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	2015	专业（方向）	移动互联网
学号	15352344	姓名	吴文标
电话	13112312320	Email	wwb.bill@qq.com
开始日期	2017.11.29	完成日期	2017.11.29

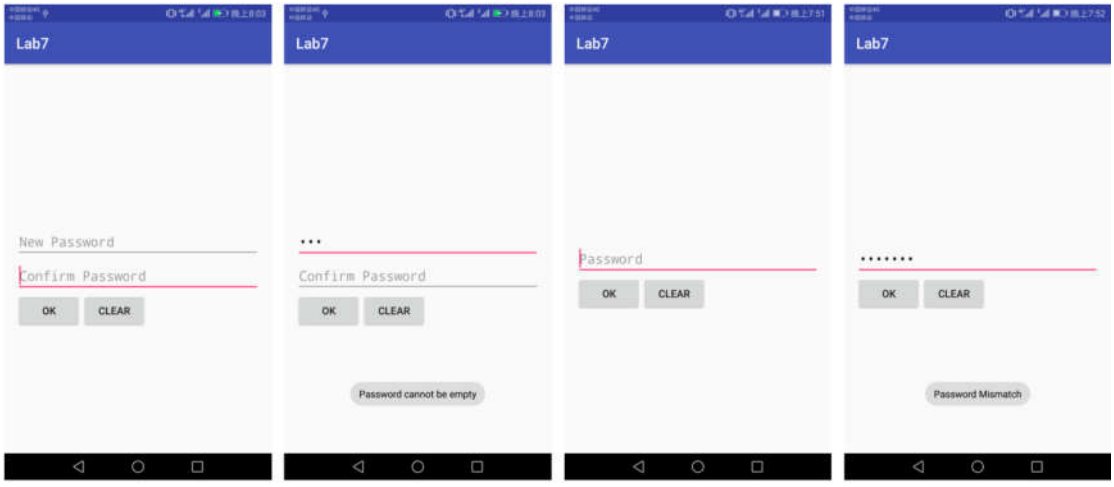
一、 实验题目

实验七 数据存储（ 一）

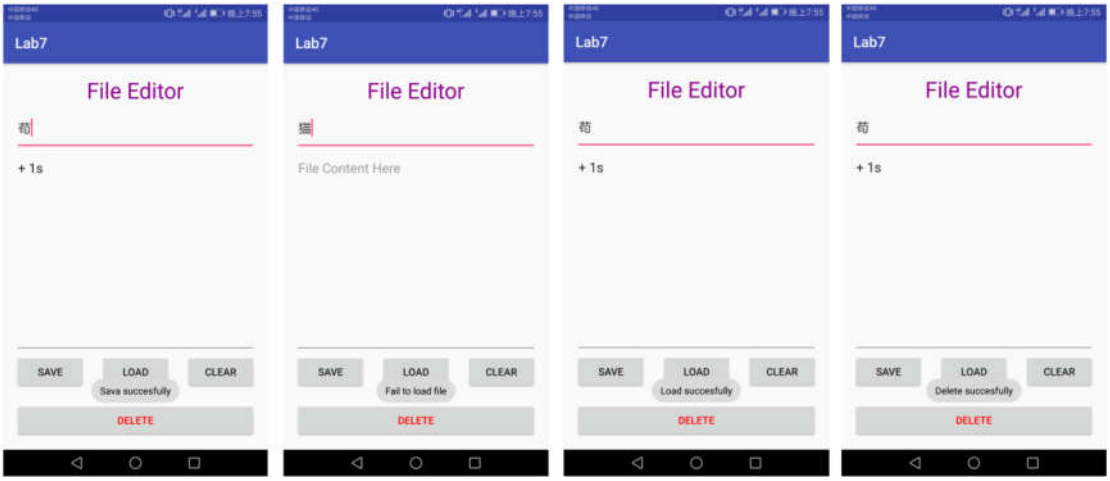
二、 实验目的

a、 学习 SharedPreferences 的基本使用；
b、 学习 Android 中常见的文件操作方法；
c、 复习 Android 界面编程。

三、 实现内容



从左至右， 依次为：初始密码界面、密码为空提示、 密码匹配后重新进入界面、 密码错误提示。



从左至右，依次为：保存成功提示、写入失败提示、 写入成功提示、删除成功提示

1、 如图所示，本次实验需要实现两个 activity；

2、 首先，需要实现一个密码输入 activity：

a、 如果应用首次启动，则界面呈现出两个输入框，分别为新密码输入和确认密码输入框；

b、 输入框下方有两个按钮：

- OK 按钮，点击之后：
- 若 new password 为空，则弹出密码为空的提示；
- 若 new password 与 comfirm password 不匹配，则弹出不匹配的提示；
- 若密码不为空且互相匹配，则保存密码，进入文件编辑界面。
- CLEAR 按钮，点击之后清除所有输入框的内容。

c、 完成创建密码后，退出应用再进入应用，则只呈现一个密码输入框；

- 点击 OK 按钮后，如果输入的密码与保存的密码不匹配，则弹出 Toast 提示；
- 点击 CLEAR 按钮后，清除密码输入框的内容。

d、 出于学习的目的，我们使用 SharedPreferences 来保存密码，但是在实际应用中我们会用更加安全的机制来保存这些隐私信息。

3、 然后，实现一个文件编辑 activity：

a、 界面底部有两行四个按钮， 第一行三个按钮高度一致，顶对齐，按钮水平均匀分布。 按钮上方除了 ActionBar 和 StatusBar 之外的空间由标题和两个 EditText 占据， 文件内容编辑的 EditText 需要占据除去其他控件的全部屏幕空间，且内部文字竖直方向置顶，左对齐；

b、 在文件名输入框内输入文件名， 在文件内容编辑区域输入任意内容， 点击 SAVE 按钮后能够保存到指定文件，成功保存后弹出 Toast 提示；

c、 点击 CLEAR 按钮，能够清空文件内容编辑区域内的内容；

d、 点击 LOAD 按钮，能够按照文件名从内存中读取文件内容，并将文件内容写入到编辑框中。如果成功导入，则弹出成功的 Toast 提示， 如果导入失败（例如： 文件不存在） ， 则弹出读取失败的 Toast 提示。

