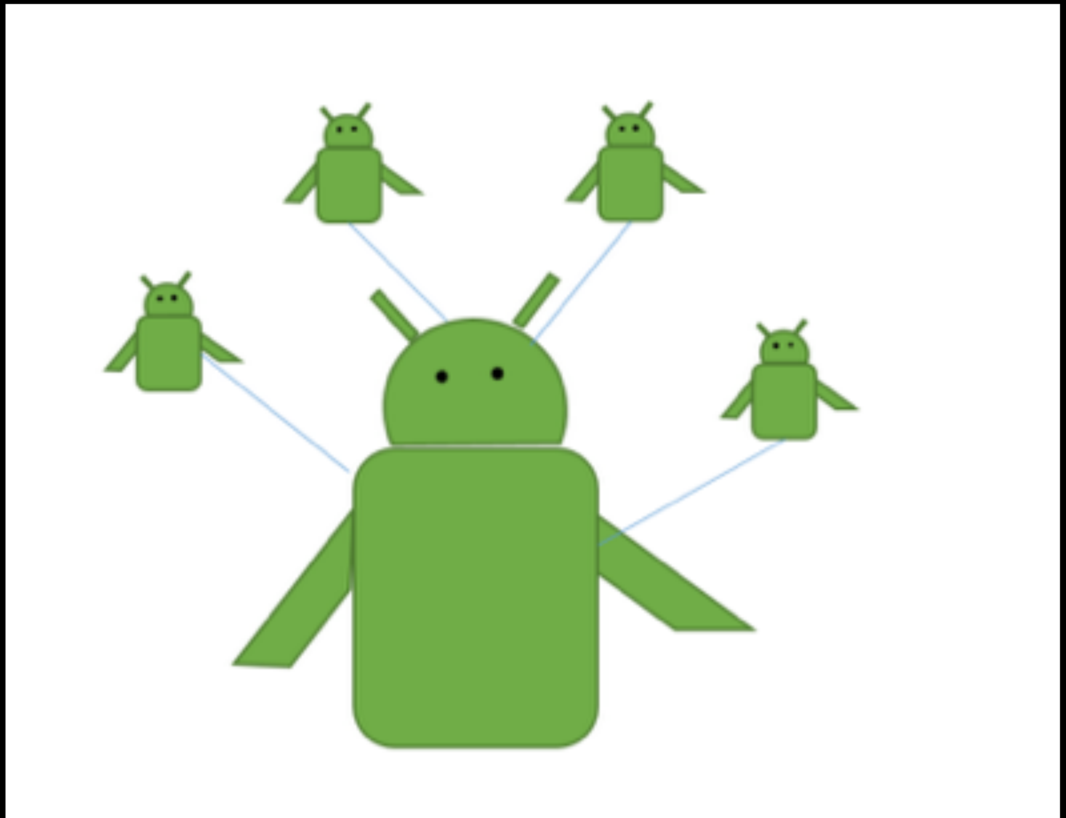
Android插件化技术(一)——插件化介绍及相关技术

什么是插件化

- 将应用程序分成独立的部分，按需加载
- 优点：减小体积、动态升级、节省流量等等
- 缺点：难度大、适配难、部分特性无法支持等



准备知识

- 反射、静态代理、动态代理
- Android的几个相关的ClassLoader原理
- 四大组件的相关原理、包括启动、生命周期相关
- 资源加载、资源打包、资源冲突相关知识
- 其他
- 涉及到的代码都经过Nexus 5(dalvik，6.0)测试

反射、代理

- 反射可以在运行过程中获取类的信息、修改对象的字段，调用类或者对象的方法
- 代理-为其他对象提供一种代理来控制这个对象的行为

反射的例子

```java
private static void reflect(Person person) {
    try {
        Field field = person.getClass().getDeclaredField("name");
        field.setAccessible(true);
        String name = (String) field.get(person);
        System.err.println(name);
        Method method = person.getClass().getDeclaredMethod("doSomething");
        method.invoke(person);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

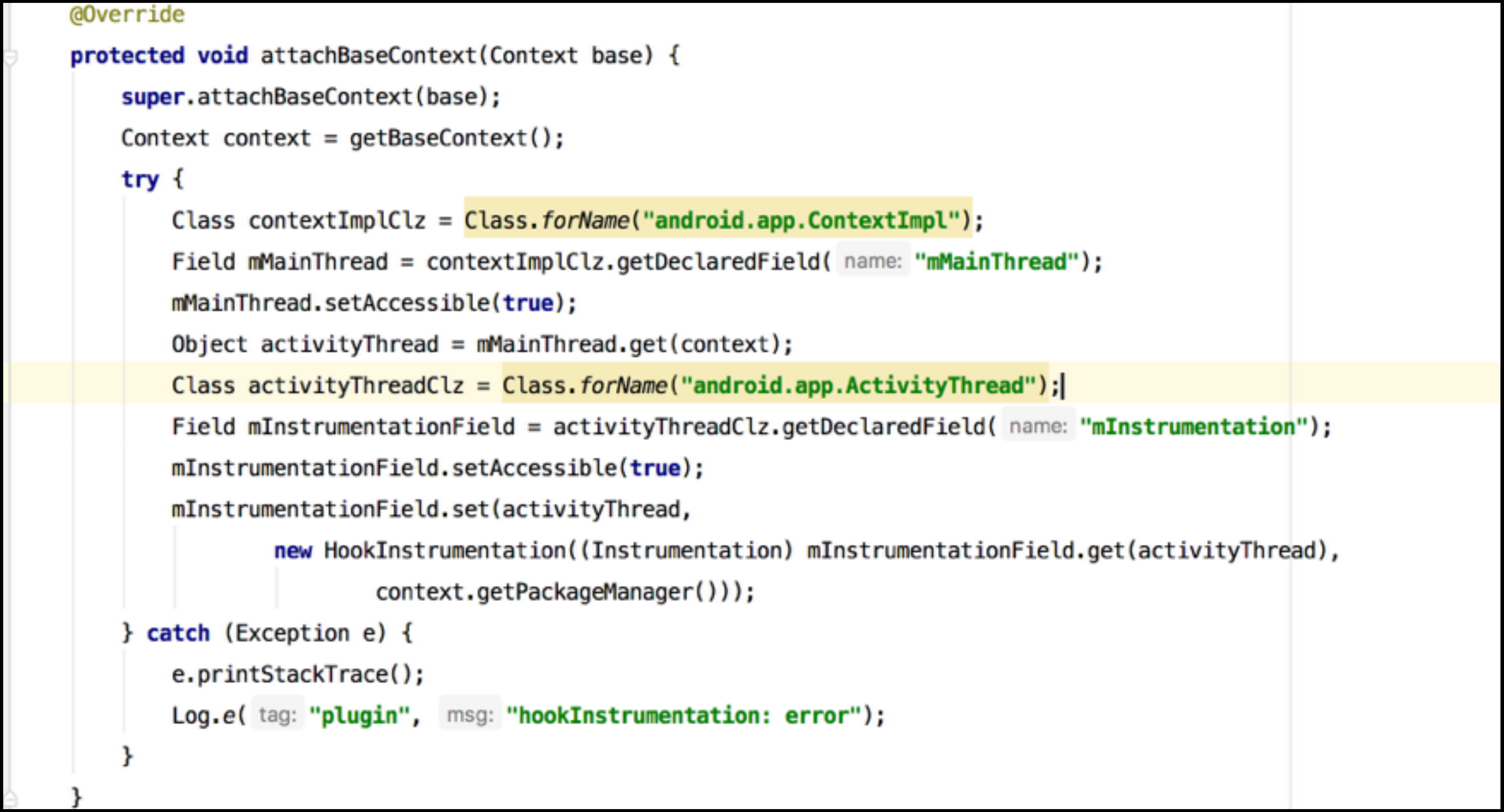
动态代理的例子

