItem(position);
        holder = new ViewHolder(convertView,commodity);
        //保存view的位置position
        location.put(position,convertView);
        convertView.setTag(holder);
    }else{
        convertView = location.get(position);
        holder = (ViewHolder) convertView.getTag();
    }
    return convertView;
}
```

至于其他几个 Adapter 的构造，也和上述方法类似，差别主要在于 ***ViewHolder 包含的元素不同***（因为不同页面展示的内容也不尽相似），例如 ReviewAdapter 的话需要展示用户的学号，发表的内容等。它们的 UML 图如下：

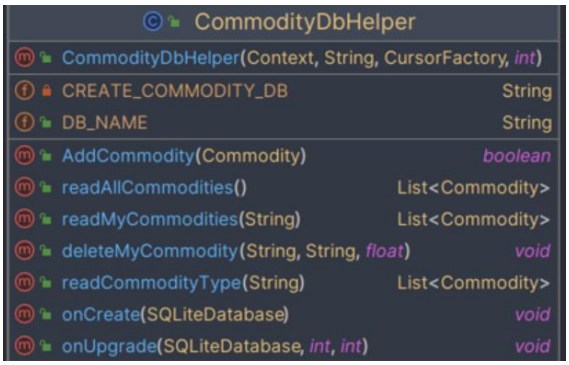
3.3 实体类的实现细节

在实体类的实现中，主要设置了每个实体类对应的***成员变量***以及***修改、获取这些变量的成员函数***，是为了之后***数据库中***对每个实体类进行的***增删改查***实现做铺垫，其本身内容比较简单，可以清晰直观地从 UML 图中读出，故这里不再做过多的阐述（仅附上 Student 实体类的图片，其余实体类和这个基本类似，无非

是每个成员变量有所不同。）

3.4 DataBase 的实现细节

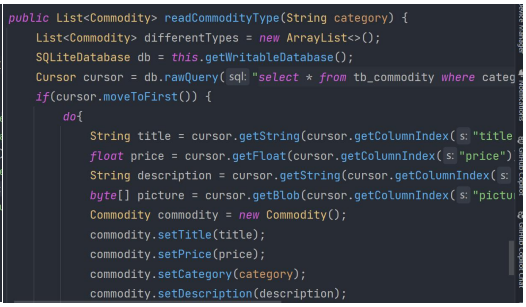
接下来是本项目的底层基础：数据库的实现。因为涉及到的数据量并不是特别大，所以我使用的是 ***Sqlite*** 这款轻量型的数据库。以 CommodityDbHelper 为例，去详细解释下我数据库类的构造。



首先，我定义了一个名为 tb_commodity 的商品表，包含了商品的各种***属性，例如标题、类别、价格、联系方式***等。创建表的 SQL 语句在 CREATE_COMMODITY_DB 常量中进行定义。将商品唯一的 ***ID 作为主键***，且每次创建新产品时 ID 都会***自动递增。***在该类中，其构造方法接受上下文（Context）、数据库名称（name）、游标工厂（factory）和数据库版本（version）作为参数，并通过调用父类的构造方法进行初始化。

该类通过 onCreate()进行第一次创建数据库。onUpgrade()是为了当数据库需要升级时调用。在 AddCommodity()中实现了将商品对象的信息***插入***到数据库中（对应着用户发布商品）。在 readMyCommodities()实现了根据学号***查询用户发布***的商品信息，并***返回一个商品列表***，包含了该用户发布商品的所有信息（对应在个人信息里面“我的发布”）。在 readAllCommodities():实现了***查询所有商品***的功能，并返回一个商品列表包含了所有商品的信息（对应在首页“所有商品的展示”）。在 deleteMyCommodity()实现了根据商品的标题、描述和价格***删除用户发布***的商品（对应在“我的发布”中删除商品）。在 readCommodityType():实现了根据***商品类别查询***商品信息，并返回一个商品列表（对应在首页支持按照商品类别进行筛选）。以下是节选的代码截图，readMyCommodities 是根据学号读取所有商品并按照价格排序，然后获取商品的标题，价格等信息；readCommodityType 是根据商品类别查询并获取商品标题，价格等信息。

