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	1501	专业（方向）	移动信息工程
学号	15352015	姓名	曹广杰
电话	13727022190	Email	1553118845@qq.com

Table of Contents

一、 实验题目 1
二、 实现内容 1
三、 课堂实验结果 2
实验截图 2
实验步骤以及关键代码 2
实验遇到困难以及解决思路 5
四、 课后实验结果 6
五、 实验思考及感想 7

一、 实验题目

事件处理

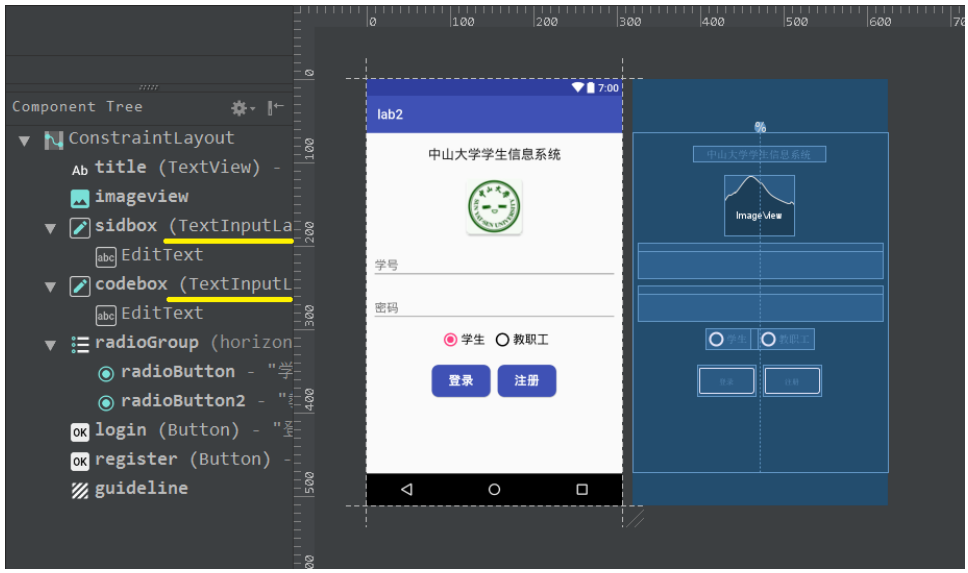
二、 实现内容

实现一个 Android 应用，界面呈现与实验一基本一致，要求：

1) 点击图片添加上传头像的功能；
2) 切换 RadioButton 的选项，弹出 Snackbar 提示已选择的单选按钮；
3) 依次判断学号是否为空，密码是否为空，用户名和密码是否正确；
4) 添加 Snackbar 信息“学生注册功能尚未启用”以及“教职工注册功能尚未启用”；

三、 课堂实验结果

实验截图



实验步骤以及关键代码

切换 RadioButton 的选项，弹出 Snackbar 提示已选择的单选按钮：

1. 设置 RadioButton 控件，初始化的时候“学生”选项被选中：
   在表示学生的布局文件里添加语句：android:checked="true"。
2. 切换的时候弹出 Snackebar 的信息：

确定添加函数的位置

在“app\MainActivity.java”布局文件中有一个 protected 函数，名为 OnCreate，这个函数表明在布局文件运行的时候，将会运行的相关操作，而笔者将会在这个函数中添加 RadioButton 的函数；

获取需要组件的实体化信息

```
stu = (RadioButton) findViewById(R.id.radioButton);
```

其中 stu 是 RadioButton 的实例化，笔者在 OnCreate 函数外部进行声明，以便能够影响到所有的后续操作；

实体化信息调用 setOnClickListener 函数

当这个 RadioButton 被 Click 的时候，应该弹出信息——这时候我们之前获取的实体化信息就用于调用 setOnClickListener 函数。

1） **setOnClickListener 函数，设置触碰监听。**
所谓“监听”就是监视触碰操作，一旦触碰操作被触发，则执行该函数内部设置的命令。内部操作是一个函数 OnClickListener，外部调用了 set 函数，则内部就应该是被设置的函数，顾名思义，应该是函数 OnClickListener。

2） **OnClickListener 函数的使用。**
该函数不需要参数传入，只需要在{}中添加我们期望进行的操作。至此，“监听 Listener”的任务就已经算是圆满完成了。在 OnClickListener 中添加的操作已经是属于非监听的范畴了。其内部调用的实际函数是外部空间可以使用的必须要重载（override）的 public 函数 OnClick。

[1]

