

通过按键去启动一个app

LauncherActivity()
frameworks/base/core/java/android/app/LauncherActivity.java

launcher桌面

onListItemClick()

用户点击某个apk图标

startActivity()
frameworks/base/core/java/android/app/Activity.java

Launch a new activity

startActivityForResult()
frameworks/base/core/java/android/app/Activity.java

这个函数会返回一个值，表示启动这个apk的状态，如果app退出，会在onActivityResult中收到返回值

mInstrumentation.execStartActivity()

ActivityManagerNative.getDefault().startActivity
frameworks/base/core/java/android/app/ActivityManagerNative.java

通知ActivityManager，这个app已经启动，并把这个app加入到ActivityManager管理中

通过binder与ActivityManagerService通信

startActivity()

activityManagerService启动某一个app

为这个APP设置一些属性，比如权限什么的。

startActivityLocked

接下页

startActivityLocked(真正启动app)

startActivityLocked的实现

把当前的APP放到ActivityManagerService
栈顶，这样启动的时候就能够启动这个app

mService.mWindowManager.setAppStartingWindow

启动这个app的界面

具体实现

首先通过binder获取到现在运行的
window的Handle，然后用将要
运行的app的window的handle代替
之，这样就会显示出这个app的
window界面了

resumeTopActivityLocked

这里会启动app(具体的会
在后面详细分析)