```
    Person person = new Person("guolei", 24);
//        reflect();
        Dothing dothing = (Dothing) Proxy.newProxyInstance(person.getClass().getClassLoader(),
                person.getClass().getInterfaces(), new CustomHandler(person));
        dothing.doSomething();
public class CustomHandler implements InvocationHandler {

    private Object target;

    CustomHandler(Object target) {
        this.target = target;
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        System.err.println("被代理拦截掉了");
        return method.invoke(target, args);
    }
}
```

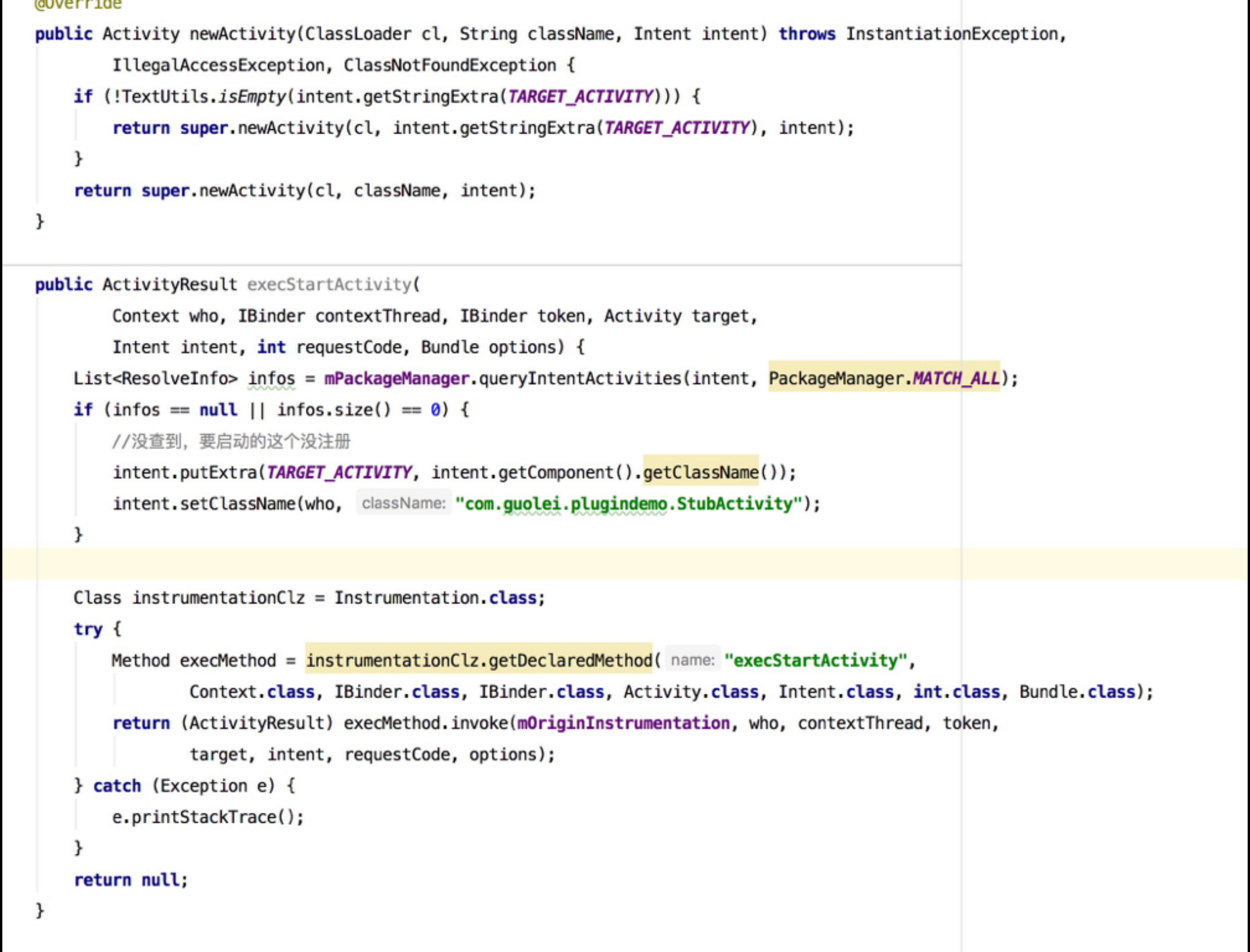
如何启动未注册的Activity

- 未注册指的是当前要启动的Activity未注册，想要注册一个没在Manifest文件中注册的Activity是不可能的事情
- 我们需要明白Activity是如何启动的 http://gityuan.com/2016/03/12/start-activity/
- 根据启动过程的源码分析，我们知道要解决的问题是如何绕后PMS检查，以及替换成真正要启动的组件。

重写Instrumentation的newActivity,execStartActivity方法,然后替换掉系统默认的。



