《手机平台应用开发（与 Google 共建）实验》

实验七：数据存储（一）

实验报告

学 院 名 称 ：	数据科学与计算机学院
专　　业	软件工程（计应）
学 生 姓 名 ：	张凯鑫
学　　号 ：	14331362
班　　级 ：	周三上午 4-5 节、周五下午 7-8 节

【实验目的】

1. 学习 SharedPreferences 的基本使用；
2. 学习 Android 中常见的文件操作方法；
3. 复习 Android 界面编程。

【实验内容】

Figure 1: 首次进入，呈现创建密码界面

Figure 2: 若密码不匹配，弹出 Toast 提示

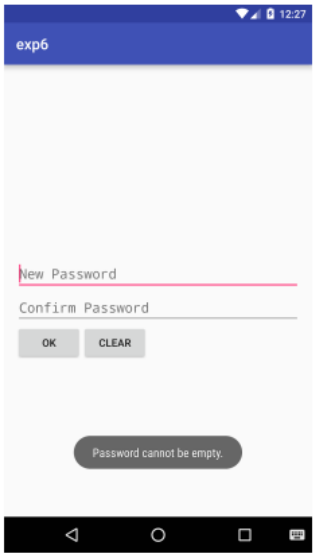


Figure 3: 若密码为空，弹出 Toast 提示

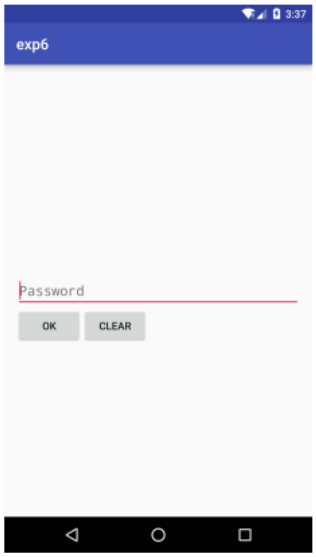


Figure 4: 退出后第二次进入呈现输入密码界面

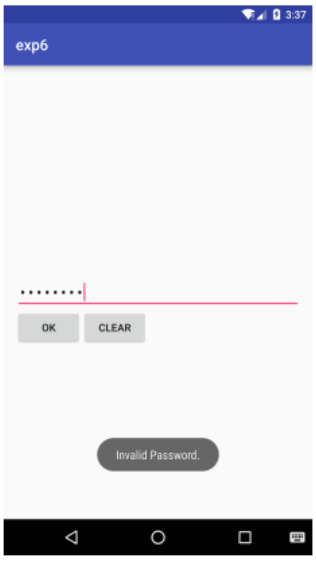


Figure 5: 若密码不正确，弹出 Toast 提示

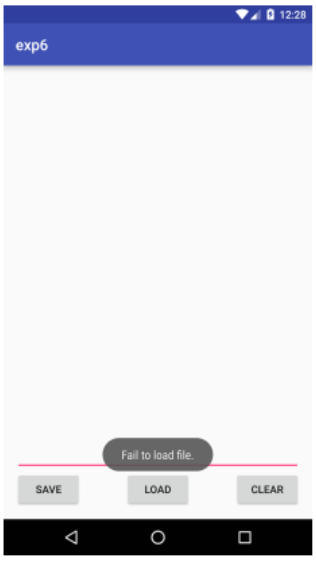


Figure 6: 文件加载失败，弹出 Toast 提示

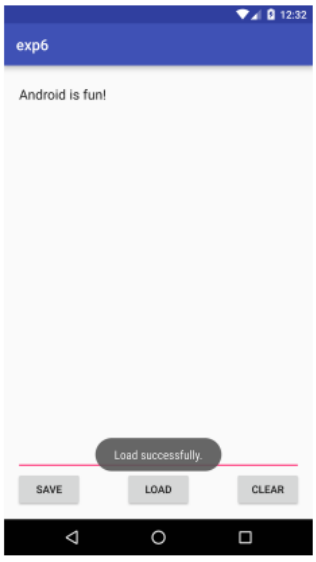


Figure 7: 成功导入文件，弹出 Toast 提示

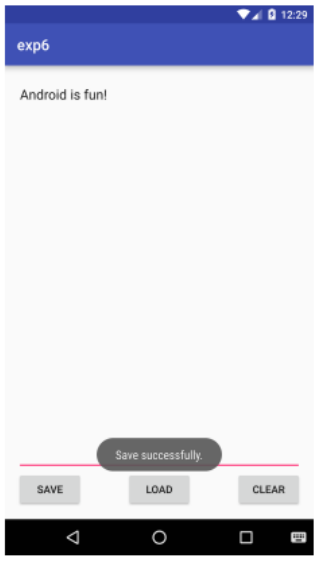


Figure 8: 成功保存文件，弹出 Toast 提示

1. 如 Figure 1 至 Figure 8 所示，本次实验演示应用包含两个 Activity.
2. 首先是密码输入 Activity:
   - 若应用首次启动，则界面呈现出两个输入框，分别为新密码输入框和确认密码输入框。

- 输入框下方有两个按钮：
