Android 应用开发技术 实验报告

专业 网络工程 班级 1 班 学号 2220201979 姓名 张帅

实验时间 2023 年 3 月 指导教师 葛新

实验名称 电商 App 的购物车设计与实现 实验成绩

电商 App 的购物车设计与实现

——ShoppingCartSoftware

一、 程序介绍

随着 Android 智能机的普及，移动应用的需求也在与日俱增。我基于 Android Studio 开发平台，开发一款电商 App 的购物车功能。现在万物互联的时代，网上购物早已经是大家的习惯了。足不出户地选择自己想要的商品进行购买十分方便，利用已经学过的知识例如 EditText、ImageButtom、SqLite 等实现购物车的基本功能。

思路来源：电商 App 的购物车可谓是司空见惯了，以京东商城的购物车为例，一开始没有添加任何商品，此时空购 物车如图 1 所示，而且提示去逛秒杀商场；加入几件商品之后，购物车页面如图 2 所示。

图 1：京东 App 购物车的初始页面

图 2：京东 App 购物车加了几件商品

可见购物车除了底部有个结算行，其余部分主要是已加入购物车的商品列表，然后每个商品行左边是商品小图，右边是商品名称及其价格。

二、 功能介绍

据此仿照京东的购物车功能，第一次进入购物车页面，购物车里面是空的，同时提示去逛手机商场， 如图 3 所示。接着去商场页面选购手机，随便挑了几部手机加入购物车，再返回购物车页面，即可看 到购物车的商品列表，如图 4 所示，有商品图片、名称、数量、单价、总价等等信息。当然购物车并 不仅仅只是展示待购买的商品，还要支持最终购买的结算操作、支持清空购物车等功能。

图 3：首次打开购物车页面

图 4：选购商品后的购物车

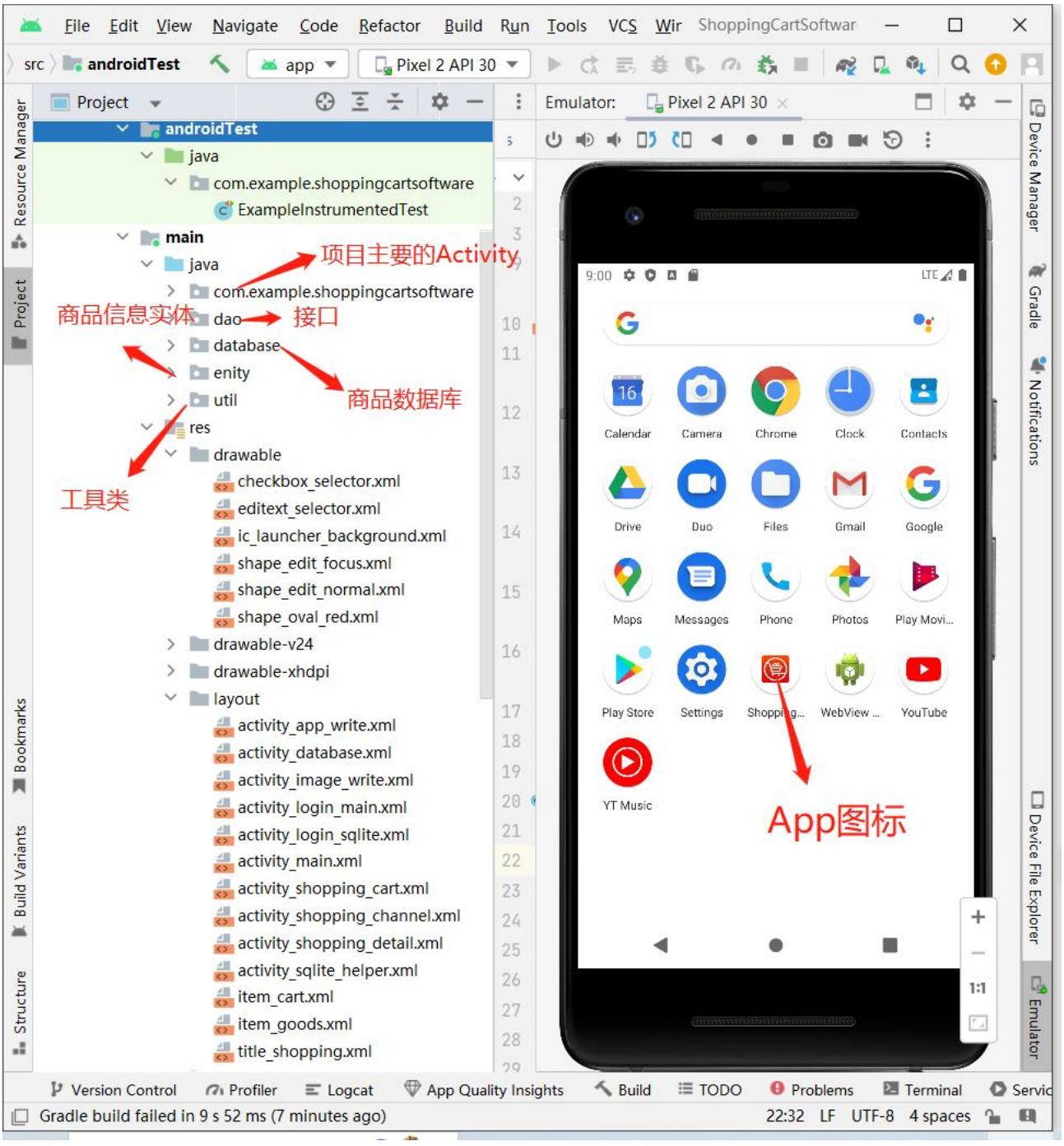
购物车的存在感很强，不仅仅在购物车页面才能看到购物车。往往在商场页面，甚至商品详情页面，都会看到某个角落冒出购物车图标。一旦有新商品加入购物车，购物车图标上的商品数量立马加一。当然，用户也能点击购物车图标直接跳到购物车页面。商场页面除了商品列表之外，页面右上角还有一个购物车图标，如图 5 所示，有时这个图标会在页面右下角。商品详情页面通常也有购物车图标，如图 6 所示，倘使用户在详情页面把商品加入购物车，那么图标上的数字也会加一。

