**Lab 2. Android App Reverse II**

* **Task A**

(1) Your Answer

[If you choose to write an app to send the Intent, please submit the APK to the Git server and copy the Intent-sending code here.]

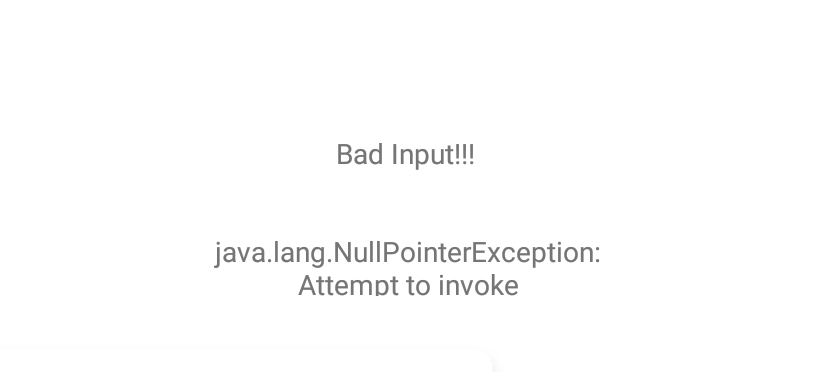
1. 第一类
   1. adb shell am start -n com.example.lab2/.MainActivity --ei type 0
      1. 0可以改为除了-1以外的其他任意数字
2. 第二类
   1. adb shell am start -n com.example.lab2/.MainActivity --ei type 1 --es id a
   2. adb shell am start -n com.example.lab2/.MainActivity --ei type 2 --es name a
      1. 以上的“a”都可以改成任意字符串
3. 第三类
   1. adb shell am start -n com.example.lab2/.MainActivity --ei type 1 --es password a
   2. adb shell am start -n com.example.lab2/.MainActivity --ei type 2 --es password a
   3. adb shell am start -n com.example.lab2/.MainActivity --ei type 2 --es birthday a
   4. adb shell am start -n com.example.lab2/.MainActivity --ei type 1 --es birthday a

以上的“a”都可以改成任意字符串

得到结果如下:

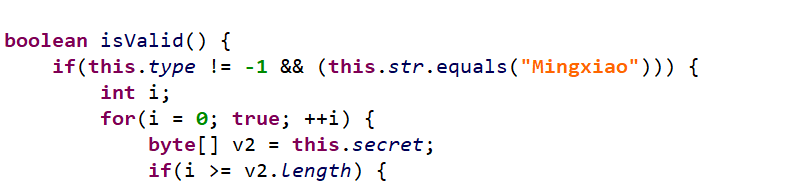
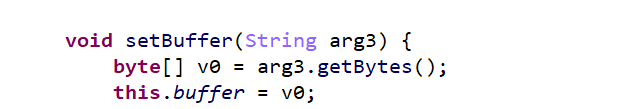


1. 第四类：
   1. adb shell am start -n com.example.lab2/.MainActivity --ei type 1 --es name a
   2. adb shell am start -n com.example.lab2/.MainActivity --ei type 2 --es id a
      1. 以上的“a”都可以改成任意字符串