e、 点击 DELETE 按钮，能够按照文件名从内容中删除文件，删除文件后再载入文件，弹出导入失败的 Toast 提示。

4、 **特殊要求：进入文件编辑的 Activity 之后，如果点击返回按钮，则直接返回 Home 界**

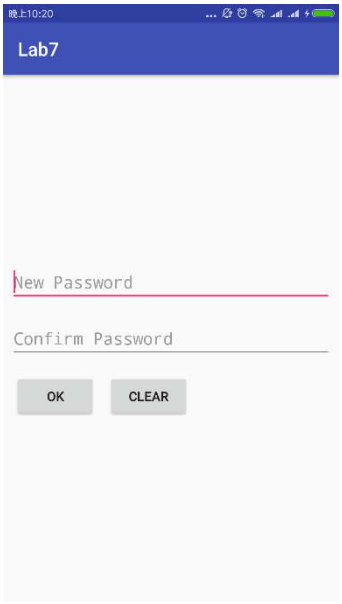
面，不再返回密码输入界面

四、 课堂实验结果

（1） 实验截图

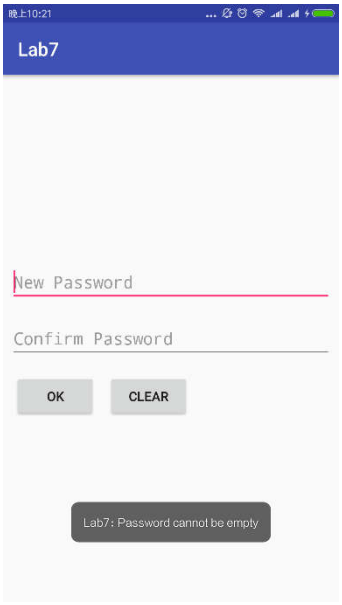
首先，需要实现一个密码输入 activity：

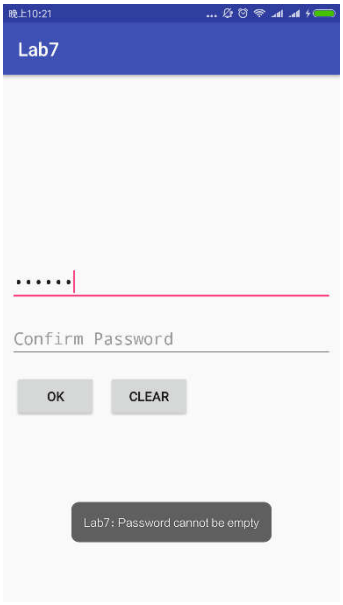
a、 如果应用首次启动，则界面呈现出两个输入框，分别为新密码输入和确认密码输入框；



b、 输入框下方有两个按钮：

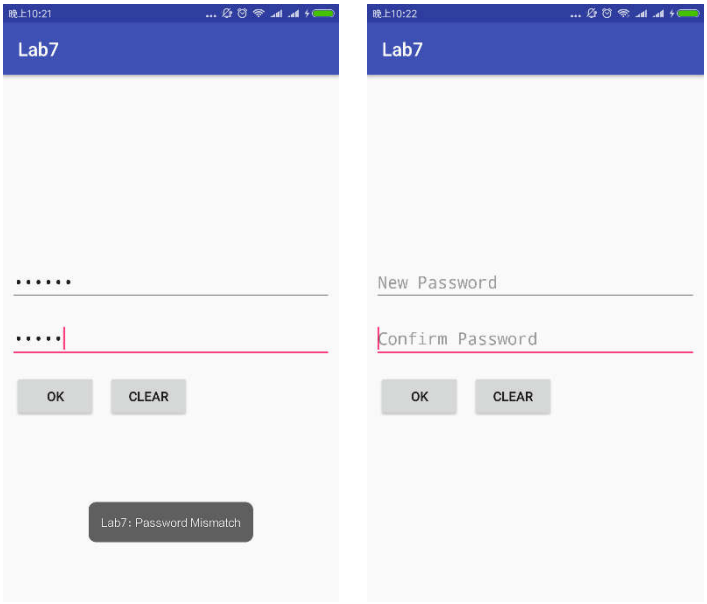
- OK 按钮，点击之后：
- 若 new password 为空，则弹出密码为空的提示；



