Report for Lab 6

Group 1

May 21, 2017

Team Name:	Smartboys
Team Leader:	Wang Duo
Partners:	Li Mingze
	Chai Zhenghao
	Jia Xiaosong

1 Prototype System Introduction

1.1 Function

Receiving users' raw data of physical experiments from touch screens, and displaying final results of experiments.

1.2 Running Environment

Windows 10

1.3 Developing Environment

Android Studio

2 Task Allocation

User Interface Component

- Adapter Design – Wang Duo
- Data Acquisition and Transmission – Jia Xiaosong
- Layout and Data Display – Li MingZe

User Interface Component

- Chai Zhenghao

3 System Architecture

3.1 User Interface Component

3.1.1 Adapter Design

The primary widget we choose for information display is ListView. ListView is an Android widget which can organize all the subwidgets in the form of a list. It can create a series of assembly components dynamically, as is shown below:

(a) Figure 1

(b) Figure 2

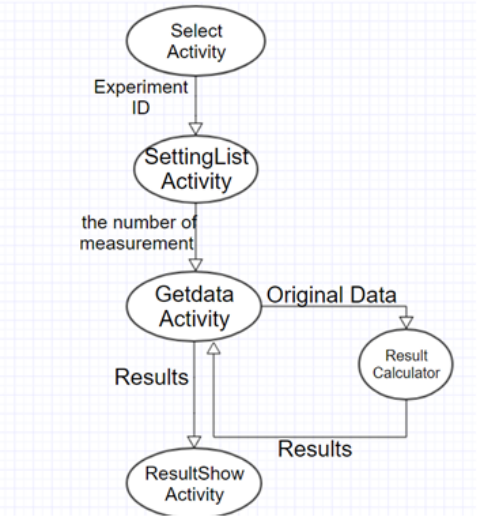
As in Figure 2, each row of the list serve as a single component, which consists of two TextViews: the variable name and the value. The key function of ListView is to create a certain number of assembly components repeatedly, and specify them with different contents.
The ListView widget requires an specially-designed object to specify the composition and layout of each component, which is called Adapter. In our project, two different types of Adapters are used.
The first one is SimpleAdapter, which is provided by Android. The contents are assigned in the form of key-value pairs, which is a parameter required to initiate a SimpleAdapter. Once the contents are assigned, each TextView in this page will display different information, and eventually form a list.
SimpleAdapter is seemingly competent enough, but there are some underlying issues. Due to the item-recycling mechanism of SimpleAdapter, some information may cannot be assigned dynamically to the components. As is shown below, the characters in the EditTexts may be lost or altered when dragging the scroll bar.

(c) Figure 3

(d) Figure 4

To solve the problem, we will need a self-defined Adapter, which we call as Custom Adapter. By using this Adapter, any information typed in by users will immediately be stored in a container. Despite the reusing of items, the contents wont be changed.

3.1.2 Data Acquisition and Transmission



First, in the Select Activity, user needs to choose the experiment and the activity will pass Experiment ID to the SettingList Activity
In the SettingList Activity: user needs to input the number of measurement. It will pass it to the GetData Activity.
In the GetData Activity: user needs to input the original data and it will be passed to the Result Calculator(A function). The calculator will return the result to GetData Activity. And then the result

will be sent to the ResultShow Activity.
Thats the structure of the data acquisition and transmission in our app.

3.2 Calculation Component

The implementation of calculation processes the raw data and outputs results of the experiments and the process of calculation, which users need to complete their experiment reports.
So far we have planned to finish the implementation of 4 experiments, so we design four classes and their get_result member functions to process and data, which are class Electron_Charge_Mass_Ratio, class Focal_Distance, class Optical_Angel_Gauge and class Moment_of_Inertia. When someone needs to use the classes to get result, they only need to call the get_result function of these classes.
Then there are three parts of the implementation designing: input designing, data process and output designing. For the input designing, we use a two-dimensional array data[][] with double type as input. In the first dimension data[] we storage different kinds of raw data, and in the second dimension data[*][], we storage data of the same kinds with different times of experiments. For the output designing, we use a array res[] to output multiple results users need.

4 Algorithm Description

4.1 User Interface Component

4.1.1 Adapter Design

4.1.1.1 Simple Adaptor

SimpleAdapter is an easy-to-use Adapter provided by Android. It requires an XML file to define the display format of each component and uses key-value pairs as parameters to specify all the contents of each component.
The key function of ListView is to create a certain number of assembly components repeatedly, and specify them with the given key-value pairs. To further explain this, Id like to quote a line of the source code in our project:

```
SimpleAdapter listAdapter = new SimpleAdapter(this,list,R.layout.setting_list_sub,
new String[]{"variable_name"}, new int[]{R.id.settinglist_text});
```

As is shown above, the construction of SimpleAdapter requires 5 parameters. The first one is a pointer of current activity, the third one is an object attached to the XML file which defines the format of each component. The rest three tells the essential mechanism of SimpleAdapter. The second parameter is a container which contains all the key-value pairs needed for the specification of components. The fourth is an array containing all the keys and the last one is an array containing the ID of all the widgets in the XML file. After such construction, the display pattern is specified by the parameters and the Adapter will organize the information accordingly on the activity.