(2) Writeup

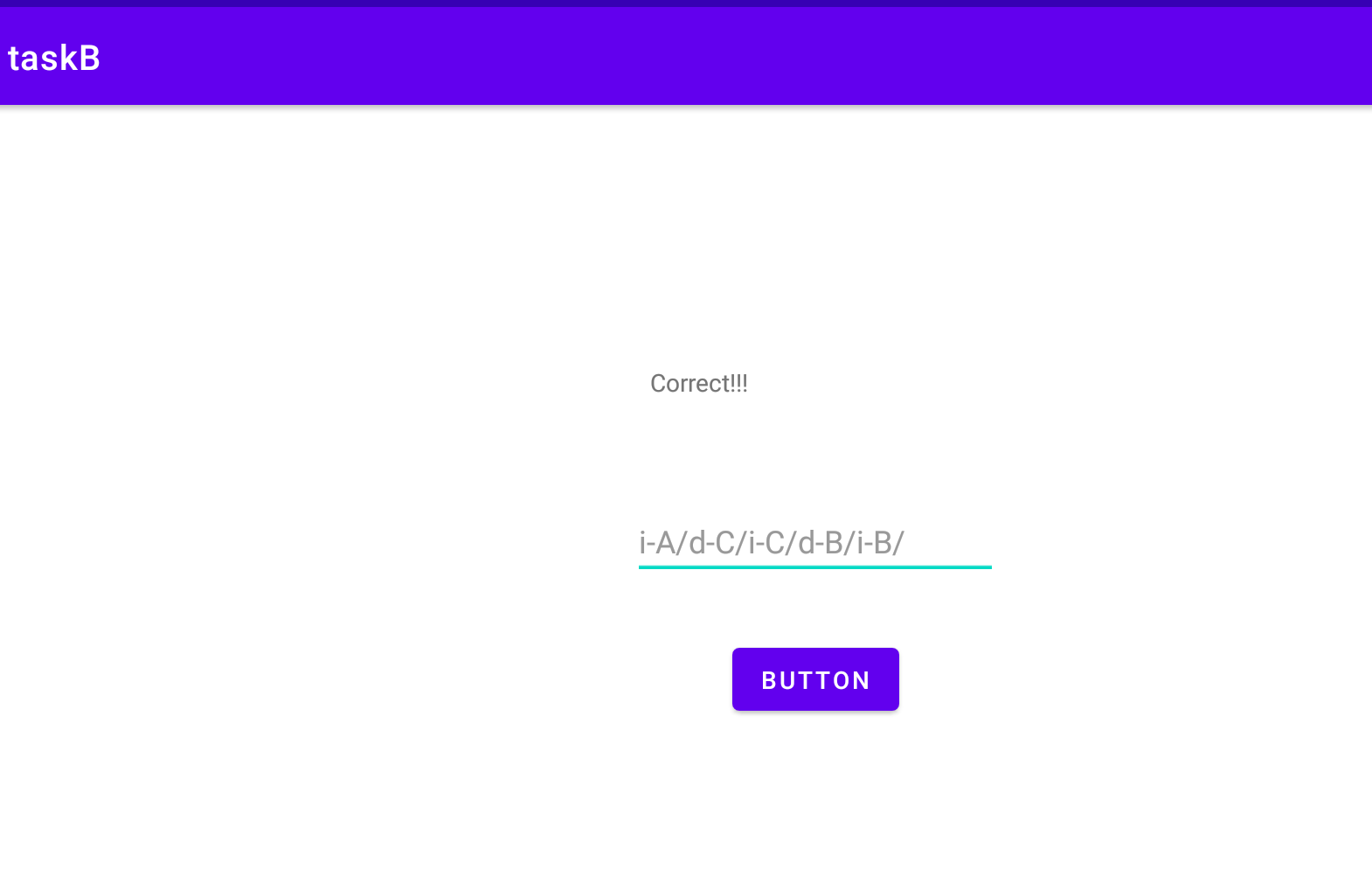
[Record how you solve this task here.]

1. 对于adb shell am start -n com.example.lab2/.MainActivity --ei type 0 （除了-1）类的：
   1. 
   2. 如果type！=-1，并且没有输入str，那么str==null，调用equals方法会报错。
2. 对于第二，三类：
   1. 
   2. 和第一类几乎同样的问题。不过第二类的问题是buffer为空；
   3. 第三类是直接忽略了第一个参数类型，参数列表错。
3. 对于第四类：
   1. Str没问题，但是buffer为空。
4. 综合上述的情况，基本都是因为参数列表错误。

