实验四—分析与测试实验

191220138 杨飞洋

1.实验环境

操作系统：win10

jdk：jdk1.8.0_231

Android Studio：Android Studio 4.1

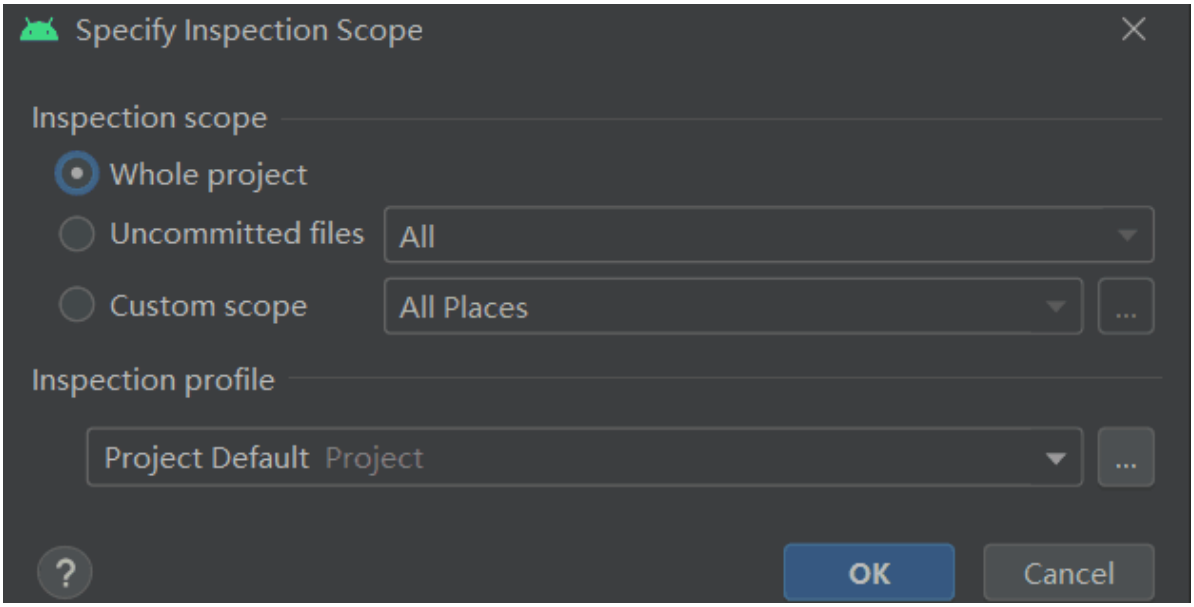
Appuim：Appium 1.21.0

2.实验目的与要求

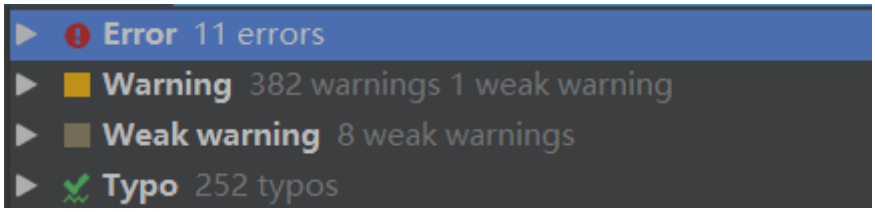
本次实验主要是为了掌握基本的程序分析方法和测试方法，从静态分析、白盒测试、黑盒测试三个方面出发，进行测试分析。

3.静态分析

在Android Studio中点击Analyze->Inspect Code，弹出以下窗口：



选择whole project进行分析，得到结果如下：



进心具体分析，得到表格如下：

类别	数目
compliance error	1
shrinker error	3
Android Resources Validation	7
Accessibility	1
Correctness	27
Internationalization	69
Performance	155
Security	1
Usability	42
Weak warning	8

4.单元测试

首先需要在build.gradle中添加依赖，所使用的是junit4，如下：

```
compile 'junit:junit:4.12'
```

但是由于本项目中的方法基本都和文件系统有关，如果要验证的话还要去访问模拟器，感觉用junit来做不大方便。

于是我就在SortUtils这个类中定义了一些容易用junit实现的方法

1. 方法一

   乘法，如下：

   ```
   public int multiply(int a,int b){
       return a*b;
   }
   ```

2. 方法二

   用空格对字符串进行拆分 ，如下:

   ```
   public String[] SplitBySpace(String s){
       return s.split(" ");
   }
   ```

3. 方法三

冒泡排序，如下：

```
public int[] BubbleSort(int[] arr){
    for (int i=0;i<arr.length-1;i++){
        for (int j=0;j<arr.length-1-i;j++){
            int temp = 0;
            if (arr[j]>arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
    return arr;
}
```

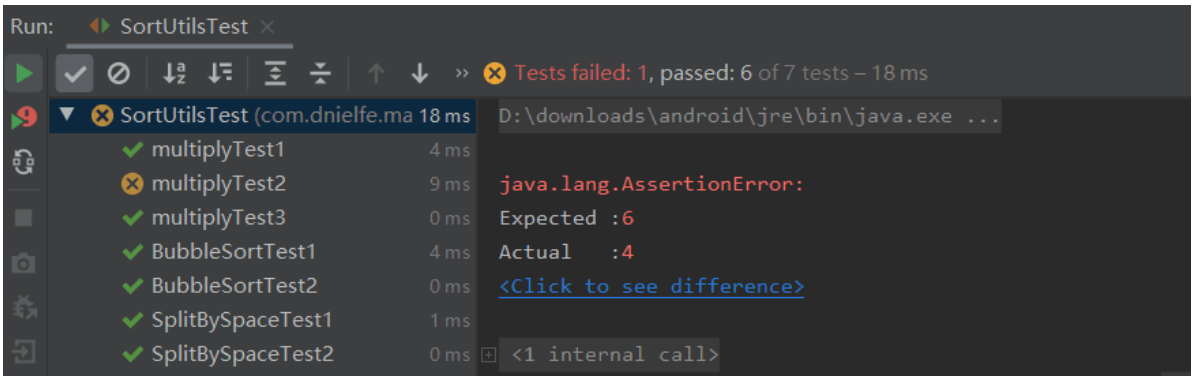
一共编写了7个用例，如下

```
public class SortUtilsTest {
    private SortUtils example;
    @Before
    public void setup() throws Exception{
        example = new SortUtils();
    }
    @Test
    public void multiplyTest1() throws Exception{
        assertEquals(8,example.multiply(2,4));
    }
    @Test
    public void multiplyTest2() throws Exception{
        assertEquals(6,example.multiply(1,4));
    }
    @Test
    public void multiplyTest3() throws Exception{
        assertEquals(4,example.multiply(2,2));
    }
    @Test
    public void BubbleSortTest1() throws Exception{
        int []src = {2,4,1,3};
        int []dest = {1,2,3,4};
        assertArrayEquals(dest,example.BubbleSort(src));
    }
    @Test
    public void BubbleSortTest2() throws Exception{
        int []src = {2,4,1,3,0,7,5};
        int []dest = {0,1,2,3,4,5,7};
        assertArrayEquals(dest,example.BubbleSort(src));
    }
    @Test
    public void SplitBySpaceTest1() throws  Exception{
        String[] dest = {"a","b","c","d"};
        assertArrayEquals(dest,example.SplitBySpace("a b c d"));
    }
    @Test
    public void SplitBySpaceTest2() throws  Exception{
```

```
        String[] dest = {"aa","bb","cc","ddd"};
        assertArrayEquals(dest,example.SplitBySpace("aa bb cc ddd"));
    }
}
```