3) **OnClick 函数的使用。**

在 OnClick 中，代码就直接表达我们将要进行的操作了。此处我们使用的操作正是 Snackbar——Snackbar 弹出提示信息，正如我们所期望的。

下图为以上函数综合使用的框架：

```
RadioButton 的实例化 = (RadioButton) findViewById(R.id. 将被设置的控件 ID);
RadioButton 的实例化.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {“BALABALABALA”}
```

以上代码添加在 OnCreate 函数内，实例化信息声明在函数外。

设置 Snackbar 的提示信息：

```
Snackbar.make(OnClick 的参数,调用的字符串信息, Snackbar.LENGTH_SHORT)
        .setAction("按键文本", new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplication(), "点击按键后出现的提示文本",
                        Toast.LENGTH_SHORT)
                        .show();
            }
        })
        .show();
```

1) **Snackbar 的要求格式。**

Snackbar.make('触发 Snack 的 view 实例','弹出信息字符串',显示时长)
.show();

触发的实例一般就是该函数向外一层的参数；

2) **为 Snack 添加右侧按键。**

在函数 make 与 show 之间使用.setAction 函数，在 Snackbar 只是一个显示框的基础上为其添加操作。

.setAction（“按键的文本信息”， 该按键的触碰监听 OnClickListener 函数）；

剩下的内容与之前的操作一样，OnClickListener 函数内置触碰时候的触发操作。；

Toast 显示文本。

Toast 函数在本次实验中使用在 OnClick 内部，与 Snackbar 的作用类似，但是只用用于产生字符串显示，不再配有更多的按钮：

Toast 函数的使用结构：

```
Toast.makeText(getApplication(), "显示的字符串",Toast.LENGTH_SHORT)
      .show();
```

至此，Snackbar 的提示信息设置就完成了，这些操作可以适用于 ImageView 等，因为这几种函数的使用方法是固定的并具有可移植性。

依次判断学号是否为空，密码是否为空，用户名和密码是否正确：

分析：

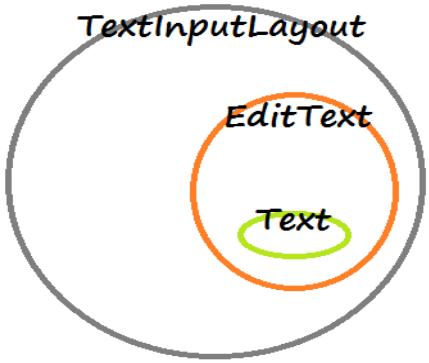
a. 密码与学号输入框是 TextInputLayout；
b. 如果两个输入框中有任何一方是空的，则按下 login 的时候会提示不能为空；
c. 如果都非空，但是不匹配，按下 login 的时候会提示不匹配；
d. 如果符合则 login；

[1]

要求 b（“不能为空”的提示信息）：

```
TextInputLayout Box = (TextInputLayout) findViewByld(R.id.sidbox);
if( TextUtils.isEmpty(Box.getEditText().getText()) ){
    Box.setErrorEnabled(true);
    Box.setError("不能为空");
}
```

我们要处理的对象是 TextInputLayout，至于是否可以为空，自然要按照人家的规矩来，好在该控件中有检查是否为空的设定。



三者为层叠包含关系，所以为了获得 text 信息，需要调用两次 get 函数，将返回值传入 isEmpty 函数中进行检验。若不合要求，则在输入框附近报错信息。

输入框检查函数：TextInputLayout 的实例化.setErrorEnabled(是否检查);因为空的故默认为错。

输入框报错函数：TextInputLayout 的实例化.setError("报错信息");

要求 b 完成。

要求 cd（“不匹配”的提示信息）：

```
if(TextUtils.equals(number.getText(), "设置的账号") &&
    TextUtils.equals(passwd.getText(), "设置的密码")){
    Snackbar.make(v, "LOGIN", Snackbar.LENGTH_SHORT)
            .show();
}
```

TextInputLayout 的比较函数与验空函数类似，只要知道该函数的存在，并且适当传参即可。之后的字符串匹配逻辑，就是常规的逻辑——这里的句法就介绍完了。

要求 cd 完成。

点击图片添加上传头像的功能：

```
    final AlertDialog.Builder 对话框 = new AlertDialog.Builder(MainActivity.this);
    对话框.setTitle("对话框标题");
    final String[] 对话框的选项= {"对话框选项 1", "对话框选项 2"};
    对话框.setItems(对话框的选项_字符串, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {}
    });
```