* **Task B**

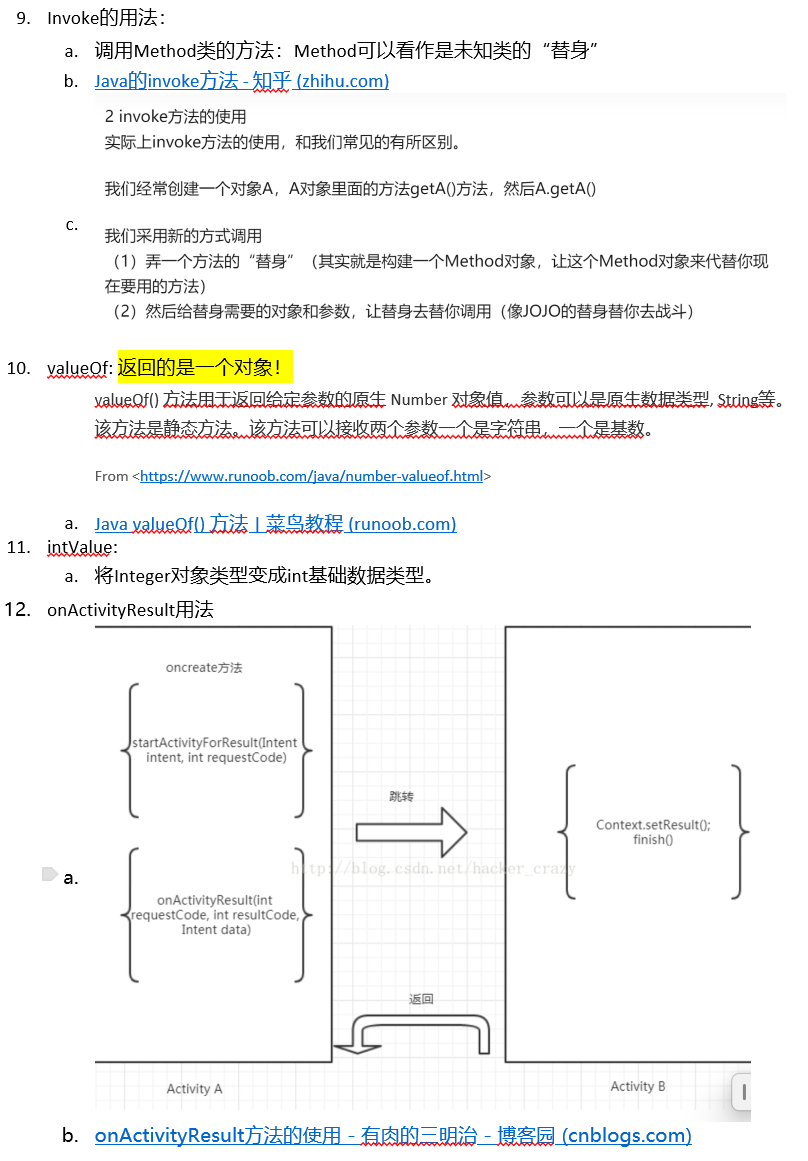
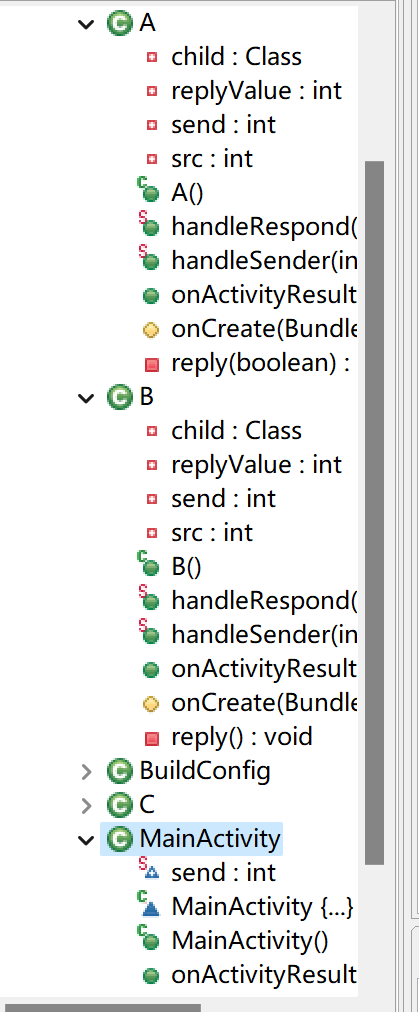
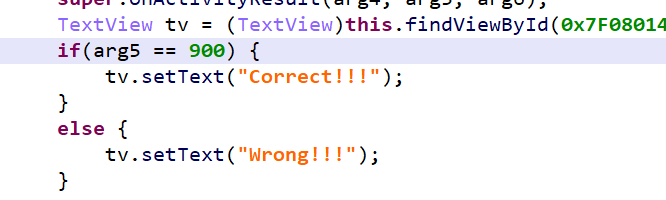
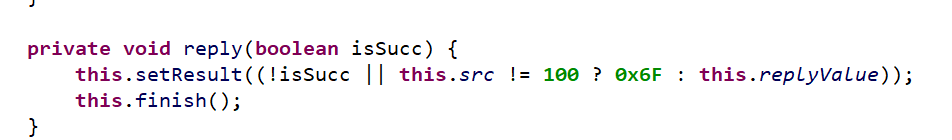
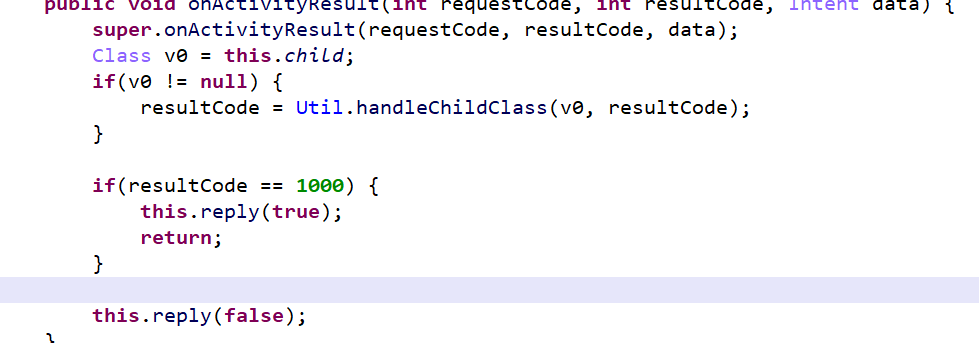
(1) Your Answer

