# 《停指测试技术》实验报告书

学生姓名：吕煜杰

学 号：31601360

班 级：软件工程1601

指导老师：杨枨

专 业：软件工程

提交时间：2018年6月2日

# 白盒测试实验报告

# 实验目的

# 通过简单程序白盒测试，熟悉测试过程，对软件测试形成初步

# 了解，并养成良好的测试习惯。熟练掌握如何运用基路径测试方法进行测试用例设计，初步熟悉如何利用程序插装技术进行逻辑覆盖率分析。

# 实验内容

## 系统地学习和理解白盒测试的基本概念、原理，掌握白盒测试的基本技术和方法

## 举例进行白盒测试

# 实验步骤

# 1.程序流程图

图标

扫描

开始

**N**

**Y**

APP信息列表

结束

**N**

包名

APP信息列表

**Y**

# 2.代码

定义一个类来代表我们扫描到的App

**public class** MyAppInfo {  
 **private** Drawable **image**;  
 **private** String **appName**;  
  
 **public** MyAppInfo(Drawable image, String appName) {  
 **this**.**image** = image;  
 **this**.**appName** = appName;  
 }  
 **public** MyAppInfo() {  
  
 }  
  
 **public** Drawable getImage() {  
 **return image**;  
 }  
  
 **public void** setImage(Drawable image) {  
 **this**.**image** = image;  
 }  
  
 **public** String getAppName() {  
 **return appName**;  
 }  
  
 **public void** setAppName(String appName) {  
 **this**.**appName** = appName;  
 }  
}

接下来再定义一个ApkTool的工具类，用来帮助我们获取当前所安装的App

*/\*\*  
 \* Created by gray\_dog3 on 16/3/3.   
 \* 扫描本地安装的应用,工具类   
 \*/***public class** ApkTool {  
 **static** String *TAG* = **"ApkTool"**;  
 **public static** List<MyAppInfo> *mLocalInstallApps* = **null**;  
  
 **public static** List<MyAppInfo> scanLocalInstallAppList(PackageManager packageManager) {  
 List<MyAppInfo> myAppInfos = **new** ArrayList<MyAppInfo>();  
 **try** {  
 List<PackageInfo> packageInfos = packageManager.getInstalledPackages(0);  
 **for** (**int** i = 0; i < packageInfos.size(); i++) {  
 PackageInfo packageInfo = packageInfos.get(i);  
 *//过滤掉系统app   
// if ((ApplicationInfo.FLAG\_SYSTEM & packageInfo.applicationInfo.flags) != 0) {   
// continue;   
// }* MyAppInfo myAppInfo = **new** MyAppInfo();  
 myAppInfo.setAppName(packageInfo.packageName);  
 **if** (packageInfo.applicationInfo.loadIcon(packageManager) == **null**) {  
 **continue**;  
 }  
 myAppInfo.setImage(packageInfo.applicationInfo.loadIcon(packageManager));  
 myAppInfos.add(myAppInfo);  
 }  
 }**catch** (Exception e){  
 Log.e(*TAG*,**"===============获取应用包信息失败"**);  
 }  
 **return** myAppInfos;  
 }  
  
}

调用ApkTool

**private void** initAppList(){  
 **new** Thread(){  
 @Override  
 **public void** run() {  
 **super**.run();  
 *//扫描得到APP列表* **final** List<MyAppInfo> appInfos = ApkTool.scanLocalInstallAppList(MainActivity.**this**.getPackageManager());  
 **mHandler**.post(**new** Runnable() {  
 @Override  
 **public void** run() {  
 mAppAdapter.setData(appInfos);  
 }  
 });  
 }  
 }.start();  
}

ListView部分

**public class** MainActivity **extends** AppCompatActivity {  
 **private** ListView **lv\_app\_list**;  
 **private** AppAdapter **mAppAdapter**;  
 **public** Handler **mHandler** = **new** Handler();  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 **lv\_app\_list** = (ListView) findViewById(R.id.lv\_app\_list);  
 **mAppAdapter** = **new** AppAdapter();  
 **lv\_app\_list**.setAdapter(**mAppAdapter**);  
 initAppList();  
 }  
  
 **private void** initAppList(){  
 **new** Thread(){  
 @Override  
 **public void** run() {  
 **super**.run();  
 *//扫描得到APP列表* **final** List<MyAppInfo> appInfos = ApkTool.scanLocalInstallAppList(MainActivity.**this**.getPackageManager());  
 **mHandler**.post(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **mAppAdapter**.setData(appInfos);  
 }  
 });  
 }  
 }.start();  
 }  
  
  
  
 **class** AppAdapter **extends** BaseAdapter {  
  
 List<MyAppInfo> **myAppInfos** = **new** ArrayList<MyAppInfo>();  
  
 **public void** setData(List<MyAppInfo> myAppInfos) {  
 **this**.**myAppInfos** = myAppInfos;  
 notifyDataSetChanged();  
 }  
  
 **public** List<MyAppInfo> getData() {  
 **return myAppInfos**;  
 }  
  
 @Override  
 **public int** getCount() {  
 **if** (**myAppInfos** != **null** && **myAppInfos**.size() > 0) {  
 **return myAppInfos**.size();  
 }  
 **return** 0;  
 }  
  
 @Override  
 **public** Object getItem(**int** position) {  
 **if** (**myAppInfos** != **null** && **myAppInfos**.size() > 0) {  
 **return myAppInfos**.get(position);  
 }  
 **return null**;  
 }  
  
 @Override  
 **public long** getItemId(**int** position) {  
 **return** 0;  
 }  
  
 @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent) {  
 ViewHolder mViewHolder;  
 MyAppInfo myAppInfo = **myAppInfos**.get(position);  
 **if** (convertView == **null**) {  
 mViewHolder = **new** ViewHolder();  
 convertView = LayoutInflater.from(getBaseContext()).inflate(R.layout.item\_app\_info, **null**);  
 mViewHolder.**iv\_app\_icon** = (ImageView) convertView.findViewById(R.id.iv\_app\_icon);  
 mViewHolder.**tx\_app\_name** = (TextView) convertView.findViewById(R.id.tv\_app\_name);  
 convertView.setTag(mViewHolder);  
 } **else** {  
 mViewHolder = (ViewHolder) convertView.getTag();  
 }  
 mViewHolder.**iv\_app\_icon**.setImageDrawable(myAppInfo.getImage());  
 mViewHolder.**tx\_app\_name**.setText(myAppInfo.getAppName());  
 **return** convertView;  
 }  
  
 **class** ViewHolder {  
  
 ImageView **iv\_app\_icon**;  
 TextView **tx\_app\_name**;  
 }  
 }  
  
}

# 黑盒测试

# 实验目的

 黑盒测试是一种常用的软件测试方法，在知道输入输出之间的关系或是程序功能的情况下来测试程序功能的正确性。

# 实验内容

1. 被测试的功能和白盒一样
2. 测试环境一样

## 三、测试方法

* 1. 等价划分和测试用例设计
     1. 等价划分：
        1. 输入条件：所以的app
        2. 有效等价类：app名字和图标
        3. 无效等价类：输入app名字不存在，图标不存在。
     2. 测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | App名字和图标 | 预期输出 | 有效/无效等价类 |
| 1 | 抖音 IMG_256 | 成功获取信息 | 有效 |
| 2 | 搜狗浏览器 IMG_256 | 成功获取信息 | 有效 |
| 3 | 王者荣耀 | 无法成功获取信息 | 无效 |
| 4 | 微信 IMG_256 | 成功获取信息 | 有效 |
| 5 | 浏览器 | 无法成功获取信息 | 无效 |

图片：