1、如果Activity被意外关闭，如何判断Activity是否被重建?

> 1. 通过onRestoreInstanceState()和onCreate方法来判断。
> 2. onRestoreInstanceStae()被调用，或者onCreate()的参数Bundle不为null。都表明Activity被重建
> 3. 因为Activity被异常关闭后，那么系统会调用onSaveInstanceState保存当前Activity的状态。

2、系统保存和恢复View层次结构的工作流程

> 1. `Activity` 被意外终止时，会调用 `onSaveInstanceState()` 去保存数据
> 2. `Activity` 去委托 `Window` 保存数据
> 3. `Window` 再委托顶层容器去保存数据(ViewGroup：一般是DecorView)
> 4. 顶层容器 最后一一通知其子元素保存数据

3、onSaveInstanceState()源码分析

```java
// Activity.java-保存状态
protected void onSaveInstanceState(Bundle outState) {
    // 1、保存Window的层次状态。
    outState.putBundle(WINDOW_HIERARCHY_TAG, mWindow.saveHierarchyState());
    outState.putInt(LAST_AUTOFILL_ID, mLastAutofillId);
    // 2、保存所有Fragment的状态
    Parcelable p = mFragments.saveAllState();
    outState.putParcelable(FRAGMENTS_TAG, p);
    if (mAutoFillResetNeeded) {
        outState.putBoolean(AUTOFILL_RESET_NEEDED, true);
        // 3、自动填充功能，保存状态信息
        getAutofillManager().onSaveInstanceState(outState);
    }
    // 4、Application执行dispatchActivitySaveInstanceState(), 内部调用ActivityLifecycleCallb
    getApplication().dispatchActivitySaveInstanceState(this, outState);
}

// PhoneWindow.java-保存层次状态
public Bundle saveHierarchyState() {
    Bundle outState = new Bundle();
    // DecorView
    if (mContentParent == null) {
        return outState;
    }
    // 1、DecorView保存层次状态
    SparseArray<Parcelable> states = new SparseArray<Parcelable>();
    mContentParent.saveHierarchyState(states);
    outState.putSparseParcelableArray(VIEWS_TAG, states);
    // 2、保存具有焦点的View的ID
    final View focusedView = mContentParent.findFocus();
    if (focusedView != null && focusedView.getId() != View.NO_ID) {
        outState.putInt(FOCUSED_ID_TAG, focusedView.getId());
    }
    // 3、保存Panel的状态(Menu中面板)。比如自定义Menu的样式，需要在AppCompatActivity的onPrepar
    SparseArray<Parcelable> panelStates = new SparseArray<Parcelable>();
    savePanelState(panelStates);
    if (panelStates.size() > 0) {
        outState.putSparseParcelableArray(PANELS_TAG, panelStates);
    }
    // 4、保存Toolbar的层次状态
    if (mDecorContentParent != null) {
        SparseArray<Parcelable> actionBarStates = new SparseArray<Parcelable>();
        mDecorContentParent.saveToolbarHierarchyState(actionBarStates);
        outState.putSparseParcelableArray(ACTION_BAR_TAG, actionBarStates);
    }
    return outState;
}

// View.java-DecorView会执行dispatchSaveInstanceState，分发层层保存状态
public void saveHierarchyState(SparseArray<Parcelable> container) {
    dispatchSaveInstanceState(container);
}

// ViewGroup.java-DecorView(ViewGroup)重写了该方法并且遍历子View
```

```
@Override
protected void dispatchSaveInstanceState(SparseArray<Parcelable> container) {
    // xxx省略xxx
    final View[] children = mChildren;
    for (int i = 0; i < mChildrenCount; i++) {
        View c = children[i];
        // 遍历view并且执行dispatchSaveInstanceState()
        c.dispatchSaveInstanceState(container);
    }
}

// View.java-执行onSaveInstanceState()保存View的状态
protected void dispatchSaveInstanceState(SparseArray<Parcelable> container) {
    // 1、执行onSaveInstanceState(), 保存View的状态。(自定义View时，可以重写该方法来保存View中
    Parcelable state = onSaveInstanceState();
    if (state != null) {
        // 2、将以mID作为key, 保存到SparseArray中。外部最终将SparseArray保存到PhoneWindow中的
        container.put(mID, state);
    }
}
```

4、请简述下关于保存和恢复Activity的状态信息，内部的工作流程是怎么样的?

1. Activity: 执行onSaveInstanceState()
    1. 保存Window的层次状态
    2. 保存所有Fragment的状态
    3. 保存自动填充功能的状态信息(Android8.0推出):Android 8.0上的自动填充功能
    4. Application.dispatchActivitySaveInstanceState()-涉及到LifeCycle这个谷歌新推出的内容。
2. Window: saveHierarchyState()保存状态信息。
    1. mContentParent.saveHierarchyState(states): 交给DecorView层层保存View的信息
    2. 保存具有焦点的View的ID
    3. 保存Panel的状态(Panel属于菜单Menu中的内容):
    4. 保存Toolbar的状态
3. DecorView(ViewGroup): 层层保存所有View的信息
    1. saveHierarchyState()->dispatchSaveInstanceState()

5、在Activity中调用onRestoreInstanceState()恢复数据和在onCreate中恢复数据区别是?

1. onRestoreInstanceState一旦被调用，那么其参数Bundle saveInstanceState一定是有值的，不需要而外判断是否为空;
2. onCreate需要额外判断 `Bundle saveInstanceState` 是否为null。
3. 官方建议采用OnRestoreInstanceState恢复数据。

6、如何利用onCreate()的参数savedInstanceState解决因为Activity的重建导致Fragment的重叠问题?

1. Activity的onCreate()需要在 `savedInstanceState==null` 时才添加Fragment

> 2. 否则会出现因为重建导致多次添加Fragment，从而导致重叠。

7、什么情况下可能会导致Activity的重建?

> 1. 按下HOME键
> 2. 按下电源键
> 3. 启动其他Activity
> 4. 横竖屏切换

8、如何在Activity禁止横竖屏切换?

> 1. 在 `AndroidManifest` 中给相应的 `Activity` 添加上属性 `android:screenOrientation="portrait"`
> 2. `portrait` 为竖屏
> 3. `landscape` 为横屏
> 4. 或者可以在 `onCreate` 中添加代码 `setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT/LANDSCAPE);`
> 5. 最终会禁止横竖屏切换，也不会触发 保存数据/恢复数据 的回调

9、如何实现Activity在横竖屏切换时，禁止Activity的重建?(通过configChanges实现)

> 1. 给 `Activity` 添加属性： `android:configChanges="orientation|screenSize"`
> 2. `Activity` 不会再销毁和重建，只会调用 `onConfigurationChanged()` 方法，可以进行特殊处理。

10、禁止了Activity的横竖屏切换(重建)后, 什么回调方法会在横竖屏切换时被调用?

> onConfigurationChanged()

11、通过禁止了横竖屏从而禁止了Activity的重建，既然Activity不再会重建，也就不需要再去处理数据的保存和恢复?

> 错误!
>
> 1. 内存不足时，依旧可能会出现Activity被杀死并且重建的情况。