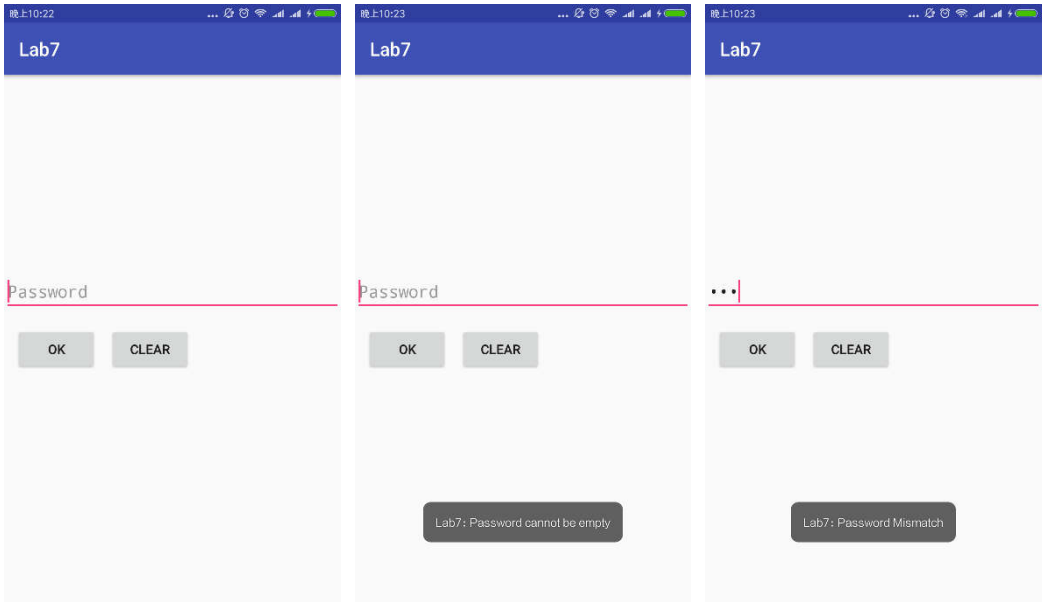


- 若 new password 与 comfirm password 不匹配，则弹出不匹配的提示；

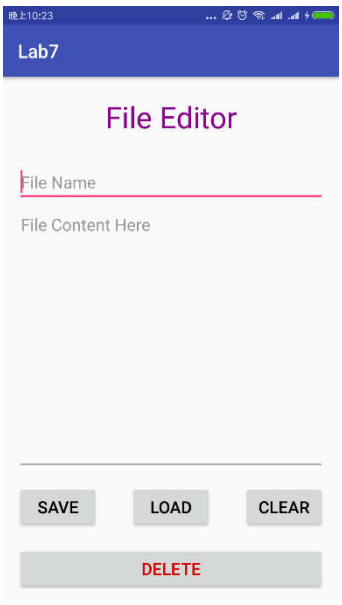
- 若密码不为空且互相匹配，则保存密码，进入文件编辑界面。
- CLEAR 按钮，点击之后清除所有输入框的内容。



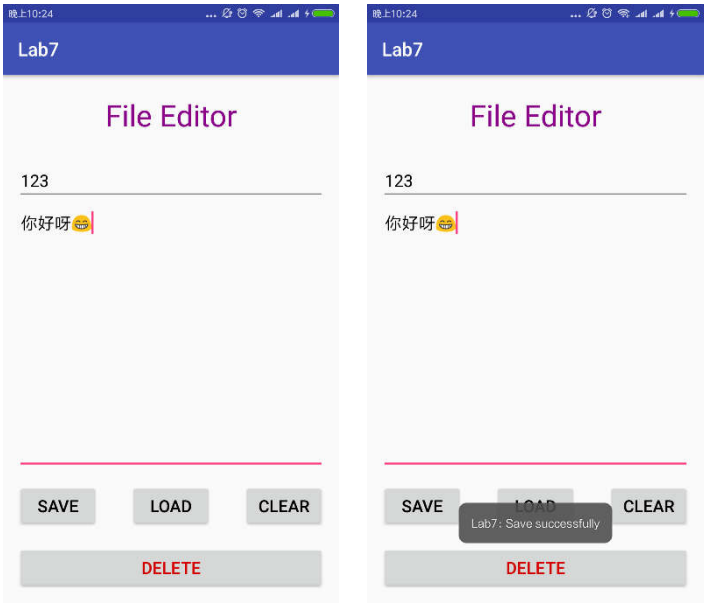
c、 完成创建密码后，退出应用再进入应用，则只呈现一个密码输入框；
- 点击 OK 按钮后，如果输入的密码与保存的密码不匹配，则弹出 Toast 提示；
- 点击 CLEAR 按钮后，清除密码输入框的内容。



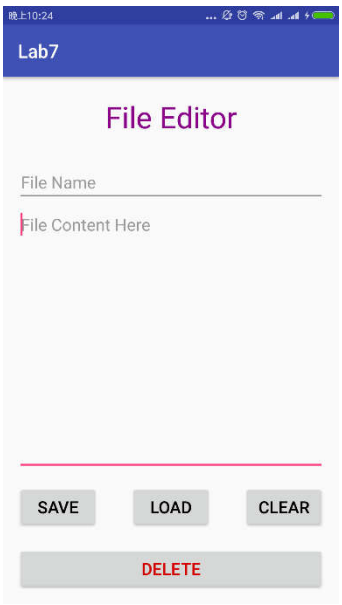
然后，实现一个文件编辑 activity：
a、 界面底部有两行四个按钮， 第一行三个按钮高度一致，顶对齐，按钮水平均匀分布。按钮上方除了 ActionBar 和 StatusBar 之外的空间由标题和两个 EditText 占据， 文件内容编辑的 EditText 需要占据除去其他控件的全部屏幕空间，且内部文字竖直方向置顶，左对齐；



b、 在文件名输入框内输入文件名， 在文件内容编辑区域输入任意内容， 点击 SAVE 按钮后能够保存到指定文件，成功保存后弹出 Toast 提示；

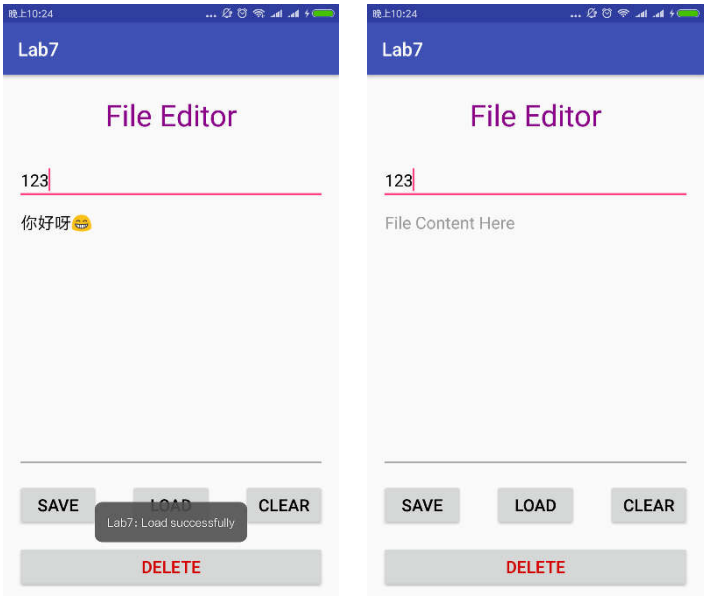


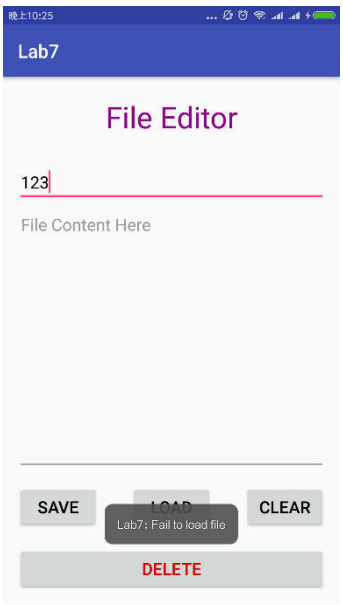
c、 点击 CLEAR 按钮，能够清空文件内容编辑区域内的内容；



d、 点击 LOAD 按钮，能够按照文件名从内存中读取文件内容，并将文件内容写入到编辑框中。如果成功导入，则弹出成功的 Toast 提示， 如果导入失败（例如： 文件不存在），则弹出读取失败的 Toast 提示。