图 5：手机商场页面

图 6：手机详情页面

三、 功能实现

1. 项目结构：



2. 界面设计：

使用的控件：

- 线性布局 LinearLayout：购物车界面从上往下排列，用到了垂直方向的线性布局。

- 网格布局 GridLayout：商场页面的陈列橱柜，允许分行分列展示商品。
- 相对布局 RelativeLayout：页面右上角的购物车图标，图标右上角又有数字标记，按照指定方位排列控件正是相对布局的特点。
- 其他常见控件尚有文本视图 TextView、图像视图 ImageView，按钮控件 Button 等。

采取的存储方式：

- 数据库 SQLite：最直观的肯定是数据库了，购物车里的商品列表一定是放在 SQLite 中，增删改查都用到它。
- 全局内存：购物车图标右上角的数字表示购物车中的商品数量，该数值建议保存在全局内存中，这 样不必每次都到数据库中执行 count 操作。
- 存储卡文件：通常商品图片来自于电商平台的服务器，此时往往引入图片缓存机制，也就是首次访 问先将网络图片保存到存储卡，下次访问时直接从存储卡获取缓存图片，从而提高图片的加载速度。
- 共享参数 SharedPreferences：是否首次访问网络图片，这个标志位推荐放在共享参数中，因为它 需要持久化存储，并且只有一个参数信息。

3. 关键代码：

1) 关于页面跳转

因为购物车页面允许直接跳到商场页面，并且商场页面也允许跳到购物车页面，所以如果用户在这两个 页面之间来回跳转，然后再按返回键，结果发现返回的时候也是在两个页面间往返跳转。出现问题的缘 由在于：每次启动活动页面都往活动栈加入一个新活动，那么返回出栈之时，也只好一个一个活动依次退出了。

解决该问题的办法参考第二章中的 Activity 的启动模式对于购物车的活动跳转需要指定启动标志 FLAG_ACTIVITY_CLEAR_TOP，表示活动栈有且仅有该页面的唯一实例，如此即可避免多次返回同一页 面的情况。比如从购物车页面跳到商场页面，此时活动跳转的代码示例如下：

```
// 从购物车页面跳到商场页面
Intent intent = new Intent(this, ShoppingChannelActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  // 设置启动标志
startActivity(intent);  // 跳转到手机商场页面
```

