**适配器设计模式的理解和运用**

龙科含、陈星霖、梁鹏、朱智豪

1. 适配器模式（Adapter）简介

适配器的定义是：将一个类的接口转换成客户希望的另外一个接口。使用适配器可以让原本由于接口不兼容而不能一起工作的那些类可以一起工作。

适配器模式运用的一般场景有：

1、已经存在的类的接口不符合我们的需求；

2、创建一个可以复用的类，使得该类可以与其他不相关的类或不可预见的类（即那些接口可能不一定兼容的类）协同工作；

3、在不对每一个都进行子类化以匹配它们的接口的情况下，使用一些已经存在的子类。

使用适配器模式的一般过程为：

1. 客户对象通过接口调用适配器的方法发出请求
2. 适配器把请求转换为被适配者的接口
3. 客户接收到调用的结果，对转换过程没有察觉
4. 通过适配器模式解决我们的问题

我们组基于一个简陋的开源餐厅管理程序进行修改，希望完成一个实用的、功能强大的、有针对性的、能在android手机上运行的食堂（以二食堂为参考）管理系统。

然而，这个开源的java程序不是在android上运行的。为此，我们需要将该程序移植到androd环境中运行。

在移植过程中，我们发现该程序使用arraylist链表的数据结构来储存数据，如dish（菜品），order（订单）和comment（评价）等。这种数据结构虽然简单、易于操作，且符合我们项目的需求，但不利于在android上运行。经过学习，我们知道在android上进行显示需要import特定的类，如android.view中的android.view.LayoutInflater和android.view.View等来进行显示和布局。但该程序并没有使用这些类，也没有实现所需接口。为了更好的进行移植，我们考虑使用适配器模式（Adapter）来解决这个问题。

1. 适配器模式在本项目中的具体实现

适配器可以完美地作为两个不兼容的接口之间的桥梁，在本项目中，我们实现了从普通java到使用android接口的java的转变。具体来说，在android的UI界面中，我们需要把要显示的数据和所看到的显示的内容做对应；将数据显示在相应行的控件做对应。而这个操作，使用简单的Arraylist是无法实现的。

在具体实现过程中，我们使用了若干个适配器，如DishListAdapter, OrderListAdapter, CommentAdapter等，将这些适配器中的getView（）函数进行重载（这些Adapter的基类均为android.widget.BaseAdapter，是android自带的一个适配器基类，），使其返回一个android.view.View类，使用该类即可进行相应物品在android屏幕上的显示和布局操作。

以DishListAdapter为例，其getview()函数实现为：

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

DataList dl = **null**;

**if** (convertView == **null**) {

dl = **new** DataList();

convertView = LayoutInflater.*from*(context).inflate(

R.layout.*dish\_item*, **null**);

dl.image = (ImageView) convertView.findViewById(R.id.*menu\_image*);

dl.name = (TextView) convertView.findViewById(R.id.*menu\_name*);

dl.price = (TextView) convertView.findViewById(R.id.*menu\_price*);

dl.discount = (ImageView) convertView.findViewById(R.id.*menu\_discount*);

dl.select = (ImageView) convertView.findViewById(R.id.*menu\_select*);

convertView.setTag(dl);

} **else** {

dl = (DataList) convertView.getTag();

}

ImageLoaderUtil.*displayImage*(HttpUtil.*SERVER*+catadata.get(position).getImage(), dl.image,

context);

dl.name.setText(catadata.get(position).getDishes\_name());

dl.price.setText("¥" + catadata.get(position).getPrice() + "/个");

**if** (catadata.get(position).getDiscount().equals("1")) {

dl.discount.setVisibility(View.*VISIBLE*);

} **else** {

dl.discount.setVisibility(View.*GONE*);

}

**if** (catadata.get(position).isIsselect()) {

dl.select.setVisibility(View.*VISIBLE*);

}**else** {

dl.select.setVisibility(View.*INVISIBLE*);

}

**return** convertView;

}

1. 取得的效果

使用适配器模式，我们成功地将源程序移植到了android环境中，这样既最大程度上保留了原始软件的功能和设计，为我们的项目开发节省了很多时间、提供了学习材料，又使其能够完美地在android环境中运行，很好地符合了我们项目的需求。