

Jeb plugin/script report of PoRE

Student ID 20307130350

Name 陈丹纯

- Task
- Which part of Jeb do you plan to improve?

Task1 Rename with DeGuard

[DeGuard](#) offers a statistical deobfuscation service. Upload your apk, and it will give you a **mapping.txt** to reveal possible real names for classes and methods. You should write a plugin/script to load this file and rename those classes and methods accordingly.

- How does your plugin work?

1. 使用方法：名字是 `Jeb2AutoRenamePlugin`，用的是 ppt 中提供的生成 jar 的文档中所使用的方法。此外，测试时注意将 `mapping.txt` 文件路径改一下。为了防止总是打开 `jeb` 和 `GlobalLog` 打印太占内存的问题，主要是用输出文件流将打印的信息输出到其他的文件上了。
2. 重命名整体思路：通过文件流读取 `mapping.txt` 文件，一行一行读取，根据读取内容划分为类、字段、方法，将 java 语言下的包名/全类名/字段名/和方法名转换成 smali 格式，通过 `IDexUnit.getPackage(String)`，`IDexUnit.getClass(String)`，`IDexUnit.getMethod(String)`，`IDexUnit.getField(String)` 得到对应的包/类/字段/方法，再 rename 即可。改名可以用 `IDexClass`，`IDexPackage`，`IDexField` 和 `IDexMethod` 下的 `setName()`，也可用 `executeAction` 来改名。

```

6 usages
private void rename_Allkinds(IDexUnit unit, long itemId, String address, String sourceStr) {
    GlobalLog.getLogger().info("s: \n"+sourceStr+"\n");
    ActionContext actionContext = new ActionContext(unit, Actions.RENAME, itemId, address);
    ActionRenameData actionRenameData = new ActionRenameData();
    actionRenameData.setNewName(sourceStr);
    if(unit.prepareExecution(actionContext, actionRenameData)){
        try{
            boolean result = unit.executeAction(actionContext, actionRenameData);
            if(result){
                GlobalLog.getLogger().info("s: "rename to "+sourceStr+" success!");
            }
            else{
                GlobalLog.getLogger().info("s: "rename to "+sourceStr+" failed!");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

3. 根据 mapping.txt 可分成两大类改名，第一类是包名和类名，第二类是字段名和方法名。

```

com.sina.weibo.sdk.api.MultimlImageObject -> com.sina.weibo.android.configurations.Mult
com.airbnb.lottie.e$c -> com.airbnb.lottie.Observable$26
com.airbnb.lottie.d a -> this$0
com.ticktick.task.manager.HolidayRegistry -> com.ticktick.task.manager.HolidayRegistry
com.airbnb.lottie.e$a -> com.airbnb.lottie.Widget
java.lang.String a -> d
com.airbnb.lottie.e$b -> com.airbnb.lottie.k
java.lang.String a -> a
cn.jiguang.an.b -> cn.jiguang.curve.a
android.content.Context a -> c
java.lang.String b -> h
java.util.concurrent.LinkedBlockingQueue c -> d

```

因此每次循环都要保存一下 IDexClass 变量,便于后面修改字段名和方法名。

4. 修改包名:

包名的 smali 格式是: **Lcom/sina/weibo/sdk/api/**

在 mapping 中的格式是 java 格式 **com.sina.weibo.sdk.api**

只需要用 split 函数分割就可以得到各级包,然后一级一级匹配修改即可。

Api: IDexUnit.getPackage(smali 格式的包名);

Eg. IDexUnit.getPackage("Lcom/sina/weibo/sdk/api/");

```
String curPkgName="L";
for (int i = 0; i < Math.min(srcStr.length, renameStr.length)-1; i++) {
    if(srcStr[i]!=renameStr[i]){
        curPkgName=curPkgName+srcStr[i]+"/";
        String curPkgName="L"+strings[0].substring(0,strings[0].indexOf(srcStr[i+1])).replace(".", "/");
        bufferedWriter.write( str: "原包名: "+curPkgName+"\n");
        IDexPackage aPackage = iDexUnit.getPackage(pkgName.toString());
        IDexPackage aPackage = iDexUnit.getPackage(curPkgName.toString());
        if(aPackage!=null) {
            bufferedWriter.write( str: "原包名" + aPackage.getSignature() + "\n");
            rename_Allkinds(iDexUnit,aPackage.getItemId(),aPackage.getAddress(),renameStr[i]);
        }
    }
}
}
```

5. 修改类名：

类名的 smali 格式是：Lcom/sina/weibo/sdk/api/MultiImageObject;

用 IDexUnit.getClass 获取，再 rename；

