移动应用开发实验报告

lab3：Intent、Bundle的使用以及RecyclerView、ListView的应用

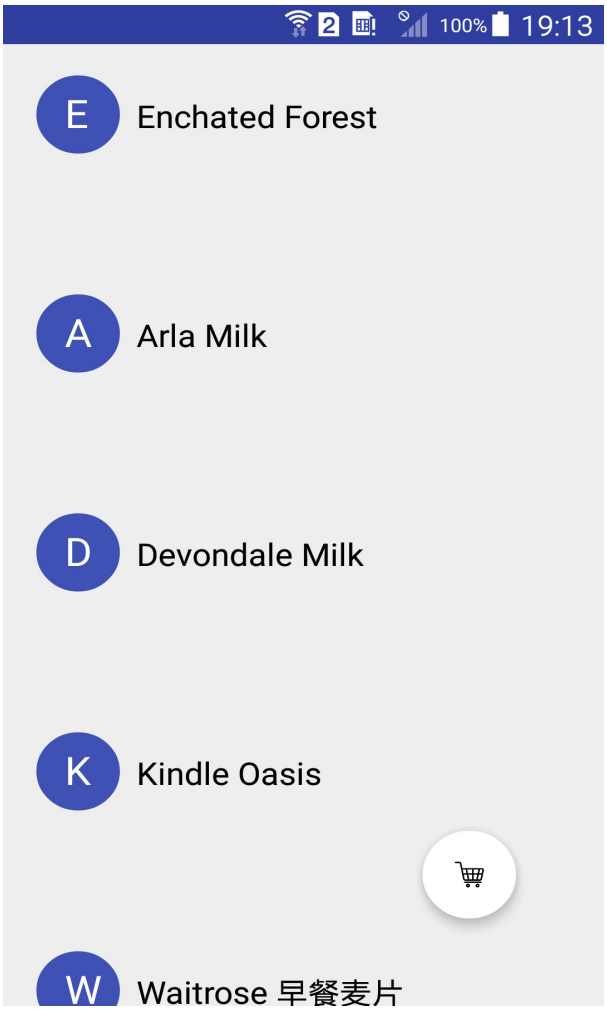
姓名	学号	年级	专业	电话	邮箱	起止日期
王若曦	15352319	15级	软件工程	15354222552	1511330303@qq.com	2017.10.15-2017.10.21

实验目的

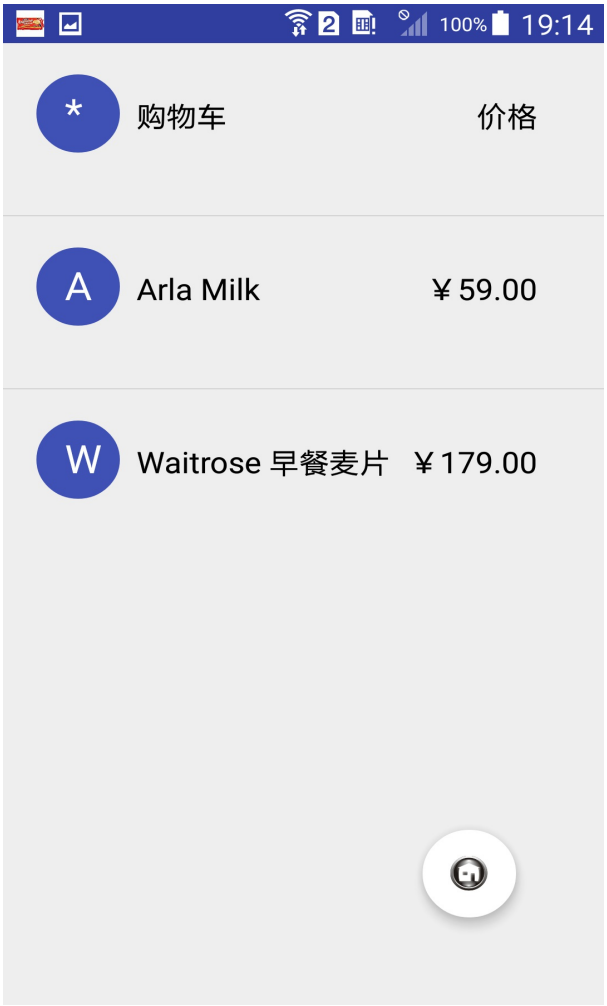
1. 复习事件处理
2. 学习Intent、Bundle在Activity跳转中的应用
3. 学习RecyclerView 、ListView 以及各类适配器的用法