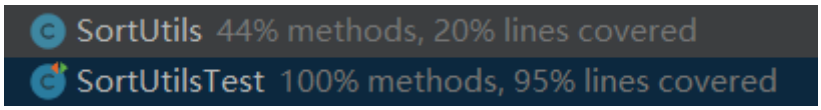
只有`multiplyTest2()`这个用例是fail的

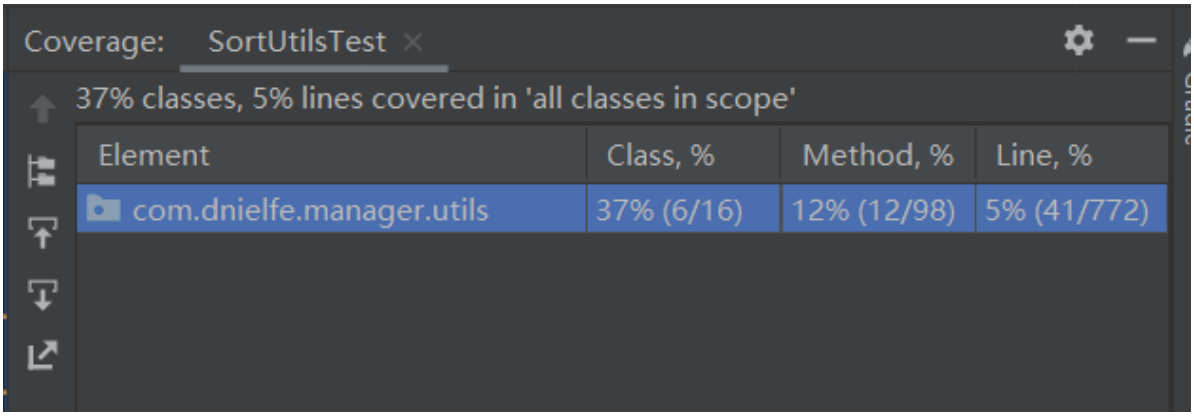
测试结果截图如下：



测试方法	功能描述	测试用例数目	pass用例数目	fail用例数目
SortUtils.multiply(int a,int b)	两个整数乘法	3	2	1
SortUtils.SplitBySpace(String s)	用空格对字符串进行拆分	3	3	0
SortUtils.BubbleSort(int []arr)	冒泡排序	2	2	0

在覆盖度模式下测试，结果如下



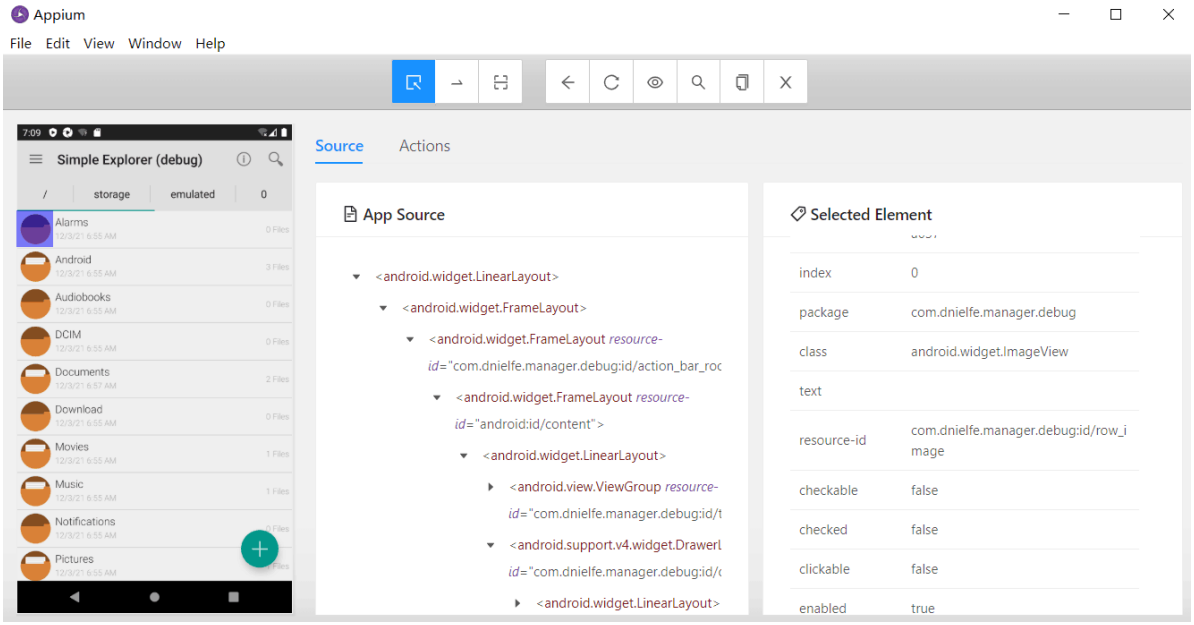


5.GUI测试

安装、使用工具

主要使用appium作为测试工具，需要下载1.21或之前的版本，因为1.22之后的版本inspector就不在appium里面了，是一个独立的软件，我在本地没有成功运行1.22。