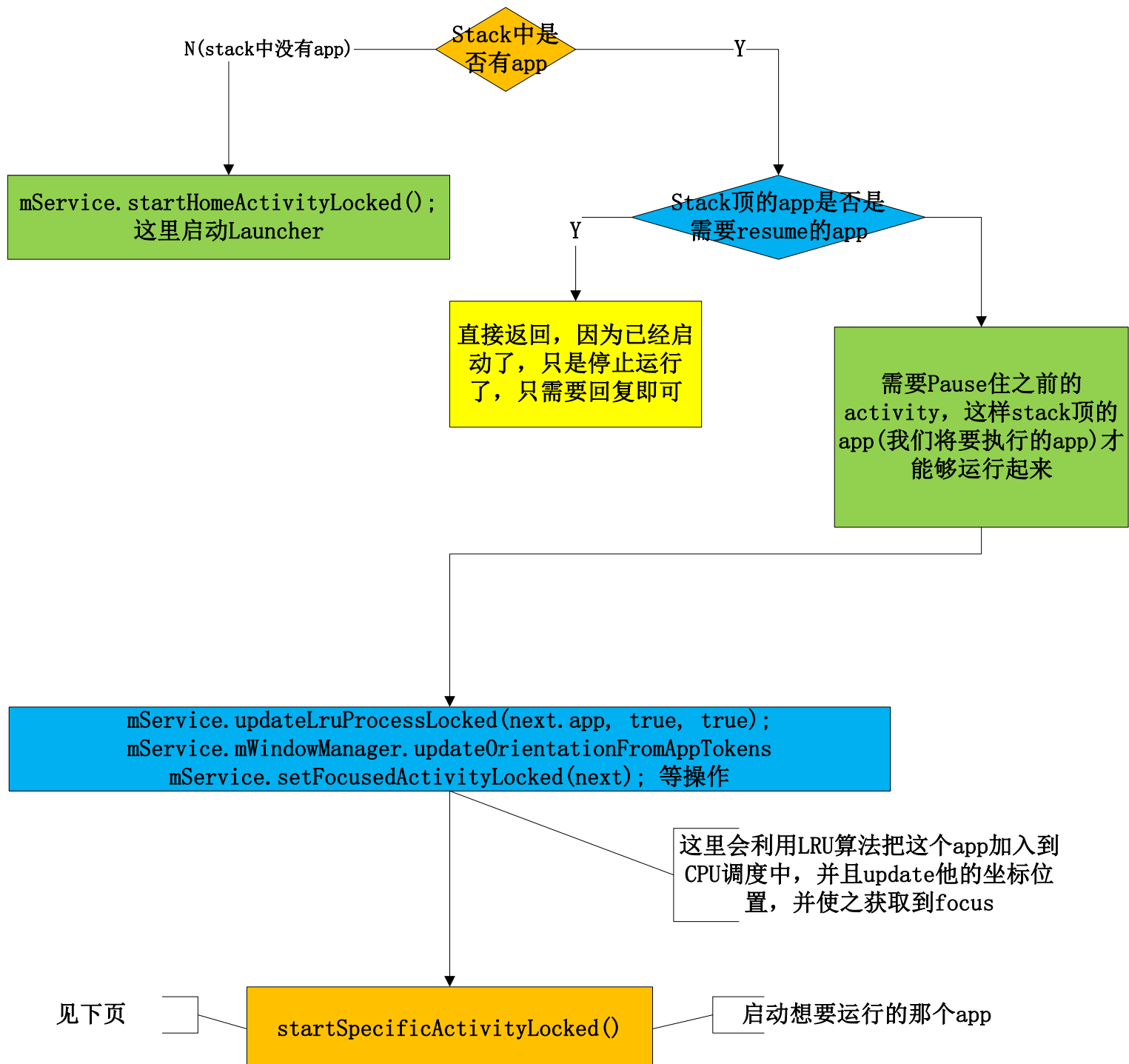
resumeTopActivityLocked

resumeTopActivityLocked
(frameworks/base/services/java/com/android/server/am/ActivityStack.java)

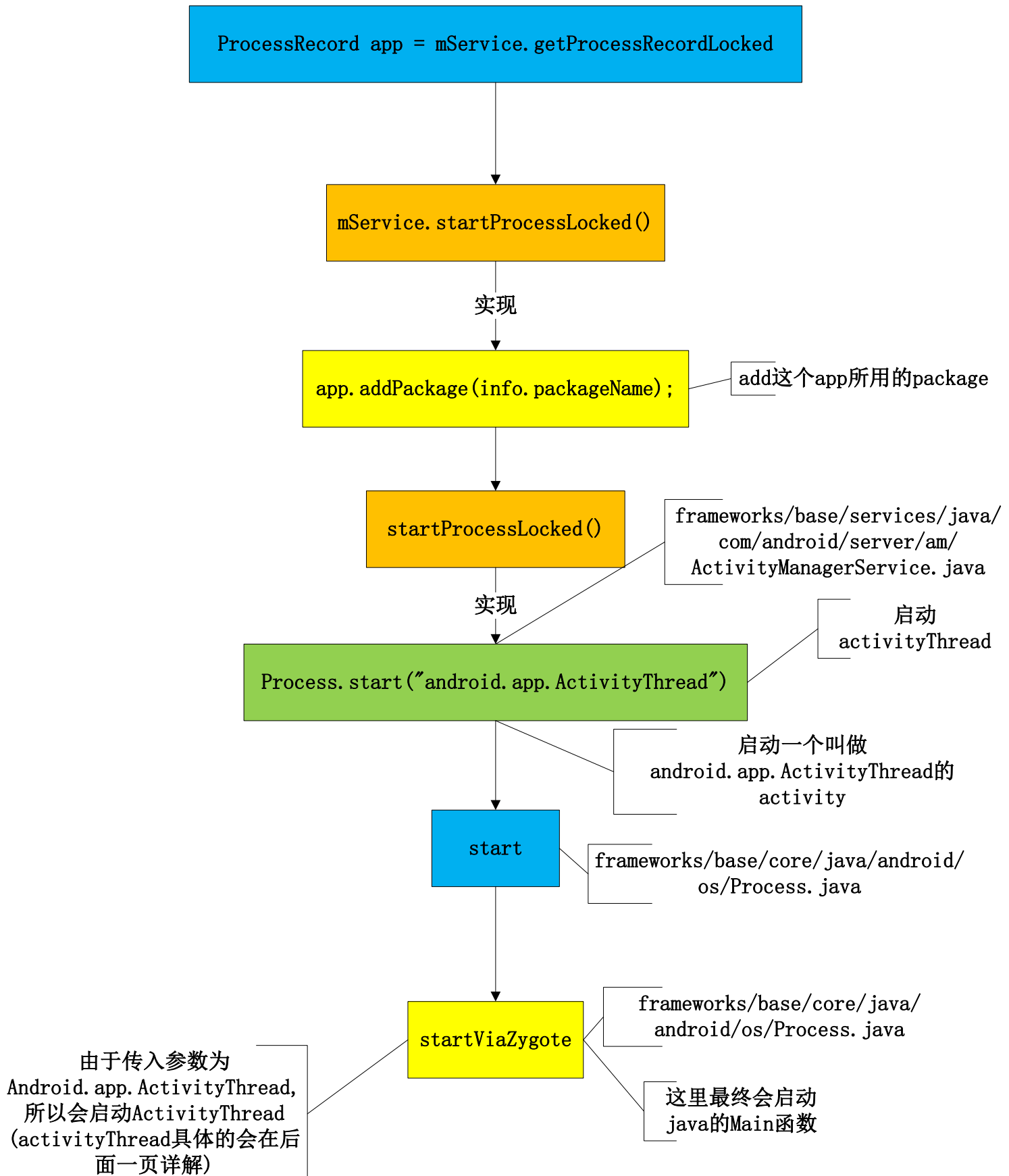
Start和resume是一个接口

```
ActivityRecord next = topRunningActivityLocked(null);
```