```
public List<Commodity> readMyCommodities(String stuId) {
    List<Commodity> myCommodities = new ArrayList<>();
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery( sql: "select * from tb_commodity where stuI
    if(cursor.moveToFirst()) {
        do {
            String title = cursor.getString(cursor.getColumnIndex( s: "titl
            String category = cursor.getString(cursor.getColumnIndex( s: "ca
            float price = cursor.getFloat(cursor.getColumnIndex( s: "price"
            String phone = cursor.getString(cursor.getColumnIndex( s: "phon
            String description = cursor.getString(cursor.getColumnIndex( s:
            byte[] picture = cursor.getBlob(cursor.getColumnIndex( s: "pictu
            Commodity commodity = new Commodity();
            commodity.setTitle(title);
            commodity.setCategory(category);
            commodity.setPrice(price);
            commodity.setDescription(description);
```



```
public List<Commodity> readCommodityType(String category) {
    List<Commodity> differentTypes = new ArrayList<>();
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery( sql: "select * from tb_commodity where categ
    if(cursor.moveToFirst()) {
        do{
            String title = cursor.getString(cursor.getColumnIndex( s: "title
            float price = cursor.getFloat(cursor.getColumnIndex( s: "price")
            String description = cursor.getString(cursor.getColumnIndex( s:
            byte[] picture = cursor.getBlob(cursor.getColumnIndex( s: "pictu
            Commodity commodity = new Commodity();
            commodity.setTitle(title);
            commodity.setPrice(price);
            commodity.setCategory(category);
            commodity.setDescription(description);
```

至于其他几个数据库类的构造，也与 CommodityDbHelper 类似，实现的功能都是基于增删改查实现的，无非是对数据库中***操作的对象有区别***，与***前端绑定的***

UI 不同。例如 Commodity 是对商品进行增删改查，而 User 是对用户进行增加，更新等操作。下面是它们的 UML 类图：

3.5 Activity 的实现

接下来，我们来到了本次项目的重头戏：Activity 的实现。限于篇幅原因，我会挑选几个比较重要的 Activity 进行描述。首先是 APP 的首页：MainActivity。

虽然其内容看着很少，但是重头戏全部放在了 onCreate()中。首先，调用 setContentView()方法设置该 Activity 使用的布局文件为 activity_main.xml，接着通过 findViewById 方法初始化了 ***ListView***（lvAllCommodity 在***首页展示所有商品***）、ImageButton（IbAddProduct 和 IbPersonalCenter 对应着***发布商品，个人中心***）、四个类别按钮（ibLearning、ibElectronic、ibDaily、ibSports，也就是首页展示的***四个商品类别图标***），获取了 TextView（tvStuNumber）的引用（维护登录的用户，实现***数据的持久化***）。同时，在 MainActivity 中有着***初始化数据库***相关操作，创建了 CommodityDbHelper 实例（dbHelper）用于数据库操作，AllCommodityAdapter 实例（adapter）用于绑定商品数据到 ListView，通过 getIntent().getExtras() 获取传递给该 Activity 的数据，主要是用户登录后的用户名，然后通过 TextView 显示欢迎信息。以下是部分代码截取。

关于不同 Activity 之间的跳转，我通过 IbAddProduct.setOnClickListener 方法设置了点击事件，点击按钮会跳转到 AddCommodityActivity，同时传递了学号等用户的个人信息（***将发布的商品与用户绑定***）。通过 IbPersonalCenter.setOnClickListener 方法设置了点击事件，点击按钮会跳转到 PersonalCenterActivity，可以让用户进行***个人信息的维护***。通过 lvAllCommodity.setOnItemClickListener 方法设置了商品列表项的点击事件，点击某个商品后会跳转到 ReviewCommodityActivity，支持用户进行***收藏***、***评论***等操作。为四个类别按钮（ibLearning、ibElectronic、ibDaily、ibSports）设置了点击事件，点击按钮会跳转到 CommodityTypeActivity，支持用户***查看不同类别的商品***。通过数据库操作，***获取全部商品***数据，并使用 AllCommodityAdapter 将数据绑定到 ListView 上。saveBitmapToFile()将商品图片的***字节数组***保存到文件，并返回保存的文件路径。

之后比较重要的便是 AddCommodityActivity 了，实现的功能是支持用户发布商品，也是本项目的核心功能。

首先通过 setContentView()设置了 Activity,其使用的布局文件为 activity_add_commodity.xml，

接着通过 findViewById 方法初始化了一系列***界面组件***，包括显示学号的 TextView（tvStuId）、返回按钮（btnBack）、商品图片按钮（ivPhoto）、商品标题、价格、联系方式、商品描述的输入框（etTitle、etPrice、etPhone、etDescription）、

商品类别的下拉列表框（spType）、发布按钮（btnPublish）等。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_commodity);
    //取出学号
    tvStuId = findViewById(R.id.tv_student_id);
    tvStuId.setText(this.getIntent().getStringExtra( name: "user_id"));
    Button btnBack = findViewById(R.id.btn_back);
    //返回按钮点击事件
    btnBack.setOnClickListener((v) -> { finish(); });
    ivPhoto = findViewById(R.id.iv_photo);
    ivPhoto.setOnClickListener((v) -> {
            Intent intent = new Intent(Intent.ACTION_PICK, uri: null);
            intent.setDataAndType(MediaStore.Images.Media.EXTERNAL_CONTE
            startActivityForResult(intent, requestCode: 1);
    });
    etTitle = findViewById(R.id.et_title);
    etPrice = findViewById(R.id.et_price);
    etPhone = findViewById(R.id.et_phone);
```