实验题目

本次实验模拟实现一个商品标，有两个界面，第一个界面用于呈现商品，如下：



点击右下方的悬浮按钮切换到购物车界面：



点击任意一项商品跳转到对应商品的详情界面：

实验具体要求就不在这里赘述了，下面是具体的实现过程。

实验过程

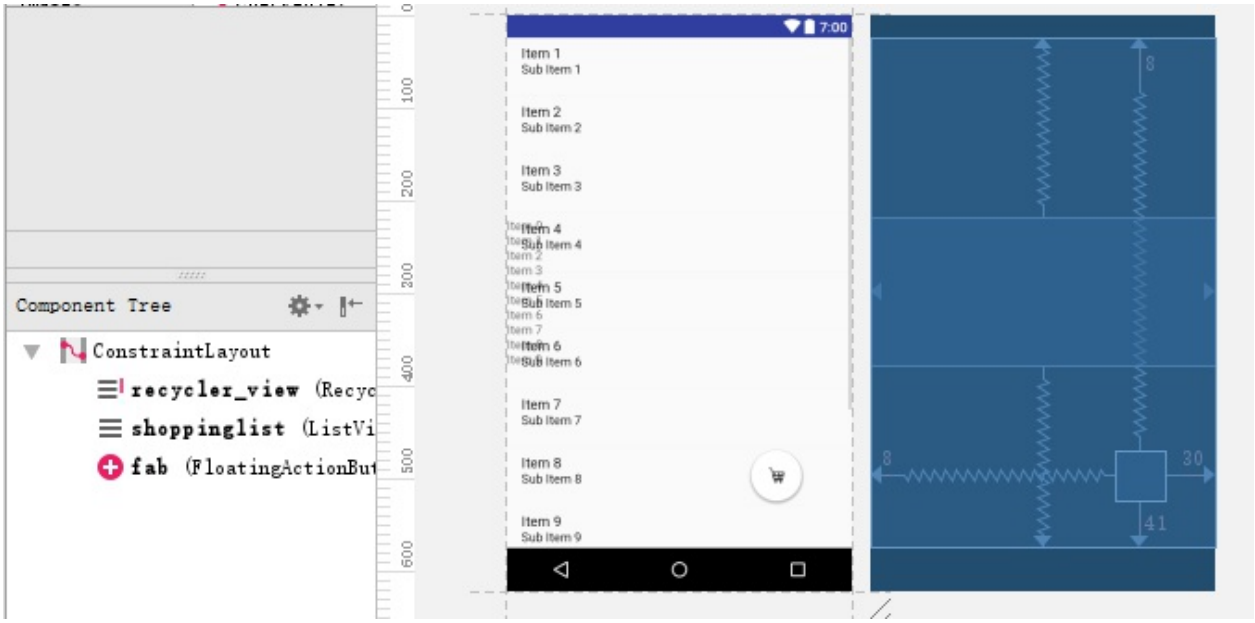
首先是这次实验的整体思路：

> 1. 明确有几个Activity。有几个界面即有几个Activity，这次实现的app一共有三个界面：商品列表，购物车，商品详情。但是要注意由于右下键的FloatingActionButton是在商品列表和购物车界面都存在的，所以说明这两个界面是在一个Activity里，也在同一个xml布局文件里。它们之间的切换是通过fab设置RecyclerView 和ListView 可见与不可见实现的。所以我们一共有两个Activity，一个用来显示商品列表和购物车界面，另一个用来显示商品详情页。
> 2. 明确有几个布局文件。首先，一个布局用来放RecyclerView，ListView 和FloatingActionButton；一个用来定义商品列表Item的布局；一个用来定义购物车Item的布局；一个用来定义商品详情页的布局。所以这次我们一共需要四个xml布局文件来定义我们全部的布局。
> 3. 明确实现过程。首先，利用RecyclerView 并自定义适配器来完成商品列表界面的显示；然后利用ListView 实现购物车界面，这里利用自带的simpleAdapter适配器即可，不需自定义；再利用FloatingActionButton设置可见与不可见实现前两个界面的交替显示，这里要在开始定义先显示RecyclerView，点击fab之后进行切换；然后实现商品详情页的布局和Activity，并通过Intent和Bundle实现这两个Activity之间的跳转；最后实现具体点击，长按等事件处理。

