Android Coding Assessment Test Prepare

deepwaterooo

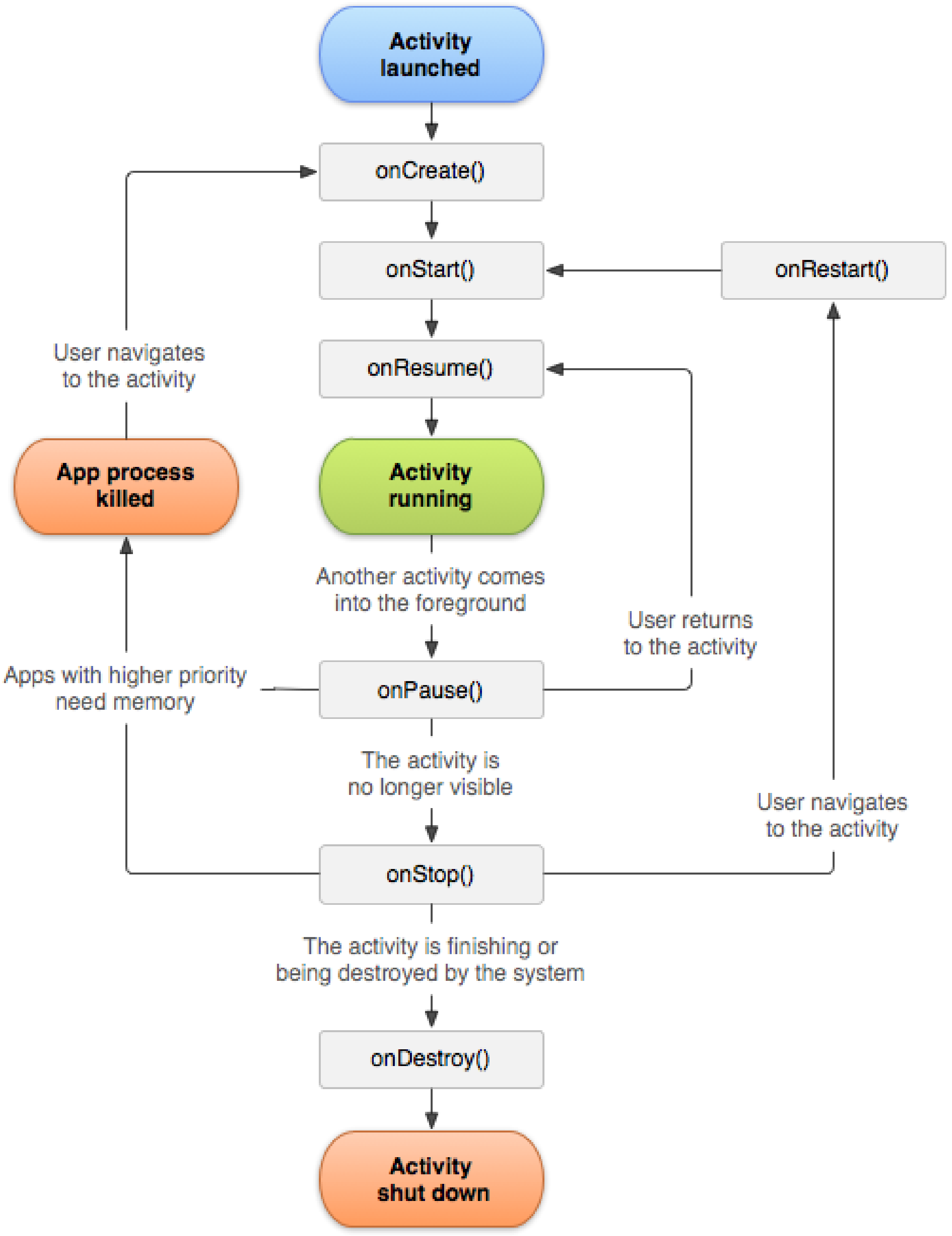
2021 年 12 月 14 日

目录

Chapter 1

Activity 生命周期探讨

- https://www.jianshu.com/p/1b3f829810a1


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

1.1 Activity 生命周期相关函数说明

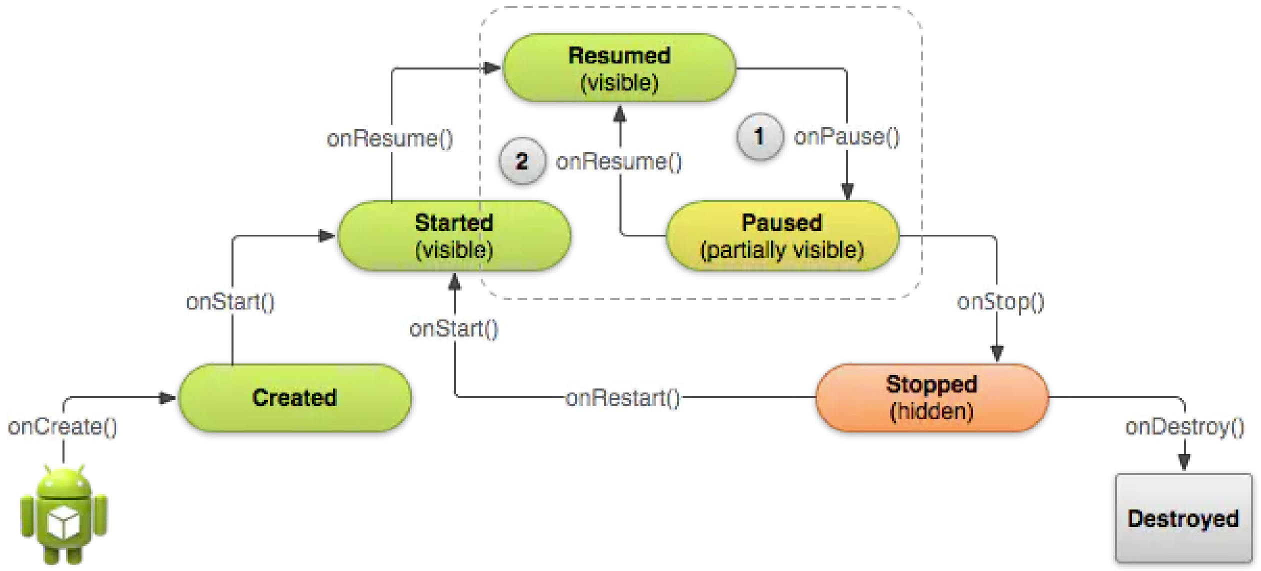
- (1) onCreate() 是 activity 正在被创建，也就是说此时的 UI 操作不会更新 UI，比如 setText() 操作，所以此时在子线程调用 setText() 不会报线程错误。详解可见 Android 子线程更新 View 的探索, 在这个方法内我们可以做一些初始化工作。
- (2) onRestart() 需要注意的是：activity 正在重新启动，一般情况下，activity 从不可见状态到可见状态，onRestart() 才会被调用，但是一定要注意的是一般来说这是用户行为导致 activity 不可见的时候，此时变为可见的时候才会调用