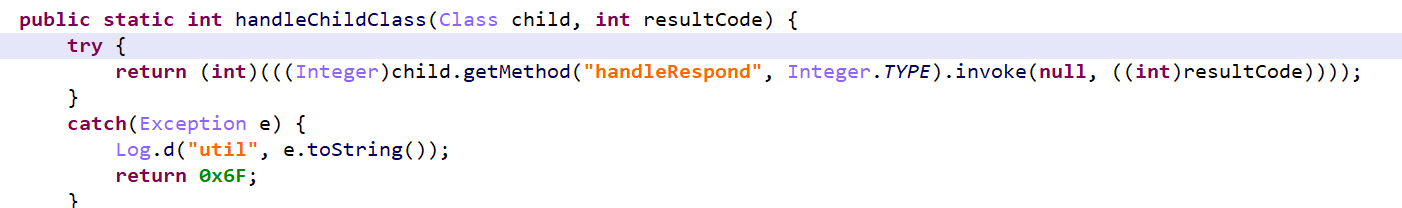
**i-A/d-C/i-C/d-B/i-B/**

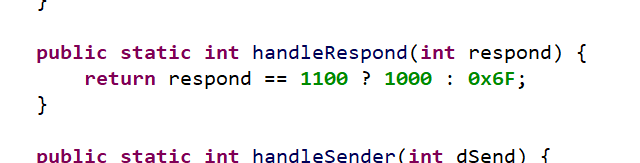


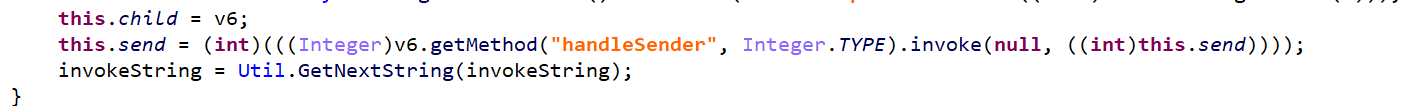
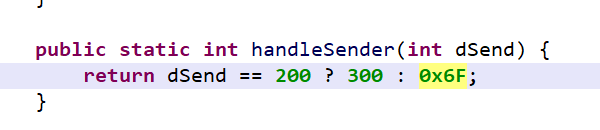
(2) Writeup

[Record how you solve this task here.]