布局文件的设计就不详细贴代码介绍了，根据要求用相应控件把布局设计好即可。

RecyclerView实现商品列表

使用RecyclerView 要先添加依赖，关于添加依赖的问题在后面的实验思考中具体谈一下。
首先布局文件中，添加RecyclerView 控件，为了使其布局不会乱掉我使用约束布局做这个界面，同时ListView 和FloatingActionButton也在这个约束布局里，感觉比较好用也比较合适。
这里就不贴布局的代码了，贴一张布局的预览。



RecyclerView 的Item的布局，要实现两个TextView，一个用来显示带圆圈的首字母，一个用来显示商品名称。这里为了简单我用了两层嵌套的线性布局，因为一层的话两个TextView 会上下排列，两层嵌套起来的话同一层LinearLayout中的就会左右并排排列。

View Holder

由于需要为RecyclerView 自定义Adapter，而实现时必须遵循View Holder设计模式。
View Holder通常出现在适配器里，为的是ListView 、RecyclerView 滚动的时候快速设置值，而不必每次都重新创建很多对象，从而提升性能。
所以首先我们需要自定义一个View Holder：
使用一个SparseArray数组存储RecyclerView 的Item的子View，使用一个View 存储RecyclerView 的Item，并以此实现构造函数

```
private SparseArray<View> mViews;
private View mConvertView;
public ViewHolder(Context context, View itemView, ViewGroup parent)
{
    super(itemView);
    mConvertView = itemView;
    mViews = new SparseArray<View>();
}
```

get函数用来获取View Holder实例：

```
public static ViewHolder get(Context context, ViewGroup parent, int layoutId)
{
    View itemView = LayoutInflater.from(context).inflate(layoutId,parent,false);
    ViewHolder holder = new ViewHolder(context, itemView, parent);
    return holder;
}
```

View Holder尚未将子View 缓存到SparseArray数组中时，仍然需要通过findView ById()创建View 对象，如果已缓存，直接返回：

```
public <T extends View> T getView(int viewId)
        {
            View view = mViews.get(viewId);
            if(view == null)
            {
                view = mConvertView.findViewById(viewId);
                mViews.put(viewId, view);
            }
```

```
            return (T) view;
        }
```

这里View Holder的自定义实验文档里都有，主要是理解它的作用和用法。下面不再详细写实验文档中有的内容，主要写没有的内容。

Adapter

下面开始自定义RecyclerView 的适配器CommenAdapter（这里写的时候手误写成了commen，忽略掉就好）
首先构造函数由上下文context,布局id和要绑定的数据data三个参数组成：

```
public CommenAdapter(Context context, int layoutId, List<T> datas)
{
    mContext = context;
    mLayoutId = layoutId;
    mDatas = datas;
}
```

在这里适配器主要做了两个工作，一个是为View 创建响应的Item-layout，用来显示每一项；另一个工作是访问数据集合并将数据绑定到响应的View 上。所以首先需要我们实现这两个函数：

```
@Override //创建ViewHolder
public MainActivity.ViewHolder onCreateViewHolder(final ViewGroup parent, int viewType)
{
   MainActivity.ViewHolder viewHolder = MainActivity.ViewHolder.get(mContext,parent,mLayoutId);
   return viewHolder;
}
@Override //数据绑定
public void onBindViewHolder(final MainActivity.ViewHolder holder, int position)
{
   convert(holder, mDatas.get(position));
   if(mOnItemClickListener != null){
       holder.itemView.setOnClickListener(new View.OnClickListener(){
           @Override
           public void onClick(View v){
               mOnItemClickListener.onClick(holder.getAdapterPosition());
           }
       });
       holder.itemView.setOnLongClickListener(new View.OnLongClickListener(){
           @Override
           public boolean onLongClick(View v){
               mOnItemClickListener.onLongClick(holder.getAdapterPosition());
               return false;
           }
       });
   }
}
```