```
if(!srcStr[srcStr.length-1].contains("$")) {
    bufferedWriter.write( str: "要修改的名字: " + renameStr[renameStr.length - 1] + "\n");
    GlobalLog.getLogger().info( s: "修改的名字: " + renameStr[renameStr.length - 1]);
    rename_Allkinds(iDexUnit, aClass.getItemId(), aClass.getAddress(), renameStr[renameStr.length - 1]);
    bufferedWriter.write( str: "改名成功: " + aClass.getSignature() + "\n");
}
else{
    IDexPackage aPackage = iDexUnit.getPackage(pkgName.toString());
    if(aPackage!=null) {
        String outerClazzName=aPackage.getSignature()+srcInner[0];
        bufferedWriter.write( str: "外部name: "+outerClazzName+"\n");
        IDexClass outerClazz = iDexUnit.getClass( s: outerClazzName+ ";");
        if(outerClazz!=null) {
            bufferedWriter.write( str: "外部类: "+outerClazz.getSignature()+"\n");
            rename_Allkinds(iDexUnit,outerClazz.getItemId(),outerClazz.getAddress(),renameInner[0]);
            bufferedWriter.write( str: "外部类修改成功\n");
        }
    }
}
```

6. 修改内部类名：

修改内部类，跟包名差不多，但是是根据“\$”分级。

```
String curInnerClassName=outerClazzName;
String renameInnerName=renameInner[0];
for (int i = 1; i < Math.min(renameInner.length,srcInner.length); i++) {
    String curInnerClass="L"+aPackage.getSignature()+srcStr[srcStr.length-1].substring(0,srcStr[srcStr.length-1].indexOf(srcStr[srcStr.length-1].indexOf(renameInnerName)+1));
    curInnerClassName=curInnerClassName+"$"+srcInner[i];
    IDexClass curInnerClazz = iDexUnit.getClass( s: curInnerClassName + ";");
    String renameInnerStr=renameStr[renameStr.length-1].substring(0,renameStr[renameStr.length-1].indexOf(renameInnerName)+1);
    renameInnerName=renameInnerName+"$"+renameInner[i];
    bufferedWriter.write( str: "\n=====修改内部类=====\\n"+renameInnerName+"\n");
    if(curInnerClazz!=null) {
        bufferedWriter.write( str: "原内部类名: " + curInnerClassName + "\n");
        bufferedWriter.write( str: "原内部类名全: " + curInnerClazz.getSignature() + "\n");
        rename_class(iDexUnit,curInnerClazz.getItemId(),curInnerClazz.getAddress(),renameInner[i]);
        rename_Allkinds(iDexUnit, curInnerClazz.getItemId(), curInnerClazz.getAddress(), renameInnerName);
    }
}
}
```

7. 修改字段名和方法名：

字段名的 smali 格式是：类->字段名： 字段类型

Lcn/jiguang/curve/b;->a:Landroid/content/Context;

方法名的 smali 格式是：

Lcom/airbnb/lottie/e\$bk;->a(Landroid/content/Context;Ljava/lang/String;)V
Lcom/airbnb/lottie/e\$bk;->aaaa(Landroid/content/Context;Ljava/lang/String
;)|||V

这两个在 java2smali 函数中进行格式转换。最后带入回去改名即可。