1. 最初，先大致浏览了一下ppt，再回来看代码时发现还是一脸懵。于是决定边做边查找具体的方法，学习相关的reflation等机制。
2. 大致浏览一下全部代码，逐个查找不认识的方法并加以理解，如Log.d，getIntExtra，invoke等。目标是明确理解其参数列表和返回值，正确理解这些方法。尤其是onActivityResult的用法，理解多个Activity相互传递的逻辑。
   1. 一些资料查找的笔记：
   2. 掌握这些函数的相关用法之后，再回来看代码就不会觉得看不下去了。
3. 全局观：掌握各个类的结构：发现A,B,C的field和method都一样。再快速浏览一下这三个java源码，发现里面的代码也几乎如出一辙。因此推测三者是并行关系，都会给mainActivity反馈（这时候对Activity之间数据的传递的理解还有较大偏差——也直接导致了后面完全绕晕了的状态）。。
4. 再仔细查阅工具util类，明确大概的用法。
5. 然后从MainActivity入手开始逐行理解代码的逻辑。推断其逻辑是：页面显示一个文本编辑框EditText，要求输入某个唯一的字符串invokeString，结果正确显示Correct!!! 错误会显示Wrong!!!或者**Invoke String Format Error!** 此外，再根据util类GetNextString方法，推测该字符串的长度应该是4的倍数，且满足满足当invokeString.CharAt[i%4==1]=“-”，invokeString.CharAt[i%4==3]=“/”。第一个字符=i；然后调用startActivityForResult函数。
6. 寻找反馈的类。发现A,B,C三个都有onActivityResult，setResult和startActivityForResult，且startActivityResult的第二个参数都是0，因此无法凭此判定究竟哪个类监听调用哪个类。
7. 进一步了解onActivityResult等方法在多个Activity间传递数据的机制：还是针对两两传递，只是多个Activity可以有多种传递方式。但确定的是，一定是从MainActivity开始最后回到MainActivity的。下一步要做的就是确定传递是顺序，如何传递。
8. 逆向推理：从各个类的OnActivityResult方法推出。首先是MainActivity，如果其resultCode==900，则结果正确。因此寻找setResult有可能返回900的类，发现只有A。A.replyValue=900；因此可以确定主活动中调用的startActivityForResult是面向A的。进一步推测，A如果要返回900，需要同时满足其isSucc=true和src=100；由于A.src=MainActivity.send,且出了Activity传递其send之后没有再修改src的值，进一步确定MainActivity.send传给A。而要满足succ=true，需要其resultValue==1000，因此寻找可能返回1000的类，发现

现C,B的replyValue都不是1000,返回来看A，只有更改其resultCode才有可能，因此推出A.child!=null，并且根据handleRespond可以确定只有C可以将A的resultCode变成1000，可以推断出A传递给C。



进一步分析A如何传递给C（便于确定某些字符）。因为C返回给A还需要同时满足其isSuc=true和src=300。因为isSuc涉及C的进一步传递问题，先分析src。由于A.send=200，与C.src=300不符，再次检查B的send，B.send=600;而C的handSender方法继续沿着前面的思路分析，因为src在本类执行过程中不会改变，唯一的可能就是更改A.send，定位到了handleSender方法——可以确定，A中先创建一个C类对象并赋给A.child;（正好可以满足前面其child不为null的约束） 

正好满足；然后A再发出startActivityForResult传递给C；

回到对C.isSucc=true的分析，也推断出C.child!=null，推测C传递给B的方式同A—>C,先创建一个B类child对象，然后再更改C.send,然后再传递给B;

过程分析同A->C。推测合理。

最后再依次回调给MainActivity。

综上：传递顺序是MainActivity——>A(先创建child对象，再传递)——>C(先创建child对象，再传递)——>B——>C——>A——>MainActivity.

推断出invokeString.length=20，且根据

得出最终结果i-A/d-C/i-C/d-B/i-B/

流程分析图如下：