关于不同 Activity 间的跳转如下所示：通过 btnBack.setOnClickListener 方法设置了***返回按钮***的点击事件，点击按钮会调用 finish() 方法结束当前 Activity。通过 ivPhoto.setOnClickListener 方法设置了商品图片按钮的点击事件，点击按钮会打开***相册***，用户可以***选择一张图片***作为商品的展示图片。通过 btnPublish.setOnClickListener 方法设置了发布按钮的点击事件，点击按钮会执行***发布商品***的操作。在点击发布按钮时，首先调用 ***CheckInput 方法***检查用户输入的商品***信息是否合法***。在这里将用户上传的图片转换成 ***Bitmap*** 格式，压缩成质量为 100 的 PNG 格式图片，并把输出流转化为二进制数组。如果输入合法，获取用户输入的***商品信息***，包括图片、标题、类别等信息。***创建 Commodity 对象，***将获取到的商品信息设置到对象中。最后调用数据库操作类 CommodityDbHelper 中的 AddCommodity 方法将***商品信息添加***到数据库中。如果添加成功，显***示发布成功***的提示，结束当前 Activity；否则，显示发布失败的提示。通过 onActivityResult 方法处理从相册返回的结果，主要是获取选择的图片的 ***URI***，并将其设置为商品图片。

```
CheckInput()) {
    CommodityDbHelper dbHelper = new CommodityDbHelper(getApplicationContext()
    Commodity commodity = new Commodity();
    //把图片先转化成bitmap格式
    BitmapDrawable drawable = (BitmapDrawable) ivPhoto.getDrawable();
    Bitmap bitmap = drawable.getBitmap();
    //二进制数组输出流
    ByteArrayOutputStream byStream = new ByteArrayOutputStream();
    //将图片压缩成质量为100的PNG格式图片
    bitmap.compress(Bitmap.CompressFormat.PNG, quality: 100, byStream);
    //把输出流转换为二进制数组
    byte[] byteArray = byStream.toByteArray();
    commodity.setPicture(byteArray);
    commodity.setTitle(etTitle.getText().toString());
    commodity.setCategory(spType.getSelectedItem().toString());
    commodity.setPrice(Float.parseFloat(etPrice.getText().toString()));
    commodity.setPhone(etPhone.getText().toString());
```

以上就是关于核心 Activity 的介绍，其他部分 Activity 的 UML 类图如下所示：

3.6 XML 布局的实现

因为从 layout 包下面可以看到有关布局的文件总共十多个，因此在这里也是挑重要的 XML 布局去展示实现细节。首先是 activity_main.xml 文件

TextView（@+id/tv_title）：
显示文本："猜你喜欢！"，位于布局顶部，水平居中

TextView（@+id/tv_student_number）：
用于显示学生学号，位于标题下方，水平居中

ImageButton（@+id/ib_learning_use、@+id/ib_electric_product、@+id/ib_daily_use、@+id/ib_sports_good）：
分别位于学号文本下方，水平排列