e、 点击 DELETE 按钮，能够按照文件名从内容中删除文件，删除文件后再载入文件，弹出导入失败的 Toast 提示。

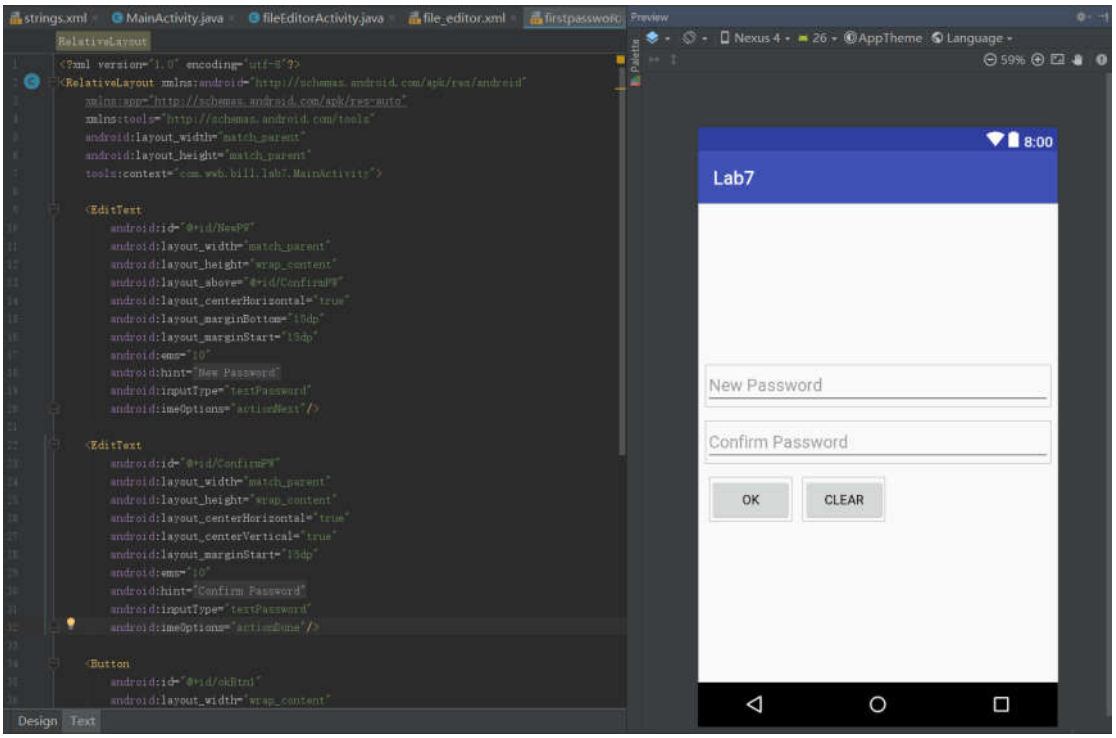


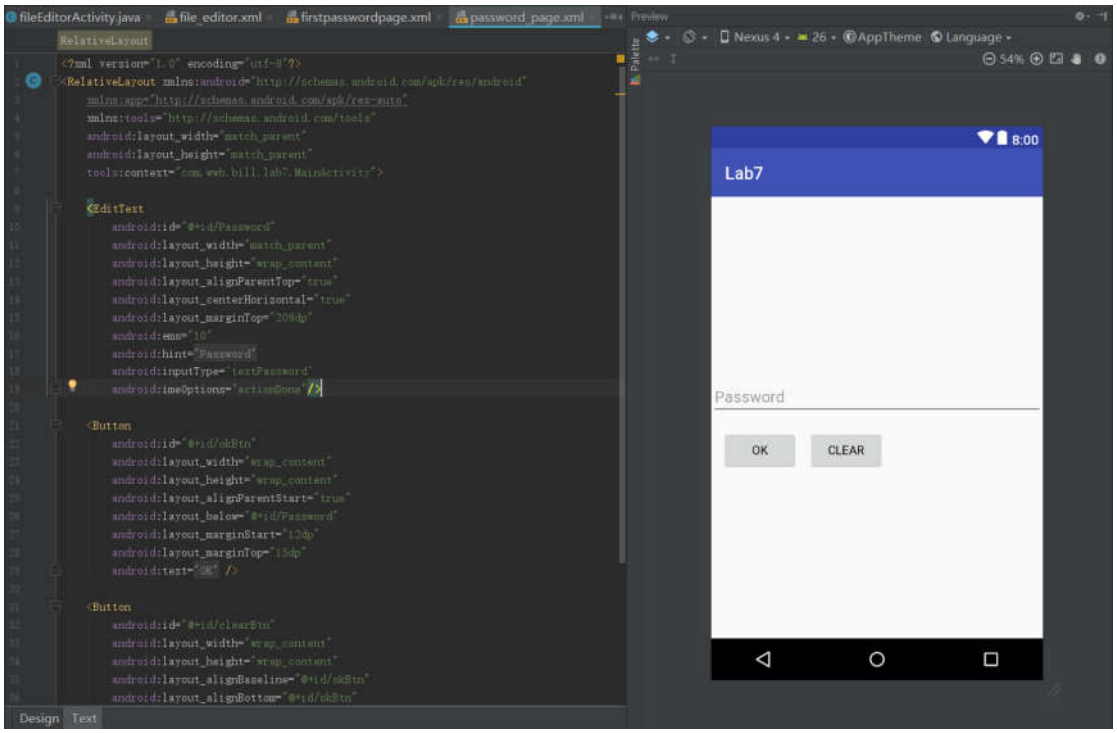


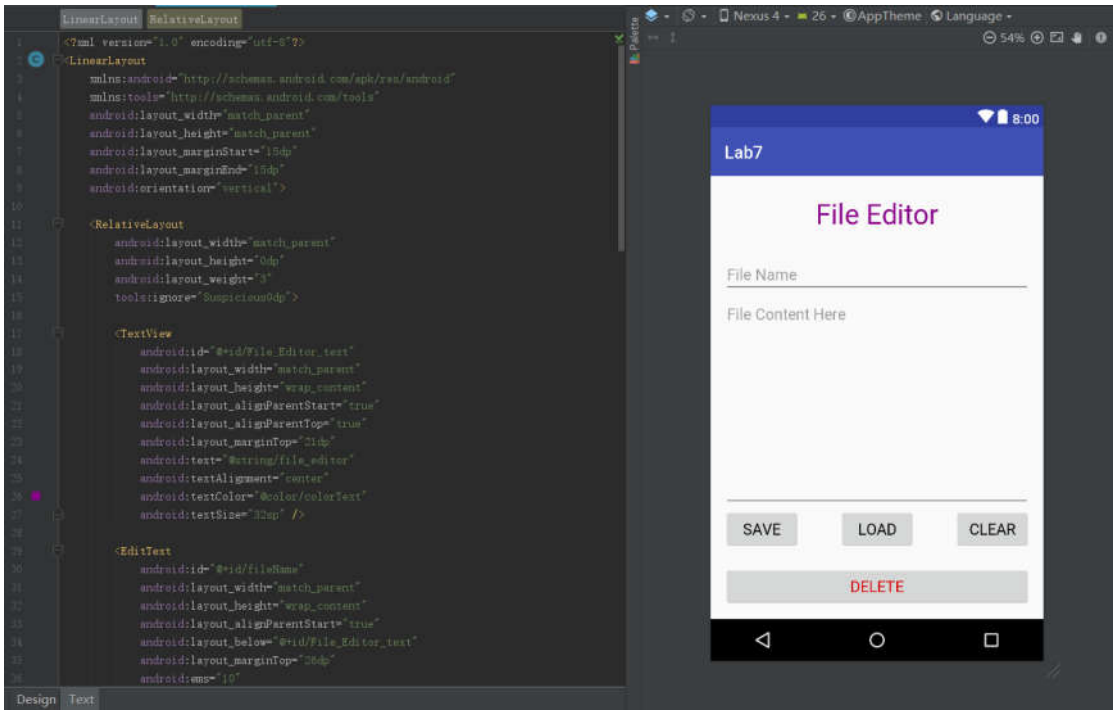
（2） 实验步骤以及关键代码

步骤 1：新建 project，布局界面实现。

1）页面布局以及部分代码







步骤 2：Activity 创建

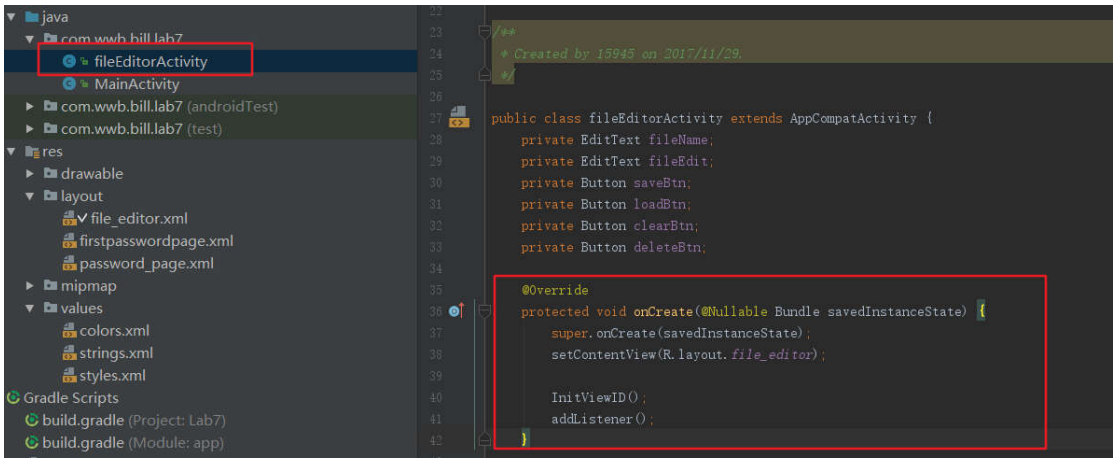
1） 清单文件 AndroidMainfest.xml

android:noHistory="true"后台自动退出 activity 栈，实现实验特殊要求：进入文件编辑的 Activity 之后，如果点击返回按钮，则直接返回 Home 界面，不再返回密码输入界面。

密码输入界面

```
manifest
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.wwb.bill.lab7">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Lab7"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity" android:noHistory="true">
            <!--android:noHistory="true"后台自动退出activity栈-->
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".fileEditorActivity" android:noHistory="true">

        </activity>
    </application>

</manifest>
```

2） 创建 fileEditorActivity 的 java 文件。



步骤 3：密码输入功能实现

1） 首先的退出应用再进入应用时只呈现一个密码输入框功能

这里我们可以通过判断版本号的方式来判断一个应用是不是第一次安装启动或者更新后启动，代码如下：

相关常量、变量：

```
public final static String PACKAGE_NAME = "com.wwb.bill.lab7";
private String VERSION_KEY = "myKey";
private boolean isfirstPage = true;
```

