**Android 分屏原理**

点击近期任务事件：

com.android.systemui.recents.RecentsImpl#startRecentsActivity(android.app.ActivityOptions, com.android.systemui.shared.recents.view.AppTransitionAnimationSpecsFuture)

点击近期任务卡片的点击事件：

com.android.systemui.recents.views.TaskView#onClick

长安近期任务卡片的事件：

com.android.systemui.recents.views.TaskView#onLongClick

1. **设备是否支持分屏**

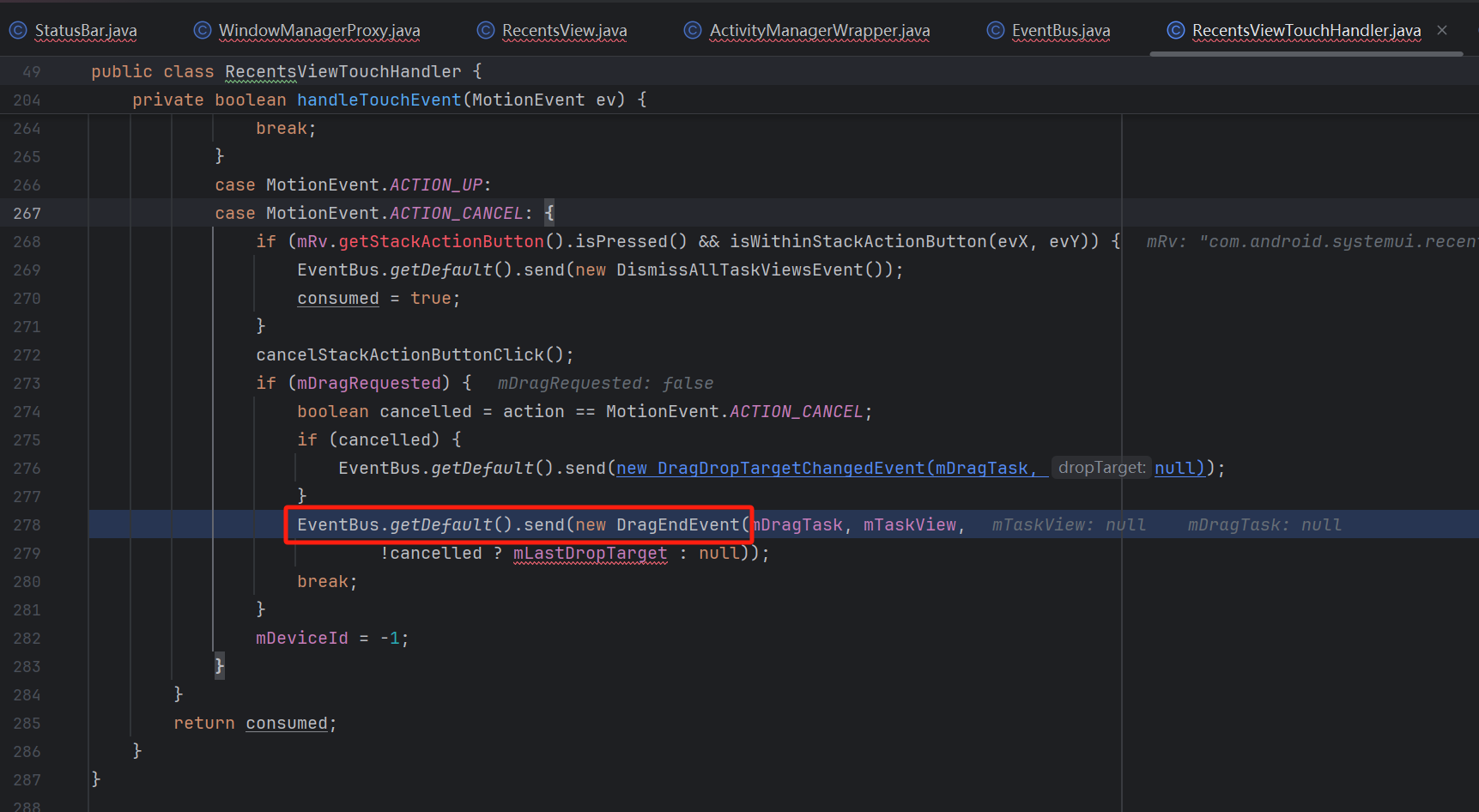
framework的base/core/res/res/values/config.xml文件中配置config\_supportsSplitScreenMultiWindow属性。

|  |
| --- |
| Java  <!-- True if the device supports at least one form of multi-window.  E.g. freeform, split-screen, picture-in-picture. -->  <bool name="config\_supportsMultiWindow">true</bool>   <!-- True if the device supports split screen as a form of multi-window. -->  <bool name="config\_supportsSplitScreenMultiWindow">true</bool>   <!-- True if the device supports running activities on secondary displays. -->  <bool name="config\_supportsMultiDisplay">true</bool> |