又如从商场页面跳到购物车页面，此时活动跳转的代码示例如下：

```
// 从商场页面跳到购物车页面
Intent intent = new Intent(this, ShoppingCartActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  // 设置启动标志
startActivity(intent);  // 跳转到购物车页面
```

2) 关于商品图片的缓存

通常商品图片由后端服务器提供，App 打开页面时再从服务器下载所需的商品图。可是购物车模块的多 个页面都会展示商品图片，如果每次都到服务器请求图片，显然既耗时间又耗流量非常不经济。因此 App 都会缓存常用的图片，一旦从服务器成功下载图片，便在手机存储卡上保存图片文件。然后下次界 面需要加载商品图片时，就先从存储卡寻找该图片，如果找到就读取图片的位图信息，如果没找到就再 到服务器下载图片。 以上的缓存逻辑是最简单的二级图片缓存，实际开发往往使用更高级的三级缓存机制，即“运行内存→存 储卡→网络下载”。当然就初学者而言，先从掌握最简单的二级缓存开始，也就是“存储卡→网络下载”。 按照二级缓存机制，可以设计以下的缓存处理逻辑：

（1）先判断是否为首次访问网络图片。

（2）如果是首次访问网络图片，就先从网络服务器下载图片。

（3）把下载完的图片数据保存到手机的存储卡。

（4）往数据库中写入商品记录，以及商品图片的本地存储路径。

（5）更新共享参数中的首次访问标志。

按照上述的处理逻辑，编写的图片加载代码示例如下：

```
private String mFirst = "true"; // 是否首次打开
// 模拟网络数据，初始化数据库中的商品信息
private void downloadGoods() {
    // 获取共享参数保存的是否首次打开参数
    mFirst = SharedUtil.getIntance(this).readString("first", "true");
    // 获取当前App的私有下载路径
    String path =
getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS).toString() + "/";
    if (mFirst.equals("true")) { // 如果是首次打开
        ArrayList<GoodsInfo> goodsList = GoodsInfo.getDefaultList(); // 模拟网络图
片下载
        for (int i = 0; i < goodsList.size(); i++) {
            GoodsInfo info = goodsList.get(i);
            long rowid = mGoodsHelper.insert(info); // 往商品数据库插入一条该商品的记
录
            info.rowid = rowid;
            Bitmap pic = BitmapFactory.decodeResource(getResources(), info.pic);
            String pic_path = path + rowid + ".jpg";
            FileUtil.saveImage(pic_path, pic); // 往存储卡保存商品图片
            pic.recycle(); // 回收位图对象
            info.pic_path = pic_path;
            mGoodsHelper.update(info); // 更新商品数据库中该商品记录的图片路径
        }
    }
    // 把是否首次打开写入共享参数
    SharedUtil.getIntance(this).writeString("first", "false");
}
```

3) 关于各页面共同的标题栏

购物车、手机商场、手机详情三个页面顶部都有标题栏，而且这三个标题栏风格统一，当然 App 界面支持局部的公共布局，以购物车的标题栏为例，公共布局的实现过程包括以下两个步骤：

步骤一，首先定义标题栏专用的布局文件，包含返回箭头、文字标题、购物车图标、商品数量表等，具 体内容如下所示：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:background="#aaaaff" >

    <ImageView
        android:id="@+id/iv_back"
        android:layout_width="50dp"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
        android:padding="10dp"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_back" />

    <TextView
        android:id="@+id/tv_title"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_centerInParent="true"
        android:gravity="center"
        android:textColor="@color/black"
        android:textSize="20sp" />
```

```
    <ImageView
        android:id="@+id/iv_cart"
        android:layout_width="50dp"
        android:layout_height="match_parent"
        android:layout_alignParentRight="true"
        android:scaleType="fitCenter"
        android:src="@drawable/cart" />