在点击头像图片之后必然会触发后续的程序——提示更换头像的程序。上述代码就是用于描述对话框的程序。该代码块应该在头像图片的 Click 监听器的内部。

AlertDialog 创建新的对象，创建对象的时候应该使用 AlertDialog 的 Builder。运行到 setItem 函数的时候会显示出对话框信息，第 2 个参数则表示点击这几个选项将会产生的结果，就是普通的 Click 监听器——不过调用的对象不一样了。

添加 Snackbar 信息“学生注册功能尚未启用”以及“教职工注册功能尚未启用”：

分析：当触碰“注册”按钮的时候，弹出 Snackbar 的信息。

[1]

所用句法：Snackbar

特点：根据 RadioButton 的信息确定输出信息。

```
if(checkedId == RadioButton 的实例化对象.getId()){BALABALABALA}
```

由于 Snackbar 的 Toast 信息基于 RadioButton 的状态，所以有一个不能被避免的操作——就是监听当前 RadioButton 的状态。

a. 监听当前 RadioButton 的状态。

```
RadioGroup 的对应实例化.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){
    @Override
    public void onCheckedChanged(RadioGroup group, @IdRes final int checkedId) {
        if(checkedId == RadioButton 实例化之一.getId()){}
        else if(checkedId == RadioButton 实例化之二.getId()){}
    }
});
```

设置监听函数 setOnCheckedListener 的使用：

监听切换函数 OnCheckListener 的使用；

切换函数触发 onCheckedChanged 的使用；

都和之前的 click 触发很类似，只不过本次实验中的切换触发函数的参数没有起到实质性的作用，此处便不予赘述。

获取当前 RadioButton 的状态信息：

使用 checkid 进行分类讨论，将实例化调用 getid 函数即可。

b. 监听顺序的影响：

以上，我们已经讨论了所有关于 Click 监听与 Change 监听的信息操作。下面结合当前的实验要求——按下“注册”按钮的一瞬间，如果 RadioButton 在“学生”，则输出“学生信息”，否则输出“教职工信息”。

设想情况一：Click 监听在外部，而 RadioButton 监听在内部嵌套。

那么在 click 监听触发之后，RadioButton 才有被触发的机会。这就直接导致，我们在按下过一次后，每次切换 RadioButton 都会触发信息的显示。

设想情况二：RadioButton 监听在外部，而 Click 监听在内部嵌套。

按下 RadioButton 之后，一旦按下注册按钮，就会弹出显示信息——正是我们期望的效果。

至此，笔者发现监听的顺序对最后的效果其实也有影响的，期望各位在开发的时候可以引以为戒。

实验遇到困难以及解决思路

披荆斩棘笔记：

尽管在实验之初笔者就知道要添加函数，但是在哪里添加，如何添加，添加什么函数，笔者都是一头雾水的。

以设置控件监听为例

控件监听表示某控件被触发后才启动特定的程序，那么控件监听要怎么与在 XML 布局文件中的控件关联起来呢？

其实有多种方法，但是笔者只发现了两种：

在布局文件中调用设置的触发函数。

就是说要修改布局文件（一般是 ActivityMain.xml）和函数文件。在布局文件中我们想添加函数的控件模块内部添加调用函数的语句：

```
Activity_main.xml:
<Button
    android:Balabala
```

```
    android:onClick="要调用的函数"/>
```

在函数文件中编写函数需要执行的操作：

```
MainActivity.Java
public void Register_click(View target){}
```

这样特定的控件与特定的函数就可以关联到一起了。

不修改布局文件，直接在 MainActivity.Java 中添加函数。

需要使用根据 ID 查找的功能进行查找并返回我们需要的控件的实例化。之后的操作与上面方法的操作一样——包括对函数的设计，如何触发，触发之后的行为。唯一不同的就是所有这些，都要写到 OnCreate 函数中去。声明变量一定要在某一个函数中声明才符合语法规则。

设置控件监听的时候，各个函数的理解与使用。

在笔者获悉了用于监听的几个函数之后，并不知道在什么情况下如何使用它们。而在进行了对于一个控件的完整编程之后，笔者又重新端详了这几个函数，才发现原来这些函数的作用以及使用方法都是在命名中体现出来的：