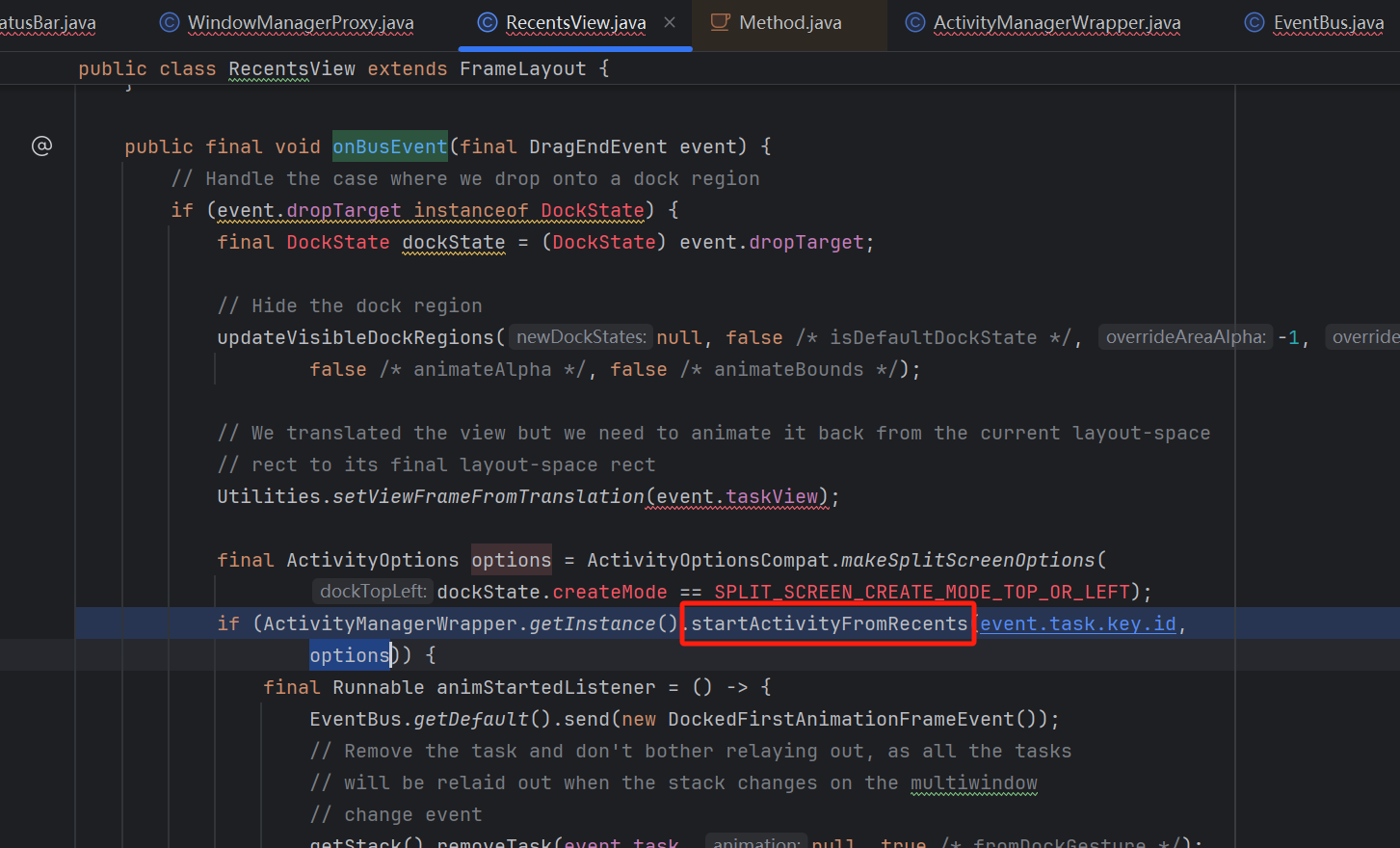
2. **如何进入分屏**

长按近期任务卡片，显示可拖动，拖动到左侧或者右侧。触发执行DragEndEvent事件

RecentsViewTouchHandler代码



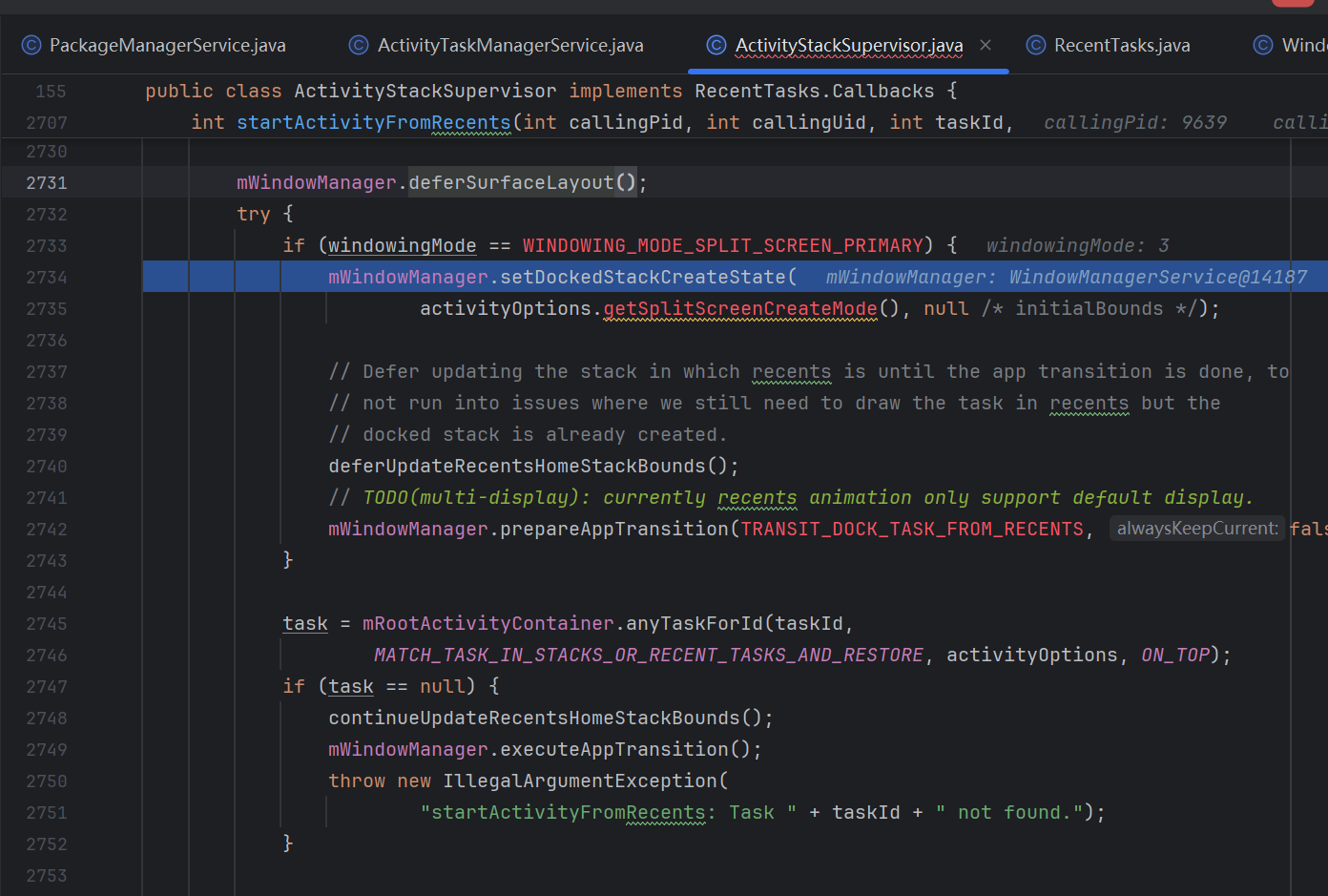
RecentsView中存在onEventBus方法，EventBus会分发到这里。



通过执行startActivityFromRecents，传入要执行分屏的taskId和分屏options进入framework-service执行分屏操作。

startActivityFromRecents方法需要android.permission.START\_TASKS\_FROM\_RECENTS权限（系统签名等级和privilege等级，或者和rencents进程具有相同的uid）

