# 中山大学移动信息工程学院本科生实验报告

（2017年秋季学期）

课程名称：移动应用开发 任课教师：郑贵锋

|  |  |  |  |
| --- | --- | --- | --- |
| 年级 | 15M1 | 专业（方向） | 移动互联网 |
| 学号 | 15352008 | 姓名 | 蔡荣裕 |
| 电话 | 13727021990 | Email | 897389207@qq.com |
| 开始日期 | 2017/10/18 | 完成日期 | 2017/10/22 |

## 实验题目

Intent、Bundle的使用以及RecyclerView、ListView的应用

## 实验目的

1. 复习事件处理
2. 学习Intent、Bundle在Activity跳转中的应用
3. 学习RecyclerView、ListView以及各类适配器的用法

## 实验内容

本次实验模拟实现一个商品表， 有两个界面， 第一个界面用于呈现商品，如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 图1界面为应用启动后看到的第一个界面(数据在素材中有给出) 2. 点击右下方的悬浮按钮可以切换到购车如图2 3. 上面两个列表点击任意一项后，可以看到详细的信息如图3 |  |  |  |
| 图1 | 图2 | 图3 |

**实验要求布局方面的要求**

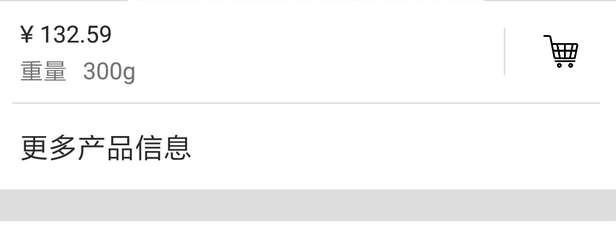
1. 商品表界面

每一项为一个圆圈和一个名字，圆圈与名字均竖直居中。圆圈中为名字的首字母，首字母要处于圆圈的中心，首字母为白色，名字为黑色，圆圈的颜色自定义即可，建议用深色的颜色，否则白色的首字母可能看不清。

1. 购物车列表界面

在商品表界面的基础上增加一个价格，价格为黑色。

1. 商品详清界面顶部

顶部占整个界面的1/3。每个商品的图片在商品数据中已给出，图片与这块view等高。返回图标处于这块View的左上角，商品名字处于左下角，星标处于右下角，它们与边距都有一定距离，自己调出合适的距离即可。需要注意的是，返回图标与名字左对齐，名字与星标底边对齐。这一块建议大家使用Relative Layout实现以便熟悉Relativelayout的使用。

1. 商品详情界面中部

使用的黑色argb编码值为#D5000000, 稍微偏灰色一点的重量”、"300g"的argb编码值为#8A000000。注意，价格那一栏的下边有以条分割线， argb编码值为#1E000000, 右边购物车符号的左边也有一条分割线， argb编码值也是#1E000000, 这条分割线要求高度与聊天符号的高度一致，并且竖直居中。字体的大小看着调就可以了。“更多资料＇底部的分割线高度自己定， argb编码值与前面的分割线一致。

1. 商品详情页面底部

这个没什么说的，内容和样式都很清楚。

1. 这次的两个界面顶部都没有标题栏，要用某些方法把它们去掉。

**逻辑方面的要求：**

1. 使用RecyclerView实现商品列表。点击商品列表中的某一个商品会跳转到该商品详情界面，呈现该商品的详细信息； 长按商品列表中的第个商品会删除该商品，并且弹出Toast提示“移除第i个商品”。
2. 点击右下方的FloatingActionButton,从商品列表切换到购物车或从购物车切换到商品列表，并且FloatingActionButton的图片要做相应改变。可通过设置RecyclerView不可见ListView可见来实现从商品列表切换到购物车。可通过设置RecyclerView可见ListView不可见来实现从购物车切换到商品列表。(此处为了附加功能魔改)
3. 使用ListView实现购物车。点击购物车的某一个商品会跳转到商品详情界面，呈现该商品的详细信息；长按购物中的商品会弹出对话框询问是否移除该商品，点击确定则移除该商品，点击取消则对话框消失。注意对话框中的商品名为被长按的商品。
4. 商品详清界面中点击返回圈标会返回上—层，点击星标会切换状态，如果原先是空心星星，则会变成实心星星；如果原先是实心星星，则会变成空心星星。点击购物车圈标会将该商品添加到购物车中并弹出Toast提示“商品已添加到购物车”。

注不要求判断购物车是否已有该商品即如果已有一件该商品添加之后则显示两个即可。未退出商品详细界面时点击多次购物车图标可以只添加一件商品也可以添加多件到购物车中(此处再次魔改)