    <TextView
        android:id="@+id/tv_count"
        android:layout_width="20dp"
        android:layout_height="20dp"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/iv_cart"
        android:layout_marginLeft="-20dp"
        android:gravity="center"
        android:background="@drawable/shape_oval_red"
        android:text="0"
        android:textColor="@color/white"
        android:textSize="15sp" />

</RelativeLayout>
```

（完整代码见 src\main\res\layout\title_shopping.xml）

步骤二，然后在购物车页面的布局文件中添加如下一行 include 标签，表示引入 title_shopping.xml 的布局内容：（完整代码见 src\main\res\layout\activity_shopping_cart.xml）

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/orange"
    android:orientation="vertical">

    <include layout="@layout/title_shopping" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
```

```
<LinearLayout
    android:id="@+id/ll_content"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:visibility="visible">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:layout_width="85dp"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="图片"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:gravity="center"
            android:text="名称"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="3"
            android:gravity="center"
            android:text="名称"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="数量"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="数量"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="数量"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="单价"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="总价"
            android:textColor="@color/black"
            android:textSize="15sp" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/ll_cart"
```

```
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="100dp"
                android:layout_marginBottom="100dp"
                android:gravity="center"
                android:text="哎呀，购物车空空如也，快去选购商品吧"
                android:textColor="@color/black"
                android:textSize="17sp" />

            <Button
                android:id="@+id/btn_shopping_channel"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="逛逛手机商场"
                android:textColor="@color/black"
                android:textSize="17sp" />
        </LinearLayout>
    </RelativeLayout>
</ScrollView>

</LinearLayout>
```

之后重新运行测试 App，即可发现购物车页面的顶部出现了公共标题栏，商场页面、详情页面的公 共标题栏可参考购物车页面的 include 标签。

4) 关于商品网格的单元布局

商场页面的商品列表，呈现三行二列的表格布局，每个表格单元的界面布局雷同，都是商品名称在上、 商品图片居中、商品价格与添加按钮在下，看起来跟公共标题栏的处理有些类似。但后者为多个页面引 用同一个标题栏，是多对一的关系；而前者为一个商场页面引用了多个商品网格，是一对多的关系。因 此二者的实现过程不尽相同，就商场网格而言，它的单元复用分为下列 3 个步骤：

步骤一，在商场页面的布局文件中添加 GridLayout 节点，如下所示：

（完整代码见 src\main\res\layout\activity_shopping_channel.xml）

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/orange"
    android:orientation="vertical" >

    <include layout="@layout/title_shopping" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <GridLayout
            android:id="@+id/gl_channel"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:columnCount="2" />
    </ScrollView>

</LinearLayout>
```

步骤二，为商场网格编写统一的商品信息布局，XML 文件内容示例如下：

（完整代码见 src\main\res\layout\item_goods.xml）

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/ll_item"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:background="@color/white"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="@color/black"
        android:textSize="17sp"
        tools:text="小米手机" />

    <ImageView
        android:id="@+id/iv_thumb"
        android:layout_width="180dp"
        android:layout_height="150dp"
        android:scaleType="fitCenter"
        tools:src="@drawable/xiaomi" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="45dp"
```

步骤三，在商场页面的 Java 代码中，先利用下面代码获取布局文件 item_goods.xml 的根视图：

```
View view = LayoutInflater.from(this).inflate(R.layout.item_goods, null);
```

再从根视图中依据控件 ID 分别取出网格单元的各控件对象：

```
ImageView iv_thumb = view.findViewById(R.id.iv_thumb);
TextView tv_name = view.findViewById(R.id.tv_name);
TextView tv_price = view.findViewById(R.id.tv_price);
Button btn_add = view.findViewById(R.id.btn_add);
```