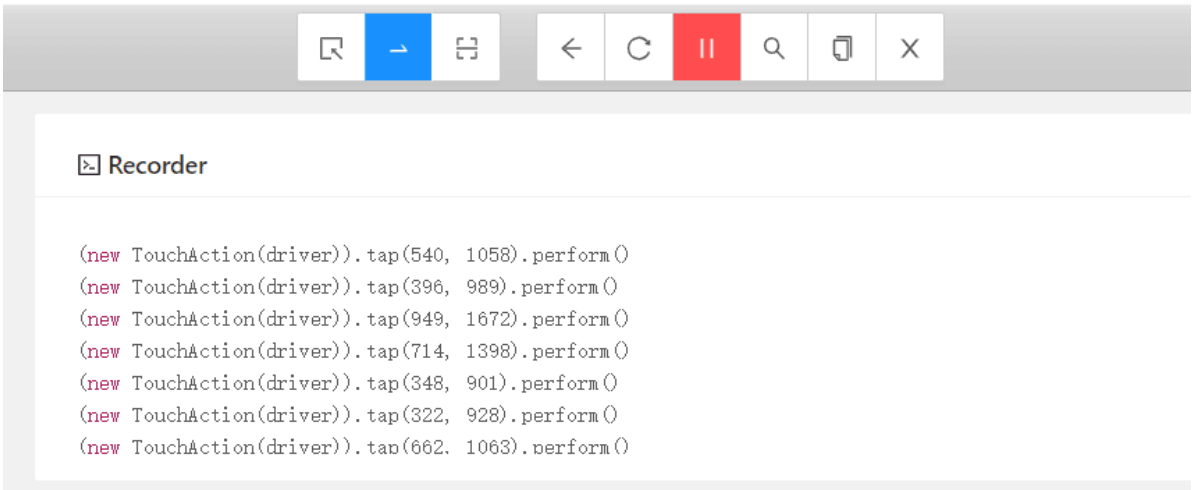
效果如下：



appium其他功能的探索：

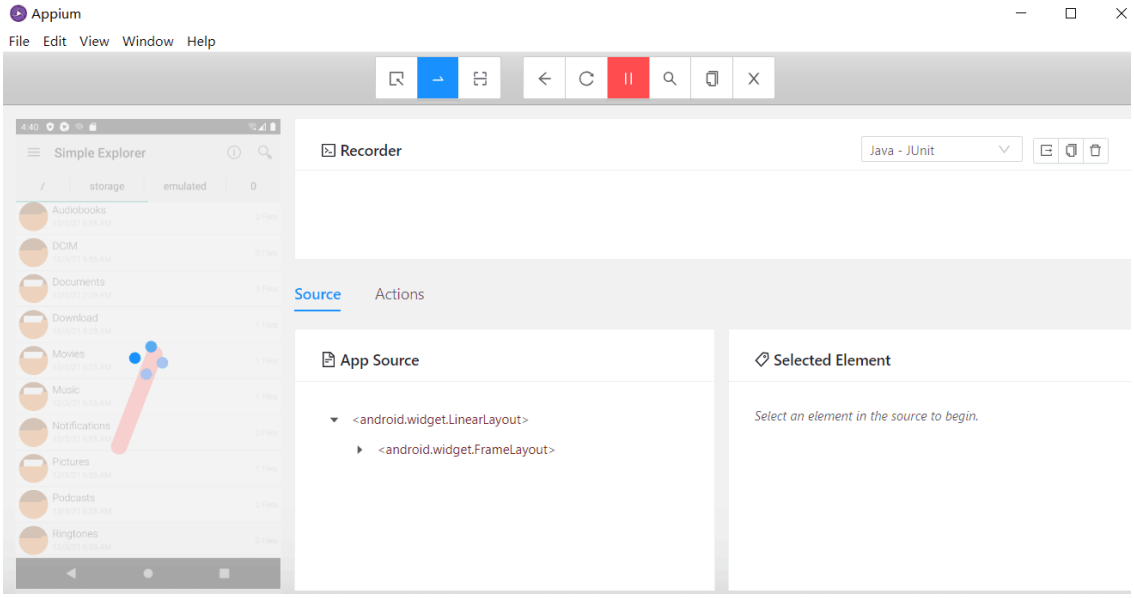
- recording

  点击上方第六个按钮，就可以记录下操作了。



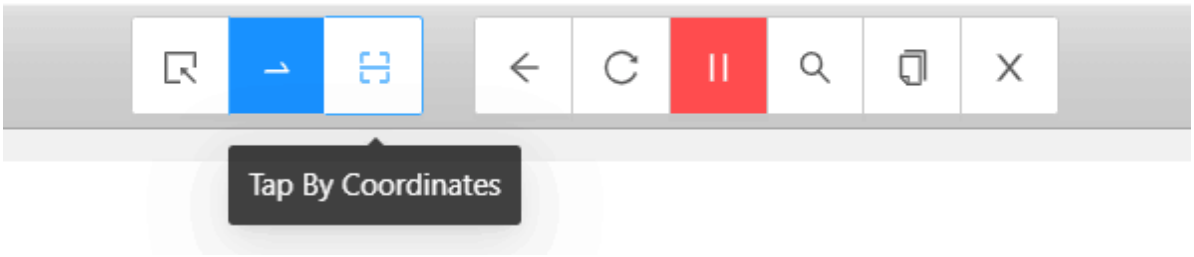
- 滑动

  点击上方第二个按钮即可开始在屏幕上进行滑动操作



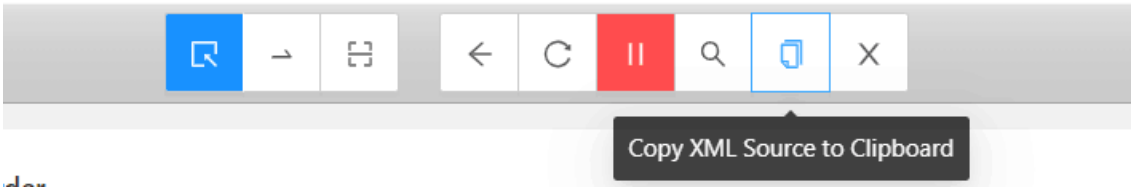
- 点击

  点击第三个按钮就可以在左边屏幕上模拟点击了，可以在上方recording功能截图上看到很多的click操作。



- 复制布局文件

  点击倒数第二个按钮就可以把布局文件拷贝到剪贴板了。