## 课堂实验结果

1. 实验截图：
   * 1. 虚拟机上效果如下：

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. 关键步骤：

实验中修改了一下使用了3个界面相互转跳用，此处所有数据存储在Good商品类中，由Data类进行管理，这两个类的实现见课后部分Good实现了界面中传参数。

1. 首先是主界面
   1. 首先是主界面问题，主界面是商品列表界面，需要使用RecyclerView实现，此外还需要一个FloatingActionButton。以下是主界面布局文件：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context="cn.chonor.lab3.MainActivity">   <android.support.v7.widget.RecyclerView  android:id="@+id/myRecyclerView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.0" />   <android.support.design.widget.FloatingActionButton  android:id="@+id/floatingActionButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="8dp"  android:layout\_marginEnd="8dp"  android:layout\_marginLeft="8dp"  android:layout\_marginRight="8dp"  android:layout\_marginStart="8dp"  android:layout\_marginTop="8dp"  android:clickable="true"  android:visibility="visible"  app:backgroundTint="@android:color/background\_light"  app:elevation="10dp"  app:fabSize="normal"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintHorizontal\_bias="0.95"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.95"  app:rippleColor="@android:color/background\_light"  app:srcCompat="@mipmap/shoplist"  tools:layout\_constraintLeft\_creator="1"  tools:layout\_constraintTop\_creator="1" />  </android.support.constraint.ConstraintLayout> |

此处需要的这两个控件都需要在build.gradle添加依赖，不过Android studio会在你引入控件的时候自动添加依赖：

|  |
| --- |
| compile 'com.android.support:recyclerview-v7:26.+'  compile 'com.android.support:design:26.+' |

这样就创建了RecyclerView，然后因为在java文件中再填充数据，所以id是一定要设的。在java文件中获得这个RecyclerView之后，使用Adapter为这个ListView填充数据，此处因为RecyclerView所以我们需要自定义一个Adapter。

* 1. RecyclerView要求必须使用ViewHolder模式，一般我们在使用过程中，都需要去建立一个新的ViewHolder然后作为泛型传入Adapter。那么想要建立通用的Adapter，必须有个通用的ViewHolder。（此处定义在我们自定义的Adapter中以防止冲突到之后的ListView）

|  |
| --- |
| public static class ViewHolder extends RecyclerView.ViewHolder {  private View mConvertView;//存储 list\_Irem  private SparseArray<View>mView;//存储 list\_Item的子View  public ViewHolder(Context context,View view,ViewGroup viewGroup){  super(view);  mConvertView=view;  mView = new SparseArray<View>();  }  //获取viewHolder实例  public static ViewHolder get (Context context,ViewGroup viewGroup,int layoutId){  View itemView = LayoutInflater.*from*(context).inflate(layoutId,viewGroup,false);  ViewHolder viewHolder = new ViewHolder(context,itemView,viewGroup);  return viewHolder;  }  //viewHolder尚未将子View缓存到SparseArray数组中时仍然需要通过findViewByld()创建View对象如果已缓存直接返回  public <T extends View>T getView(int viewId){  View view=mView.get(viewId);  if(view == null){  view=mConvertView.findViewById(viewId);  mView.put(viewId,view);  }  return (T)view;  } } |

有了Adapter用的ViewHolder之后，Adapter也可以开始写。

* 1. 我们的每次使用过程中，针对的数据类型肯定是不同的，那么这里最好是要引入泛型代表， 但是因为此处为了简化一下直接实例化，为我们之后需要用到的一个数据类，此时也省的重载。

Adapter扮演着两个角色。一是，根据不同ViewType创建与之相应的的Itemlayout, 二是，访问数据集合并将数据绑定到正确的View上。这就需要我们重写以下两个函数:

public ViewHolder onCreateViewHolder(final ViewGroup viewGroup,int viewType)

public void onBindViewHolder(final ViewHolder viewHolder, int position)

