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：

年级	15M1	专业（方向）	互联网
学号	15352006	姓名	蔡丽芝
电话	13538489980	Email	314749816@qq.com
开始日期	2017/12/10	完成日期	2017/12/10

一、 实验题目

数据存储（一）

二、 实现内容

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，更多可以参考链接一和链接二。

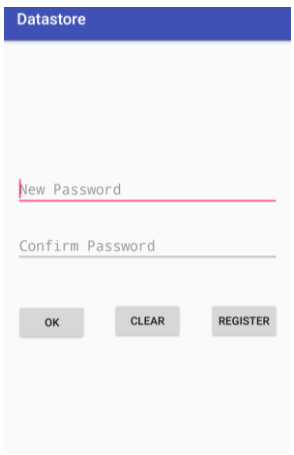
3、 然后，实现一个文件编辑 activity： a、 界面底部有两行四个按钮，第一行三个按钮高度一致，顶对齐，按钮水平均匀分布。按钮上方除 了 ActionBar 和 StatusBar 之外的空间由标题和两个 EditText 占据，文件内容编辑的 EditText 需 要占据除去其他控件的全部屏幕空间，且内部文字竖直方向置顶，左对齐； b、 在文件名输入框内输入文件名，在文件内容编辑区域输入任意内容，点击 SAVE 按钮后能够保存 到指定文件，成功

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。

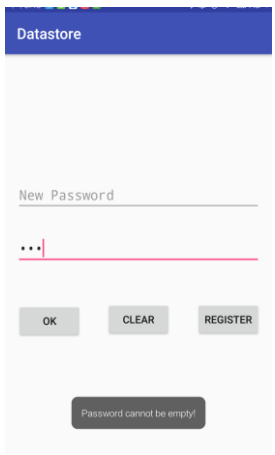
4、 特殊要求：进入文件编辑的 Activity 之后，如果点击返回按钮，则直接返回 Home 界面，不再返回 密码输入界面

三、 课堂实验结果

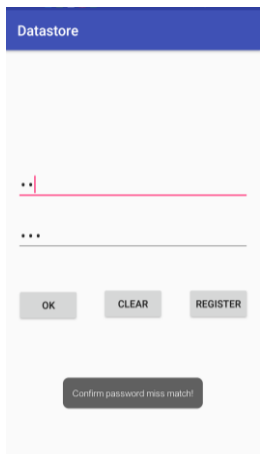
（1） 实验截图



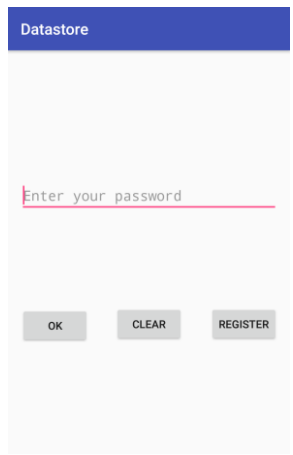
初始密码界面



密码为空的提示



密码不匹配提示

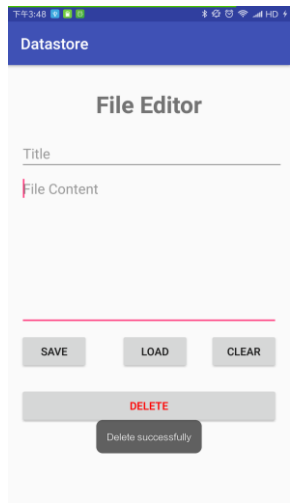


密码匹配后重新进入界面

保存成功

写入失败提示

写入成功提示



删除成功提示

（2） 实验步骤以及关键代码

MainActivity.java 文件

```
private SharedPreferences myPassSp;
private SharedPreferences.Editor mspfEditor;
```

为了能过够使用 SharedPreferences 存储数据密码，必须先声明 sharedPreference 对象

实例以及 sharedPreference 对应的实例 Editor

```
btn_ok.setOnClickListener((v) -> {
        String msg = "";
        String pw1_str = pw1.getText().toString();
        if(!is_registered) {
            String pw2_str = pw2.getText().toString();    // 获取控件pw2处的内容
            if(pw1_str.isEmpty()) {                       // 当控件pw1的内容为空时
                msg = "Password cannot be empty!";    // 进行相应的提示
            }
            else if(pw2_str.isEmpty() || !pw2_str.equals(pw1_str)) {  //如果两个密码框中的内容不一致
                msg = "Confirm password miss match!";             // 提示不匹配，重新输入
            }
            else {                                            // 来到这表示两次输入的密码已经一致，
                msg = "Register successful";                  // 注册成功，将密码保存进SharedPreferences
                mspfEditor.putString("password", pw1_str);
                mspfEditor.commit();
                Intent mIntent = new Intent(MainActivity.this, EditActivity.class);
                startActivity(mIntent);
            }
        }
```