TextView（@+id/tv_learning_use、@+id/tv_electric_product、@+id/tv_daily_use、@+id/tv_sports_good）：
分别用于显示四个商品类别的名称，即学习用品、电子用品、生活用品、体育用品，位于相应的类别按钮下方

ListView（@+id/lv_all_commodity）：
用于显示所有商品的列表

Views（@+id/view1、@+id/view2、@+id/view3、@+id/view4）：
用于显示横向和纵向的分割线，通过 View 元素设置

ImageButton（@+id/ib_***home***_page、@+id/ib_add_product、@+id/ib_personal_center）：
三个底部导航按钮，分别为首页、发布闲置、个人中心，位于布局的底部，水平排列

TextView（@+id/tv_home_page、@+id/tv_add_product、@+id/tv_personal_center）：
三个底部导航按钮对应的文本，位于相应的导航按钮上方，水平排列

View（@+id/view4）：
底部导航栏和内容区域的分隔线，高度为 2dp
位于布局的底部。

在 layout_all_commodity.xml 中，基础布局使用了 LinearLayout，

ImageView（@+id/iv_commodity）：
显示商品图片

TextView（@+id/tv_name）：
用于显示商品名称的 TextView

TextView（@+id/tv_type）：
用于显示商品类型的 TextView

TextView（@+id/tv_description）：
用于显示商品描述的 TextView

TextView（@+id/tv_price）：
用于显示商品价格的 TextView。
其余布局就不在此一一介绍了，具体内容都在 layout 包下面。

3.7 用户注册的密码与加密

在课堂视频演示之后，陈辰老师也提出了密码是否有加密的形式，这一点确实是我疏忽了，因此在这份报告并且在代码中补上：

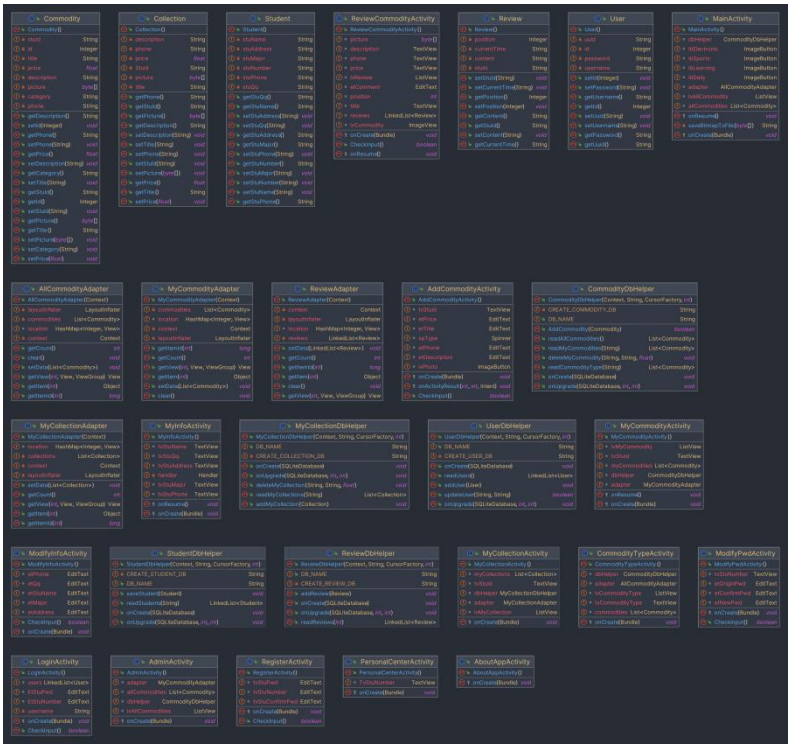
```
public class PasswordHasher {

    public static String hashPassword(String password) {
        try {
            MessageDigest md = MessageDigest.getInstance( algorithm: "SHA-256")
            byte[] hashBytes = md.digest(password.getBytes());
            // Convert the byte array to a hexadecimal representation
            StringBuilder str = new StringBuilder();
            for (byte b : hashBytes) {
                str.append(String.format("%02x", b));
            }
            return str.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            // Handle the exception according to your needs
            return null;
```