```
@Override
protected void attachBaseContext(Context base) {
    super.attachBaseContext(base);
    Context context = getBaseContext();
    try {
        Class contextImplClz = Class.forName("android.app.ContextImpl");
        Field mMainThread = contextImplClz.getDeclaredField( name: "mMainThread");
        mMainThread.setAccessible(true);
        Object activityThread = mMainThread.get(context);
        Class activityThreadClz = Class.forName("android.app.ActivityThread");
        Field mInstrumentationField = activityThreadClz.getDeclaredField( name: "mInstrumentation");
        mInstrumentationField.setAccessible(true);
        mInstrumentationField.set(activityThread,
                new HookInstrumentation((Instrumentation) mInstrumentationField.get(activityThread),
                        context.getPackageManager()));
    } catch (Exception e) {
        e.printStackTrace();
        Log.e( tag: "plugin",  msg: "hookInstrumentation: error");
    }
}
```

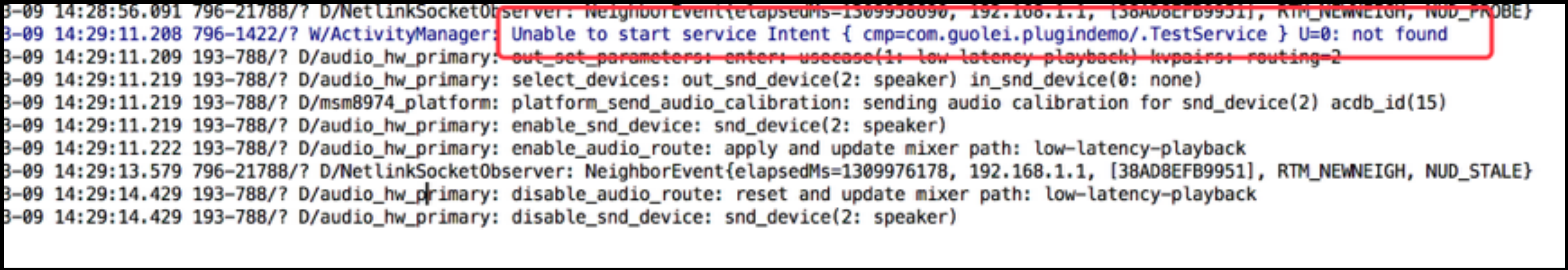


```
@Override
public Activity newActivity(ClassLoader cl, String className, Intent intent) throws InstantiationException,
        IllegalAccessException, ClassNotFoundException {
    if (!TextUtils.isEmpty(intent.getStringExtra(TARGET_ACTIVITY))) {
        return super.newActivity(cl, intent.getStringExtra(TARGET_ACTIVITY), intent);
    }
    return super.newActivity(cl, className, intent);
}

public ActivityResult execStartActivity(
        Context who, IBinder contextThread, IBinder token, Activity target,
        Intent intent, int requestCode, Bundle options) {
    List<ResolveInfo> infos = mPackageManager.queryIntentActivities(intent, PackageManager.MATCH_ALL);
    if (infos == null || infos.size() == 0) {
        //没查到，要启动的这个没注册
        intent.putExtra(TARGET_ACTIVITY, intent.getComponent().getClassName());
        intent.setClassName(who, className: "com.guolei.plugindemo.StubActivity");
    }

    Class instrumentationClz = Instrumentation.class;
    try {
        Method execMethod = instrumentationClz.getDeclaredMethod( name: "execStartActivity",
                Context.class, IBinder.class, IBinder.class, Activity.class, Intent.class, int.class, Bundle.class);
        return (ActivityResult) execMethod.invoke(mOriginInstrumentation, who, contextThread, token,
                target, intent, requestCode, options);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