数据绑定函数中的convert函数用来进行实际的绑定数据到对应的view Holder上。因为我们需要设计的Item不是简单的一个TextView，所以不能通过直接setText()来做，所以这里利用一个convert函数可以对不同的Item进行数据绑定，比较方便。具体的方法重写在调用数据绑定的onCreate里。
同时在这里实现了Item被点击时的事件处理，当某一项被点击的时候回返回对应位置的position信息，即点击的是列表中的哪一项，进而进行具体的数据传送和跳转。
但是由于RecyclerView 没有OnItemClickListener方法，所以我们需要在适配器里再实现该方法，即需要我们添加接口和方法：

```
public  interface OnItemClickListener{
    void onClick(int position);
    void  onLongClick(int position);
```

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

为了实现长按移除，我们还需要写一个remove函数，调用notifyItemRemoved来实现对某一项的移除：

```
public void removeItem(int position){
    mDatas.remove(position);
    notifyItemRemoved(position);
}
```

最后，我们还需要在这里获取列表中Item的总数：

```
public int getItemCount(){
    return mDatas.size();
}
```

ListView实现购物车

ListView 的布局和RecyclerView 放在同一个xml文件中。
ListView 不需要我们自定义适配器，要注意的是适配器的选择与使用。因为我们的Item含有多项，所以需要选择simpleAdapter，当然可以自定义，但是还是不给自己找事了。
因为不需要自定义适配器等方法，所以在onCreate里即可完成ListView 的设置。onCreate在后面详细介绍。
这里主要要注意的是设置simpleAdapter适配器时的各个参数：

1. 上下文context
2. 要绑定到ListView 的List类型的数据
3. ListView 的Item布局
4. 用来提取数据时对应的key
5. 获取的数据要绑定到哪些控件上，通过id设置

FloatingActionButton进行界面切换

fab的布局和上面的RecyclerView，ListView 在同一个布局文件中。
浮动按钮用来进行商品列表和购物车界面的切换，这里是通过setVisibility函数设置RecyclerView 和ListView 一个可见一个不可见实现界面的显示和切换。所以在初始化的时候我们要设置ListView 不可见（在oncreate中）。

```
public void change(View v){
        if(mRecyclerView.getVisibility()==View.VISIBLE)
        {
            mRecyclerView.setVisibility(View.GONE);
            mListView.setVisibility(View.VISIBLE);
            fab.setImageResource(R.drawable.mainpage);

        }
        else
        {
            mRecyclerView.setVisibility(View.VISIBLE);
            mListView.setVisibility(View.GONE);
            fab.setImageResource(R.drawable.shoplist);
        }
    }
```

这里我是在fab的布局中设置onclick属性，然后再Activity中进行设置。
还有是fab的图标切换，通过setImageResource函数实现。

数据的设置和初始化

列表中每一项的数据和详情中的所有数据需要我们利用一个List类型的变量data存储，而每一项的数据类型需要我们定义一个Good数据类型，含有图片，名称，价格，详情等数据。

```
public class Goods implements Serializable {

    private int imageId;
    private String name;
    private String price;
    private String info;
    private char first;

    public  Goods(int image, String name, String price, String info){...}

    public String getFirstLetter(){...}

    public int getImageId(){...}

    public String getName() { return name; }

    public String getPrice() { return price; }

    public String getInfomation() { return info; }

}
```

其中还需要我们实现一些函数用来返回其中的某一项信息。如getFirstLetter，用来获得name中的首字母，用于列表中的第一项首字母的显示。这里用到了substring函数，用来获得字符串中指定的几个字符。其他的就直接返回就好。

```
public String getFirstLetter(){
    String first;
    first = name.substring(0,1);
    return first;
}
```