  - OK 按钮点击后：
    * 若 New Password 为空，则发出 Toast 提示。见 Figure 3.
    * 若 New Password 与 Confirm Password 不匹配，则发出 Toast 提示。见 Figure 2。
    * 若两密码匹配，则保存此密码，并进入文件编辑 Activity.
  - CLEAR 按钮点击后： 清除两输入框的内容。
- 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。见 Figure 4.
  - 点击 OK 按钮后，若输入的密码与之前的密码不匹配，则弹出 Toast 提示。见 Figure 5
  - 点击 CLEAR 按钮后，清除密码输入框的内容。
- 出于演示和学习的目的，本次实验我们使用 SharedPreferences 来保存密码。

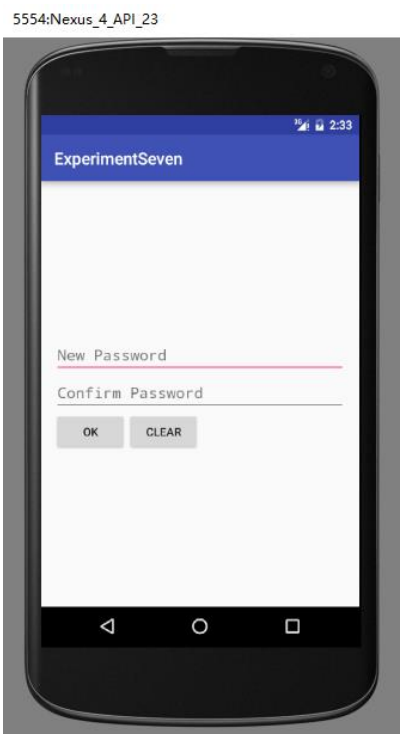
3. 文件编辑 Activity:
   - 界面底部有三个按钮，高度一致，顶对齐，按钮水平均匀分布。三个按钮上方除 ActionBar 和 StatusBar 之外的全部空间由一个 EditText 占据（保留 margin)。EditText 内的文字竖直方向置顶，左对齐。
   - 在编辑区域输入任意内容，点击 SAVE 按钮后能保存到指定文件（文件名随意)。成功保存后，弹出 Toast 提示。见 Figure 8.
   - 点击 CLEAR 按钮，能清空编辑区域的内容。
   - 点击 LOAD 按钮，能够从同一文件导入内容，并显示到编辑框中。若成功导入，则弹出 Toast 提示。见 Figure 7. 若读取文件过程中出现异常（如文件不存在)，则弹出 Toast 提示。见 Figure 6.
4. 特殊要求：进入文件编辑 Activity 后，若点击返回按钮，则直接返回 Home 界面，不再返回密码输入 Activity.