最开始需要在appium inspector中打开这个app，需要一系列的参数，其中包名和界面名可以通过以下方式获得：

1. 打开Android虚拟机，打开app。
2. 在命令行中输入以下命令即可，前提是配置好adb

```
adb shell dumpsys window | findstr mCurrentFocus
```

如下：

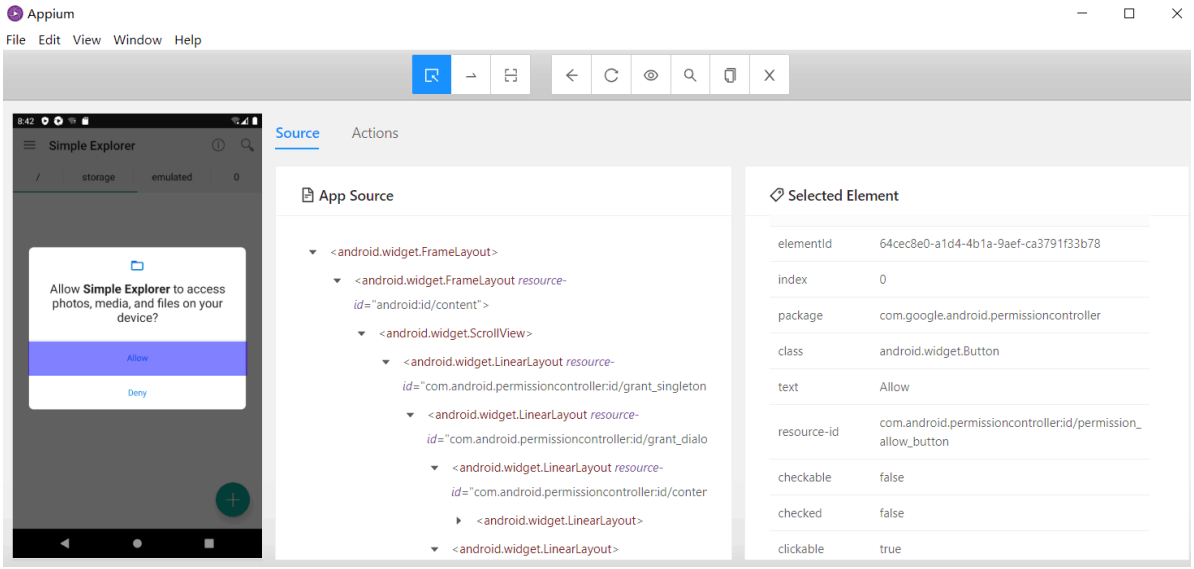
```
C:\Users\86182>adb shell dumpsys window | findstr mCurrentFocus
  mCurrentFocus=Window{c5c0769 u0 com.dnielfe.manager/com.dnielfe.manager.BrowserActivity}
```