数据的初始化首先需要我们把所有数据打表打进去（利用add函数），注意打进去的数据要和Good数据类型的各参数顺序保持一致。然后再根据需要给RecyclerView 和ListView 的Item数组进行数据添加。

```
protected void initData()
{
    data.add(new Goods(R.drawable.ef,"Enchated Forest","￥5.00", "作者 Johanna Basford"));
    data.add(new Goods(R.drawable.am,"Arla Milk","￥59.00", "产地 德国"));
    data.add(new Goods(R.drawable.dm,"Devondale Milk","￥79.00", "产地 澳大利亚"));
    data.add(new Goods(R.drawable.kindle,"Kindle Oasis","￥2399.00", "版本 8GB"));
    data.add(new Goods(R.drawable.waitrose,"Waitrose 早餐麦片","￥179.00", "重量 2kg"));
    data.add(new Goods(R.drawable.mctitie,"Mcvitie's 饼干","￥14.90", "产地 美国"));
    data.add(new Goods(R.drawable.fedo,"Ferrero Rocher","￥132.59", "重量 300g"));
    data.add(new Goods(R.drawable.maltess,"Maltesers","￥141.43", "重量 118g"));
    data.add(new Goods(R.drawable.lin,"Lindt","￥139.43", "重量 249g"));
    data.add(new Goods(R.drawable.borg,"Borggreve","￥28.90", "重量 640g"));

    listItem = new ArrayList<>(); //RecyclerView
    shoppingitem = new ArrayList<>(); //ListView
    for(Goods c : data){
        Map<String, Object> Item = new LinkedHashMap<>();
        Item.put("name",c.getName());
        Item.put("firstLetter",c.getFirstLetter());
```

```
            listItem.add(Item); //用来给适配器添加数据，即商品列表中的一项
        }
        {
            Goods c =new Goods(0,"购物车","价格",null);
            Map<String,Object> Item = new LinkedHashMap<>();
            Item.put("first","*");
            Item.put("name",c.getName());
            Item.put("price",c.getPrice());
            shoplist.add(c); //shoplist用来存储购物车中商品信息（Goods类型）
            shoppingitem.add(Item); //shoppingitem用来给适配器添加数据，即购物车列表中的一项
        }
    }
```

RecyclerView 中需要每一项商品的信息，所以使用for循环进行数据添加；ListView 是在点击购物按钮后才会添加信息，所以这里只是初始化了第一栏的购物车的标题栏。
这里Map用来适配我们定义的Item的类型，然后利用put和add函数进行数据的添加。

事件的处理

首先，点击商品列表中的某一项，跳转到相应的详情页以及长按商品列表中的某一项，移除对应商品：
即需要利用我们定义的RecyclerView 的监听器，这里用到的是短按，即click。
跳转过程：首先new intent来定义从MainActivity页面跳转到Goods_info页面；然后定义bundle来存储跳转过程信息；调用putSerializable函数存储被点击Item的位置position，然后通过data和position来找到被点击项；然后利用putExtras函数传送bundle中的数据；最后利用startActivityForResult进行页面跳转。
然后在Goods_info页面，接受到来自MainActivity传来的数据，然后在数据初始化中利用Goods类型的变量获取从MainActivity传来的数据，并得到相应的图片，名字等数据，并完成商品详情页面的显示。

```
MainActivity:
commenAdapter.setOnItemClickListener(new CommenAdapter.OnItemClickListener(){
    @Override
    public void onClick(int position){
        Intent intent = new Intent(MainActivity.this, Goods_info.class);
        Bundle bundle = new Bundle();
        bundle.putSerializable("goods",data.get(position));
        intent.putExtras(bundle);//传送数据
        startActivityForResult(intent,1);//启动跳转

    }
    @Override
    public void onLongClick(int position){
        commenAdapter.removeItem(position);
        data.remove(position);
        Toast.makeText(MainActivity.this,"移除第"+position+"个商品",Toast.LENGTH_SHORT).show();
    }
});

Goods_info:
private void initialData(){

        goods = (Goods) getIntent().getExtras().get("goods");
        if(goods != null){
            pic.setImageResource(goods.getImageId());
            name.setText(goods.getName());
            price.setText(goods.getPrice());
            info.setText(goods.getInfomation());
        }
        star.setTag(0);
    }
```

长按的效果为移除商品列表中的某一项，利用我们自定义适配器时写的removeItem函数。要注意的是不要忘记同时移除掉data中的对应项，因为上面是通过data和position找到被点击项的，如果只移除列表中的Item的话那么data利用position对应到列表中时就会出现混乱。即点开的商品详情不是被点击的商品。

其次，商品详情页点击购物车图标将对应商品添加到购物车：

首先将购物车图标利用setClickable设为可点击，然后添加监听器器，利用setResult函数进行页面跳转，并将跳转信息即对应Goods类型商品信息传回去。此时是在商品详情页面Goods_info，然后再MainActivity中通过onActivityResult获得跳转信息，并将对应商品添加到购物车列表。

```
buy.setClickable(true);
buy.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
        setResult(1,new Intent(Goods_info.this,MainActivity.class).putExtra("goods",goods));
        Toast.makeText(Goods_info.this,"商品已加入到购物车",Toast.LENGTH_SHORT).show();
    }
});
```