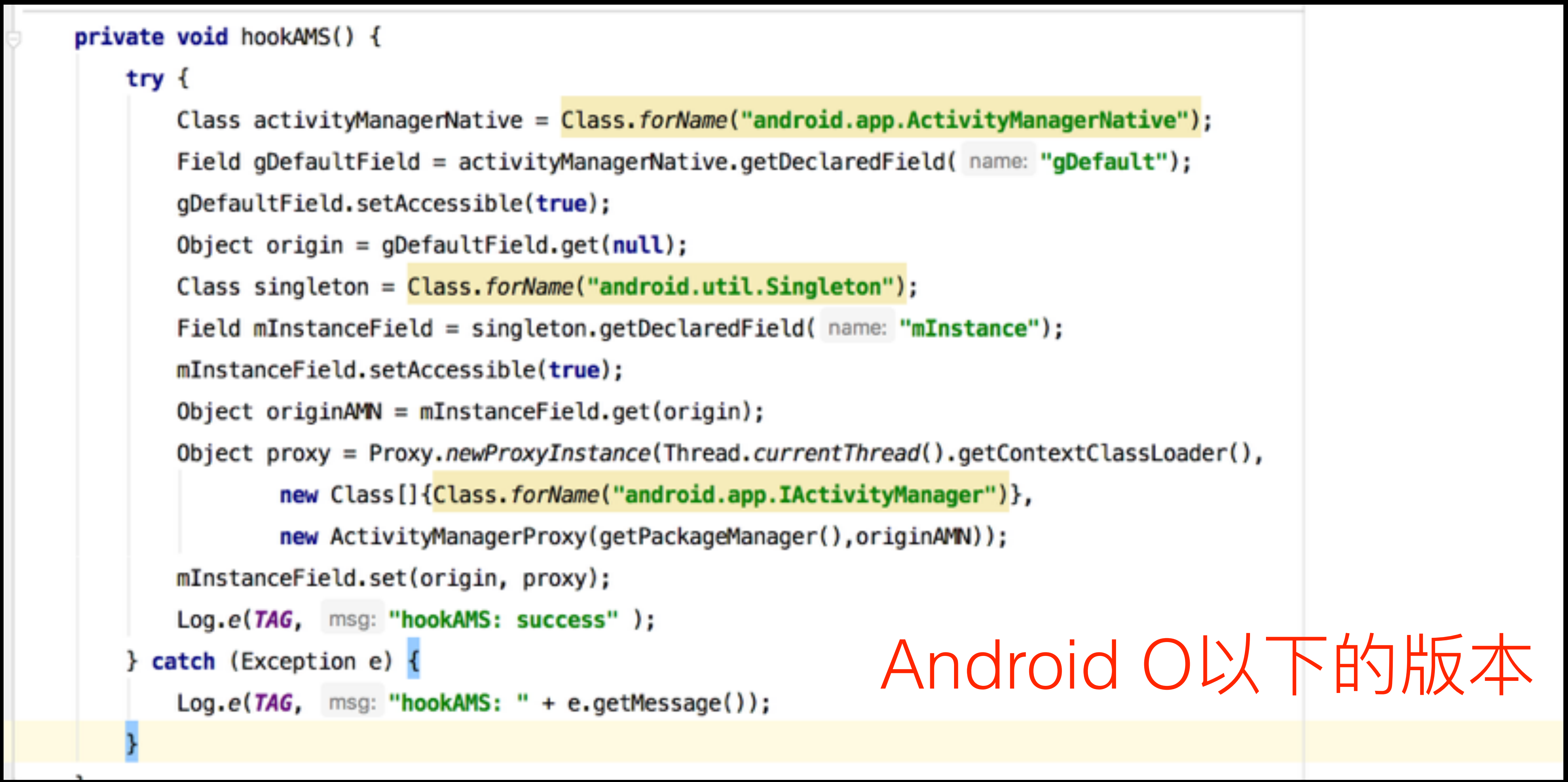
启动未注册的Service

- 不会像启动未注册的Activity一样，发生奔溃。只有如下日志。

- 和Activity的启动略有不同，但切入点是一样的，不同的是，不再通过Instrumentation了，而且某些声明周期方法，要我们自己去调用

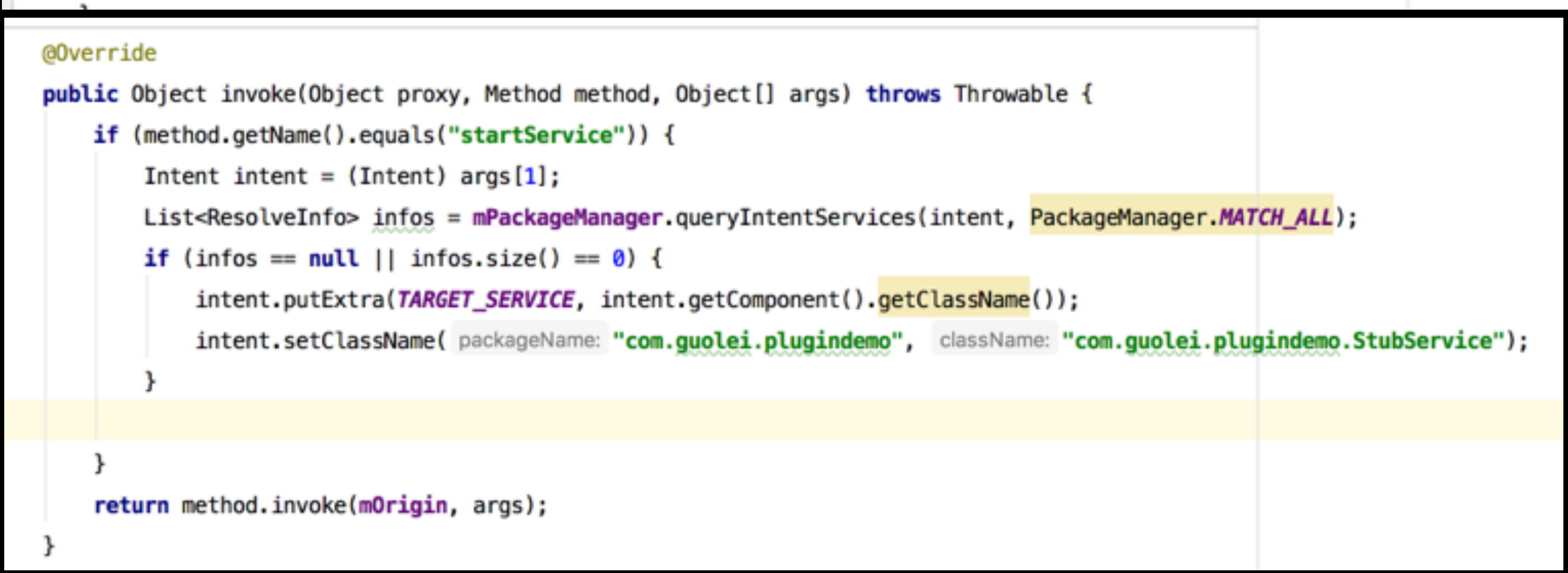


```
3-09 14:28:56.091 796-21788/? D/NetlinkSocketObserver: NeighborEvent{elapsedMs=1309958090, 192.168.1.1, [38AD8EFB9951], RTM_NEWNEIGH, NUD_PROBE}
3-09 14:29:11.208 796-1422/? W/ActivityManager: Unable to start service Intent { cmp=com.guolei.plugindemo/.TestService } U=0: not found
3-09 14:29:11.209 193-788/? D/audio_hw_primary: out_set_parameters: enter: usecase(1: low-latency-playback) kvpairs: routing=2
3-09 14:29:11.219 193-788/? D/audio_hw_primary: select_devices: out_snd_device(2: speaker) in_snd_device(0: none)
3-09 14:29:11.219 193-788/? D/msm8974_platform: platform_send_audio_calibration: sending audio calibration for snd_device(2) acdb_id(15)
3-09 14:29:11.219 193-788/? D/audio_hw_primary: enable_snd_device: snd_device(2: speaker)
3-09 14:29:11.222 193-788/? D/audio_hw_primary: enable_audio_route: apply and update mixer path: low-latency-playback
3-09 14:29:13.579 796-21788/? D/NetlinkSocketObserver: NeighborEvent{elapsedMs=1309976178, 192.168.1.1, [38AD8EFB9951], RTM_NEWNEIGH, NUD_STALE}
3-09 14:29:14.429 193-788/? D/audio_hw_primary: disable_audio_route: reset and update mixer path: low-latency-playback
3-09 14:29:14.429 193-788/? D/audio_hw_primary: disable_snd_device: snd_device(2: speaker)
```