```
1 usage
public String java2smali(IDexClass iDexClass, String javaStr){
    String[] temp = javaStr.substring( beginIndex: 4).split( regex: "["] ); //前面都是4个空格开始的
    String smaliStr="";
    if(temp[1].indexOf("(")==-1) { //Field
        smaliStr = iDexClass.getSignature() + "->" + temp[1] + ":L" + temp[0].replace( target: ".", replacement: "/" ) + ";";
        //eg:Lcn/jiguang/curve/b;->a:Landroid/content/Context;
        System.out.println(smaliStr);
    }
    else{ //Method
        String returnTypeWithNoArray=temp[0];
        if(temp[0].indexOf("[")!=1){
            returnTypeWithNoArray=temp[0].substring(0,temp[0].indexOf("["]);
        }
        returnTypeWithNoArray=java2smaliTypeTable(returnTypeWithNoArray);
        int num=(temp[0].lastIndexOf( str: "[")-temp[0].indexOf("[])/2+1;
        if(temp[0].indexOf("[")==-1)
            num=0;
        temp[0]=String.join( delimiter: " ", Collections.nCopies(num, "[")+returnTypeWithNoArray; //返回类型
        System.out.println("返回类型"+temp[0]);
        String[] parameters=temp[1].substring(temp[1].indexOf("(")+1,temp[1].lastIndexOf( str: ")")).split( regex: "["] );
        StringBuffer paramsBuffer=new StringBuffer("");
        for (String param :parameters) {
            paramsBuffer.append(java2smaliTypeTable(param));
        }
        paramsBuffer.append("");
        smaliStr=iDexClass.getSignature() + "->" + temp[1].substring(0,temp[1].indexOf("("))+paramsBuffer.toString()+temp[0];
    }
}
```

```
if(srcSmaliSignature.indexOf("(")==-1) { //字段
    String srcSmaliName = srcSmaliSignature.substring(srcSmaliSignature.indexOf(">")+1,srcSmaliSignature.indexOf(":"));
    if(srcSmaliName!=strings[1]) {
        bufferedWriter.write( str: "原字段名: " + srcSmaliName + "\n");
        IDexField iDexField = iDexUnit.getField(srcSmaliSignature);
        if (iDexField != null) {
            bufferedWriter.write( str: "字段名: " + iDexField.getSignature() + "\n");
            GlobalLog.getLogger().info("字段名: " + iDexField.getSignature());
            String destStr=strings[1];
            bufferedWriter.write( str: "要改的字段名: " + destStr + "\n");
            rename_Allkinds(iDexUnit, iDexField.getItemId(), iDexField.getAddress(), destStr);
            boolean isSetName = iDexField.setName(destStr);
            GlobalLog.getLogger().info( str: "\n用setName改字段名! "+isSetName);
            bufferedWriter.write( str: "改字段名成功: " + iDexField.getSignature() + "\n");
            GlobalLog.getLogger().info("改字段名: " + iDexField.getSignature());
        }
    }
}
```