另外还需要重写另一个方法，像ListView-Adapter那样，同样地告诉RecyclerView-Adapter列表Items的总数这就需要public int getItemCount()

|  |
| --- |
| public class CommonAdapter extends RecyclerView.Adapter<CommonAdapter.ViewHolder> {  private Context mContext;  private List<Good> mDatas;  private int mLayoutId;  //构造函数  public CommonAdapter(Context mContext,int mLayoutId,List<Good>mDatas) {  this.mLayoutId=mLayoutId;  this.mContext = mContext;  this.mDatas = mDatas;  }  @Override  public ViewHolder onCreateViewHolder(final ViewGroup viewGroup,int viewType) {  ViewHolder viewHolder=ViewHolder.*get*(mContext,viewGroup,mLayoutId);  return viewHolder;  }  //将数据与界面进行绑定的操作  @Override  public void onBindViewHolder(final ViewHolder viewHolder, int position) {  convert(viewHolder,mDatas.get(position));  if(mOnItemClickListener !=null){  viewHolder.itemView.setOnClickListener(new View.OnClickListener(){  @Override  public void onClick(View view){  mOnItemClickListener.onClick(viewHolder.getAdapterPosition());  }  });  viewHolder.itemView.setOnLongClickListener(new View.OnLongClickListener() {  @Override  public boolean onLongClick(View view) {  mOnItemClickListener.onLongClick(viewHolder.getAdapterPosition());  return false;  }  });  }  }  //获取数据的数量  @Override  public int getItemCount() {  return mDatas.size();  }  } |

此处我们需要重写适配器中的convert方法进行数据绑定，此处因为我们直接实现了专用的Adapter所以此时可以直接在Adapter类中写上这个方法。

|  |
| --- |
| public void convert(ViewHolder viewHolder, Good good){  TextView first=viewHolder.getView(R.id.*first*);  first.setText(good.getGoodName\_First());  TextView name=viewHolder.getView(R.id.*name*);  name.setText(good.getGoodName()); } |

RecyclerView没有On ltemClicklistener方法需要在适配器中实现。实现的方法为：在Adapter中设置一个监听器当itemView被点击时候调用该监听器并且将itemView的position作为参数传递出去。

|  |
| --- |
| //注册点击事件  public interface OnItemClickListener{  void onClick(int position);  void onLongClick(int position); } private OnItemClickListener mOnItemClickListener = null; public void setOnItemClickListener(OnItemClickListener onItemClickListener){  this.mOnItemClickListener=onItemClickListener; } |

此处对应的点击实现包括public void onBindViewHolder (final ViewHolder viewHolder, int position)中的onClick和onLongClick。

* 1. 此时我们可以在main中创建RecyclerView，我们先要创建一个用来代表每一项的item的xml如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:background="?android:attr/selectableItemBackground"  android:clickable="true"  android:focusable="true"  android:id="@+id/items"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:paddingTop="25dp"  android:paddingBottom="25dp"  android:paddingLeft="15dp"  android:paddingRight="15dp"  >  <TextView  android:id="@+id/first"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:background="@drawable/buttonsytle"  android:textAlignment="center"  android:textColor="@android:color/white"  android:textSize="25sp"  android:textStyle="bold" />  <TextView  android:id="@+id/name"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:padding="10dp"  android:textColor="@android:color/black"  android:textSize="20sp" /> </LinearLayout> |

此时我们可以开始创建RecyclerView并进行动画设置

|  |
| --- |
| mRecyclerView = (RecyclerView)findViewById(R.id.*myRecyclerView*); //初始化RecyclerView  mRecyclerView.setLayoutManager(new LinearLayoutManager(this));//这里用线性显示 mRecyclerView.setHasFixedSize(true); //初始化适配 //mRecyclerView.setAdapter(commonAdapter); commonAdapter = new CommonAdapter(MainActivity.this,R.layout.*items*,data.getGood\_list()); animationAdapter=new ScaleInAnimationAdapter(commonAdapter);  animationAdapter.setDuration(1000); //设置开始动画 mRecyclerView.setAdapter(animationAdapter);//填充数据 mRecyclerView.setItemAnimator(new OvershootInLeftAnimator()); mRecyclerView.getItemAnimator().setRemoveDuration(300); //设置移除延时 |

到此为止我们主界面的RecyclerView创建和准备工作完成。

然后我们需要为RecyclerView创建对应的点击事件打开新的界面和删除，此处打开新的界面传入两个类数据都是基于商品类实现的商品列表和购物车列表：

|  |
| --- |
| commonAdapter.setOnItemClickListener(new CommonAdapter.OnItemClickListener(){  @Override  public void onClick(int position){  if(position >= 0) {  Intent i = new Intent(MainActivity.this, Good\_Info.class);  Bundle bundle = new Bundle();  bundle.putInt("position", position);  bundle.putParcelableArrayList("data", data.getGood\_list());  bundle.putParcelableArrayList("cart", data.getCart\_list());  i.putExtra("mainActivity", bundle);  startActivityForResult(i, 0);  }  }  @Override  public void onLongClick(int position) {  if (position >= 0) {  data.removeGood\_list\_index(position);  animationAdapter.notifyItemRemoved(position);  mRecyclerView.setItemViewCacheSize(data.getGood\_list().size());  Toast.*makeText*(MainActivity.this, "移除了第"+String.*valueOf*(position)+"件商品", Toast.*LENGTH\_SHORT*).show();  }  } }); |