onRestart(), 这里所说的用户行为就是用户按 home 键，或者进入“新”的 activity。这样的操作会使 activity 先执行 onPause(), 后执行 onStop()，这样回到这个 activity 会调用 onRestart()。为什么我这里强调说用户行为导致的不可见状态，等下我会说。。。

- (3) onStart() 的时候，activity 才可见，但是没有出现在前台，无法与用户交互
- (4) onResume() 的时候，activity 已经可见，并且出现在前台开始活动，与 onStart() 相比，activity 都已经可见，但是 onStart() 的时候 activity 还在后台，onResume() 才显示在前台
- (5) onPause() 主要注意的是：此时的 activity 正在被停止，接下来马上调用 onStop()。特殊情况下快速回到该 activity，onStop() 不会执行，会去执行 onResume()。
  - 一般在这个生命周期内做存储数据、停止动画工作，但不能太耗时。
  - 为什么特殊强调呢，因为该 activity 的 onPause() 执行完了，才回去执行新的 activity 的 onResume()，一旦耗时，必然会拖慢新的 activity 的显示。
- (6) onStop()：此时的 activity 即将停止。在这里可以做稍微重量级的操作，同样也不能耗时。
- (7) onDestroy()：此时的 activity 即将被回收，在这里会做一些回收工作和最终资源释放。