```

else if(srcSmaliSignature.indexOf("(")!=-1){ //method
    String srcSmaliName = srcSmaliSignature.substring(srcSmaliSignature.indexOf(">")+1,srcSmaliSignature.indexOf("("));
    if(srcSmaliName!=strings[1]) {
        IDexMethod iDexMethod = iDexUnit.getMethod(srcSmaliSignature);
        if (iDexMethod != null) {
            bufferedWriter.write( str: "方法名: " + iDexMethod.getSignature() + "\n");
            GlobalLog.getLogger().info("方法名: " + iDexMethod.getSignature());
            String destStr = srcSmaliSignature.substring(0,srcSmaliSignature.indexOf(">")+1) + strings[1] + srcSmaliSignature
            String destStr=strings[1];
            rename_Allkinds(iDexUnit, iDexMethod.getItemId(), iDexMethod.getAddress(), destStr);
            boolean isSetName = iDexMethod.setName(destStr);
            GlobalLog.getLogger().info( str: "\n用setName改方法名! "+isSetName);
            bufferedWriter.write( str: "改方法名成功: " + iDexMethod.getSignature() + "\n");
            GlobalLog.getLogger().info("改方法名: " + iDexMethod.getSignature());
        }
    }
}

```

8. 局限:

a) 尚无法改变静态成员的名字

尝试过 `override load` 函数也无法更改。因为静态成员在加载阶段 `clinit` 函数中进行连接等操作加载的，目前找不到响应的接口。

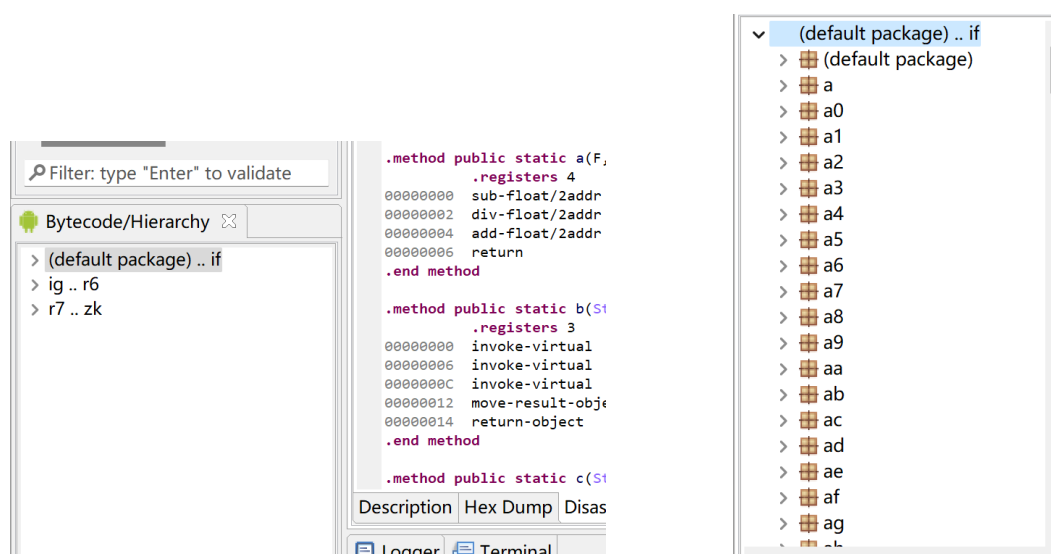
b) 对于已经同一个类下重复的字段名/方法名无法修改

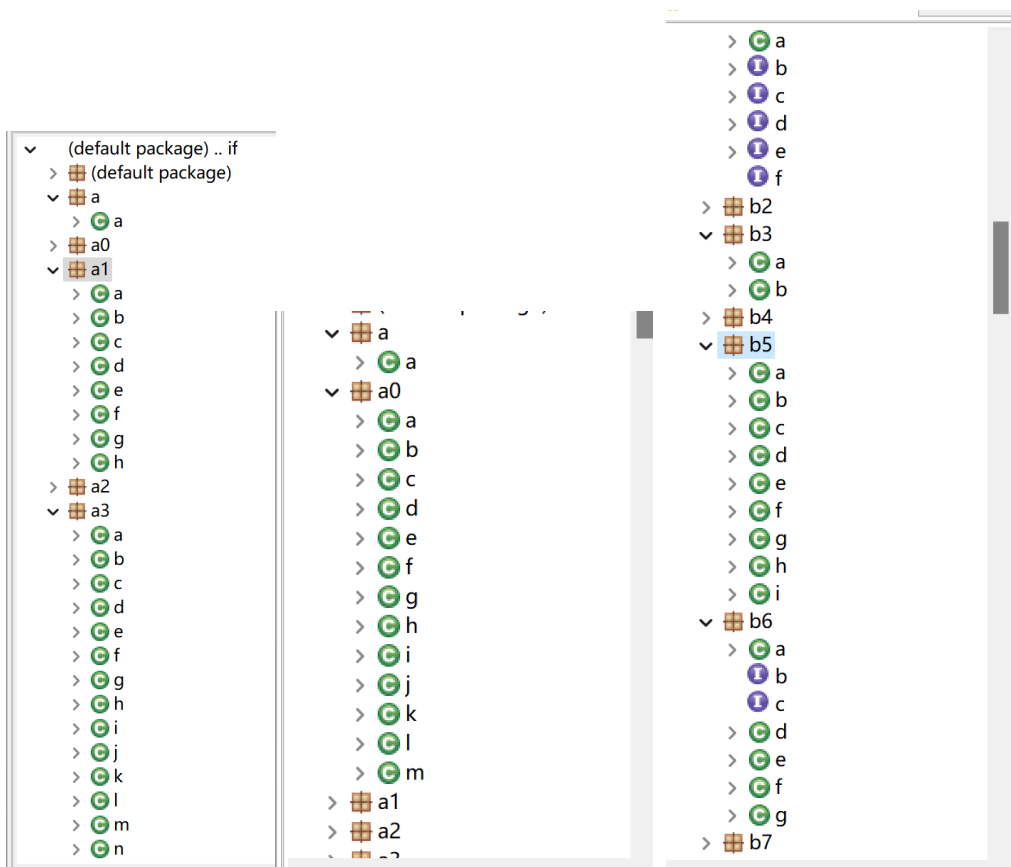
这个在于更改的顺序是读取 `mapping` 的顺序，更改的名字已经被后面的占有，但是后面的名字之后才会改，所以不成功。

● Screenshots of your plugin's execution result.

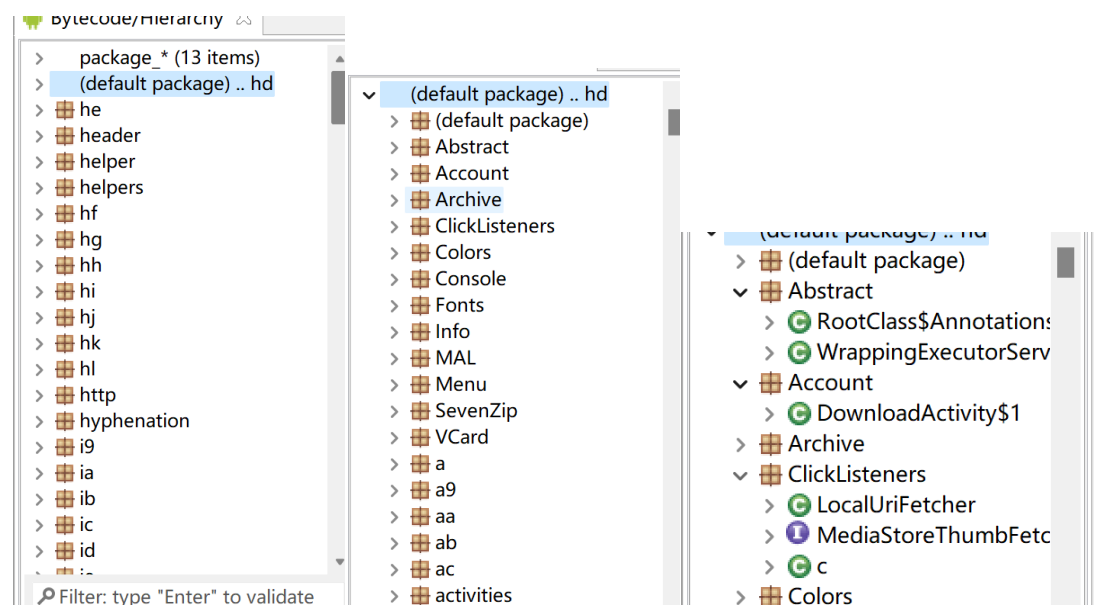
测试用的是滴答清单.apk 和盒马.apk;这里只截前者;

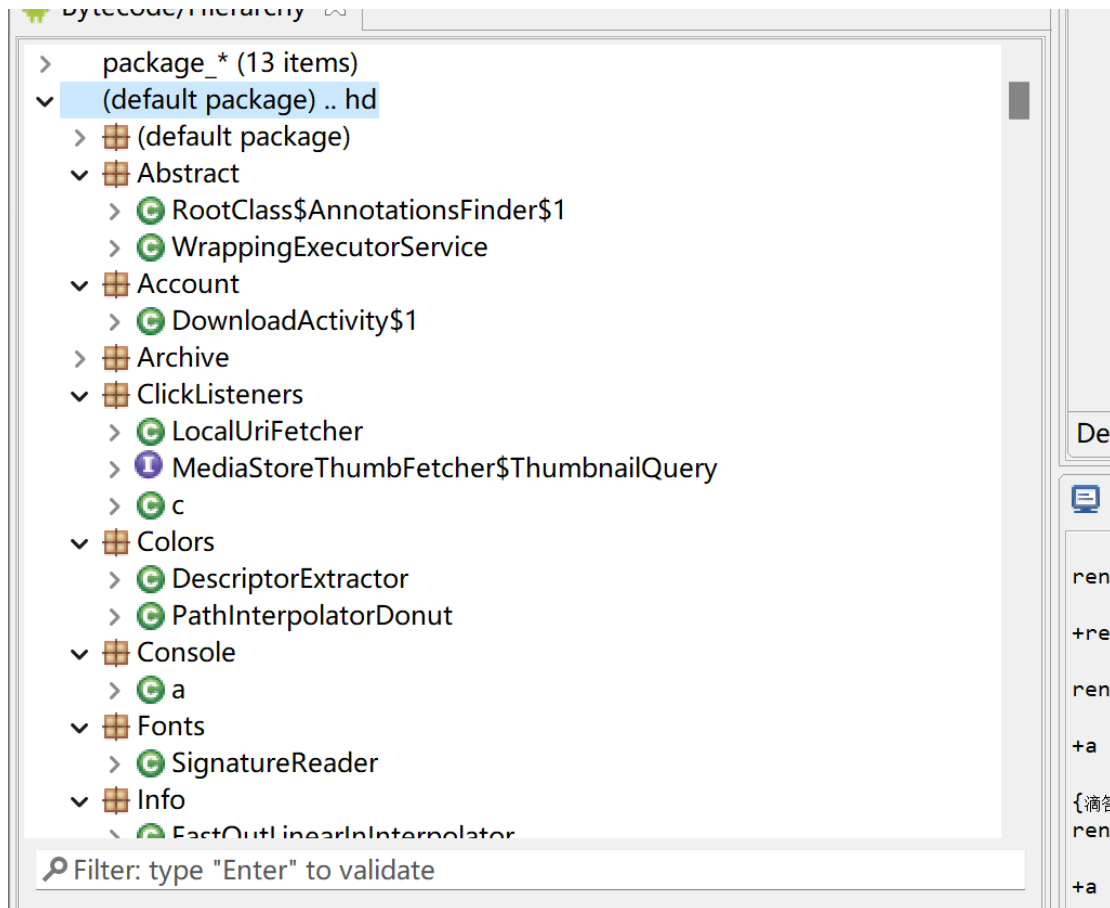
未修改前:





修改之后:



































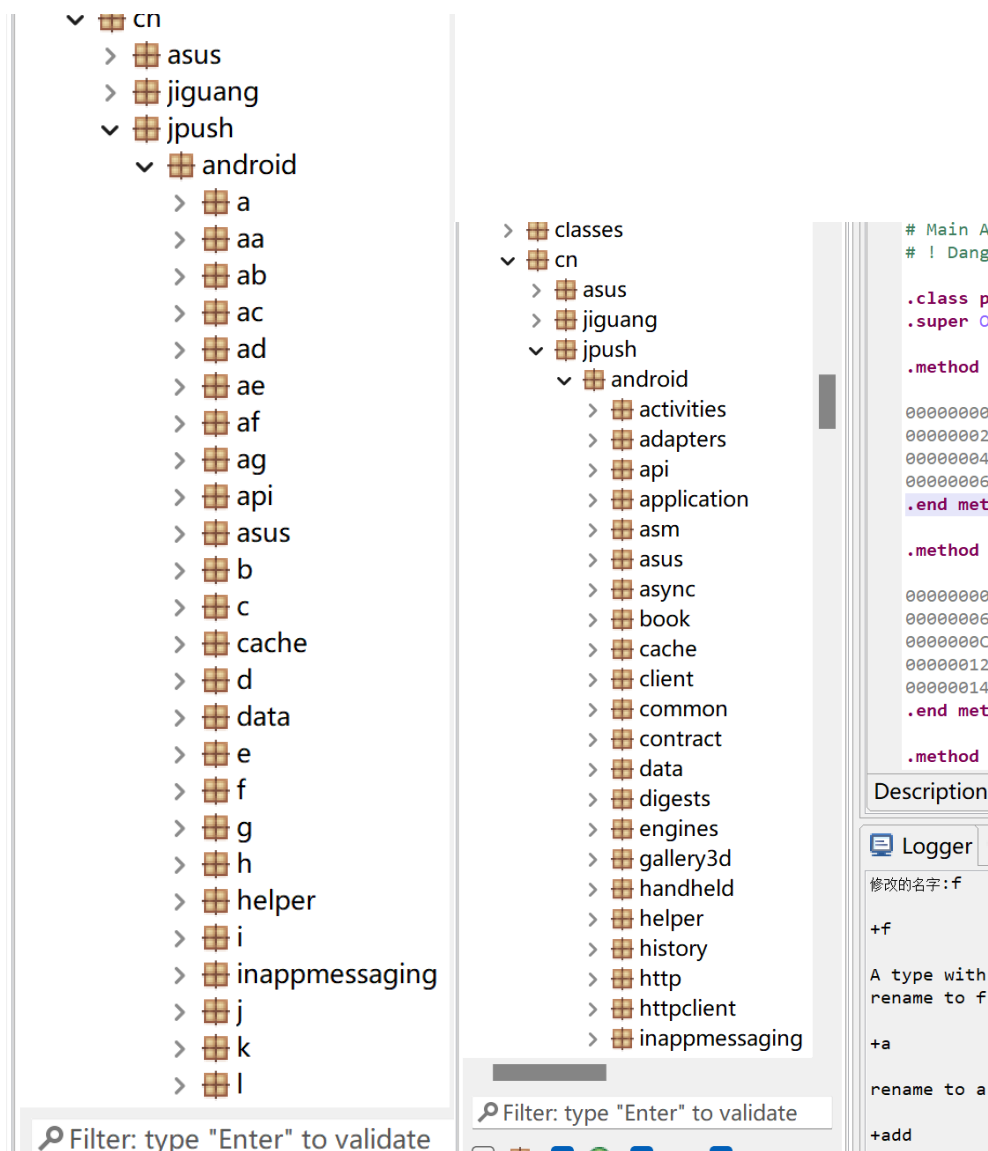
用 mapping 匹配看一下