然后就能按照寻常方式操纵这些控件对象了，下面便是给网格布局加载商品的代码例子：

（完整代码见 src\main\java\com\example\chapter06\ShoppingChannelActivity.java）

```
1   package com.example.shoppingcartsoftware;
2
3   import ...
21
    3 usages
22  public class ShoppingChannelActivity extends AppCompatActivity implements View.OnClickListener {
23
24      // 声明一个商品数据库的帮助器对象
        7 usages
25      private ShoppingDBHelper mDBHelper;
        3 usages
26      private TextView tv_count;
        3 usages
27      private GridLayout gl_channel;
28
29      @Override
30      protected void onCreate(Bundle savedInstanceState) {
31          super.onCreate(savedInstanceState);
32          setContentView(R.layout.activity_shopping_channel);
33          TextView tv_title = findViewById(R.id.tv_title);
34          tv_title.setText("手机商场");
35
36          tv_count = findViewById(R.id.tv_count);
37          gl_channel = findViewById(R.id.gl_channel);
38          findViewById(R.id.iv_back).setOnClickListener(this);
39          findViewById(R.id.iv_cart).setOnClickListener(this);
40
41          mDBHelper = ShoppingDBHelper.getInstance( context: this);
42          mDBHelper.openReadLink();
43          mDBHelper.openWriteLink();
44
45          // 从数据库查询出商品信息，并展示
46          showGoods();
47      }
48
49      @Override
50      protected void onResume() {
51          super.onResume();
52          // 查询购物车商品总数，并展示
53          showCartInfoTotal();
54      }
55
56      // 查询购物车商品总数，并展示
        1 usage
57      private void showCartInfoTotal() {
58          int count = mDBHelper.countCartInfo();
59          MyApplication.getInstance().goodsCount = count;
60          tv_count.setText(String.valueOf(count));
61      }
```

```java
        // 商品条目是一个线性布局，设置布局的宽度为屏幕的一半
        int screenWidth = getResources().getDisplayMetrics().widthPixels;
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams( width: screenWidth / 2, Line
        // 查询商品数据库中的所有商品记录
        List<GoodsInfo> list = mDBHelper.queryAllGoodsInfo();

        // 移除下面的所有子视图
        gl_channel.removeAllViews();

        for (GoodsInfo info : list) {
            // 获取布局文件item_goods.xml的根视图
            View view = LayoutInflater.from( context: this).inflate(R.layout.item_goods,  root: null);
            ImageView iv_thumb = view.findViewById(R.id.iv_thumb);
            TextView tv_name = view.findViewById(R.id.tv_name);
            TextView tv_price = view.findViewById(R.id.tv_price);
            Button btn_add = view.findViewById(R.id.btn_add);

            // 给控件设置值
            iv_thumb.setImageURI(Uri.parse(info.picPath));
            tv_name.setText(info.name);
            tv_price.setText(String.valueOf((int) info.price));

            // 添加到购物车
            btn_add.setOnClickListener(v -> {
                addToCart(info.id, info.name);
            });

            // 点击商品图片，跳转到商品详情页面
            iv_thumb.setOnClickListener(v -> {
                Intent intent = new Intent( packageContext: ShoppingChannelActivity.this, ShoppingDetail
                intent.putExtra( name: "goods_id", info.id);
                startActivity(intent);
            });

            // 把商品视图添加到网格布局
            gl_channel.addView(view, params);
        }
    }

    // 把指定编号的商品添加到购物车
    1 usage
    private void addToCart(int goodsId, String goodsName) {
        // 购物车商品数量+1
        int count = ++MyApplication.getInstance().goodsCount;
        tv_count.setText(String.valueOf(count));
        mDBHelper.insertCartInfo(goodsId);
        ToastUtil.show( ctx: this,  desc: "已添加一部" + goodsName + "到购物车");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        mDBHelper.closeLink();
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.iv_back:
                // 点击了返回图标，关闭当前页面
                finish();
                break;

            case R.id.iv_cart:
                // 点击了购物车图标
                // 从商场页面跳到购物车页面
                Intent intent = new Intent( packageContext: this, ShoppingCartActivity.class);
                // 设置启动标志，避免多次返回同一页面的
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
                break;
        }
    }
```

弄好了商场页面的网格单元，购物车页面的商品行也同理，不同之处在于购物车页面的商品行使 用线性布局而非网格布局，其余实现过程依然分成上述 3 个步骤。

5) 部分源码

ShoppingDetailActivity 类

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_shopping_detail);
        tv_title = findViewById(R.id.tv_title);
        tv_count = findViewById(R.id.tv_count);
        tv_goods_price = findViewById(R.id.tv_goods_price);
        tv_goods_desc = findViewById(R.id.tv_goods_desc);
        iv_goods_pic = findViewById(R.id.iv_goods_pic);
        findViewById(R.id.iv_back).setOnClickListener(this);
        findViewById(R.id.iv_cart).setOnClickListener(this);
        findViewById(R.id.btn_add_cart).setOnClickListener(this);

        tv_count.setText(String.valueOf(MyApplication.getInstance().goodsCount));

        mDBHelper = ShoppingDBHelper.getInstance( context: this);