* 1. 此时主界面还差最后一步就完工了就是FloatingActionButton的点击事件和对于转跳之后回传数据的处理。

FloatingActionButton的点击事件：

|  |
| --- |
| floatingActionButton = (FloatingActionButton) findViewById(R.id.floatingActionButton); //初始化按钮  floatingActionButton.setOnClickListener(new View.OnClickListener() {  @Override //跳转至 购物车界面  public void onClick(View view) {  Intent i = new Intent(MainActivity.this, Shoppingcart.class);  Bundle bundle = new Bundle();//传输数据  bundle.putParcelableArrayList("data",data.getGood\_list());  bundle.putParcelableArrayList("cart",data.getCart\_list());  i.putExtra("mainActivity",bundle);  startActivityForResult(i,1);  } }); |

转跳后回传数据的处理：

|  |
| --- |
| protected void onActivityResult(int requestCode, int resultCode, Intent intent) {  super.onActivityResult(requestCode, resultCode, intent); //数据回传  if (resultCode == *RESULT\_OK*) {  Bundle extras = intent.getExtras();  ArrayList<Good> tmp= extras.getParcelableArrayList("data");  ArrayList<Good> tmp1= extras.getParcelableArrayList("cart");  data.getGood\_list().clear();  animationAdapter.notifyDataSetChanged();  mRecyclerView.removeAllViews();  data.setGood\_list(tmp);  data.setCart\_list(tmp1);  init();  init\_listener();  } } |

1. 第二个部分购物车界面。
   1. 首先还是购物车界面的Layout，此处和之前的主界面是完全不同的两个界面用于实现附加功能。此处需要在tools:context添加对应的activity的类，同时manifest要注册这个类。

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"   tools:context="cn.chonor.lab3.Shoppingcart">  <android.support.design.widget.FloatingActionButton  android:id="@+id/floatingActionButton1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="8dp"  android:layout\_marginEnd="8dp"  android:layout\_marginLeft="8dp"  android:layout\_marginRight="8dp"  android:layout\_marginStart="8dp"  android:layout\_marginTop="8dp"  android:clickable="true"  android:visibility="visible"  app:backgroundTint="@android:color/background\_light"  app:elevation="10dp"  app:fabSize="normal"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintHorizontal\_bias="0.95"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.95"  app:rippleColor="@android:color/background\_light"  app:srcCompat="@mipmap/mainpage"  tools:layout\_constraintLeft\_creator="1"  tools:layout\_constraintTop\_creator="1" />  <ListView  android:id="@+id/myListView"  android:layout\_width="0dp"  android:layout\_height="0dp"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintHorizontal\_bias="0.0"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintVertical\_bias="0.0" /> </android.support.constraint.ConstraintLayout> |

此处还需Item需要的xml文件，但是跟之前的类似不在展示。

* 1. 此处使用ListView的自定义Adapter进行填充数据，具体看课后实现部分。
  2. 首先我们从主界面中接收传过来的数据，然后增加上购物车那个特有的一行

|  |
| --- |
| private void get\_set\_data(){ //接收数据  Bundle extras = getIntent().getBundleExtra("mainActivity");  if (extras != null) {  ArrayList<Good> tmp = extras.getParcelableArrayList("data");  ArrayList<Good> tmp1 = extras.getParcelableArrayList("cart");  data.setGood\_list(tmp);  data.setCart\_list(tmp1);  }  //购物车时增加一列空  if (data.getCart\_list().size() == 0 || !data.getCart\_list().get(0).getGoodName().equals("购物车")) {  Good good = new Good("购物车", "价格", " ", " ");  good.setGoodFisrt("\*");  data.addCart\_list(0, good);  } } |

* 1. 之后初始化控件和ListView

|  |
| --- |
| private void init(){//初始化  builder = new AlertDialog.Builder(this);  listView = (ListView) this.findViewById(R.id.*myListView*);  myAdapter = new MyAdapter(this, data.getCart\_list(), 0);  listView.setAdapter(myAdapter);  floatingActionButton = (FloatingActionButton) findViewById(R.id.*floatingActionButton1*); } |