OnCheckChange：On 表触发，“CheckChange”表触发条件。该函数意为，一旦切换按钮被切换了，执行什么样子的操作，操作内容放在后续的{}中。

OnCheckChangeListener：在上文的基础上，添加了一个 Listener 表监听——表明调用这个函数的触发实例。

setOnCheckChangeListener：对之前的函数的设置。

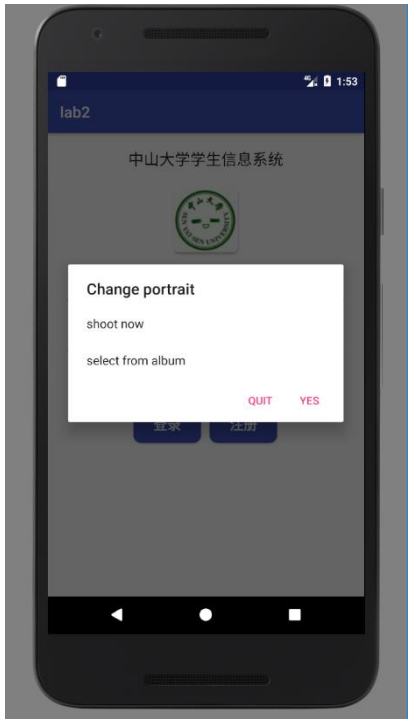
这种设计框架可以使得一个控件做到多重触发，即可以在执行的行为之前添加多个不同种类的条件——如：只有当条件 A，条件 B……满足的时候，才……。正是由于层次多，使得后续的条件限制的时候更加灵活。

使用 TextInputLayout 的过程。

TextInputLayout 的数据结构有三层，最里层是字符串，也就是我们真正可以使用的部分。其他的外层都是由地址的信息与 16 进制的信息，可读性不强。但是笔者之前使用的时候，就是因为没有分清三者的包含关系，导致浪费了很多时间。

四、 课后实验结果

如图：



这个……具体的功能就不一张一张截屏了，到时验收再解释吧。

以下是关于实现真正的拍照功能与从相机选择相片的功能：

```
void SelectPhoto(){
    Intent intent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, 1);
}
void TakePhoto(){
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(intent, 2);
}
```

a. 关于 Intent：

Intent 是 Android 程序中各组件之间进行交互的一种重要方式，不仅可以指明当前组件想要执行的动作，还可以在不同组件之间传递数据。所以在笔者需要为当前的窗口调用其他的进程的时候，就需要使用这一语法。

b. 根据我们需要的不同，传入不同的参数——或者是拍照行为（ACTION_PICK）或者是图片抓取（ACTION_IMAGE_CAPTURE），设置结束后就使用 start 运行这部分功能。

五、 实验思考及感想

本次实验在 Android 应用界面设计的基础上添加了很多要求，从布局结构的修改到新的控件的介绍，PDF 中的信息又不乏有片面错乱之嫌，总之在开发的过程中还是浪费了很多时间。

从 Toast 到 Snackbar，关于提示框的更新笔者已经不想再去说些什么了。新的技术迭代出现，如果可以大大减轻代码量、提高可读性、降低使用难度或者提高运行效率，那么都是值得学习的。笔者在使用 Snackbar 的时候出现的情况就是 Click 监听嵌套，Click 监听中又可以 Click 监听（因为笔者在 Click 的 Snackbar 中添加了按钮嘛），这种嵌套的格式可以说提高了程序的设计深度，添加了新的功能。

但是在 PDF 的讲解中，这种特点就没有很好地表现出来——很抱歉，笔者认为是这样的。PDF 的讲解效果甚至不如一些小白的博客。比如对于 Snackbar，给一个使用模板就可以快速地掌握，但是用于教学的 PDF 中却只给出代码的半个截图，这种隐藏是毫无意义的，我们不需要完整的代码，但是我们需要的关于函数调用的一整行在教学文件中一定要完整地给出，时间就是生命谢谢。

[1]

还有一点就是在 TextInputLayout 中，本来这个工具是用来代替 EditText 的，笔者期望看到的是几行——如何获取 TextInputLayout 中的字符串占一行，如何比较与设定的字符串占一行，如何检验为空占一行，如何开启输入框提示占一行，如何设定提示信息占一行，对于技术类的教学文档，文不在多而在精。

对于本次实验的一些感受，笔者出于信任，在使用的过程中努力去理解教学文件的内容，结果发现效率并不高，对于函数的解释也并不到位，不能说撰写人没有认真，只能说撰写者没有雕琢过。在大家都在歌颂实验的选材很好的时候，笔者很抱歉地提示各位，笔者身边的同学大多不能在课堂上获取什么营养这是其一，而作为重头戏的课后文档，在大家草草浏览过一遍之后也不会认真去读，虽然都不说，但是真的，他们宁可去网页上一张一张地翻博客也不会回来读与本次实验密切相关的只有几页的 PDF 文档，只有一个原因——比例。关联度高的信息占全文的比例。很抱歉这是一个令人失望的信息，但是事情还没有就此结束，笔者愿与君共勉。