    }

    @Override
    protected void onResume() {
        super.onResume();
        showDetail();
    }

    1 usage
    private void showDetail() {
        // 获取上一个页面传来的商品编号
        mGoodsId = getIntent().getIntExtra( name: "goods_id",  defaultValue: 0);
        if (mGoodsId > 0) {
            // 根据商品编号查询商品数据库中的商品记录
            GoodsInfo info = mDBHelper.queryGoodsInfoById(mGoodsId);
            tv_title.setText(info.name);
            tv_goods_desc.setText(info.description);
            tv_goods_price.setText(String.valueOf((int) info.price));
            iv_goods_pic.setImageURI(Uri.parse(info.picPath));
        }
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.iv_back:
                finish();
                break;

            case R.id.iv_cart:
                Intent intent = new Intent( packageContext: this, ShoppingCartActivity.class);
                startActivity(intent);
                break;

            case R.id.btn_add_cart:
                addToCart(mGoodsId);
                break;
        }
    }

    1 usage
    private void addToCart(int goodsId) {
        // 购物车商品数量+1
        int count = ++MyApplication.getInstance().goodsCount;
        tv_count.setText(String.valueOf(count));
        mDBHelper.insertCartInfo(goodsId);
        ToastUtil.show( ctx: this,  desc: "成功添加至购物车");
    }
}
```

ShoppingChannelActivity 类

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_shopping_channel);
        TextView tv_title = findViewById(R.id.tv_title);
        tv_title.setText("手机商场");

        tv_count = findViewById(R.id.tv_count);
        gl_channel = findViewById(R.id.gl_channel);
        findViewById(R.id.iv_back).setOnClickListener(this);
        findViewById(R.id.iv_cart).setOnClickListener(this);

        mDBHelper = ShoppingDBHelper.getInstance( context: this);
        mDBHelper.openReadLink();
        mDBHelper.openWriteLink();

        // 从数据库查询出商品信息，并展示
        showGoods();
    }

    @Override
    protected void onResume() {
        super.onResume();
        // 查询购物车商品总数，并展示
        showCartInfoTotal();
    }

    // 查询购物车商品总数，并展示
    1 usage
    private void showCartInfoTotal() {
        int count = mDBHelper.countCartInfo();
        MyApplication.getInstance().goodsCount = count;
        tv_count.setText(String.valueOf(count));
    }

    1 usage
    private void showGoods() {
        // 商品条目是一个线性布局，设置布局的宽度为屏幕的一半
        int screenWidth = getResources().getDisplayMetrics().widthPixels;
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams( width: screenWidth / 2, LinearLayout.LayoutPara
        // 查询商品数据库中的所有商品记录
        List<GoodsInfo> list = mDBHelper.queryAllGoodsInfo();

        // 移除下面的所有子视图
        gl_channel.removeAllViews();

        for (GoodsInfo info : list) {
            // 获取布局文件item_goods.xml的根视图
            View view = LayoutInflater.from( context: this).inflate(R.layout.item_goods,  root: null);
            ImageView iv_thumb = view.findViewById(R.id.iv_thumb);
            TextView tv_name = view.findViewById(R.id.tv_name);
            TextView tv_price = view.findViewById(R.id.tv_price);
            Button btn_add = view.findViewById(R.id.btn_add);

            // 给控件设置值
            iv_thumb.setImageURI(Uri.parse(info.picPath));
            tv_name.setText(info.name);
            tv_price.setText(String.valueOf((int) info.price));

            // 添加到购物车
            btn_add.setOnClickListener(v -> {
                addToCart(info.id, info.name);
            });
```