【实验过程】

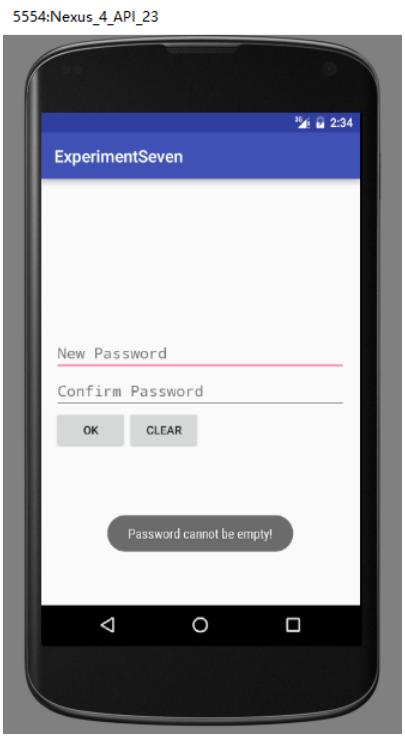
1. 创建新的 Android Studio 项目，命名为：ExperimentSeven。
2. 在布局文件 activity_main.xml 中，按照界面要求添加控件：3 个 EditText（一个默认隐藏，两个默认显示)，2 个 Button，并按要求设置控件属性。
3. 在布局文件 activity_file_editor.xml 中，按照界面要求添加控件：1 个 EditText，3 个 Button，并按要求设置控件属性。
4. 在 java 文件 MainActivity.java 中，实现 Button 控件“OK”、“CLEAR”的事件点击处理（详见实验结果部分)。“OK”按键的点击处理包含多种逻辑判断：注册新密码时，密码输入不能为空，两次输入需要匹配；新密码注册后，密码输入不能为空，密码需与之前设置的密码匹配。
5. 在 java 文件 FileEditorActivity.java 中，实现 Button 控件“SAVE”、“LOAD”、“CLEAR”的事件点击处理（详见实验结果部分)，实验中使用内部文件存储写入或读取数据。
6. 在配置文件 AndroidManifest.xml 中设置活动 MainActivity.java 的属性 noHistory 为 true，实现 Activity 的不可见，即从 stack 中除去。
7. 运行并调整实验代码直到实验完成。