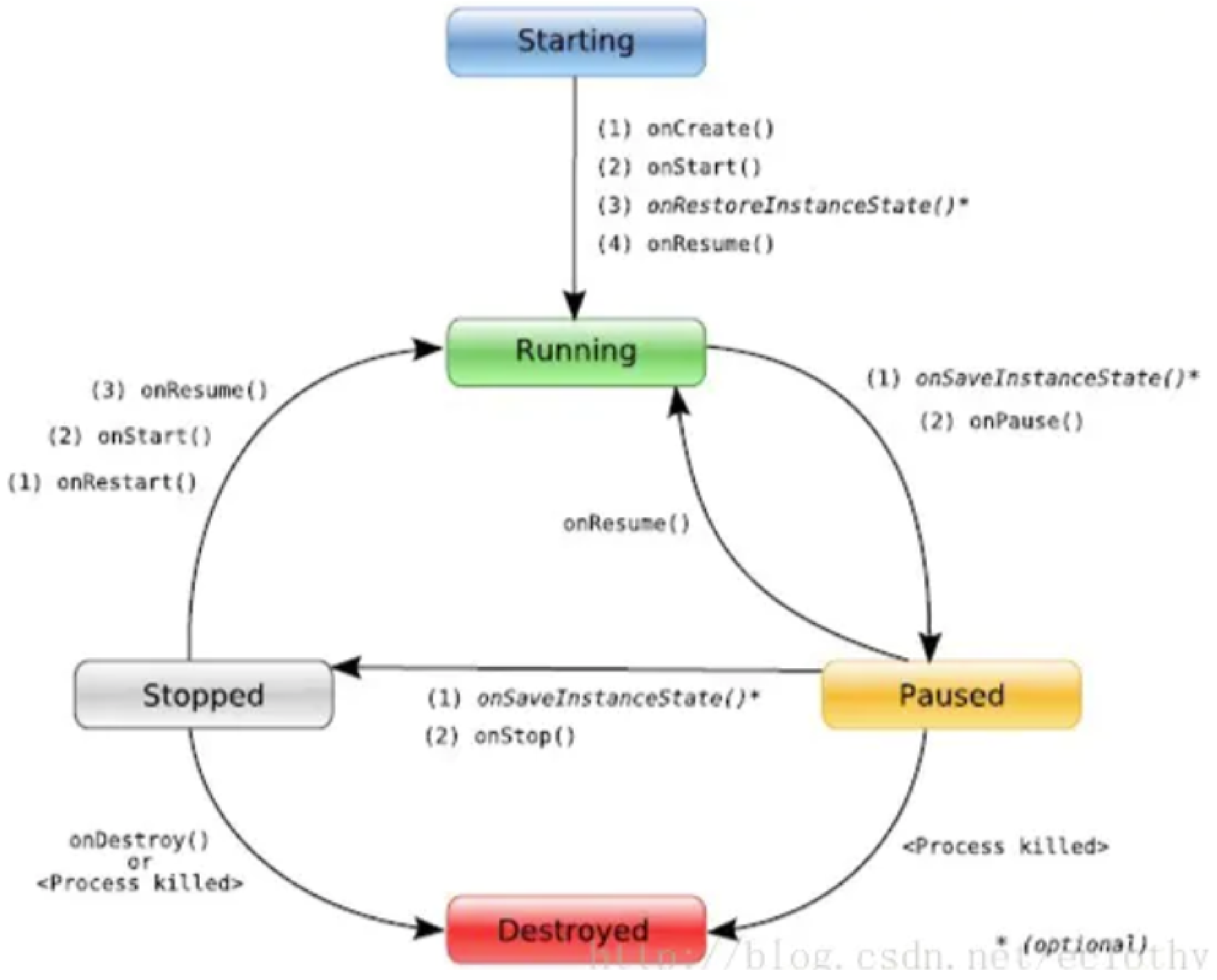
1.1.1 在这里着重讲解一下 **onStart** 与 **onResume**，**onPause** 与 **onStop** 区别

onStart 与 onResume 两种状态虽都可见，但 onStart 时还无法与用户交互，并未获得焦点。onResume 时页面已获得焦点，可与用户交互；onPause 时页面还在前台，只不过页面已失去焦点，无法与用户交互了，onStop 时已不可见了。