ShoppingCartActivity 类

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_shopping_cart);
        TextView tv_title = findViewById(R.id.tv_title);
        tv_title.setText("购物车");
        ll_cart = findViewById(R.id.ll_cart);
        tv_total_price = findViewById(R.id.tv_total_price);

        tv_count = findViewById(R.id.tv_count);
        tv_count.setText(String.valueOf(MyApplication.getInstance().goodsCount));

        mDBHelper = ShoppingDBHelper.getInstance( context: this);

        findViewById(R.id.iv_back).setOnClickListener(this);
        findViewById(R.id.btn_shopping_channel).setOnClickListener(this);
        findViewById(R.id.btn_clear).setOnClickListener(this);
        findViewById(R.id.btn_settle).setOnClickListener(this);
        ll_empty = findViewById(R.id.ll_empty);
        ll_content = findViewById(R.id.ll_content);
    }

    @Override
    protected void onResume() {
        super.onResume();
        showCart();
    }

    // 展示购物车中的商品列表
    1 usage
    private void showCart() {
        // 移除下面的所有子视图
        ll_cart.removeAllViews();
        // 查询购物车数据库中所有的商品记录
        mCartList = mDBHelper.queryAllCartInfo();
        if (mCartList.size() == 0) {
            return;
        }

        for (CartInfo info : mCartList) {
            // 根据商品编号查询商品数据库中的商品记录
            GoodsInfo goods = mDBHelper.queryGoodsInfoById(info.goodsId);
            mGoodsMap.put(info.goodsId, goods);

            // 获取布局文件item_cart.xml的根视图
            View view = LayoutInflater.from( context: this).inflate(R.layout.item_cart,  root: null);
            ImageView iv_thumb = view.findViewById(R.id.iv_thumb);
            TextView tv_name = view.findViewById(R.id.tv_name);
            TextView tv_desc = view.findViewById(R.id.tv_desc);
            TextView tv_count = view.findViewById(R.id.tv_count);
            TextView tv_price = view.findViewById(R.id.tv_price);
            TextView tv_sum = view.findViewById(R.id.tv_sum);

            iv_thumb.setImageURI(Uri.parse(goods.picPath));
            tv_name.setText(goods.name);
            tv_desc.setText(goods.description);
            tv_count.setText(String.valueOf(info.count));
            tv_price.setText(String.valueOf((int) goods.price));
            // 设置商品总价
            tv_sum.setText(String.valueOf((int) (info.count * goods.price)));

            // 给商品行添加长按事件。长按商品行就删除该商品
            view.setOnLongClickListener(v -> {
                AlertDialog.Builder builder = new AlertDialog.Builder( context: ShoppingCartActivity.this);
                builder.setMessage("是否从购物车删除" + goods.name + "？");
                builder.setPositiveButton( text: "是", (dialog, which) -> {
                    // 移除当前视图
                    ll_cart.removeView(v);
                    // 删除该商品
                    deleteGoods(info);
                });
                builder.setNegativeButton( text: "否",  listener: null);
                builder.create().show();
                return true;
            });
```

实验总结：

改进之处：

为了提高用户体验，准备接下来将购物车图标右上角的数字保存在全局内存中，而不是每次都到数据库中执行 count 操作。这样，当用户在商品详情页面将商品加入购物车时，图标上的数字也会自动加一，而不需要刷新整个页面。这样可以减少对数据库的操作，提高应用程序的运行效率。此外，还可以通过定时刷新或事件触发等方式，实现购物车图标数字的实时更新。

本次实验是一次关于 Android 应用开发的实践，主题是电商 App 的购物车的设计与实现。通过本次实验，我们学习了 Android 应用开发的相关知识和技术，包括控件的使用、存储卡的操作、网络请求等。在程序的设计中，仿照了京东商城的购物车，实现了展示商品列表、支持结算操作和清空购物车等功能。通过本次实验，了解到了 Android 应用开发的基本思路和方法，也深入了解了购物车功能的实现原理。同时，也遇到了一些问题和挑战，例如如何提高程序的运行效率、如何优化用户体验等。通过不断的实践和探索， 在未来的学习和工作中，将继续深入学习 Android 应用开发的相关知识和技术，不断提高自己的实践能力和创新能力，为移动应用的发展做出贡献。