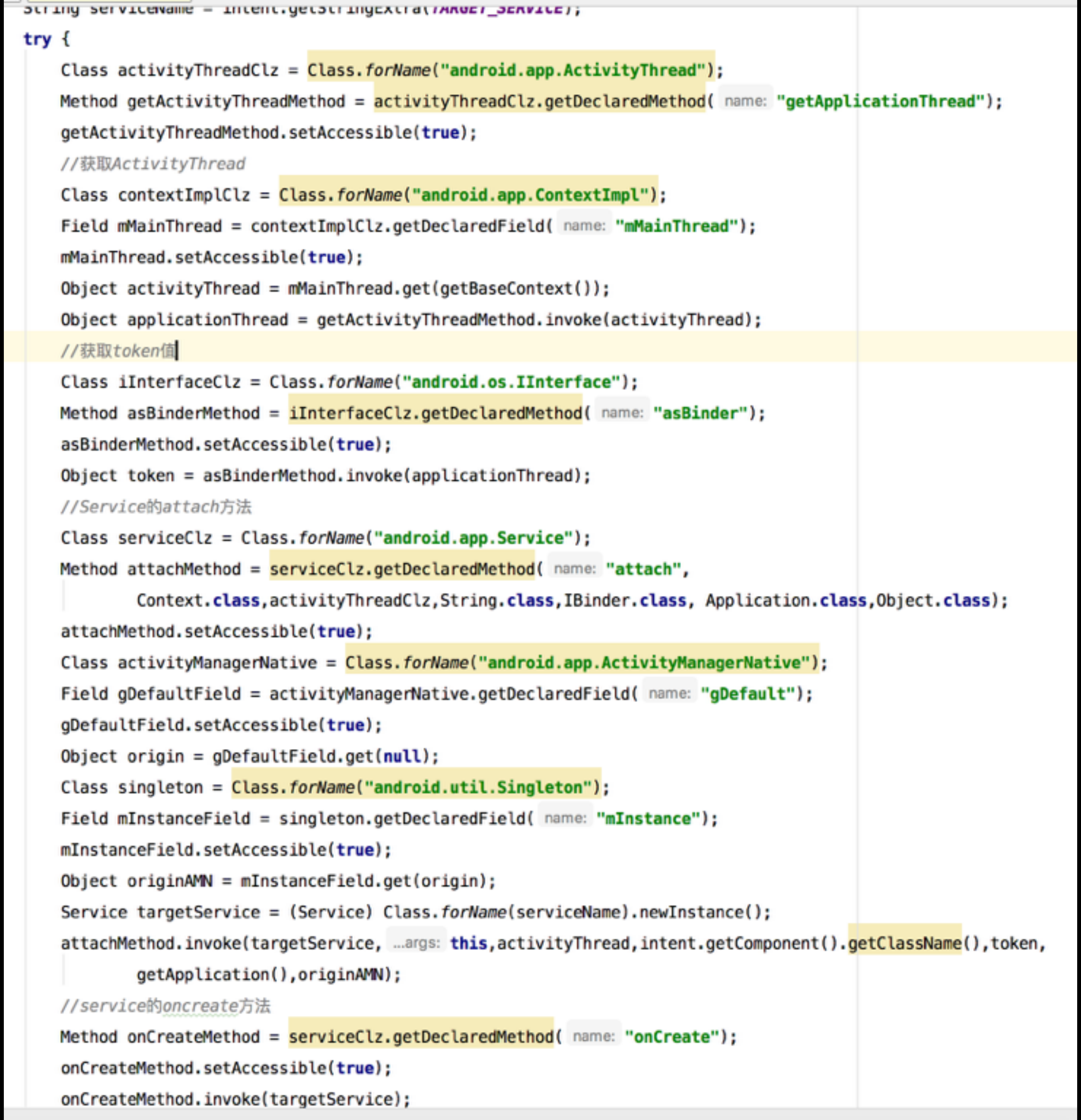


```java
private void hookAMS() {
    try {
        Class activityManagerNative = Class.forName("android.app.ActivityManagerNative");
        Field gDefaultField = activityManagerNative.getDeclaredField( name: "gDefault");
        gDefaultField.setAccessible(true);
        Object origin = gDefaultField.get(null);
        Class singleton = Class.forName("android.util.Singleton");
        Field mInstanceField = singleton.getDeclaredField( name: "mInstance");
        mInstanceField.setAccessible(true);
        Object originAMN = mInstanceField.get(origin);
        Object proxy = Proxy.newProxyInstance(Thread.currentThread().getContextClassLoader(),
                new Class[]{Class.forName("android.app.IActivityManager")},
                new ActivityManagerProxy(getPackageManager(),originAMN));
        mInstanceField.set(origin, proxy);
        Log.e(TAG, msg: "hookAMS: success" );
    } catch (Exception e) {
        Log.e(TAG, msg: "hookAMS: " + e.getMessage());
    }
```

Android O以下的版本



```java
@Override
public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
    if (method.getName().equals("startService")) {
        Intent intent = (Intent) args[1];
        List<ResolveInfo> infos = mPackageManager.queryIntentServices(intent, PackageManager.MATCH_ALL);
        if (infos == null || infos.size() == 0) {
            intent.putExtra(TARGET_SERVICE, intent.getComponent().getClassName());
            intent.setClassName( packageName: "com.guolei.plugindemo", className: "com.guolei.plugindemo.StubService");
        }

    }
    return method.invoke(mOrigin, args);
}
```

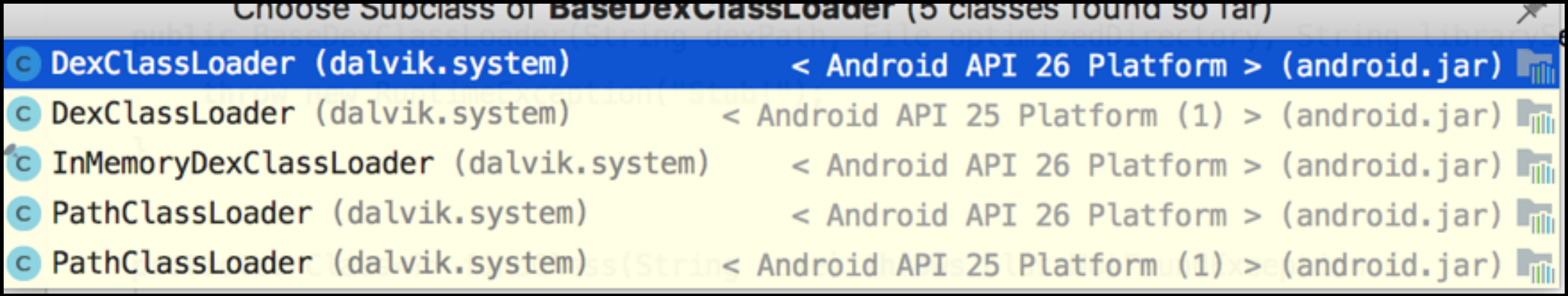


```
String serviceName = intent.getStringExtra(TARGET_SERVICE);
try {
    Class activityThreadClz = Class.forName("android.app.ActivityThread");
    Method getActivityThreadMethod = activityThreadClz.getDeclaredMethod( name: "getApplicationThread");
    getActivityThreadMethod.setAccessible(true);
    //获取ActivityThread
    Class contextImplClz = Class.forName("android.app.ContextImpl");
    Field mMainThread = contextImplClz.getDeclaredField( name: "mMainThread");
    mMainThread.setAccessible(true);
    Object activityThread = mMainThread.get(getBaseContext());
    Object applicationThread = getActivityThreadMethod.invoke(activityThread);
    //获取token值
    Class iInterfaceClz = Class.forName("android.os.IInterface");
    Method asBinderMethod = iInterfaceClz.getDeclaredMethod( name: "asBinder");
    asBinderMethod.setAccessible(true);
    Object token = asBinderMethod.invoke(applicationThread);
    //Service的attach方法
    Class serviceClz = Class.forName("android.app.Service");
    Method attachMethod = serviceClz.getDeclaredMethod( name: "attach",
            Context.class,activityThreadClz,String.class,IBinder.class, Application.class,Object.class);
    attachMethod.setAccessible(true);
    Class activityManagerNative = Class.forName("android.app.ActivityManagerNative");
    Field gDefaultField = activityManagerNative.getDeclaredField( name: "gDefault");
    gDefaultField.setAccessible(true);
    Object origin = gDefaultField.get(null);
    Class singleton = Class.forName("android.util.Singleton");
    Field mInstanceField = singleton.getDeclaredField( name: "mInstance");
    mInstanceField.setAccessible(true);
    Object originAMN = mInstanceField.get(origin);
    Service targetService = (Service) Class.forName(serviceName).newInstance();
    attachMethod.invoke(targetService, ...args: this,activityThread,intent.getComponent().getClassName(),token,
            getApplication(),originAMN);
    //service的oncreate方法
    Method onCreateMethod = serviceClz.getDeclaredMethod( name: "onCreate");
    onCreateMethod.setAccessible(true);
    onCreateMethod.invoke(targetService);
```