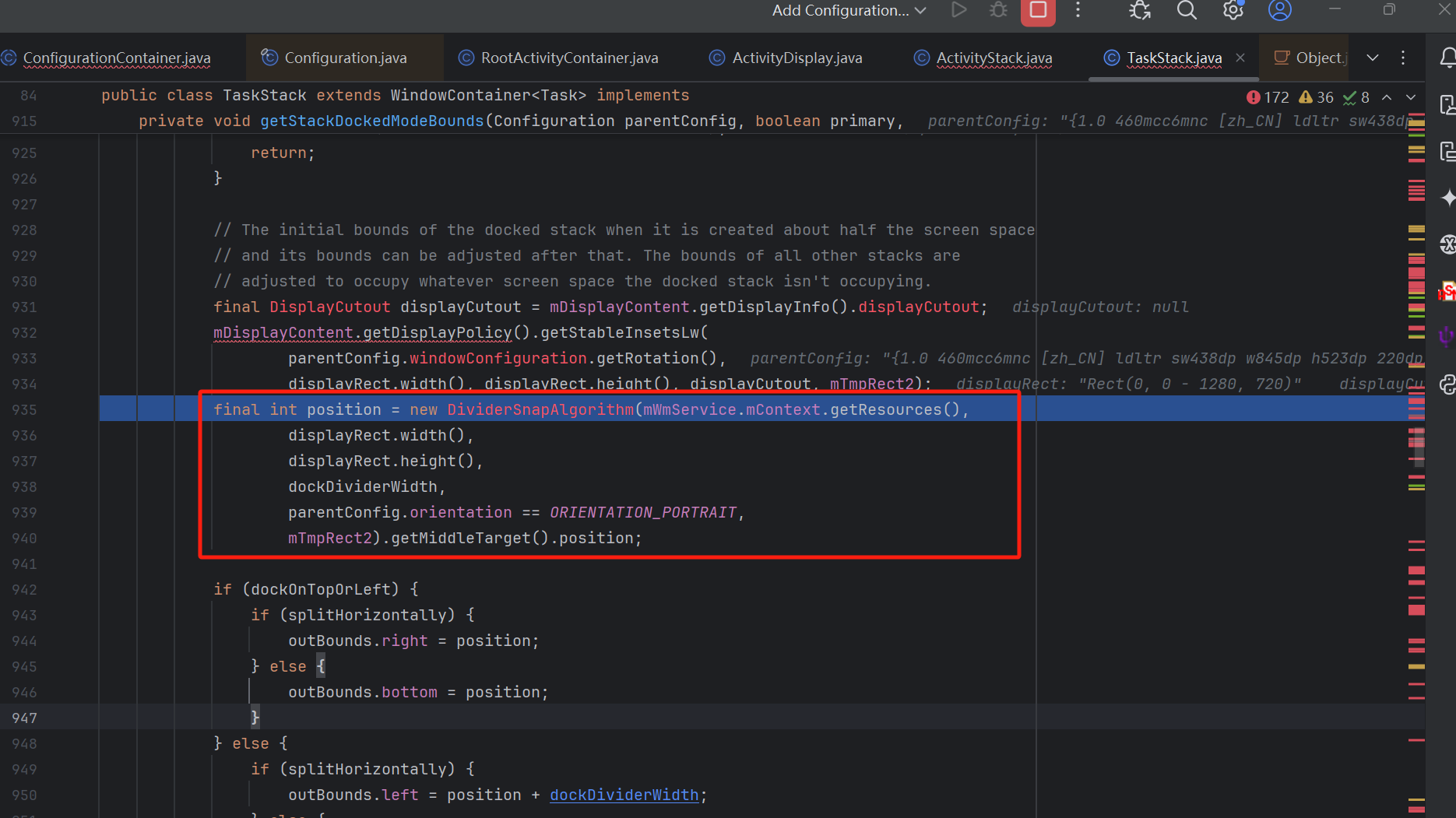
ams会调用进入com.android.server.wm.ActivityStackSupervisor#startActivityFromRecents，触发windowingMode == WINDOWING\_MODE\_SPLIT\_SCREEN\_PRIMARY的判断



......

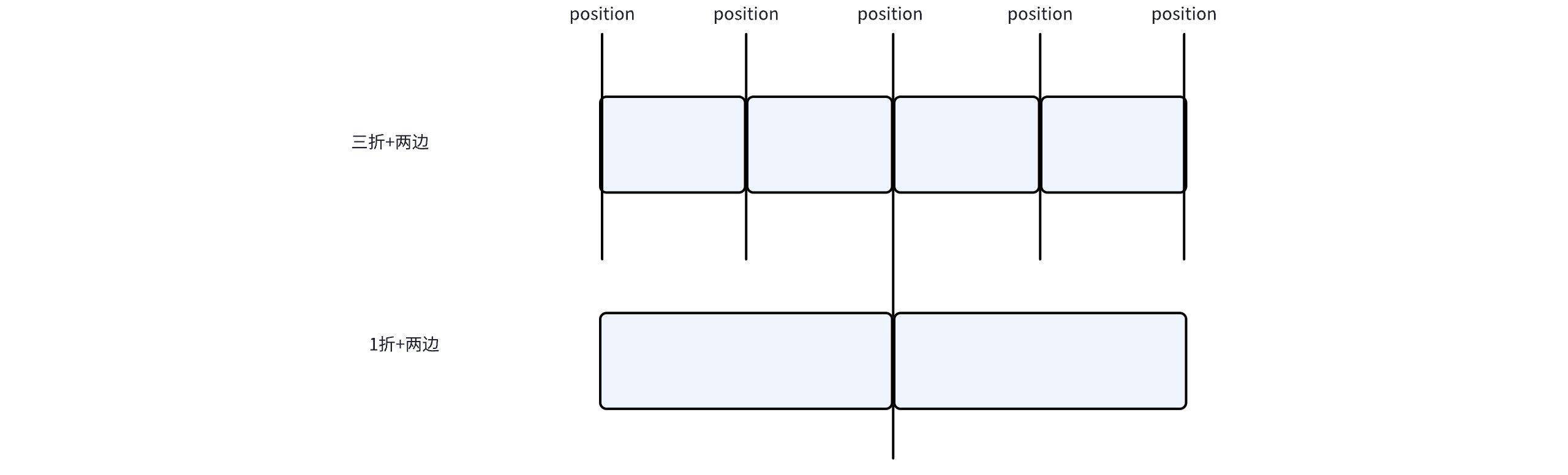
......