* 1. 然后注册监听事件，此处传递到商品信息界面的也是两个good类list

|  |
| --- |
| floatingActionButton.setOnClickListener(new View.OnClickListener() { //按钮转跳  @Override  public void onClick(View view) { //设置数据  Intent i = new Intent();  Bundle bundle = new Bundle();  bundle.putParcelableArrayList("data", data.getGood\_list());  bundle.putParcelableArrayList("cart", data.getCart\_list());  i.putExtras(bundle);  setResult(*RESULT\_OK*, i);  finish();  } }); listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  @Override  public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {//单击进入详细  if (data.getCart\_list().get(i).getGoodName().equals("购物车")) {  } else {//数据传输  Intent intent = new Intent(Shoppingcart.this, Good\_Info.class);  Bundle bundle = new Bundle();  bundle.putInt("position", i);  bundle.putParcelableArrayList("data", data.getGood\_list());  bundle.putParcelableArrayList("cart", data.getCart\_list());  intent.putExtra("shoppingcart", bundle);  startActivityForResult(intent, 2);  }  } }); listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {//长按提示删除  @Override  public boolean onItemLongClick(AdapterView<?> adapterView, View view, int i, long l) {  if (data.getCart\_list().get(i).getGoodName().equals("购物车")) {  } else {  alertdialog\_build(i);  AlertDialog dialog = builder.create(); //完成创建AlertDialog并显示  dialog.show();  }  return true;  } });  public void alertdialog\_build(final int position) {  //alertdialog 初始化  builder.setTitle("移除商品 ");  final String tmp = data.getCart\_list().get(position).getGoodName();  builder.setMessage("从购物车移除" + tmp);//设置显示文本  builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {  @Override //设置确定按钮动作  public void onClick(DialogInterface dialogInterface, int i) {  data.removeCart\_list\_index(position);  myAdapter.notifyDataSetChanged();  Toast.*makeText*(getApplicationContext(), "成功删除 " + tmp, Toast.*LENGTH\_SHORT*).show();  }  });  builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {  @Override //设置取消按钮动作  public void onClick(DialogInterface dialogInterface, int i) {  Toast.*makeText*(getApplicationContext(), "您选择了取消", Toast.*LENGTH\_SHORT*).show();  }  });  builder.setCancelable(true); //允许取消 } |

* 1. 之后设置一下对于商品信息类回传数据的处理就大功告成

|  |
| --- |
| @Override  protected void onActivityResult(int requestCode, int resultCode, Intent intent) { //接收返回数据  super.onActivityResult(requestCode, resultCode, intent);  if (resultCode == *RESULT\_OK*) {  Bundle extras = intent.getExtras();  ArrayList<Good> tmp= extras.getParcelableArrayList("data");  ArrayList<Good> tmp1= extras.getParcelableArrayList("cart");  data.setGood\_list(tmp);  data.setCart\_list(tmp1);  myAdapter = new MyAdapter(this, data.getCart\_list(), 0);  listView.setAdapter(myAdapter); //接收返回数据重新建表  } } |

1. 商品信息界面
   1. 商品信息界面xml的建立，此处使用RelativeLayout，同时和之前一样也需要组成绑定对应的activity类：

此处因为数据较多直接展示一下布局结果：



同样的这里也有一个listView,所以也需要一个Item的xml，不过这个简单不展示

* 1. 然后设计商品信息界面对应的activity类，首先是初始化和建立那个固定的listView，这里为了使得少写一个适配器，直接把这个固定的listView当商品装进去。

|  |
| --- |
| private void init(){ //初始化  String []hits ={"一键下单","分享商品","不感兴趣","查看更多商品促销信息"};  hit=new ArrayList<>();  for(int i=0;i<4;i++){//假装商品数据装进去  Good good = new Good();  good.setGoodName(hits[i]);  hit.add(good);  }  imageView=(ImageView)findViewById(R.id.*imageView*);  star = (ImageView)findViewById(R.id.*imageView2*);  cart = (ImageView)findViewById(R.id.*imageView3*);  back = (ImageView)findViewById(R.id.*imageView4*);  name = (TextView)findViewById(R.id.*info\_name*);  price = (TextView)findViewById(R.id.*info\_price*);  info = (TextView)findViewById(R.id.*info\_info*);  list = (ListView)findViewById(R.id.*myListView1*); } |