先跳过BroadcastReceiver
和ContentProvider,这个后面再说

如何加载一个外部的jar包

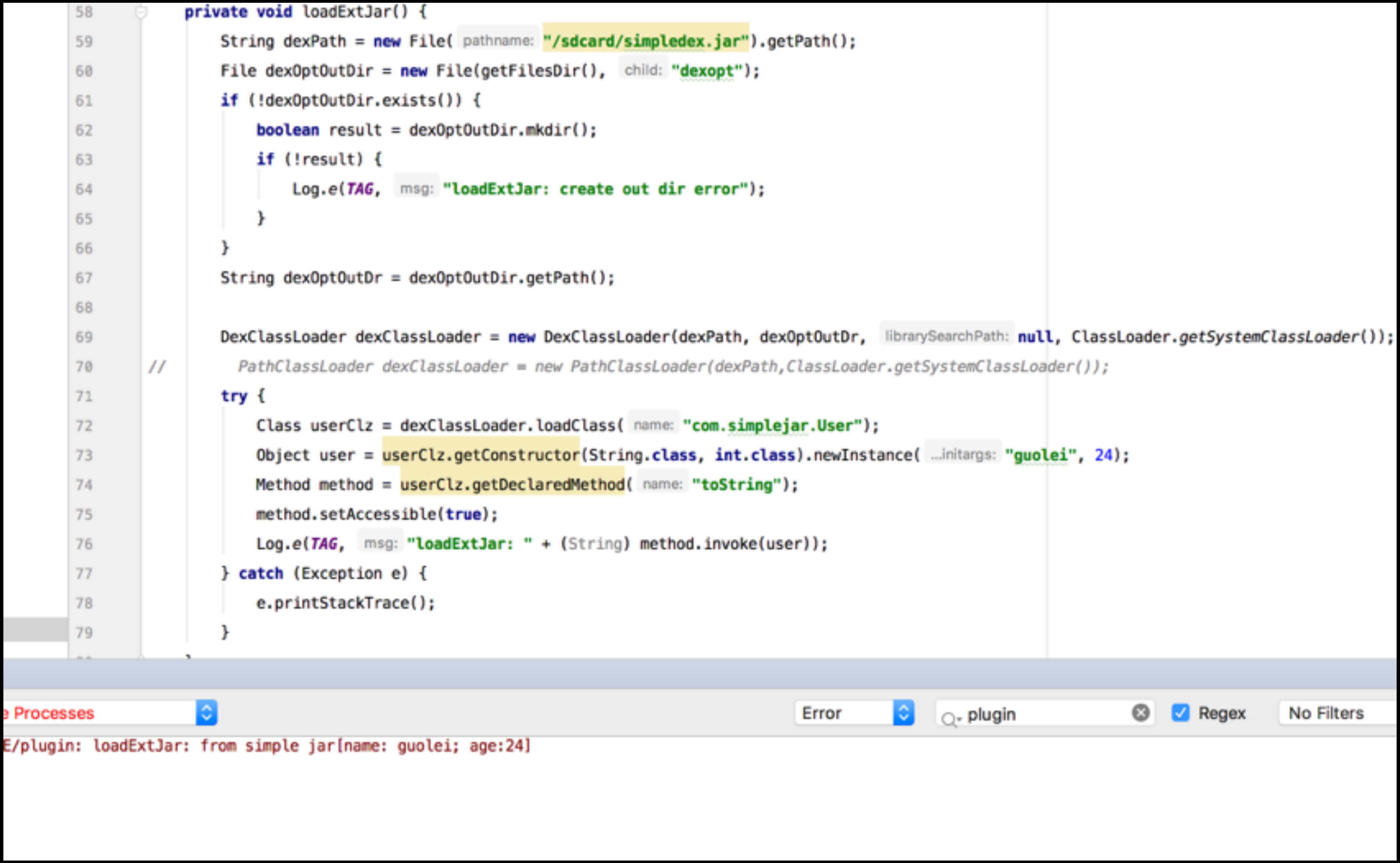
- BaseDexClassLoader
- DexClassLoader，加载jar、apk等
- PathClassLoader，如果加载apk，必须是已安装的(dalvik虚拟机，网上文章错误，art虚拟机是可以的)
- InMemoryDexClassLoader，



Choose Subclass of BaseDexClassLoader (5 classes found so far)

```
DexClassLoader (dalvik.system)                < Android API 26 Platform > (android.jar)
DexClassLoader (dalvik.system)            < Android API 25 Platform (1) > (android.jar)
InMemoryDexClassLoader (dalvik.system)        < Android API 26 Platform > (android.jar)
PathClassLoader (dalvik.system)               < Android API 26 Platform > (android.jar)
PathClassLoader (dalvik.system)           < Android API 25 Platform (1) > (android.jar)
```