【实验结果】

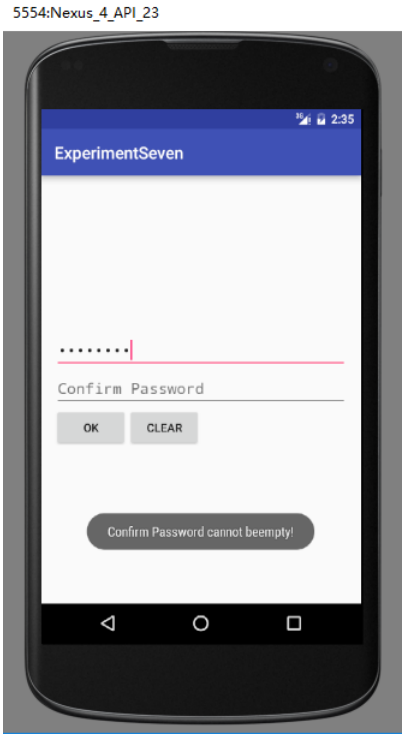
实验效果图如下（实验代码详见 lab7_code 文件夹)：



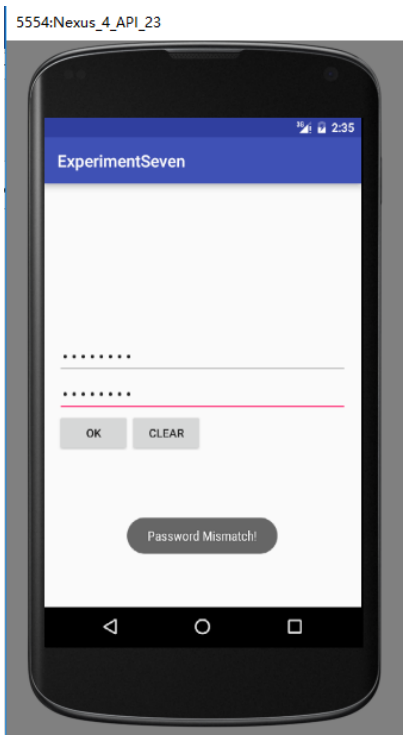
首次进入，
呈现创建密码界面



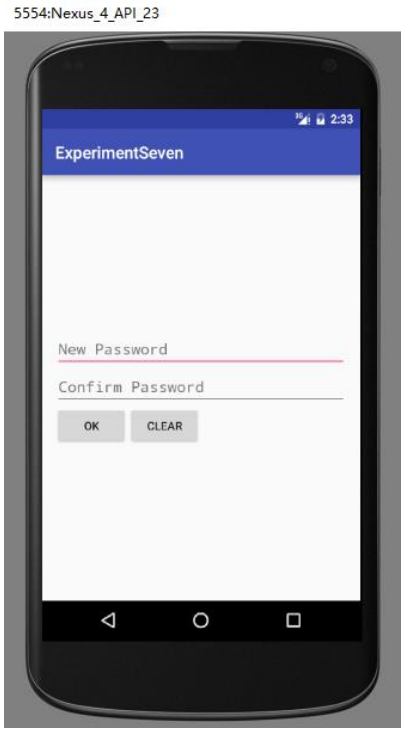
密码为空，
弹出 Toast 提示



确认密码为空
弹出 Toast 提示



密码不匹配，
弹出 Toast 提示



点击 CLEAR 按键，
输入框内容被清除

密码匹配，创建成功
成功跳转到文件编辑 Activity

关键代码如下：

```
SharedPreferences pref = getSharedPreferences("data", MODE_PRIVATE);
final String password = pref.getString("password", "");
final Boolean success = pref.getBoolean("success", false);
if (success) {
    pas.setVisibility(View.VISIBLE);
    newPas.setVisibility(View.GONE);
    conPas.setVisibility(View.GONE);
}
```

此处，success 的布尔值表示是否已创建密码成功，进而选择显示或隐藏哪些 EditText 控件，success 的值是从 SharedPreferences 中获得的 Boolean 值，默认为 false，这个 true 值的写入是在创建密码成功后。

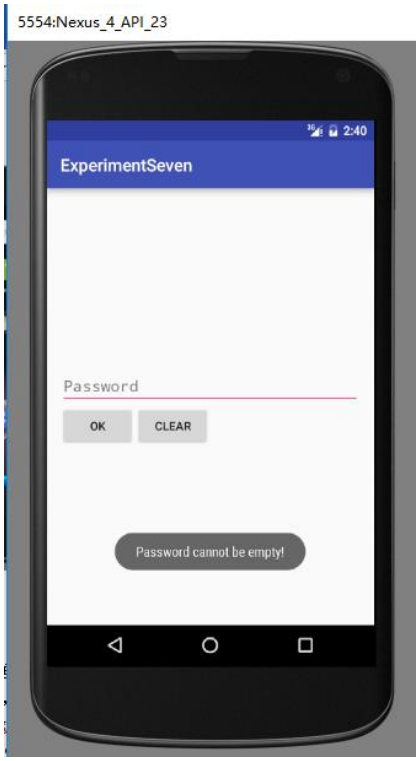
创建密码时，按键“OK”的点击事件处理逻辑如下，要求有二：不能为空，必须匹配。成功创建时跳转到文件编辑 Activity，同时，使用 SharedPreferences 来保存密码，更新 success 作为创建密码成功的标记。

```
if (!success) {
    if (TextUtils.isEmpty(newPas.getText().toString())) {
        Toast.makeText(MainActivity.this, "Password cannot be empty!",
Toast.LENGTH_SHORT).show();
    } else if (TextUtils.isEmpty(conPas.getText().toString())) {
        Toast.makeText(MainActivity.this, "Confirm Password cannot beempty!",
Toast.LENGTH_SHORT).show();
    } else if (!newPas.getText().toString().equals(conPas.getText().toString())) {
        Toast.makeText(MainActivity.this, "Password Mismatch!", Toast.LENGTH_SHORT).show();
    } else if (newPas.getText().toString().equals(conPas.getText().toString())){
        SharedPreferences.Editor editor = getSharedPreferences("password",
MODE_PRIVATE).edit();
        editor.putString("password", conPas.getText().toString());
        editor.putBoolean("success", true);
        editor.commit();
        Intent intent = new Intent(MainActivity.this, FileEditorActivity.class);
        startActivity(intent);
    }
}
```

“CLEAR”按键的点击事件处理如下：将所有的 EditText 控件的 text 设为 null 即可。

```
Button btn_clear = (Button)findViewById(R.id.clear);
btn_clear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        newPas.setText(null);
        conPas.setText(null);
        pas.setText(null);
    }
});
```