这里是我的参数列表：

JSON Representation

```
{
  "platformName": "Android",
  "platformVersion": "11",
  "deviceName": "Pixel 2 API 30",
  "appPackage": "com.dnielfe.manager",
  "appActivity": "com.dnielfe.manager.BrowserActivity"
}
```

在用appium进入app之后，不同于手动点击的是，他会出现一个权限弹框，如下：



需要点击这个allow，查看resource-id为`com.android.permissioncontroller:id/permission_allow_button`

用`find_elements_by_id`定位，`click()`点击，代码如下：

```
allow = driver.find_elements_by_id("com.android.permissioncontroller:id/permission_allow_button")[0]
allow.click()
```

测试新建文件功能

首先在默认目录下新建一个atestdir文件夹，用于放置测试的一些辅助文件。

调用一个`createDir(filename)`函数，来实现这个新建文件夹的功能，把字符串`"atestdir"`传进去就好了。

下面看下这个函数的实现：

```
def createDir(dirname):
    createBtn = driver.find_elements_by_id("com.dnielfe.manager:id/fabbutton")
[0]
    createBtn[0].click()
    createOpts = driver.find_elements_by_id("android:id/title")
    for item in createOpts:
        if item.text == "Create new folder":
            item.click()
            break;
    for item in dirname:
        driver.press_keycode(ord(item)-97+29)
    FinalCreate = driver.find_elements_by_id("android:id/button1")#click create
    if FinalCreate:
        FinalCreate[0].click()
    time.sleep(1)
```

稍微解释一下

1. 整体的流程大致是：点击加号"+"，点击Create new folder，输入文件名，点击CREATE，完成
2. 基本都是根据resource-id来进行定位的，其中定位"Create new folder"，我是用字符串比较进行的，text属性就是对应的字符串
3. `press_keycode()`是传入一个int型，然后对应键盘啊输入，其中a~z对应29~54，用`ord()`得到对应的ascii码再进行转换即可。

然后进入这个文件夹，调用一个`clickFile(filename)`方法，把字符串`"atestdir"`传进去就好了。

看下实现：

```
def clickFile(filename):
    TestNewFile = driver.find_elements_by_id("com.dnielfe.manager:id/top_view")
    for item in TestNewFile:
        if item.text == filename:
            item.click()
            break
```

还是用`find_elements_by_id()`来找到一组控件，用`text`属性来挑选。

进入之后，创建几个文件和文件夹：

```
createDir("aa")
createFile("bb")
createDir("acopysrc")
createDir("acutsrc")
```

由于新建文件和新建文件夹在这里没什么区别，就不赘述了。

接下来进行测试，用一个`testfile(filename)`方法，实现如下：

```
def testfile(filename):
    TestNewFile = driver.find_elements_by_id("com.dnielfe.manager:id/top_view")
    for item in TestNewFile:
        if item.text == filename:
            print(filename,"test successed")
            return True
    return False
```