这里要记得simpleAdapter要利用notifyDataSetChanged进行更新。

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if(requestCode==1)
    {
        if(resultCode==1)
        {
            Goods c = (Goods) data.getExtras().get("goods");
            Map<String,Object> listItem = new LinkedHashMap<>();
            assert c != null;
            listItem.put("first",c.getFirstLetter());
            listItem.put("name",c.getName());
            listItem.put("price",c.getPrice());
            shoplist.add(c);
            shoppingitem.add(listItem);
            simpleAdapter.notifyDataSetChanged();
        }
    }
}
```

这里具体说一下从商品详情页面的Activity点击购物车图标后返回商品列表页面Activity时的处理方法：
如果我们想在Activity中得到新打开的Activity关闭后返回的数据，那么就需要使用系统提供的startActivityForResult方法来打开新的Activity，新的Activity关闭后会向前面的Activity返回数据，为了得到传回的数据，我们要在前面的Activity中重写onActivityResult方法，来对返回的数据进行相应的操作。这里数据的传递是通过Intent实现的，Android平台此时会调用前面Activity的onActivityResult方法，把存放了返回数据的Intent作为参数传入，此时就可以使用该数据，这里具体返回的就是一个position，即点击的商品是哪个商品（在data数组中的位置），然后将其添加到购物车的ListView 。
调用startActivityForResult,onActivityResult,setResult这些方法时有两个参数：请求码和结果码。请求码在我们调用startActivityForResult时用来标识请求来源，简单来说就是有多种打开新Activity方法时，如多个button，这时请求码会标识是哪个button打开的新Activity；结果码是用于标识返回结果的来源，当我们从一个Activity打开多个不同Activity时，setResult时会返回一个结果码，用来标识时从哪个Activity返回的，并在原Activity的onActivityResult中根据该结果码进行响应操作。
我们这次实验因为只有一种跳转情况，不需具体用到请求码和结果码，所以都设为1即可。

最后，在购物车界面，点击某一项跳转到对应商品详情页面，长按某一项弹出对话框询问是否删除，确定删除，取消返回：
这里通过ListView 的监听器实现，首先点击的事件和商品列表点击事件基本一致。区别在于这里的有两个参数i和l，i指的是被点击项在列表中的位置，l指的是被点击项的id，不过在simpleAdapter中i和l相等。所以这里我们利用shoplist和i确定被点击项。

```
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        if(i==0) return; //若i=0，即购物车中没有商品，则返回
        Intent intent = new Intent(MainActivity.this, Goods_info.class);
        Bundle bundle = new Bundle();
        bundle.putSerializable("goods",shoplist.get(i));
        intent.putExtras(bundle);//传送数据
        startActivityForResult(intent,1);//启动跳转
    }
```

```
});
```

长按的监听器中需要定义多一个对话框，上次实验中用到了对话框，所以比较简单。别忘了在外面new 。然后直接通过shoppingitem调用remove进行移除，同时simpleAdapter利用notifyDataSetChanged进行更新数据。

```
mListView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener(){
            @Override
            public boolean onItemLongClick(AdapterView<?> adapterView, View view,final int pos, long l){
                if(pos==0)return true;
                builder.setTitle("移除商品");
                builder.setNegativeButton("取消",new DialogInterface.OnClickListener(){
                    @Override
                    public void onClick(DialogInterface dialogInterface,int i){}
                });
                builder.setMessage("从购物车移除"+shoplist.get(pos).getName()+"?");
                builder.setPositiveButton("确定",new DialogInterface.OnClickListener(){
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i){
                        shoppingitem.remove(pos);
                        simpleAdapter.notifyDataSetChanged();
                    }
                }).create().show();
                return true;
            }
        });
```

`

还有一个商品详情页的返回按钮和星号收藏按钮的事件：
返回按钮通过设置监听器，然后利用finish函数即可实现返回。