TaskStack中com.android.server.wm.TaskStack#getStackDockedModeBounds方法会进行边界计算



最终计算分屏边界的类：com.android.internal.policy.DividerSnapAlgorithm

com.android.internal.policy.DividerSnapAlgorithm.SnapTarget，折痕概念



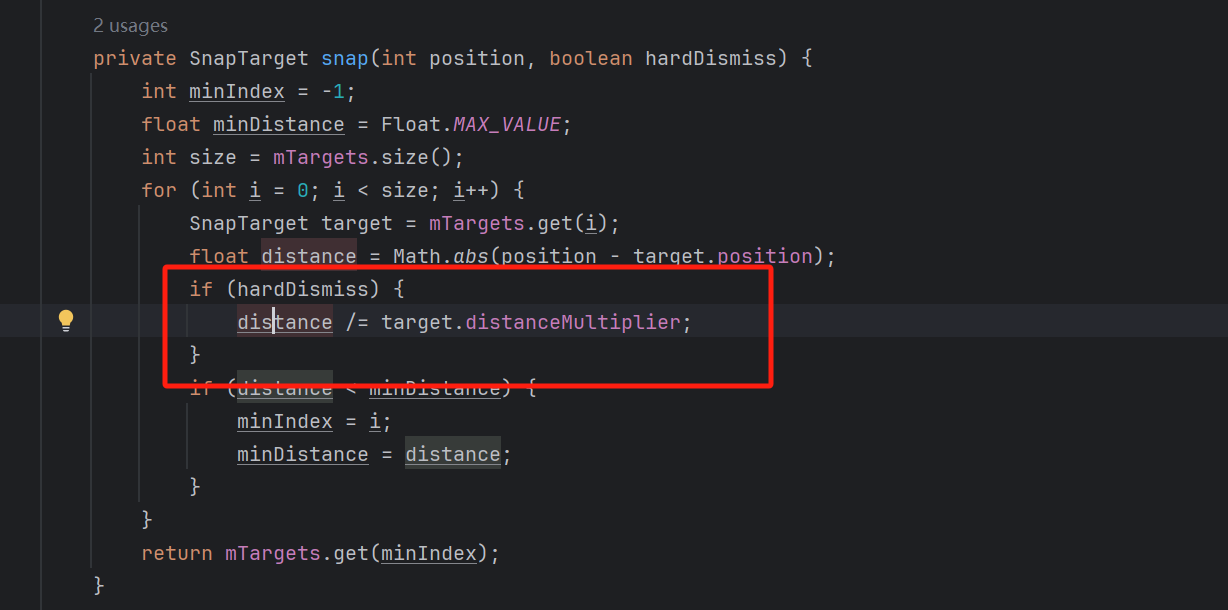
属性：

position：折痕位置

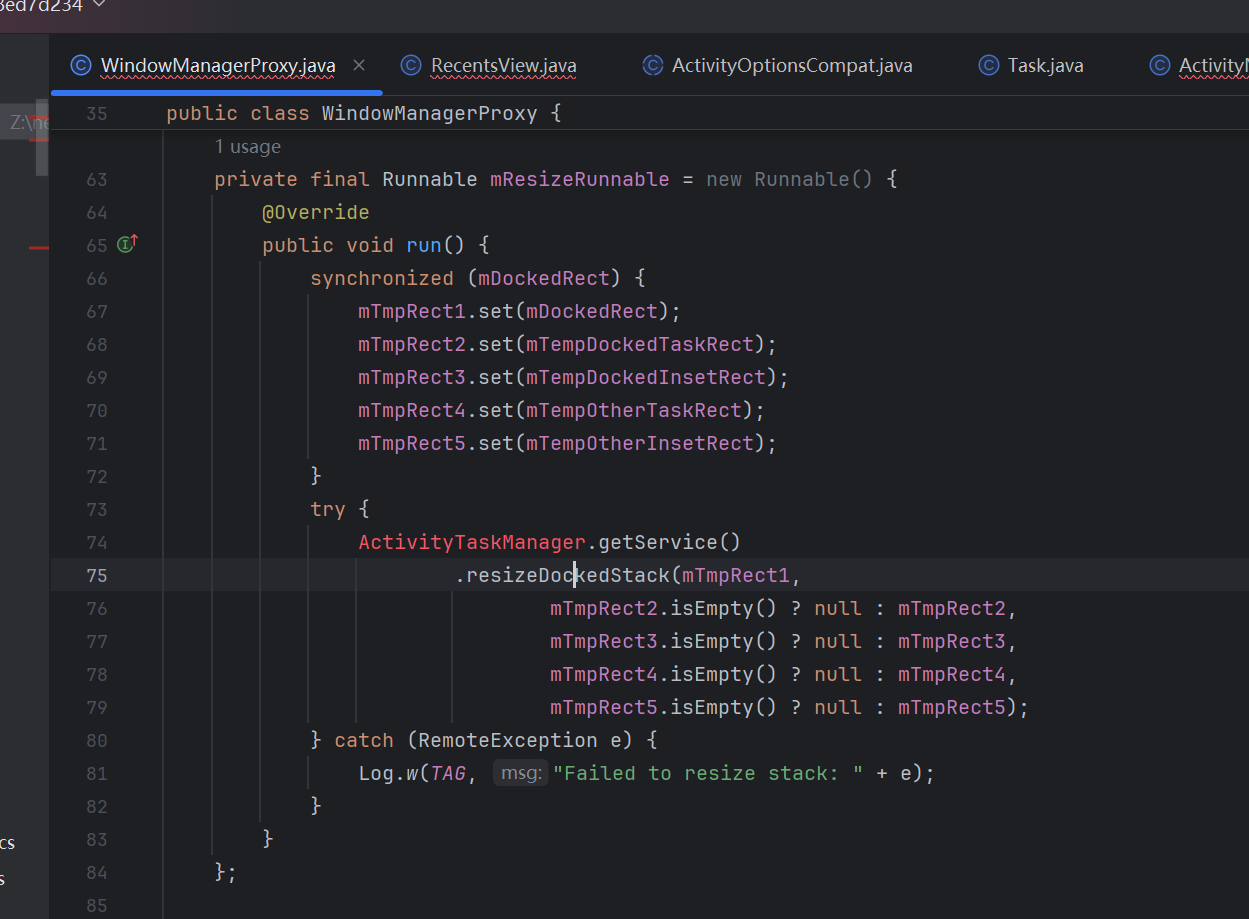