```

... int h(android.content.Context,boolean)->c
com.huawei.wearengine.device.FoundListener->
com.huawei.wearengine.device.FoundListener
com.huawei.wearengine.p2p.MessageParcelExtra->
com.huawei.wearengine.labs.MessageParcelExtra
androidx.fragment.app.Fragment$SavedState$
androidx.fragment.package_11.Fragment$SavedState$
cn.jpish.android.x.c->cn.jpish.android.params.h
... float a->z
... float b->w

```

- >  customview
- >  drawerlayout
- ▼  fragment
 - ▼  package_11
 - >  ActivityCompat\$1
 - >  Attribute
 - >  BackStackState
 - >  ByteVector
 - >  ClassWriter
 - >  Comment
 - >  DialogFragment
 - >  FileTransfer\$5
 - ▼  Fragment
 - ▼ Fragment\$SavedState .. requireHost() : Object
 - ▼  Fragment\$SavedState
 - >  new Parcelable\$ClassLoaderCreator() {...}
 -  CREATOR : Parcelable\$Creator
 -  mState : Bundle
 -  Fragment\$Fragment\$SavedState {...}
 -  Fragment\$Fragment\$SavedState(Bundle)
 -  Fragment\$Fragment\$SavedState(Parcel, ClassLoader)
 -  describeContents() : int
 -  writeToParcel(Parcel, int) : void
 - >  new Runnable() {...}
 - >  Fragment\$d
 - >  Fragment\$e
 -  Fragment\$f
 -  ACTIVITY_CREATED : int
 -  ATTACHED : int
 -  CREATED : int
 -  ...



```
cn.jpusth.android.g.a->cn.jpusth.android.maps.ClassWriter<
... java.util.concurrent.atomic.AtomicBoolean a->this$0<
... void a(java.lang.String,java.lang.String)->c<
... boolean a()->connect<
... boolean b()->a<
... boolean c()->internalDataAvailable<
```