```
private void setListener(){
    back.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view){
            finish();
        }
    });
```

星号收藏按钮主要是空心和实心的变换，这里通过setTag函数设一个tag值来标记当前是实心还是空心，当点击时改变tag值，并利用setBackgroundResource函数更换图片即可。和上次试验我设置登录模式是学生还是教职工差不多。这里要初始化时把tag设为0，表示空心。

```
star.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
        if((Integer) view.getTag()==0){
            view.setBackgroundResource(R.drawable.full_star);
            view.setTag(1);//实心
        }
        else{
            view.setBackgroundResource(R.drawable.empty_star);
            view.setTag(0);//空心
        }
    }
});
```

Activity的增加

由于我们用到了两个Activity，所以还需要在AndroidManifest.xml文件中添加一个Activity。但是由于这个不是启动页面，所以在MainActivity后只加一个Activity就好。

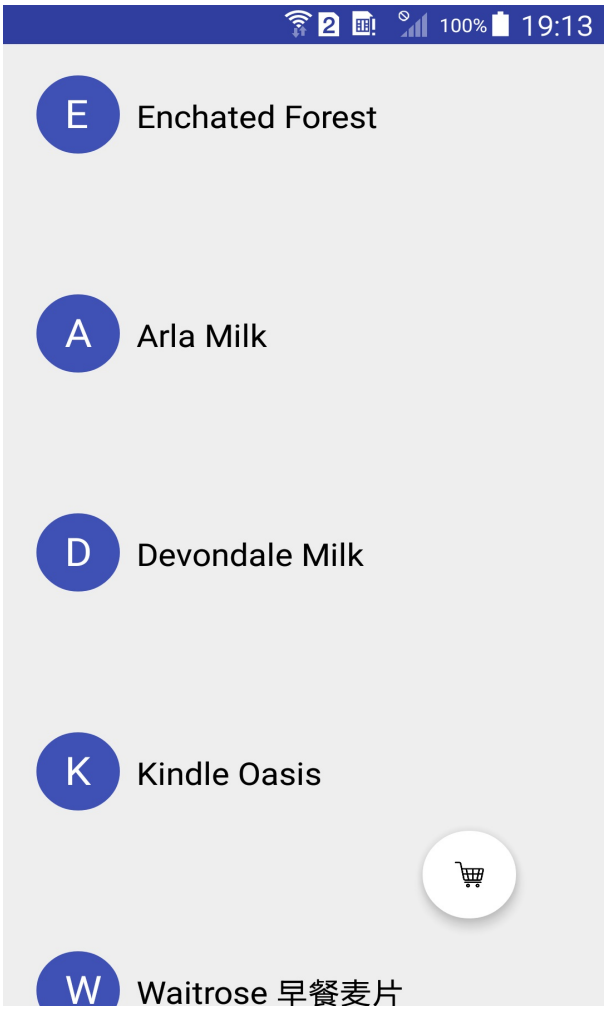
```
<activity android:name=".Goods_info">
</activity>
```

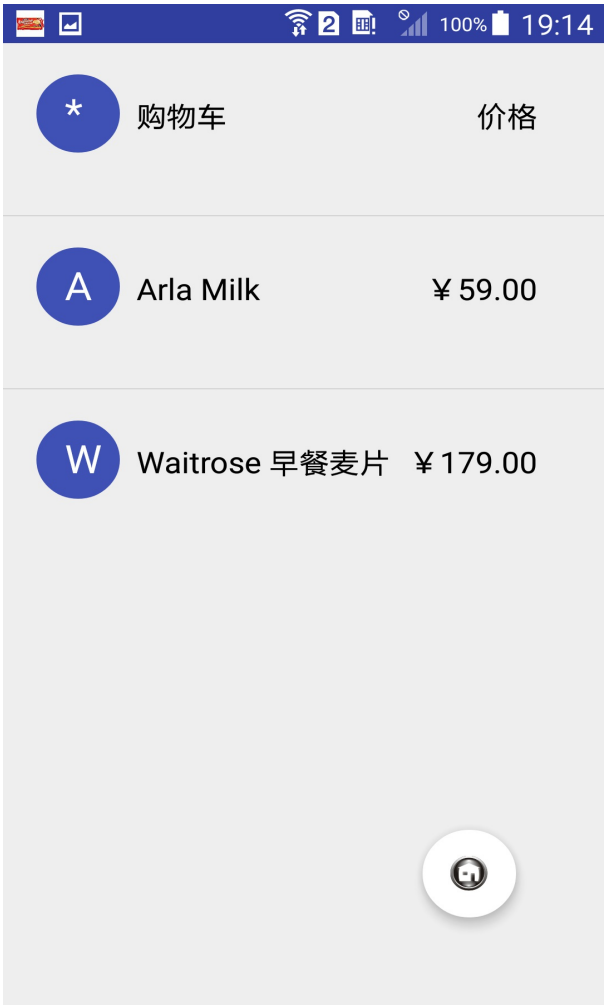
动画效果

RecyclerView 的动画效果从实验文档给的网页扒下来，添加完约束后，根据提示设置适配器即可。