退出后第二次进入，
呈现输入密码界面



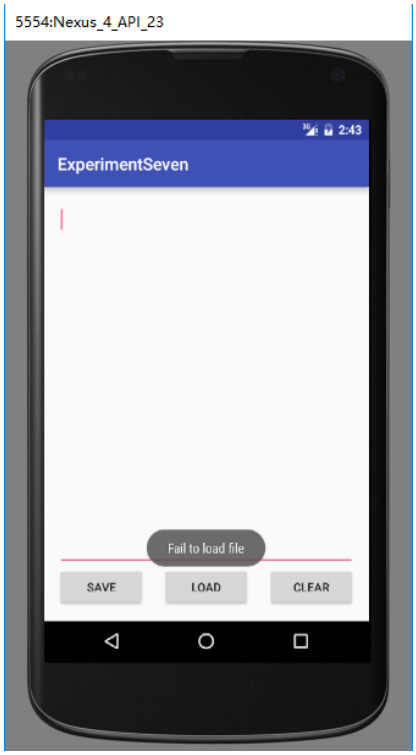
密码为空，
弹出 Toast 提示

密码与创建时密码匹配
成功跳转到文件编辑界面

关键代码如下：

二次进入界面时，success 获取到的值为 true，从 SharedPreferences 中获取 password 的值，修改 EditText 控件的显示与隐藏情况。“OK”按键的点击事件处理如下：要求有二：不能为空，须与创建密码时输入的密码 password 相匹配，异常操作会有对应的 Toast 提示，输入正确即跳转到文件编辑 Activity，不需要修改 success 或 password 的值。

```
SharedPreferences pref = getSharedPreferences("data", MODE_PRIVATE);
final String password = pref.getString("password", "");
if (success) {
    if (TextUtils.isEmpty(pas.getText().toString())) {
        Toast.makeText(MainActivity.this, "Password cannot be empty!",
Toast.LENGTH_SHORT).show();
    } else if (pas.getText().toString().equals(password)){
        Intent intent = new Intent(MainActivity.this, FileEditorActivity.class);
        startActivity(intent);
    } else {
        Toast.makeText(MainActivity.this, "invalid password", Toast.LENGTH_SHORT).show();
    }
}
```

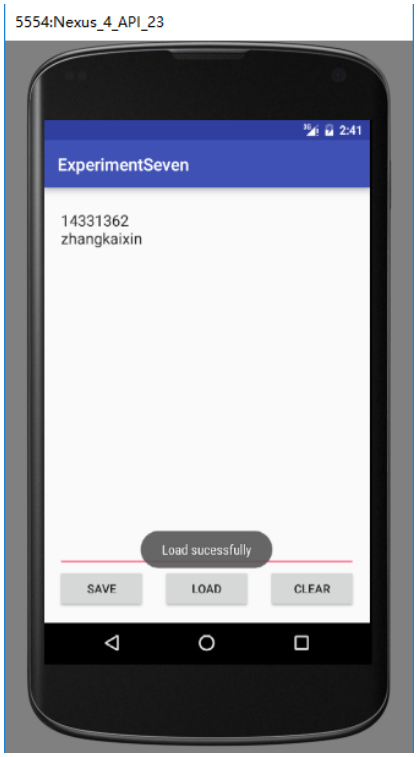


文件加载失败，
弹出 Toast 提示

点击 SAVE 按键，文件保存成功，
弹出 Toast 提示

点击返回按键，直接返回 Home 界面
不再返回密码输入 Activity

重新进入
文件编辑 Activity



点击 LOAD 按键，文件加载成功，
弹出 Toast 提示

点击 CLEAR 按键
编辑区内容被被清除

关键代码如下 ：

“SAVE”按键用于写入文件信息，此处存储是 app 内部存储。异常情况出现 Toast 提示操作失败，正常情况下将编辑框中的信息写入到文件名为“test”的文件中，Toast 提示操作成功。

```
final EditText file_edit = (EditText)findViewById(R.id.file_edit);
Button btn_save = (Button)findViewById(R.id.save);
btn_save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try (FileOutputStream fileOutputStream = openFileOutput("test", MODE_PRIVATE)) {
            fileOutputStream.write(file_edit.getText().toString().getBytes());
            fileOutputStream.close();
            Toast.makeText(FileEditorActivity.this, "Save successfully",
Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            Toast.makeText(FileEditorActivity.this, "Fail to save file",
Toast.LENGTH_SHORT).show();
        }
    }
});
```