和`clickFile(filename)`，几乎一样的手法，只是目的不同而已。

这里随便测试两个：

```
if testfile("aa"):
    print("create dir successed")
if testfile("bb"):
    print("create file successed")
```

结果显然是对的。

测试重命名功能

思路就是选中第一个文件，然后输入`"aa"`，因为进入重命名界面之后光标位置默认在开头，相当于在文件夹名字前面加了`"aa"`，然后用`testfile(filename)`进行验证

大致如下：

```
Filelist = driver.find_elements_by_id("com.dnielfe.manager:id/top_view")
OriginalFirstFilename = Filelist[0].text
rename_firstfile("aa")
if testfile("aa"+ OriginalFirstFilename):
    print("test rename successed")
```

然后讲一下`rename_firstfile(str)`方法，这个str就是加在开头的字符串。

看下它的实现：

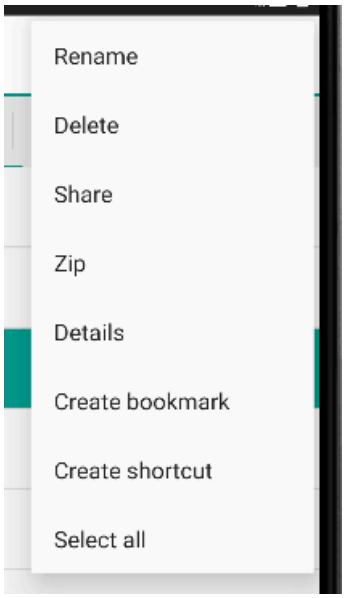
```
def rename_firstfile(newname):
    FirstFile = driver.find_elements_by_id("com.dnielfe.manager:id/top_view")

 TouchAction(driver).long_press(FirstFile[0],duration=3000).release().perform()#
长按
    driver.find_elements_by_id("com.dnielfe.manager:id/search")[0].click()
#options
    rename = driver.find_elements_by_class_name("android.widget.LinearLayout")
[0]#click rename button
    rename.click()
    for item in newname:
        driver.press_keycode(ord(item)-97+29)
    driver.find_elements_by_id("android:id/button1")[0].click()
```

长按是用`TouchAction(driver).long_press`方法实现的，第一个参数是控件，第二个参数是时间，然后在`release().perform()`

`"com.dnielfe.manager:id/search"`这个id是右上角那三个点对应的id，虽然名字的意义有点矛盾。。

`android.widget.LinearLayout`这是点击options后出现的选项的class，就是下面这几个



明显Rename排在第一个，就是第[0]号位。

然后输入、点击确认没什么可说的。

测试删除功能

用`deleteFile(filename)`删除刚才重命名的那个文件，然后用`testNofile(filename)`测试是否删除成功。

```
deleteFile("aa"+ OriginalFirstFilename)
NewFileList = driver.find_elements_by_id("com.dnielfe.manager:id/top_view")
if testNofile("aa"+ OriginalFirstFilename):
    print("delete test1 successed")
else:
    print("delete failed")
```

其中`deleteFile(filename)`的实现和重命名类似，都是要长按再点击右上角三个点，再进行操作，这里展示点击完三个点的操作。

```
delete_button =
driver.find_elements_by_class_name("android.widget.LinearLayout")[1]
delete_button.click()
driver.find_elements_by_id("android:id/button1")[0].click()
```

可以从上面看到delete排在第2个，就是数组的[1]号位，点进去之后再点击确认删除就好了。

`testNofile(filename)`和上面的`testfile(filename)`正好相反，如下：

```
def testNofile(filename):
    TestNewFile = driver.find_elements_by_id("com.dnielfe.manager:id/top_view")
    for item in TestNewFile:
        if item.text == filename:
            return False
    return True
```

测试结果正确，输出正确。

测试排序显示功能

这里就测试根据文件名称排序的情况，默认是升序的。

首先就是获得当前页面所有的文件名，从上到下。

然后遍历这个数组，确保后一项的字典序大于前一项。当然先把他们都转化为小写。

实现如下：

```
FileList = driver.find_elements_by_id("com.dnielfe.manager:id/top_view")
FilenameList = [item.text for item in FileList]
print(FilenameList)
flag = 1
for i in range(0,len(FilenameList)-1):
    if(FilenameList[i].lower() > FilenameList[i+1].lower()):
        flag = 0
        break
if not flag:
    print("sort failed")
else:
    print("sort successed")
```