主体代码：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //通过判断版本号设置启动界面
    {
        PackageInfo info = null;
        try {
            info = getPackageManager().getPackageInfo(PACKAGE_NAME, i: 0);
        } catch (PackageManager.NameNotFoundException e) {
            e.printStackTrace();
        }
        int currentVersion = info.versionCode;
        SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences( context: this);

        int lastVersion = prefs.getInt(VERSION_KEY, i: 0);
        if (currentVersion > lastVersion) {
            //如果当前版本大于上次版本，该版本属于第一次启动
            setContentView(R.layout.firstpasswordpage);
            isfirstPage = true;
            //将当前版本写入preference中，则下次启动的时候，据此判断，不再为首次启动
            prefs.edit().putInt(VERSION_KEY, currentVersion).commit();
        } else{
            setContentView(R.layout.password_page);
            isfirstPage = false;
        }
    }
```

分析：

```
PackageInfo info = null;
try {
    info = getPackageManager().getPackageInfo(PACKAGE_NAME, i: 0);
} catch (PackageManager.NameNotFoundException e) {
    e.printStackTrace();
}
int currentVersion = info.versionCode;
```

PackageManger 管理类，它的主要职责是管理应用程序包。通过它可以获取应用程序信息。这里获取了应用的版本号。

```
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences( context: this);

int lastVersion = prefs.getInt(VERSION_KEY, i: 0);
if (currentVersion > lastVersion) {
    //如果当前版本大于上次版本，该版本属于第一次启动
    setContentView(R.layout.firstpasswordpage);
    isfirstPage = true;
    //将当前版本写入preference中，则下次启动的时候，据此判断，不再为首次启动
    prefs.edit().putInt(VERSION_KEY, currentVersion).commit();
} else{
    setContentView(R.layout.password_page);
    isfirstPage = false;
}
```

红框部分为 SharedPreferences 的使用，熟悉一下它的操作便能理解这部分原理。下面主要说一下第一行。

PreferenceManager.getDefaultSharedPreferences(this)与
context.getSharedPreference(name, mode)，
他们的区别在于一个是开发者自己定义名字，另一个是使用包名
+"_preference"作为存储文件名。具体可以看一下源码：
context.getSharedPreference(name, mode)

```
public SharedPreferences getSharedPreferences(String name, int mode) {

  return mBase.getSharedPreferences(name, mode);

}
```

PreferenceManager.getDefaultSharedPreferences(this)

```
public static SharedPreferences getDefaultSharedPreferences(Context
context) {

  return
context.getSharedPreferences(getDefaultSharedPreferencesName(context),

                        getDefaultSharedPreferencesMode());

}

private static String getDefaultSharedPreferencesName(Context context) {

  return context.getPackageName() + "_preferences";

}

private static int getDefaultSharedPreferencesMode() {

  return Context.MODE_PRIVATE;

}
```

注意看标红部分。

2) 根据 isfristPage 的值判断是否第一次打开，从而确定 ViewID 的获取和监听事件的设定。

```
    private void addListener() {
        okBtn.setOnClickListener(new okBtnClick());
        clearBtn.setOnClickListener(new clearBtnClick());
        if (isfirstPage) {
            confirmPW.setOnEditorActionListener(new PWfinnsh());
        }else {
            Password.setOnEditorActionListener(new PWfinnsh());
        }
    }


    private void getViewID(){
        if (isfirstPage){
            newPW = (EditText)findViewById(R.id.NewPW);
            confirmPW  =(EditText)findViewById(R.id.ConfirmPW);
            okBtn = (Button)findViewById(R.id.okBtn1);
            clearBtn = (Button)findViewById(R.id.clearBtn1);
        }else {
            Password = (EditText)findViewById(R.id.Password);
            okBtn = (Button)findViewById(R.id.okBtn);
            clearBtn = (Button)findViewById(R.id.clearBtn);
        }
    }
```

3) OK 按钮相关功能实现

```
    private class okBtnClick implements View.OnClickListener {
        @Override
        public void onClick(View view) {
            if(isfirstPage) {
                if (TextUtils.isEmpty(newPW.getText()) || TextUtils.isEmpty(confirmPW.getText())) { //判断是否为空：null 或者 ""
                    Toast.makeText(getApplicationContext(), text: "Password cannot be empty", Toast.LENGTH_SHORT).show();
                }else{
                    if (newPW.getText().toString().equals(confirmPW.getText().toString())) { //判断字符串是否相等
                        String password = confirmPW.getText().toString();
                        SharedPreferences sharedPreferences = getSharedPreferences ( PREFERENCE_NAME, MODE );//创建SharedPreferences对象
                        SharedPreferences.Editor editor = sharedPreferences.edit();//通过SharedPreferences.Editor类进行修改
                        editor.putString( s: "user",password);
                        editor.commit();
                        changeActivity();
                    }else {
                        Toast.makeText(getApplicationContext(), text: "Password Mismatch",Toast.LENGTH_SHORT).show();
                    }
                }
            }else{
                if (TextUtils.isEmpty(Password.getText())) { //判断是否为空：null 或者 ""
                    Toast.makeText(getApplicationContext(), text: "Password cannot be empty", Toast.LENGTH_SHORT).show();
                }else{
                    SharedPreferences sharedPreferences = getSharedPreferences ( PREFERENCE_NAME, MODE );//创建SharedPreferences对象
                    String userPassword = sharedPreferences.getString( s: "user", s1: "0000");
                    if(Password.getText().toString().equals(userPassword)){ //判断字符串是否相等
                        changeActivity();
                    }else {
                        Toast.makeText(getApplicationContext(), text: "Password Mismatch",Toast.LENGTH_SHORT).show();
                    }
                }
            }
        }
    }
```

分析：

TextUtils.isEmpty()方法可以判断一个 String 对象是否为 null 或者字符串长度为 0，而 String 类下的 isEmpty()返回的只是字符串的长度是否为 0，如果字符串为 null 就会直接报空指针。

SharedPreferences 修改数据的基本操作，通过键值对的方式来修改：

```
SharedPreferences sharedPreferences = getSharedPreferences ( PREFERENCE_NAME, MODE );//创建SharedPreferences对象
SharedPreferences.Editor editor = sharedPreferences.edit();//通过SharedPreferences.Editor类进行修改
editor.putString( s: "user",password);
editor.commit();
```

如果是读取数据则只需要：