“LOAD”按键用于读取文件信息。异常情况仅出现 Toast 提示操作失败，正常情况下将文件名为“test”的文件中的信息读取到编辑框中，Toast 提示操作成功。

```
Button btn_load = (Button)findViewById(R.id.load);
btn_load.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try (FileInputStream fileInputStream = openFileInput("test")) {
            byte[] contents = new byte[fileInputStream.available()];
            fileInputStream.read(contents);
            file_edit.setText(new String(contents));
            fileInputStream.close();
            Toast.makeText(FileEditorActivity.this, "Load successfully",
Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            Toast.makeText(FileEditorActivity.this, "Fail to load file",
Toast.LENGTH_SHORT).show();
        }
    }
});
```

【思考】在实验报告中简要描述 Internal Storage 和 External Storage 的区别，以及它们的适用场景。

区别：Internal Storage 保存的文件默认情况下只有应用程序可见，其他应用程序和用户本身是无法访问这些文件的，Internal Storage 的读写是通过 openFileOutput(FILE_NAME, MODE_PRIVATE)、

openFileInput(FILE_NAME))的 read()、write()进行，这里，FILE_NAME 是一个文件名，而不是文件的路径，即不能含有“/”。卸载应用程序后，Internal Storage 的文件也会跟着被删除。再次安装后的应用程序无法获取之前的文件。External Storage 保存文件是保存在手机 sd 卡内存上的，通过调用 getExternalFilesDir(String type) 或 Environment.getExternalStoragePublicDirectory()来获取 SD 卡路径，前者指向的目录会随着程序的卸载而被删除，而后者不会随着程序的卸载而被删除，另外，使用 External Storage 写入信息时，需要在 AndroidManifest.xml 中设置权限：<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
适用场景：Internal Storage 适用于存储小文件，且该文件是私有的；External Storage 适用于存储较大的文件，持久存在的，允许共享的文件。

【遇到的问题与解决方法】

1. 密码输入 Activity 对应的界面控件显示与隐藏问题，最初是使用 setTag()来标记“OK”按键成功创建密码，但实际过程中每次 Activity 被创建时时 setTag()的值都会被初始化，不能起到标记作用，最后从密码的保存中得到灵感，将成功创建的信息 success 与密码 password 一并写入 SharedPrefences 中，每次 Activity 被创建时取出其中的值 success，并根据其布尔值决定 EditText 控件的显示与隐藏。
2. EditText 控件占据上方全部空间。最初使用的是 RelativeLayout，在 EditText 中设置属性 android:layout_above 和 android:layout_alignParentTop，但光标显示在界面的中间位置，以为不能这样设置，于是改用 LinearLayout，用实验文档中的方法设置属性 android:layout_weight，但光标显示还是在界面的中间位置，这让我意识到是 EditText 多行的光标位置问题，网上搜索得以解决：设置 EditText 的属性 android:gravity=“top”(默认 EditText 的属性 android:gravity=“center”)。因此，让 EditText 控件占据上方全部空间的方法在 LinearLayout 下代码可以如下（其它控件不设置 layout_weight 属性）：

```
<EditText
    android:id="@+id/file_edit"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:gravity="top"/>
```

让 EditText 控件占据上方全部空间的方法在 LinearLayout 下代码如下：

```
<EditText
    android:id="@+id/file_edit"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_above="@id/save"
    android:gravity="top"/>
```

3. 文件读取后，无法成功显示在 EditText 中，因为使用 FileInputStream.read(contents)读取到的不是 String，无法直接使用 setText(contents)，之后改用 setText(contents.toString())，无效，使用 setText(new String(contents))，成功使得信息显示在 EditText 控件上。
4. 进入文件编辑 Activity 后，若点击返回按键，则直接返回 Home 界面，不再返回密码输入 Activity，要求是这样的，在 AndroidManifest.xml 中设置 noHistory 属性后，仍是无效，后来发现 noHistory 的属性被我设置到 activity 中的 action 中去了，只有设置在 activity 下才起效，更改后达到要求。

【实验心得与体会】

1. 本次实验较为简单却十分重要，主要是学习数据的存储，用到了 SharedPreferences 和 Internal Storage、External Storage，每种方式都有读取和写入两种方法，其中，SharedPreferences 是数据的存储，Internal Storage 是文件的内部存储，一般用于存储小文件，不可共享，External Storage 是文件的外部存储，一般用于存储较大文件，是共享型的。
2. 本次实验还学会了一些小的技巧，比如，将 Activity 从 stack 中去除，之前一直是用 finish()，现在学会了设置 noHistory 属性；将 EditText 占据全部剩余空间，并设置光标位置；设置控件的显示与隐藏等。