* 1. 然后是接收数据，此处要分别判断数据是来自主界面还是购物车界面。

|  |
| --- |
| private void get\_set\_data(){  Bundle extras\_main = getIntent().getBundleExtra("mainActivity"); //获取数据 主界面  Bundle extras\_shop = getIntent().getBundleExtra("shoppingcart");//购物车  if (extras\_main != null||extras\_shop != null) {  if(extras\_main != null) { //主界面  flag = true; //设置flag  id=extras\_main.getInt("position"); //获得数据位置  ArrayList<Good>tmp = extras\_main.getParcelableArrayList("data"); //获得数据  ArrayList<Good>tmp1 = extras\_main.getParcelableArrayList("cart");  data.setGood\_list(tmp);//设置数据  data.setCart\_list(tmp1);  }  else{  flag=false;//购物车//设置flag  id=extras\_shop.getInt("position");//获得数据位置  ArrayList<Good>tmp = extras\_shop.getParcelableArrayList("data");//获得数据  ArrayList<Good>tmp1 = extras\_shop.getParcelableArrayList("cart");  data.setGood\_list(tmp);//设置数据  data.setCart\_list(tmp1);  }  set\_info();//设置界面  } } |

* 1. 然后根据传进来的数据设置对应的商品信息,此处对于图片的处理设置比较麻烦一点，同时因为能够记录收藏情况所以那个星星也要根据传入的收藏情况进行处理。

|  |
| --- |
| private void set\_info(){  String names; //获商品名称  if(flag)names=data.getGood\_list\_index(id).getGoodName();  else names=data.getCart\_list\_index(id).getGoodName();   //匹配图片  if(names.equals("Enchated Forest"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*enchatedforest*));  else if(names.equals("Arla Milk"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*arla*));  else if(names.equals("Devondale Milk"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*devondale*));  else if(names.equals("Kindle Oasis"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*kindle*));  else if(names.equals("waitrose 早餐麦片"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*waitrose*));  else if(names.equals("Mcvitie's 饼干"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*mcvitie*));  else if(names.equals("Ferrero Rocher"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*ferrero*));  else if(names.equals("Maltesers"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*maltesers*));  else if(names.equals("Lindt"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*lindt*));  else if(names.equals("Borggreve"))imageView.setImageDrawable(getResources().getDrawable(R.mipmap.*borggreve*));   //设置 价格 物品名称 物品提示  if(flag) {//商品列表转跳时  name.setText(data.getGood\_list\_index(id).getGoodName());  price.setText(data.getGood\_list\_index(id).getGoodPrice());  if (data.getGood\_list\_index(id).getStar() == 1)  star.setImageDrawable(getResources().getDrawable(R.mipmap.*full\_star*));  else star.setImageDrawable(getResources().getDrawable(R.mipmap.*empty\_star*));  info.setText(data.getGood\_list\_index(id).getGoodTypes() + " " + data.getGood\_list\_index(id).getGoodInfo());  }else{//购物车列表转跳时  name.setText(data.getCart\_list\_index(id).getGoodName());  price.setText(data.getCart\_list\_index(id).getGoodPrice());  if (data.getCart\_list\_index(id).getStar() == 1)  star.setImageDrawable(getResources().getDrawable(R.mipmap.*full\_star*));  else star.setImageDrawable(getResources().getDrawable(R.mipmap.*empty\_star*));  info.setText(data.getCart\_list\_index(id).getGoodTypes() + " " + data.getCart\_list\_index(id).getGoodInfo());  }  //设置下面4个items  MyAdapter myAdapter=new MyAdapter(this,hit,1);  list.setAdapter(myAdapter); } |