`"com.dnielfe.manager:id/top_view"`，所有文件名控件的resource-id都是这个。

测试复制功能

之前在`atestdir`中新建了一个acopysrc就是现在用的，把他在atestdir目录下复制，然后返回上一级目录，粘贴，然后验证。

代码如下：

```
clickFile("atestdir")
copyFile("acopysrc")
driver.back()
paste = driver.find_elements_by_id("com.dnielfe.manager:id/paste")[0]
paste.click()
time.sleep(1)
driver.swipe(540,1000,540,1550)
if testfile("acopysrc"):
    print("copy successed")
else:
    print("copy failed")
```

其中`copyFile(filename)`的实现逻辑是：

1. 长按
2. 点击复制按钮，复制按钮的resource-id是`"id:com.dnielfe.manager:id/folderinfo"`

`driver.back()`回到上一级目录

1. 点击粘贴按钮，resource-id是`"com.dnielfe.manager:id/paste"`
2. 上滑一下，因为acopysrc毕竟字典序比较小，很可能在Android文件夹上面，这里需要用`swipe()`方法上滑，前两个参数是起始位置，后两个参数是目的位置，手机是1080*1920的，x、y坐标从左上角向右下角递增。

   则起始位置(540,1000)，结束位置(540,1550)，就可以达到上滑的效果了
3. 再用`testFile(filename)`方法进行验证就好了。

测试剪切功能

剪切功能和复制功能的步骤几乎一样，只是最后需要再回到atestdir文件下检查一下文件是不是没了就行了。

代码如下：

```
clickFile("atestdir")
cutfile("acutsrc")
driver.back()
paste = driver.find_elements_by_id("com.dnielfe.manager:id/paste")[0]
paste.click()
time.sleep(1)
driver.swipe(540,1000,540,1550)
flag1 = testfile("acutsrc")
clickFile("atestdir")
flag2 = testNofile("acutsrc")
if flag1 and flag2:
```

```
    print("cut test successed")
else:
    print("cut test failed")
driver.back()
```

其中剪切按钮对应的resource-id是`"com.dnielfe.manager:id/actionmove"`

输出日志

最后把默认目录下的atestdir、acopysrc、acutsrc删除

```
driver.swipe(540,1000,540,1550)
deleteFile("atestdir")
deleteFile("acopysrc")
deleteFile("acutsrc")

if testNofile("atestdir") and testNofile("acoptsrc") and testNofile("acutsrc"):
    print("delete 3dirs successed")
else:
    print("delete 3dirs failed")
```

由于我所有的测试代码都在testcase.py这个文件里，所以输出日志就在这里统一看一下：

```
PS D:\src\2021fall\SE\实验4_191220138_杨飞洋> python -u "d:\src\2021fall\SE\实验4_191220138_杨飞洋\testcase.py"
create dir successed
create file successed
[Deprecated] 'TouchAction' action is deprecated. Please use W3C actions instead.
test rename successed
[Deprecated] 'TouchAction' action is deprecated. Please use W3C actions instead.
delete test successed
['atestdir', 'Audiobooks', 'DCIM', 'Documents', 'Download', 'Movies', 'Music', 'Notifications', 'Pictures', 'Podcasts']
sort successed
[Deprecated] 'TouchAction' action is deprecated. Please use W3C actions instead.
copy successed
[Deprecated] 'TouchAction' action is deprecated. Please use W3C actions instead.
cut test successed
[Deprecated] 'TouchAction' action is deprecated. Please use W3C actions instead.
[Deprecated] 'TouchAction' action is deprecated. Please use W3C actions instead.
[Deprecated] 'TouchAction' action is deprecated. Please use W3C actions instead.
delete 3dirs successed
```

表格

测试功能	样例数目	代码行数
新建文件	4	60
重命名	1	30
删除文件	4	39
排序显示	1	14
复制文件	1	34
剪切文件	1	37

6.实验总结

实验时长：10h

报告撰写时长：1.5h

总结：测试真是不简单。