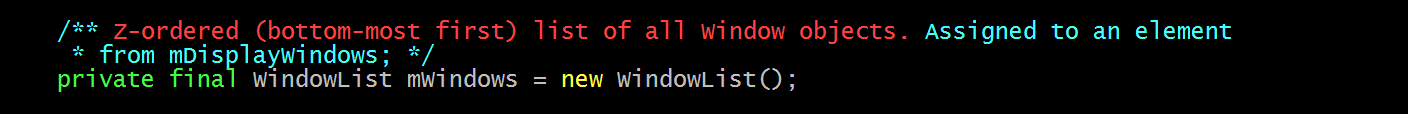
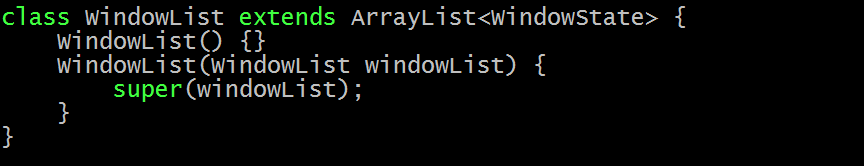
与老罗的博客相比，WindowManagerService为支持多屏特性(4.3->4.4)

原来位于WindowManagerService中的保存窗口层级的mAppTokens被移动到DisplayContent类中。



WindowList类位于WindowState中，用于封装ArrayList<WindowState>



所以可以想像，在WindowManagerService类中必然有一个保存所有DisplayContent的成员变量：

SparseArray<DisplayContent> mDisplayContents = new SparseArray<>(2);

在DisplayContent中显示次序的组织还依靠Task的顺序，这里与AMS中的task，stack一致。（多窗口特性）

每个Stack会有一个唯一的Id，在ActivityManager.java中定义了这些Stack的Id：

/\*\* First static stack ID. \*/

**public** **static** **final** int **FIRST\_STATIC\_STACK\_ID** = 0;

/\*\* Home activity stack ID. \*/

**public** **static** **final** int **HOME\_STACK\_ID** = **FIRST\_STATIC\_STACK\_ID**;

/\*\* ID of stack where fullscreen activities are normally launched into. \*/

**public** **static** **final** int **FULLSCREEN\_WORKSPACE\_STACK\_ID** = 1;

/\*\* ID of stack where freeform/resized activities are normally launched into. \*/

**public** **static** **final** int **FREEFORM\_WORKSPACE\_STACK\_ID** = **FULLSCREEN\_WORKSPACE\_STACK\_ID** + 1;

/\*\* ID of stack that occupies a dedicated region of the screen. \*/

**public** **static** **final** int **DOCKED\_STACK\_ID** = **FREEFORM\_WORKSPACE\_STACK\_ID** + 1;

/\*\* ID of stack that always on top (always visible) when it exist. \*/

**public** **static** **final** int **PINNED\_STACK\_ID** = **DOCKED\_STACK\_ID** + 1;

由此我们可以知道，系统中可能会包含这么几个Stack：

* 【Id：0】Home Stack，这个是Launcher所在的Stack。 其实还有一些系统界面也运行在这个Stack上，例如近期任务
* 【Id：1】FullScren Stack，全屏的Activity所在的Stack。 但其实在分屏模式下，Id为1的Stack只占了半个屏幕。
* 【Id：2】Freeform模式的Activity所在Stack
* 【Id：3】Docked Stack 下文中我们将看到，在分屏模式下，屏幕有一半运行了一个固定的应用，这个就是这里的Docked Stack
* 【Id：4】Pinned Stack 这个是画中画Activity所在的Stack

场景

需要调整窗口层次的时机

1. 添加窗口
2. 调整任务栈顺序
3. 调整任务栈数量

1.添加应用窗口:

先获得显示窗口

查询是否在快查列表中有应用的Window

新建WindowState对象

调用addWindowToListInOrderLocked

最后调用addAppWindowToListLocked在位置添加WindowState对象

WindowManagerService类中与层级计算相关的几个成员变量：