* 1. 注册不同按钮对应的点击事件

|  |
| --- |
| private void init\_listener(){  star.setOnClickListener(new View.OnClickListener() { //收藏按钮监听  @Override  public void onClick(View view) {  if(flag) {//商品列表转跳时  if (data.getGood\_list\_index(id).getStar() == 1) {//判断是否收藏  star.setImageDrawable(getResources().getDrawable(R.mipmap.*empty\_star*));  Toast.*makeText*(Good\_Info.this,"成功将"+data.getGood\_list\_index(id).getGoodName()+"取消收藏",Toast.*LENGTH\_SHORT*).show();  } else {  star.setImageDrawable(getResources().getDrawable(R.mipmap.*full\_star*));  Toast.*makeText*(Good\_Info.this,"成功将"+data.getGood\_list\_index(id).getGoodName()+"加入收藏",Toast.*LENGTH\_SHORT*).show();  }  data.setGood\_list\_Stat(id); //同步购物车列表收藏  for(int i=0;i<data.getCart\_list().size();i++){  if(data.getGood\_list\_index(id).getGoodName().equals(data.getCart\_list\_index(i).getGoodName())){  data.setCart\_list\_Stat(i);  }  }  }else{//购物车列表转调试  if (data.getCart\_list\_index(id).getStar() == 1) {//判断是否收藏  star.setImageDrawable(getResources().getDrawable(R.mipmap.*empty\_star*));  Toast.*makeText*(Good\_Info.this,"成功将"+data.getCart\_list\_index(id).getGoodName()+"取消收藏",Toast.*LENGTH\_SHORT*).show();  } else {  star.setImageDrawable(getResources().getDrawable(R.mipmap.*full\_star*));  Toast.*makeText*(Good\_Info.this,"成功将"+data.getCart\_list\_index(id).getGoodName()+"加入收藏",Toast.*LENGTH\_SHORT*).show();  }  data.setCart\_list\_Stat(id); //同步商品列表收藏  for(int i=0;i<data.getGood\_list().size();i++){  if(data.getCart\_list\_index(id).getGoodName().equals(data.getGood\_list\_index(i).getGoodName())){  data.setGood\_list\_Stat(i);  }  }  }  }  });  cart.setOnClickListener(new View.OnClickListener() { //购物车按钮  @Override  public void onClick(View view) {  if(flag){  int incart= -1; //确定购物车里有没有  for(int i=0;i<data.getCart\_list().size();i++){  if(data.getGood\_list\_index(id).getGoodName().equals(data.getCart\_list\_index(i).getGoodName()))  incart=i;  }  if(incart== -1) //没有加入购物车  data.addCart\_list(data.getGood\_list\_index(id));  else //有数量+1  data.setCart\_list\_Cnt(incart,data.getCart\_list\_index(incart).getCnt()+1);  Toast.*makeText*(Good\_Info.this,"成功将"+data.getGood\_list\_index(id).getGoodName()+"加入购物车",Toast.*LENGTH\_SHORT*).show();  } else {  data.setCart\_list\_Cnt(id,data.getCart\_list\_index(id).getCnt()+1);  Toast.*makeText*(Good\_Info.this,"成功将"+data.getCart\_list\_index(id).getGoodName()+"加入购物车",Toast.*LENGTH\_SHORT*).show();  }  }  });   back.setOnClickListener(new View.OnClickListener() { //返回按钮监听  @Override  public void onClick(View view) { //数据返回  Intent i = new Intent(Good\_Info.this, Shoppingcart.class);  Bundle bundle = new Bundle();  bundle.putParcelableArrayList("data",data.getGood\_list());  bundle.putParcelableArrayList("cart",data.getCart\_list());  i.putExtras(bundle);  setResult(*RESULT\_OK*,i);  Toast.*makeText*(Good\_Info.this,"back",Toast.*LENGTH\_SHORT*).show();  finish();  }  }); |

至此界面全部完成，可以上虚拟机运行。

1. 实验中遇到的困难和解决思路：
2. 最开始写java文件的时候发现想用很多控件名字打出来都是cannot resolve symbol，然后百度半天import，然后发现可以直接使用alt+enter自动补全
3. 一开始修改好界面之后，在虚拟机中运行发现只要一开启app，屏幕焦点就自动集中在密码输入框上，然后输入密码的提示消失。于是百度之，最终解决方法是在xml文件中使得一开始把焦点移走，此处选择的是把焦点移到完全没有影响的地方上，在xml文件中增加如下两句:

|  |
| --- |
| android:focusable="true"  android:focusableInTouchMode="true" |

增加之后显示效果正常。

## 课后实验结果

* + 1. 实验中要求点击照片弹出拍照和从相册选择这两个选项，然而并不要求实现，只要求输出提示，于是百度了一下如何调出相机和相册，此处只需要在AlertDialog的Items，点击中多增加几句即可，使用Intent 打开相机或者相册，然后传入参数调用处理函数。

|  |
| --- |
| if(i==0){//额外 打开相机  Intent intent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  startActivityForResult(intent, *CAMERA*); }else {//额外 打开相册  Intent intent = new Intent(Intent.*ACTION\_PICK*);  intent.setType("image/\*");  startActivityForResult(intent, *IMAGE*); } |

处理函数针对相机和相册分别处理，此处的相机处理比较简单只需要一句话丢给ImageView显示就可以了，此处因为我们不储存所以很简单，从相册获取，这个代码是像百度学习的，因为不同的系统需要不用的调用处理函数，所以此处也是不太清楚，于是就学习了适合API版本26的做法，使用输入流处理。

|  |
| --- |
| protected void onActivityResult(int requestCode, int resultCode, Intent data) {  if (requestCode == *CAMERA* && resultCode == *RESULT\_OK*) {  Bitmap photo = (Bitmap) data.getExtras().get("data");  mImage.setImageBitmap(photo); //设置图片  para = mImage.getLayoutParams(); // 设置自动宽高   para.height = height;  para.width = width;  mImage.setLayoutParams(para);   }else if(requestCode == *IMAGE* && resultCode == *RESULT\_OK*){  try { //此处提示需要抛出异常 所以加了  InputStream inputStream = getContentResolver().openInputStream(data.getData());  Bitmap photo = BitmapFactory.*decodeStream*(inputStream); //使用输入流转化图片  mImage.setImageBitmap(photo);  para = mImage.getLayoutParams();// 设置自动宽高   para.height = height;  para.width = width;  mImage.setLayoutParams(para);  }catch (FileNotFoundException e) {  e.printStackTrace();  }  }  super.onActivityResult(requestCode, resultCode, data); } |