taskPosition：task所处的位置，一般和position相同

flag：退出分屏的逻辑

distanceMultiplier：hardDismiss模式下，真实距离会除以该值，如果该值越小，会导致计算后的距离越大，就更难到达这个target



分割线拖动时，会出触发resizeDockedStack调用，不停设置stack的bounds，但是如果没有触发分割线的变化，每次传递的值都会是一样的



第一个参数是primary stack的bounds，拖动过程中持续变化

第二个参数是primary stack的附带了insets的结果，拖动过程中保持不变。拖动结束后传null

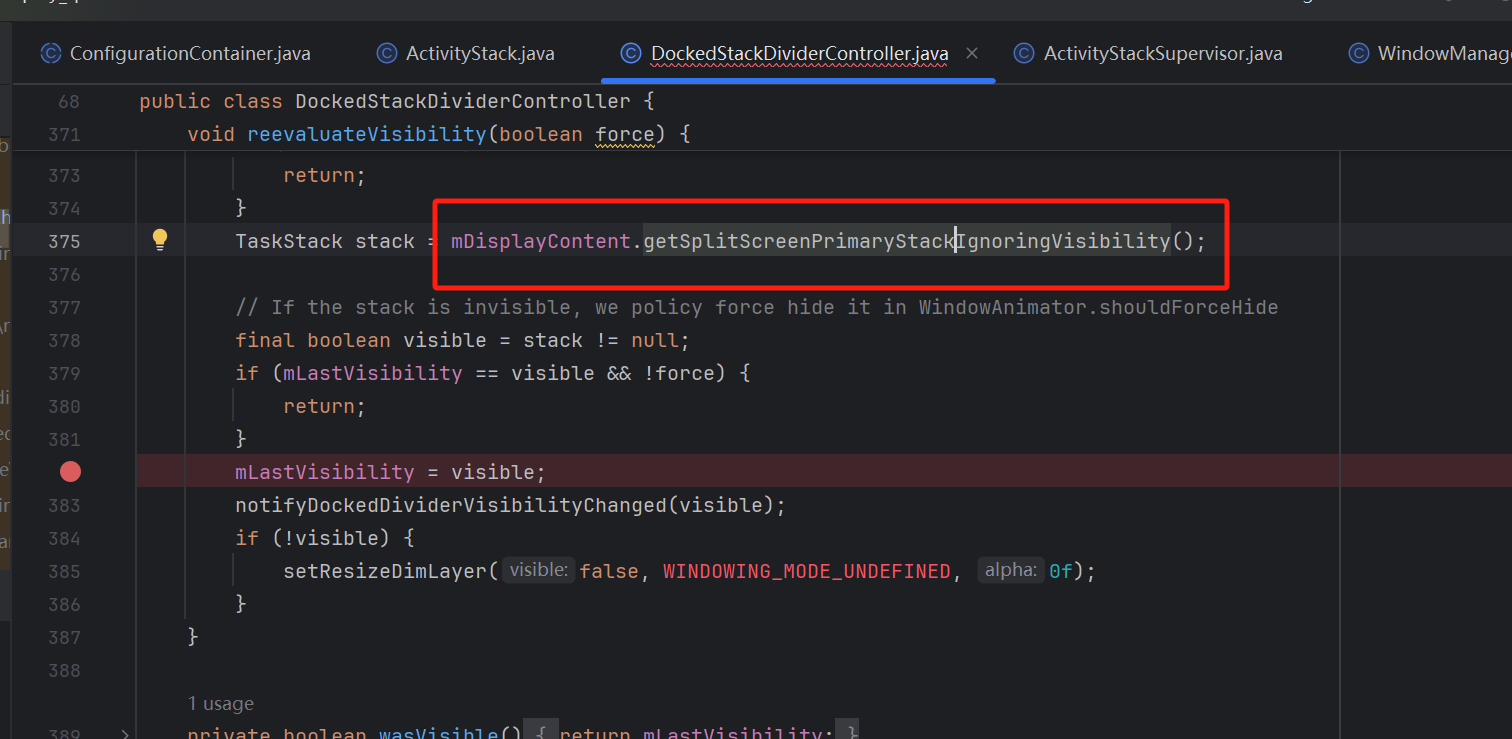
第三个参数是primary stack的task边界，拖动过程中不变

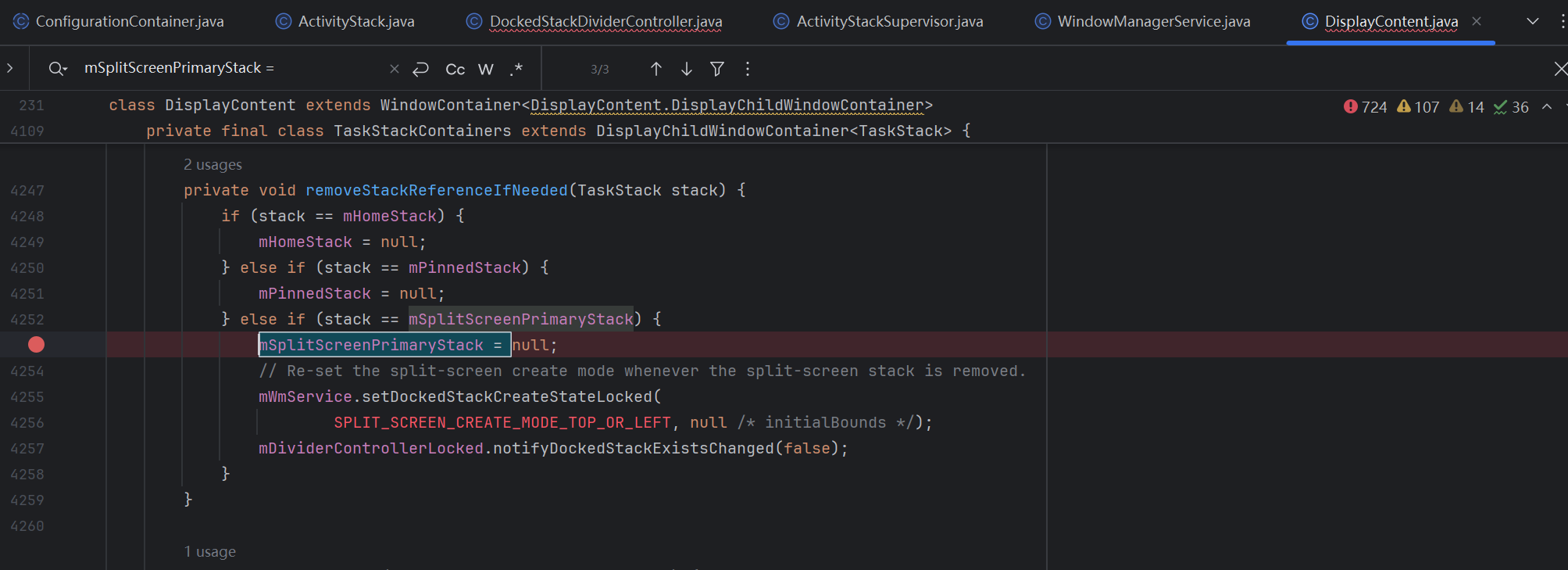
第四个参数是secondary stack的bounds，拖动过程中持续变化

第五个参数是secondary stack的附带了insets的结果，拖动过程中保持不变。拖动结束后传null

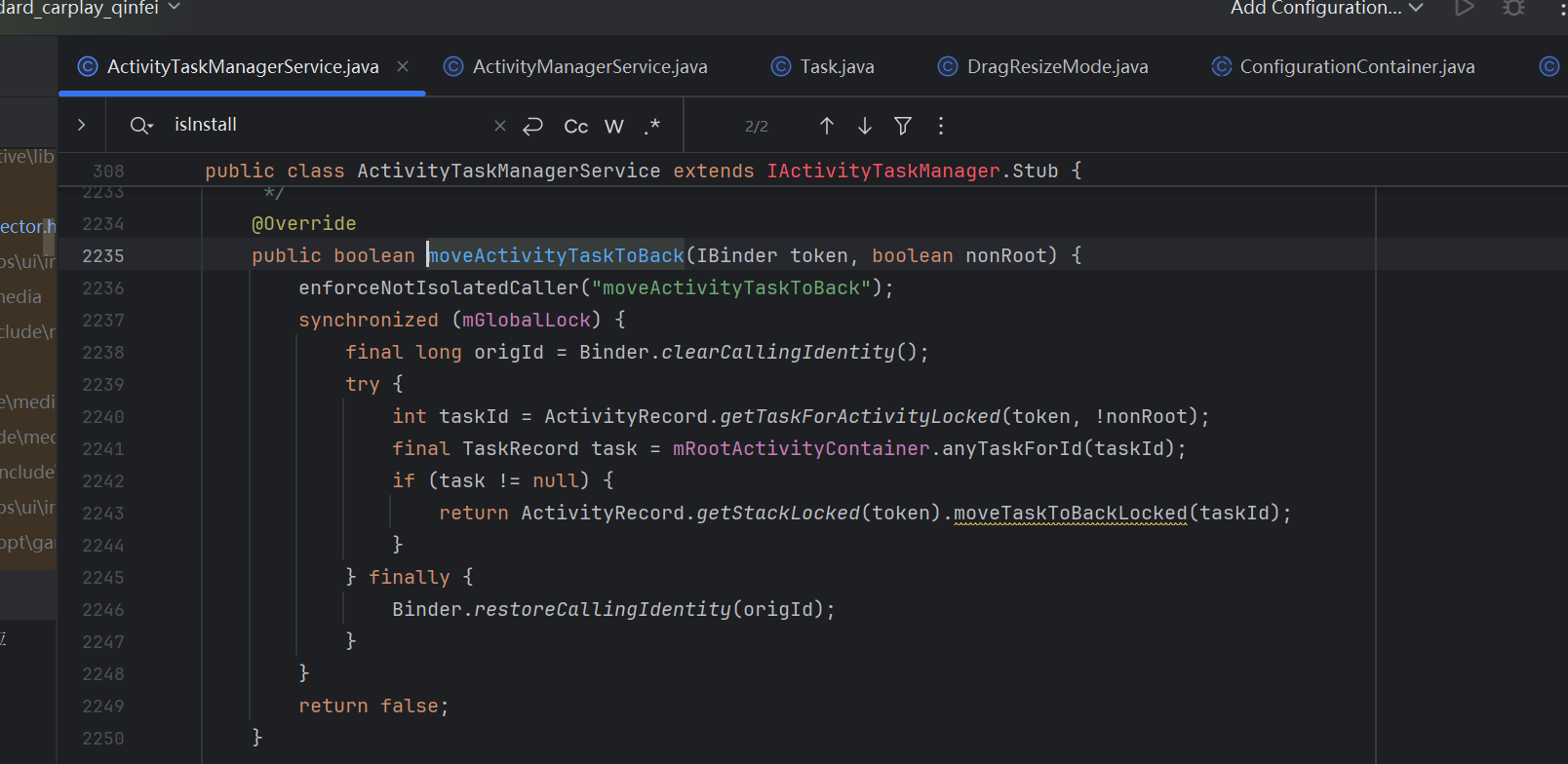
如何退出分屏：

通过反向追踪，发现只有task为null时才会导致systemui退出分屏





该方法断点发现是在moveActivityTaskToBack中触发



1. activity

IActivityTaskManager.moveActivityTaskToBack

1. ATMS

android.app.IActivityTaskManager#dismissSplitScreenMode

参考com.android.systemui.statusbar.phone.StatusBar#toggleSplitScreenMode方法

获取吸附方向：WindowManager.getDockedStackSide