先把生成的jar包用dx工具转一下dx —dex —outputh=xxx xxx，然后push到sdcard里



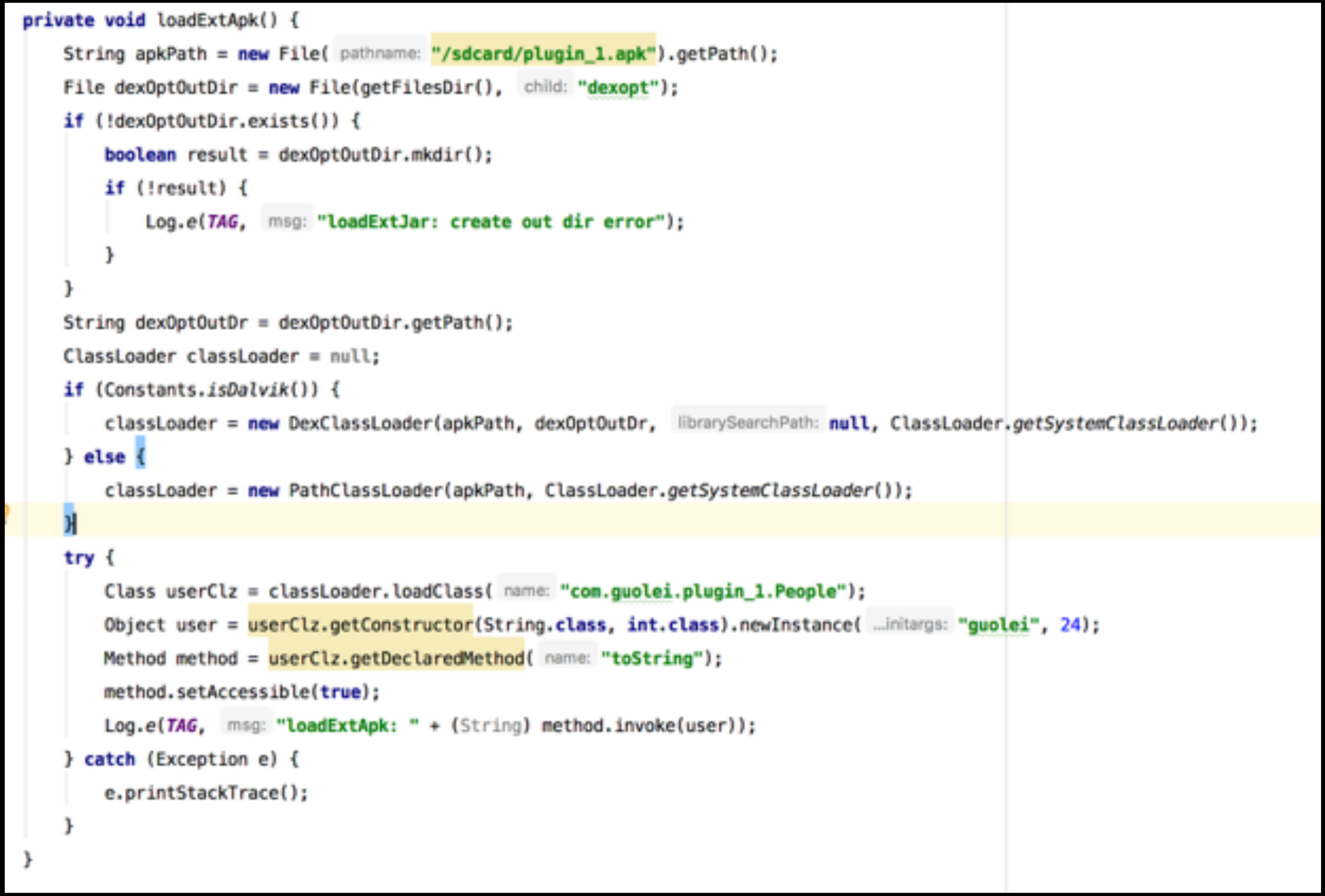
```java
private void loadExtJar() {
    String dexPath = new File( pathname: "/sdcard/simpledex.jar").getPath();
    File dexOptOutDir = new File(getFilesDir(),  child: "dexopt");
    if (!dexOptOutDir.exists()) {
        boolean result = dexOptOutDir.mkdir();
        if (!result) {
            Log.e(TAG,  msg: "loadExtJar: create out dir error");
        }
    }
    String dexOptOutDr = dexOptOutDir.getPath();

    DexClassLoader dexClassLoader = new DexClassLoader(dexPath, dexOptOutDr,  librarySearchPath: null, ClassLoader.getSystemClassLoader());
//      PathClassLoader dexClassLoader = new PathClassLoader(dexPath,ClassLoader.getSystemClassLoader());
    try {
        Class userClz = dexClassLoader.loadClass( name: "com.simplejar.User");
        Object user = userClz.getConstructor(String.class, int.class).newInstance( ...initargs: "guolei", 24);
        Method method = userClz.getDeclaredMethod( name: "toString");
        method.setAccessible(true);
        Log.e(TAG,  msg: "loadExtJar: " + (String) method.invoke(user));
    } catch (Exception e) {
        e.printStackTrace();
    }
```

e Processes　　Error　　plugin　　✓ Regex　　No Filters

```
E/plugin: loadExtJar: from simple jar[name: guolei; age:24]
```