```
SharedPreferences sharedPreferences = getSharedPreferences ( PREFERENCE_NAME, MODE );//创建SharedPreferences对象
String userPassword = sharedPreferences.getString( s: "user", s1: "0000");
```

这里的常量定义如下：

```
    //初始化SharedPreferences
    private final int MODE = MODE_PRIVATE;
    private final String PREFERENCE_NAME = "UserDatabase";
```

changeActivity()函数实现：

```
    private void changeActivity() {
        Intent intent = new Intent();
        intent.setClass( packageContext: MainActivity.this,fileEditorActivity.class);
        startActivity(intent);
    }
```

4） CLEAR 按钮相关功能实现

```
    private class clearBtnClick implements View.OnClickListener {
        @Override
        public void onClick(View view) {
            if (isfirstPage){
                newPW.setText("");
                confirmPW.setText("");
            }else{
                Password.setText("");
            }
        }
    }
```

直接设置 EditText 控件的 text 为空字符即可。

5)

步骤 4：File Editor 功能实现

1）控件 ID 获取

```
    private void InitViewID() {
        fileName = (EditText)findViewById(R.id.fileName);
        fileEdit = (EditText)findViewById(R.id.fileEdit);
        saveBtn = (Button)findViewById(R.id.saveBtn);
        loadBtn = (Button)findViewById(R.id.loadBtn);
        clearBtn = (Button)findViewById(R.id.clearfileEditBtn);
        deleteBtn = (Button)findViewById(R.id.deleteBtn);
    }
```

2）监听事件

```
    private void addListener() {
        saveBtn.setOnClickListener(new saveBtnClick());
        loadBtn.setOnClickListener(new loadBtnClick());
        clearBtn.setOnClickListener(new clearBtnClick());
        deleteBtn.setOnClickListener(new deleteBtnClick());

```

四个按钮的点击事件。

3）SAVE 按钮功能实现

```
77      private class saveBtnClick implements View.OnClickListener {
78          @Override
79          public void onClick(View view) {
80              String FILE_NAME = fileName.getText().toString();
81              String fileContent = fileEdit.getText().toString();
82
83
84
85
86  //          try(FileOutputStream fileOutputStream = openFileOutput(FILE_NAME,MODE_PRIVATE)){
87  //              fileOutputStream.write(fileContent.getBytes());
88  //              Toast.makeText(getApplicationContext(),"Save successfully",Toast.LENGTH_SHORT).show();
89  //          } catch (FileNotFoundException e) {
90  //              e.printStackTrace();
91  //          } catch (IOException e) {
92  //              e.printStackTrace();
93  //          }
94          }
95      }
```

框内为初始版本的保存文件功能，其功能便是向 Internal Storage 写入文件。如果对应的文件不存在，openFileOutput(String,int)函数就会直接新建一个文件，这里需要注意文件名里不能含有路径分隔符（也就是’ / ’），这个函数会返回一个 FileOutStream 对象，可以调用里面的 wirte 方法将文本写进文件中。

4）LOAD 按钮功能实现

```
97      private class loadBtnClick implements View.OnClickListener {
98          @Override
99          public void onClick(View view) {
100             String FILE_NAME = fileName.getText().toString();
101
102
103
104 //          int read;
105 //          StringBuffer strContent = new StringBuffer("");
106 //          try(FileInputStream fileInputStream = openFileInput(FILE_NAME)) {
107 //              while ((read = fileInputStream.read()) != -1){ //转换为字符串
108 //                  strContent.append((char)read);
109 //              }
110 //
111 //              fileEdit.setText(strContent);
112 //              Toast.makeText(getApplicationContext(),"Load successfully" ,Toast.LENGTH_SHORT).show();
113 //          } catch (FileNotFoundException e) {
114 //              e.printStackTrace();
115 //              Toast.makeText(getApplicationContext(),"Fail to load file" ,Toast.LENGTH_SHORT).show();
116 //          } catch (IOException e) {
117 //              e.printStackTrace();
118 //          }
119
120         }
121     }
```

框内为初版的读取文件内容并转换为 String 的代码，由于 fileInputStream.read()每次返回的都是一个字符的编码（或许是 GBK 码）所以在遇到如中文一类的字符串时会变成乱码，所以原版只能实现英文文本之类的读取转换字符串。

5）CLEAR 按钮功能实现

```
123     private class clearBtnClick implements View.OnClickListener {
124         @Override
125         public void onClick(View view) {
126             fileName.setText("");
127             fileEdit.setText("");
128         }
129     }
```

6）DELETE 按钮功能实现

```
131     private class deleteBtnClick implements View.OnClickListener {
132         @Override
133         public void onClick(View view) {
134             String FILE_NAME = fileName.getText().toString();
135             getApplicationContext().deleteFile(FILE_NAME);
136             Toast.makeText(getApplicationContext(), text: "Delete successfully" ,Toast.LENGTH_SHORT).show()
137
138         }
139     }
```

getApplicationContext().deleteFile(FILE_NAME);可以直接获取到文件的目录并删除选择的文件。

（3） 实验遇到困难以及解决思路

1) 初版中文 load 出现乱码的问题，看之前的源码发现之前的操作都是基于字节来进行的，而中文等特殊字符是多个字节组成的，这样读写自然是不能正常 load，后来经过一番资料查找发现了 InputStreamReader 和 OutputStreamWriter 类，他们的基于字符来进行文件读写的，而且还可以规定编码格式，所以就很方便地解决了这个问题。

五、 课后实验

（1） 实现中文以及 emoji 符号的读取与转换

```
77      private class saveBtnClick implements View.OnClickListener {
78          @Override
79          public void onClick(View view) {
80              String FILE_NAME = fileName.getText().toString();
81              String fileContent = fileEdit.getText().toString();
82
83              if (writeFile(FILE_NAME,fileContent)) {
84                  Toast.makeText(getApplicationContext(), text: "Save successfully",Toast.LENGTH_SHORT).show();
85              }
86  //...
94          }
95      }
96
97      private class loadBtnClick implements View.OnClickListener {
98          @Override
99          public void onClick(View view) {
100             String FILE_NAME = fileName.getText().toString();
101             String strContent = readFile(FILE_NAME);
102             fileEdit.setText(strContent);
103
104 //...
119
120         }
121     }
```