```
ScaleInAnimationAdapter animationAdapter = new ScaleInAnimationAdapter(commenAdapter);
animationAdapter.setDuration(1000);
mRecyclerView.setAdapter(animationAdapter);
mRecyclerView.setItemAnimator(new OvershootInLeftAnimator());
```

实验结果截图




100%
19:14
*
购物车
价格
A
Arla Milk
￥59.00
W
Waitrose 早餐麦片
￥179.00

￥2399.00

版本 8GB

更多产品信息

一键下单

分享商品

不感兴趣

查看更多商品促销信息

实验思考

实验中遇到的问题：

1. RecyclerView 自定义适配器。一开始自定义适配器时，因为把View Holder和commenAdapter写在了一起，但是实验文档中给的TA的代码是分开java写的，所以调用起来会报错。一开始写commenAdapter的时候没有注意到Item的类型是需要Map的，没有写来满足View Holder，导致报了好多错，主要是重写convert函数时。
2. convert函数的重写。绑定数据时因为Item的类型需要写一个convert函数来进行绑定，但是一开始我本来是想具体实现绑定，即把重写convert函数时的方法直接写在onBindView Holder里，但是感觉有点单一，还是用个convert好一点。只不过这次实验没有用到更多的，在onCreate里重写一个函数感觉怪怪的。
3. findview 。使用布局中的控件时一定要记得findview ，找到对应控件的id。写fab的时候设置可见不可见能正常实现，但是通过fab调用setImageResource时会报错。找了半天原来是忘了先findview 。
4. 界面的跳转。这次实验通过Intent和Bundle的通信实现界面的跳转，当时查了资料还有多种进行界面跳转的方法，如这次是通过class跳转，还可以通过name跳转，也可以为Activity设置Action，通过Action跳转等。而利用fab在加一些变量感觉可以实现多界面的跳转，不过都写在一个布局中感觉有点乱，可能会出错。
5. 布局的嵌套使用。原来以为布局文件中只能有一种布局方法，这次用到了多次布局的嵌套，如在设置RecyclerView 的Item布局时，利用双层嵌套线性布局实现两个TextView 的并列；商品详情页的布局利用线性布局中嵌套相对布局设置1/3和控件间的相对布局。
6. Activity跳转的机制。做实验的时候其实不知道要怎么实现界面间的跳转，尤其是不同Activity间的跳转。而且还有在跳转时把触发的事件即点击的是哪一项返回回去进行相关操作。也是查了好久，看了好久才懂了一点。如setResult和onActivityResult的使用。同时还有Intent和Bundle的使用。
7. Goods数据类型的定义。为了方便数据的传输，我定义了一个商品的数据类型，数据含有商品图片，名字，信息等。这里需要注意的是为了使用Intent和Bundle进行页面跳转和数据传递，需要调用Bundle的putSerializable方法，所以我们在定义Goods数据类型时需要实现序列化:
   public class Goods implements Serializable;
8. onCreate的使用。我理解的onCreate函数就是相当于C++里的main函数，我们写完各种函数，类，数据类型后，在主函数里进

行具体的操作。例如我们RecyclerView 和ListView 的适配器设置，界面跳转和事件处理等都是在这里写的，包括convert方法的重写。

本次实验主要是做Activity的跳转和数据的传递。由于跟上一次实验的差距太大，感觉确实有点难，好多不会做。首先就是RecyclerView 的自定义，然后是跳转时间的处理，布局的设计也遇到的很多问题，所以花的时间比较长。但是做出来的时候确实还是比较有成就感的。