4.1.1.2 Custom Adapter

As is mentioned above, the mechanism of SimpleAdapter has certain shortcuts. Due to the item-recycling mechanism of SimpleAdapter, some information may cannot be assigned dynamically to the components. To solve this problem, we will have to store the information immediately after input and reassign it to the components each time it is displayed. The definition of Custom Adapter is rather complicated, in order to explain the overall algorithm, Ill quote some key lines.

```
public  class Adapter extends BaseAdapter{
....
private String[] data;//Container of inputs
....
  @Override
public View getView(int position, View convertView, ViewGroup parent) {
    vh = new ViewHolder();//get the objects attached to the widgets
    vh.textview = (TextView) convertView.findViewById(R.id.settinglist_text);
    vh.edittext = (EditText) convertView.findViewById(R.id.settinglist_edittext);
....
vh.edittext.setTag(position);//attach the widget with its position in the list
....
vh.edittext.addTextChangedListener(new TextWatcher() {
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            int position = (int)vh.edittext.getTag();
                data[position] =s.toString();//store the info according to position
        }
    });
    if(!TextUtils.isEmpty(data[position]+"")){
        vh.edittext.setText(data[position]+"");
    }else{
        vh.edittext.setText("");
    }

    return convertView;
}
```

As is shown above, we define a String array data for storing the inputs. In the getView method, we set the position of the widgets as a tag. Then we set an TextChangedListener to the EditTexts to store the information. And if the item is reused and the information is crashed, we reassign the stored information to the component. By doing so, we can ensure that all the information wont get lost or altered when the widgets are reused.

4.1.2 Data Acquisition and Transmission

Data Acquisition: we use a function called getdata to get the users data from the EditView to an array. Then, we process it according to the experiment ID.
Data Transmission: we use intent to pass information between activity. We use the method of the intent putExtra including the key-value pair to realize the transmission.
In this component, the main mission is to process the data into the right form other activities and functions need.
Take the getdata activity for an example. It gets MEASUREMENT(the number of measurement), SIZE(the number of the variable), NAMES(the names of the variable) and SELECTION(the experiment ID) from the previous activity.
After processing these information, it will generate the view and then get original data from user.
After processing the original data, it will call the Calculator function and get the result.
Finally, it will pass the results and other important information to the next activity.
The following is part of the source codes of every step:

Get data from previous activity:

```
Intent intent = getIntent();
datas = intent.getIntArrayExtra("DATA");
size = intent.getIntExtra("SIZE", 0);
String[] names = intent.getStringArrayExtra("NAMES");
result_id = intent.getIntExtra("SELECTION", 0);
```

Process the data :

```
int total = 0;
for (int i = 0; i < size; ++i)
    total += datas[i];
ArrayList<HashMap<String, String>> list = new ArrayList<HashMap<String, String>>();
HashMap<String, String>[] maps = new HashMap[total];
int counter = 0;
for (int i = 0; i < size; i++)
    for (int j = 0; j < datas[i]; ++j) {
        maps[counter] = new HashMap<String, String>();
        if (j == 0)
            maps[counter].put("variable_name", names[i]);
        else
            maps[counter].put("varible_name", "      ");
        list.add(maps[counter]);
        counter++;
    }
```

Calculate the results:

```
switch (result_id) {
    case 0:
        results = Electron_Charge_Mass_Ratio.get_result(data);
        result_name = resname0;
        result_number = result_num[0];
        break;
```

Pass the results:

```
Intent intent = new Intent();
intent.putExtra("RESULTS", results);
intent.putExtra("RESULTNAME", result_name);
intent.putExtra("RESULTNUMBER",result_number);
intent.setClass(get_data.this, ShowActivity.class);
startActivity(intent);
```

4.2 Calculation Component

For processing the data, we design a class Math_Cal to help me process the data and for further plan of the project. The class Math_Cal now includes members functions average(), significance_digit() and

least_square_method(). In the further plan of our project, we would like to let users design their own template of experiment data process. At that time, we will include more functions such as function to get errors of the data and function to solve equations in this class to help users finish their own template more conveniently.

5 Demo and Testing Result

(e) Figure 6

(f) Figure 7

6 Conclusion

Physics Lab is a compulsory course for every student in SJTU majoring in Natural Science or Engineering, and the data operations are usually rather complicated in these laboratory works. We hope that our project will help some of the students to deal with these data more efficiently.