当点击 OK 的按钮的时候，我们需要通过 sharedPreferences 保存密码，通过 sharedPreferences.editor()返回的对象 mspfEditor，调用 editor 的 putString 函数，以键值对的方式，将 pw1_str 这个密码保存到 sharePreferences 的 password 中去。接着调用 commit 函数提交键值内容。这样，就可以将密码写入 sharepreference 中去。

```
@Override
protected  void onResume() {
    super.onResume();
    myPassSp = getApplicationContext()
            .getSharedPreferences("password", Context.MODE_PRIVATE);    // 从sharePreferences中获取密码
    mspfEditor=myPassSp.edit();          // 获取sharePreferences配套的编辑器
    if(myPassSp.getString("password", null) != null) {  // 如果密码不为空，说明之前已经成功注册了
        is_registered = true;                           // 所以这时，这时只需要输入密码，将注册状态变为true
        pw2.setVisibility(View.INVISIBLE);              // 隐藏pw2的输入框
        pw1.setHint("Enter your password");
        pw1.setText("");
    }
}
```

程序无论是按 back 退出，还是按 home 退出，当需要再次进入程序的时候，都会调用 onResume 函数，而实验要求当完成创建密码后，退出应用再进入应用，需要只呈现一个密码输入框，所以我们可以重写 onResume 函数达到上述要求。

EditActivity.java 文件

```
try {
    FileOutputStream localFileOutputStream =    // 创建一名为title(变量)的文件，并返回
            getApplicationContext().openFileOutput(title, 0); //一个FileOutpuStream对象
    localFileOutputStream.write(content.getBytes());  // 将编辑框中的字符串转化为bytes，并写入上面创建的文件中
    localFileOutputStream.close();
    msg = "save successfully";

} catch (IOException e) {
    msg = "save fail";
    e.printStackTrace();
}
```

当点击 SAVE 按钮的时候，向 Internal Storage 写入文件，使用 FileOutputStream 对象保存数据文件

```
try {
    // 使用OpenFileInput函数读取文件title，并返回一个FileInputStream对象
    FileInputStream localFile = getApplicationContext().openFileInput(title);
    // 文件中的内容是以byte格式存储，所以需要一个byte类型的中间变量content
    byte[] content = new byte[localFile.available()];
    localFile.read(content); // 调用FileInputStream对象中的read函数读取文件的内容
    localFile.close();
    ed_content.setText(new String(content)); // 将content转化为Sting，并写入编辑框中
    msg = "load from file successfully";
}catch (IOException e ){
    msg = "No such file";
    ed_content.setText("");
    e.printStackTrace();
}
```

当点击 LOAD 按钮的时候，使用 FileInputStream 对象读取数据文件，无论是保存数据还是读取数据，都涉及到 I/O 操作，因此我们需要通过 try.. catch 的方式处理异常情况

```
deleteFile(title);  // 删除名为title的文件
```

当点击 delete 按钮的时候，调用 deleteFile 函数删除文件

```
@Override
public void onBackPressed() {
    super.onBackPressed();
    Intent mHomeIntent = new Intent(Intent.ACTION_MAIN);
    mHomeIntent.addCategory(Intent.CATEGORY_HOME);
    mHomeIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK
            | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED);
    startActivity(mHomeIntent);
}
```

重写 onBackPressed 函数，使得当按下返回按钮，直接返回 Home 主页面

（3） 实验遇到困难以及解决思路

实验的过程中遇到的主要困难是虚拟机的问题，不知道虚拟机是出现了什么问题，同样的程序在虚拟机上跑会崩溃，而在真机上跑没问题。后来也没有找出原因，就直接在真机上跑，期间由于大意出现了不少逻辑性的错误，后来通过调试，终于解决了问题。

四、 实验思考及感想

Internal storage 和 External storage 这两个概念的理解需要相对于应用程序来说，是逻辑上的概念，主要是逻辑上的区别，当一个应用把数据存在 external storage 上的时候，那么数据就会变成共有的，任何人任何程序都可以访问，可以存在任何的地方，当应用把数据存在 Internal Storage 的时候，那么此时的数据仅仅只是面向程序的私有数据，存储在 data/应用名称/下。