加载外部的apk文件

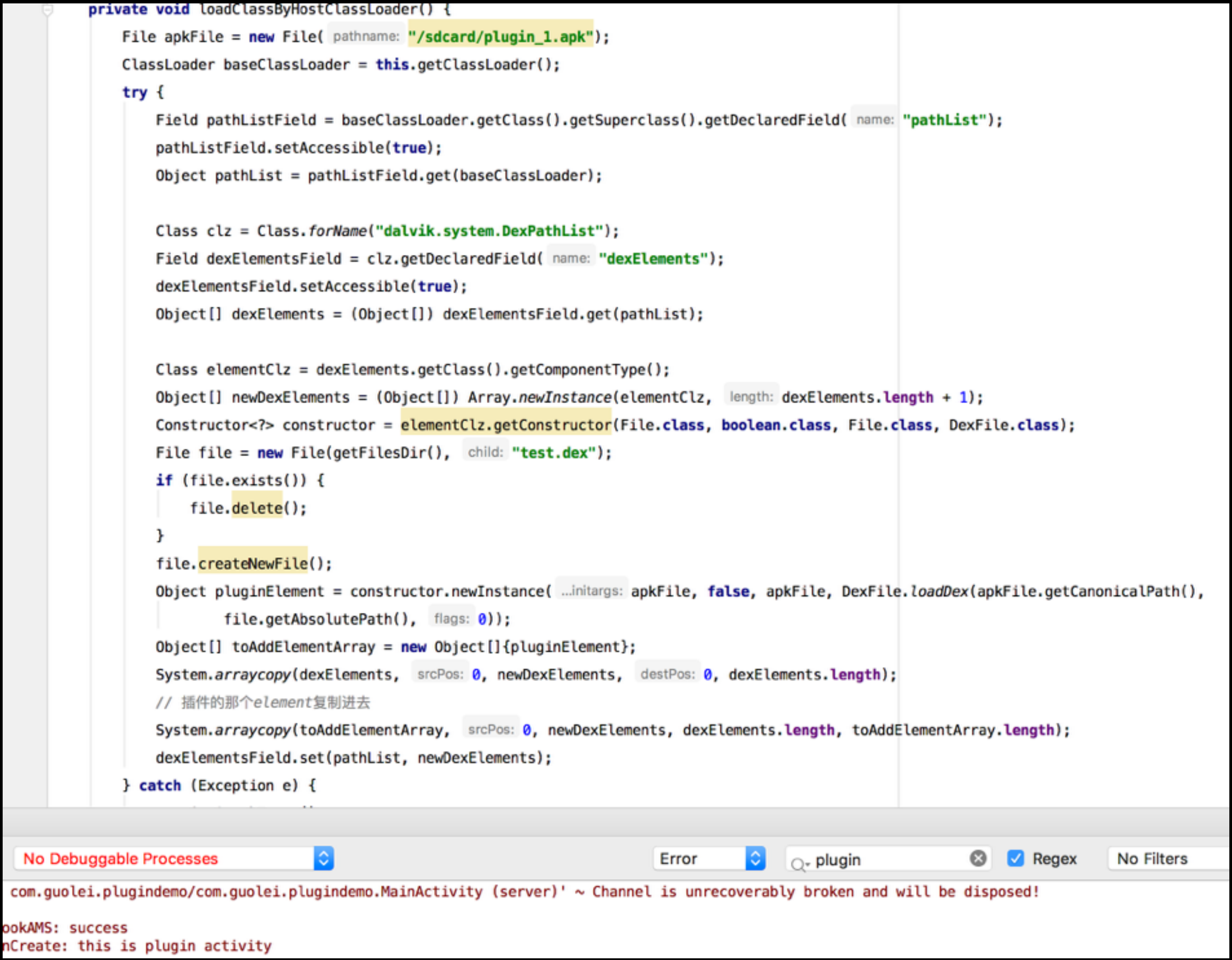
- 和加载jar基本一样，主要注意PathClassLoader在Dalvik和Art虚拟机上的区别



```java
private void loadExtApk() {
    String apkPath = new File( pathname: "/sdcard/plugin_1.apk").getPath();
    File dexOptOutDir = new File(getFilesDir(),  child: "dexopt");
    if (!dexOptOutDir.exists()) {
        boolean result = dexOptOutDir.mkdir();
        if (!result) {
            Log.e(TAG,  msg: "loadExtJar: create out dir error");
        }
    }
    String dexOptOutDr = dexOptOutDir.getPath();
    ClassLoader classLoader = null;
    if (Constants.isDalvik()) {
        classLoader = new DexClassLoader(apkPath, dexOptOutDr,  librarySearchPath: null, ClassLoader.getSystemClassLoader());
    } else {
        classLoader = new PathClassLoader(apkPath, ClassLoader.getSystemClassLoader());
    }
    try {
        Class userClz = classLoader.loadClass( name: "com.guolei.plugin_1.People");
        Object user = userClz.getConstructor(String.class, int.class).newInstance( ...initargs: "guolei", 24);
        Method method = userClz.getDeclaredMethod( name: "toString");
        method.setAccessible(true);
        Log.e(TAG,  msg: "loadExtApk: " + (String) method.invoke(user));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

如何启动外部apk的Activity

- 我们要考虑到构造Activity的时候ClassLoader的问题，前面的方法构造的时候是另外的ClassLoader，我们也可以使用宿主apk的ClassLoader去加载。
- 方法类似MultiDex



```
    private void loadClassByHostClassLoader() {
        File apkFile = new File( pathname: "/sdcard/plugin_1.apk");
        ClassLoader baseClassLoader = this.getClassLoader();
        try {
            Field pathListField = baseClassLoader.getClass().getSuperclass().getDeclaredField( name: "pathList");
            pathListField.setAccessible(true);
            Object pathList = pathListField.get(baseClassLoader);

            Class clz = Class.forName("dalvik.system.DexPathList");
            Field dexElementsField = clz.getDeclaredField( name: "dexElements");
            dexElementsField.setAccessible(true);
            Object[] dexElements = (Object[]) dexElementsField.get(pathList);

            Class elementClz = dexElements.getClass().getComponentType();
            Object[] newDexElements = (Object[]) Array.newInstance(elementClz,  length: dexElements.length + 1);
            Constructor<?> constructor = elementClz.getConstructor(File.class, boolean.class, File.class, DexFile.class);
            File file = new File(getFilesDir(),  child: "test.dex");
            if (file.exists()) {
                file.delete();
            }
            file.createNewFile();
            Object pluginElement = constructor.newInstance( ...initargs: apkFile, false, apkFile, DexFile.loadDex(apkFile.getCanonicalPath(),
                    file.getAbsolutePath(),  flags: 0));
            Object[] toAddElementArray = new Object[]{pluginElement};
            System.arraycopy(dexElements,  srcPos: 0, newDexElements,  destPos: 0, dexElements.length);
            // 插件的那个element复制进去
            System.arraycopy(toAddElementArray,  srcPos: 0, newDexElements, dexElements.length, toAddElementArray.length);
            dexElementsField.set(pathList, newDexElements);
        } catch (Exception e) {
```

No Debuggable Processes | Error | plugin | Regex ✓ | No Filters

```
com.guolei.plugindemo/com.guolei.plugindemo.MainActivity (server)' ~ Channel is unrecoverably broken and will be disposed!

ookAMS: success
nCreate: this is plugin activity
```

资源ID冲突的问题

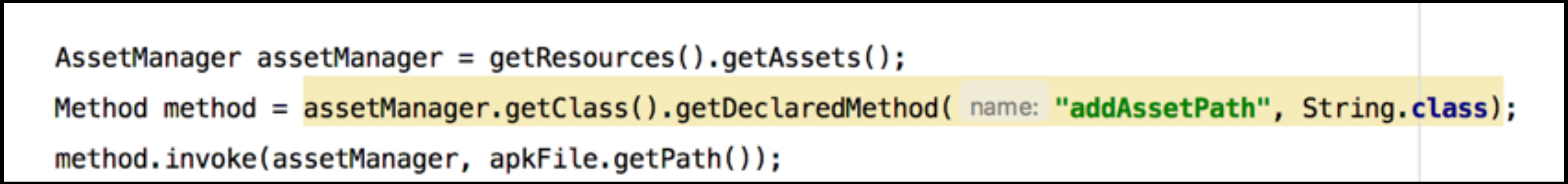
- 上面的情况，虽然我们能启动未安装Apk中的Activity，但是，会发现，启动的Activity的布局文件不对。这是因为资源id的问题

- 构建插件自己的Resource

- 共用AssetManager，将插件apk添加进来，1.修改aapt 2. 修改arsc和R文件

修改arsc的实现

- 为了简单，借助现成的修改arsc方案。我们只需要hook 宿主的AssetManager并 调用addAssetPath方法加入一个资源即可。



```
AssetManager assetManager = getResources().getAssets();
Method method = assetManager.getClass().getDeclaredMethod( name: "addAssetPath", String.class);
method.invoke(assetManager, apkFile.getPath());
```