此处需要设置固定宽高，所以要先获取图片宽高，测试了百度上的两种方式最后发现用View.MeasureSpec测量宽高，之后设置图片的时候LayoutParams固定照片大小比较方便，一下为测量图片宽高代码

|  |
| --- |
| int i = View.MeasureSpec.*makeMeasureSpec*(0, 0);  int j = View.MeasureSpec.*makeMeasureSpec*(0, 0); mImage.measure(i,j); height=mImage.getMeasuredHeight(); width=mImage.getMeasuredWidth(); |

* + 1. TextInputLayout控件在输入结束时直接判断是不是输了空账号密码，而不是需要经过button
* TextInputLayout和EditText控件

|  |
| --- |
| et\_id = (EditText) findViewById(R.id.*textid*);  et\_password = (EditText) findViewById(R.id.*textpassword*); til\_id = (TextInputLayout) findViewById(R.id.*textInputLayout*); til\_password = (TextInputLayout) findViewById(R.id.*textInputLayout1*); |

* 增加输入框输入的事件监听(此处展示输入学号密码同理)，此处同时使得提示更加友好，选择教职工的时候提示从学号xxxx变成教职工号xxxx

|  |
| --- |
| et\_id.addTextChangedListener(new TextWatcher() {  @Override  public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {  }  @Override  public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {  }  @Override  public void afterTextChanged(Editable editable) { //额外//输入完毕自主判断  if (editable.length() == 0) {  til\_id.setErrorEnabled(true);  til\_id.setError(rb\_string\_hit+"不能为空");  } else {  til\_id.setErrorEnabled(false);  }  } }); |

* 此次同时调整了一下提示，使得提示更友好：

|  |
| --- |
|  |

切换的时候回更改提示。

* + 1. 既然做了登录，所以做了一个登录后转跳的界面。页面转跳也很简单通过Intent就可以实现转跳，不过我们需要写一个新的class和一个新的xml布局文件，作为转跳用，此处增加的为学生登录成功后才进行转跳，同时使用Timer和TimerTask 进行转跳延时(只是为了留存提示。
* 此处需要先修改两个地方

首先在AndroidManifest中注册我们这个心的class

|  |
| --- |
| <activity android:name=".AfterLogin"></activity> |

然后在class中和xml文件中分别设定绑定

|  |
| --- |
| setContentView(R.layout.*after\_login*);  tools:context="cn.chonor.lab2.AfterLogin" |

* 新增的转跳以及延时代码学生登录成功后才进行转跳

|  |
| --- |
| if(rb\_string.equals("学生")){  Timer timer = new Timer();  timer.schedule(new TimerTask(){  @Override  public void run() {  Intent i = new Intent(MainActivity.this ,AfterLogin.class);  startActivity(i);  }  },2000);//延时2s执行 |

* 转跳后界面如下(反正除了退学按钮都不行C:\Users\Chonor\AppData\Roaming\Tencent\QQ\Temp\AG71%B4]V_O6PZ(P53Y_G%Y.png：

|  |
| --- |
|  |

* 转跳界面后设置退出，之后继续转跳回主登录界面，此处代码没有多少变化，所以在展示。

## 实验思考及感想

这次实验主要是写事件监听的代码，界面布局基本没有变动，从写的时候的感觉来说，这个事件处理还是很简单的，本质上就是一个注册监听，一但监听到这个动作之后就会调用相应的函数，我们只要写相应的执行部分就行了，注册监听的方式也是比较相似，就是动不动就要个improt真的麻烦。实验中还涉及了两种消息提示模式，这两种的参数也比较固定使用起来也比较简单。还是自己做点拓展好玩，那个打开相机比较简单，只需要用Intent即可打开相机返回会也可以直接使用Bitmap来实现显示，但是打开相册返回图片就比较麻烦了，还需要设计到转换，一开始套用相机的直接出运行bug，之后百度了才得到了比较好用的方式。拓展的时候还涉及了一下新的页面转跳，觉得这个转跳是挺简单，但是用来写APP的时候也是很重要的。