```
142     //UTF-8读写文件，解决中文乱码
143     public String readFile(String filePathAndName) {
144         String fileContent = null;
145         try {
146             FileInputStream fileInputStream = openFileInput(filePathAndName);
147             fileContent = "";
148             InputStreamReader read = new InputStreamReader(fileInputStream , charsetName: "UTF-8");
149             BufferedReader reader = new BufferedReader(read);
150             String line;
151             while ((line = reader.readLine()) != null) {
152                 fileContent += line;
153             }
154             read.close();
155             Toast.makeText(getApplicationContext(), text: "Load successfully" ,Toast.LENGTH_SHORT).show();
156         }
157         catch (FileNotFoundException e) {
158             e.printStackTrace();
159             Toast.makeText(getApplicationContext(), text: "Fail to load file" ,Toast.LENGTH_SHORT).show();
160             return null;
161         }catch (Exception e) {
162             e.printStackTrace();
163             return null;
164         }
165         return fileContent;
166     }
167
168     public boolean writeFile(String filePathAndName, String fileContent) {
169         try {
170             FileOutputStream fileOutputStream = openFileOutput(filePathAndName,MODE_PRIVATE);
171             OutputStreamWriter write = new OutputStreamWriter(fileOutputStream, charsetName: "UTF-8");
172             BufferedWriter writer = new BufferedWriter(write);
173             writer.write(fileContent);
174             writer.close();
175         } catch (Exception e) {
176             e.printStackTrace();
177             return false;
178         }
```

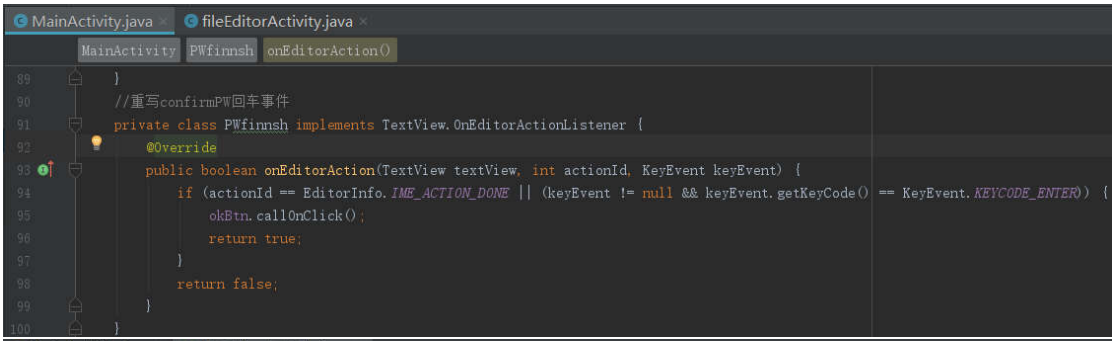
android 读取文件中文出现乱码的原因无非就是，读取文件的字符格式与写如文件的格式不一致。因此，避免中文乱码，要在写入文件的时候按照

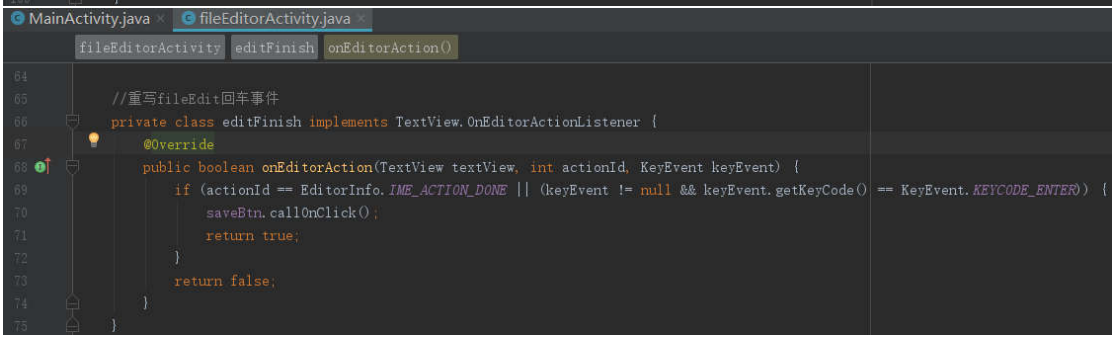
一定的格式写入，读取的时候按照一定的格式读取。这样对应就不会出现乱码。

InputStreamReader 可以将字节流转换为字符流。这里通过 InputStreamReader 和 OutputStreamWriter 来规定读写文件的编码为“UTF-8”从而避免了乱码的出现。

这里用到了 BufferedReader，通常，Reader 所作的每个读取请求都会导致对底层字符或字节流进行相应的读取请求。因此，建议用 BufferedReader 包装所有其 read()操作可能开销很高 Reader（如 FileReader 和 InputStreamReader）。

（2） 重写 EditText 回车事件，提升用户体验





这里修改了最后一个输入框的回车事件，点击回车便执行相应的按钮点击事件，如 OK 按钮、SAVE 按钮。

其他 EditText 可以直接通过 xml 设置点击回车进入下一个 EditText。

（3） 修改 file content EditText 为自动换行

```
51          fileEdit.setInputType(InputType.TYPE_TEXT_FLAG_MULTI_LINE);//设置文本显示为多行输入
52          fileEdit.setSingleLine(false);//默认单行显示设为false
53          fileEdit.setHorizontallyScrolling(false);//水平不滚动
```

六、 实验思考及感想

本次实验整体而言比较简单，毕竟是复习实验，加上一些数据文件操作，而这些在 java 在 android 里已经封装得非常完好了，只需要多读文档深挖一下原理便可很好的完成实验。本次实验感悟比较深的是，文档阅读的重要性，很多人包括我自己都不喜欢去读那些看着就很头疼的文档，然而在文档里面可以学到的东西真的很多，而且这也是一个最基本的能力。

作业要求：

1. 命名要求： 学号_姓名_实验编号，例如 15330000_林 XX_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。