首先新建了一个 PasswordHasher 类，接收用户明文密码作为输入，经过 SHA-256 的方式进行加密，并且返回经过 SHA-256 加密后的密码的十六进制表示。

在 RegisterActivty 中，将从前端传来的明文密码加密后再储存到数据库中。同时在登录的时候根据用户名去找到数据库中加密的字符串，若二者都存在且能够对得上则登录成功。

```
String hashedPassword = PasswordHasher.hashPassword(tvStuPwd.getText().toStr
user.setPassword(hashedPassword);
```

附上 IDEA 生成的所有代码的 UML 图：



四.安卓项目开发的心得与体会

本安卓项目毫无疑问是我本学期工作量最大的项目之一，附上 cloc 生成的代码行数截图：

```
Language                     files          blank        comment
          code
-------------------------------------------------------------------------------
-------------
Java                            28            311            396
         2161
XML                             30             36             18
         2110
-------------------------------------------------------------------------------
-------------
SUM:                            58            347            414
         4271
-------------------------------------------------------------------------------
```

除去代码中的重复量，板子等工作（例如数据库的增删改查有一些地方的板子都是几乎相似的，还有布局一些类似的地方），个人完成的代码量也有 ***2000行+***，耗时从***学期中一直到学期末***，期间不知道经历了多少个***不眠夜***）。因为每次写完一个 activity 相关的东西就需要***测试***这个 activity 的功能是否完善，布局方面应该做出如何的调整才能适应这个 activity，有些时候调试了半天发现不是 activity 也不是 xml，而是数据库的问题，又需要进行数据库方法的调整。最后也终于是实现了这样一个校园二手交易平台的成品。作为个人独立开发的安卓 APP 来说，我对于安卓的 activity 的理解又深入了不少，目前已经较为良好地掌握了***静态页面之间的跳转***与 activity 的***生命周期***（例如通过 onResume()实现页面的***自***

动刷新而不是手动刷新，从 ***demo 演示***也可以看到）。该 APP 的***跳转逻辑、设计***都是巨大的工作量，为我在以后的安卓开发道路上打下了夯实的基础。最近看到大厂已经开始高薪挖鸿蒙开发人才了，因为鸿蒙以后不会和安卓兼容了，说不定我以后也会从事这方面的工作呢）。

五.该安卓项目的未来展望

限于开发时间以及个人精力（大三怎么会有 8 门专业课啊啊啊），该项目仍然存在一些可以继续完善的地方。首先是登录方面，可以引入 ***UIS 登录***，类似 elearning（也就是学生自身的学号和密码而无需注册），同时，前端的页面展示使用的还仅仅是安卓最基本的框架布局，给人一种 10 年前的安卓 APP 的感觉，如果未来有机会的话，可以用一些市面上主流的***设计框架***，增加***前端 UI 的美观***程度。并且，可以像市面上主流的二手 APP 一样（例如闲鱼），支持买家向卖家***发起对话***，从而询问更多商品的有关细节。当买家在对话框中选择付款后，卖家应该能看到买家的***详细信息***例如姓名，个人电话，收货地址等信息，从而进行***商品的发货***。最后，管理员的功能也应该可以更强大一些，目前仅支持对于违禁商品的删除，之后可以引入对于多次发布违禁物品的卖家的***惩罚机制***，例如发布的违禁物品到达了一个数量上限便把这个账户关到小黑屋几天，屡次不改后直接封禁账号。甚至可以引入大数据，针对用户经常浏览的商品记录分析用户的喜好，从而在***首页推送***用户更加可能感兴趣的商品。

总结而言，这种校园交易二手 APP 还是具有很高的推广性的，因为其***只支持校内人员登录，***环境较为良好，不像网上主流的二手 APP 一样，使用人员鱼龙混杂甚至还有骗子，而且确实也利于环保，例如二手书的购买出售等。

谢谢助教，老师的阅读，向你们的工作表示感谢！！希望我最后能取到一个不错的成绩呜呜呜，花在这个 pj 上面的时间真的***特别特别长***，并且大三上真的巨忙无比 QAQ）。