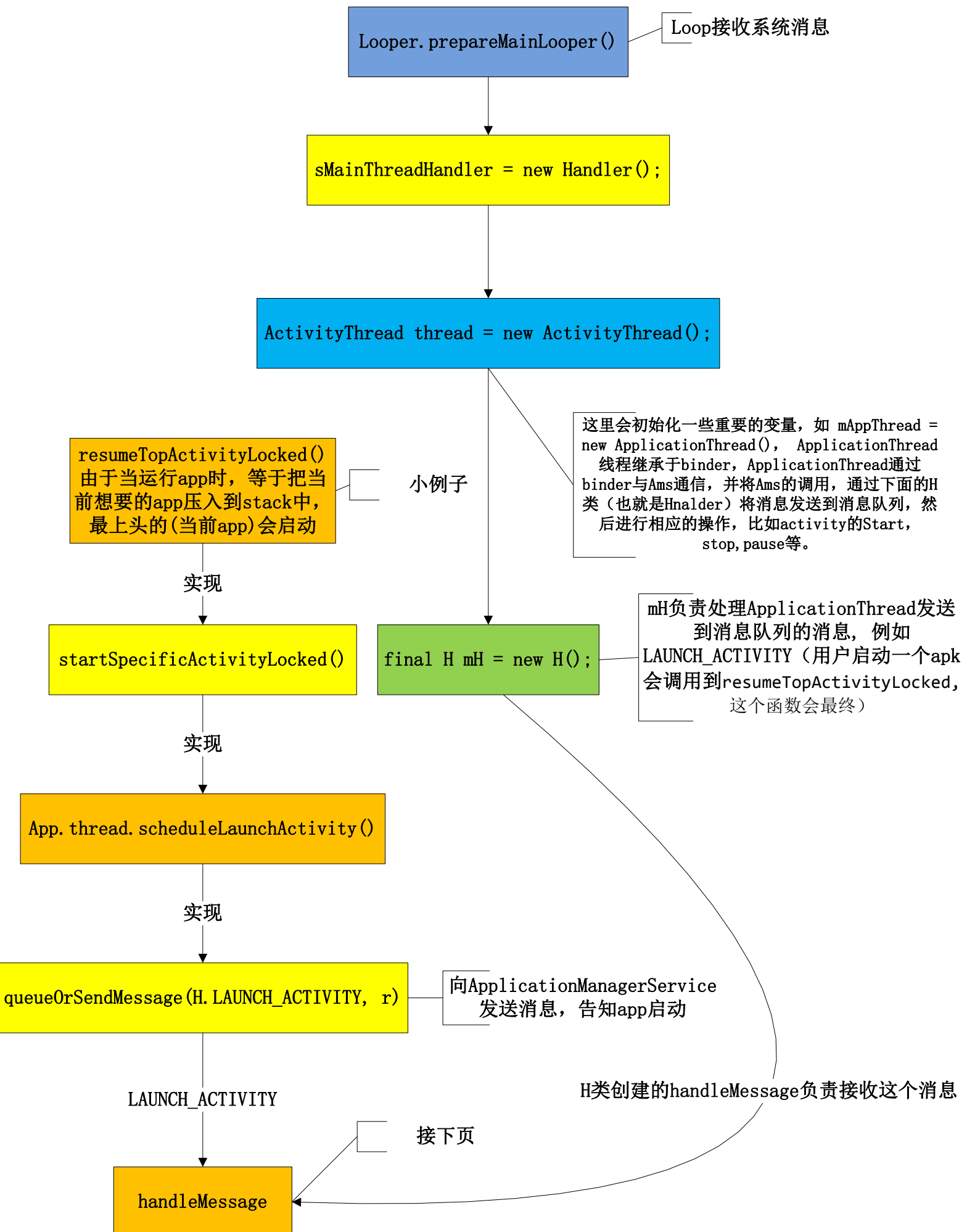
找到stack中可以运行的第一个app(即我们当前要执行的app)



startSpecificActivityLocked



ActivityThread线程及APP启动的最后一步(调用onCreate)分析



ActivityThread分析2(接上页)

handleMessage分析