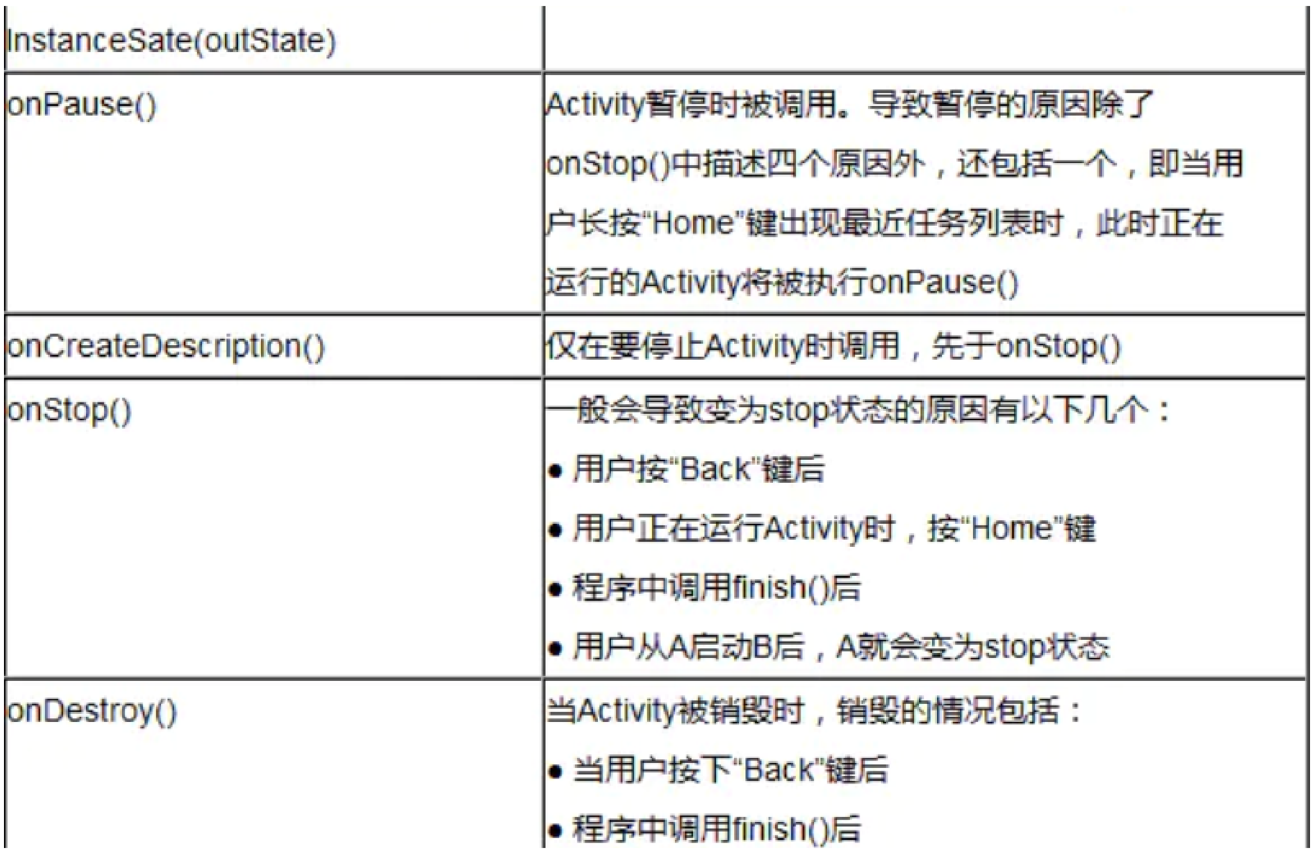
activity 四个状态所在的生命周期：

- Running 状态：一个新的 Activity 启动入栈后，它在屏幕最前端，处于栈的最顶端，此时它处于可见并可和用户交互的激活状态。
- Paused 状态：依旧在用户可见状态，但是界面焦点已经失去，此 Activity 无法与用户进行交互。当 Activity 被另一个透明或者 Dialog 样式的 Activity 覆盖时的状态。此时它依然与窗口管理器保持连接，系统继续维护其内部状态，它仍然可见，但它已经失去了焦点，故不可与用户交互。所以就解释为什么启动一个 dialogActivity 或者透明 Activity 时，原 Activity 只执行了 onPause 生命周期，并未执行 onStop
- Stopped 状态：用户看不到当前界面, 也无法与用户进行交互完全被覆盖。当 Activity 不可见时，Activity 处于 Stopped 状态。当 Activity 处于此状态时，一定要保存当前数据和当前的 UI 状态，否则一旦 Activity 退出或关闭时，当前的数据和 UI 状态就丢失了。
- Killed 状态：Activity 被杀掉以后或者被启动以前，处于 Killed 状态。这是 Activity 已从 Activity 堆栈中移除，需要重新启动才可以显示和使用。
- 4 种状态中，Running 状态和 Paused 状态是可见的，Stopped 状态和 Killed 状态时不可见的。

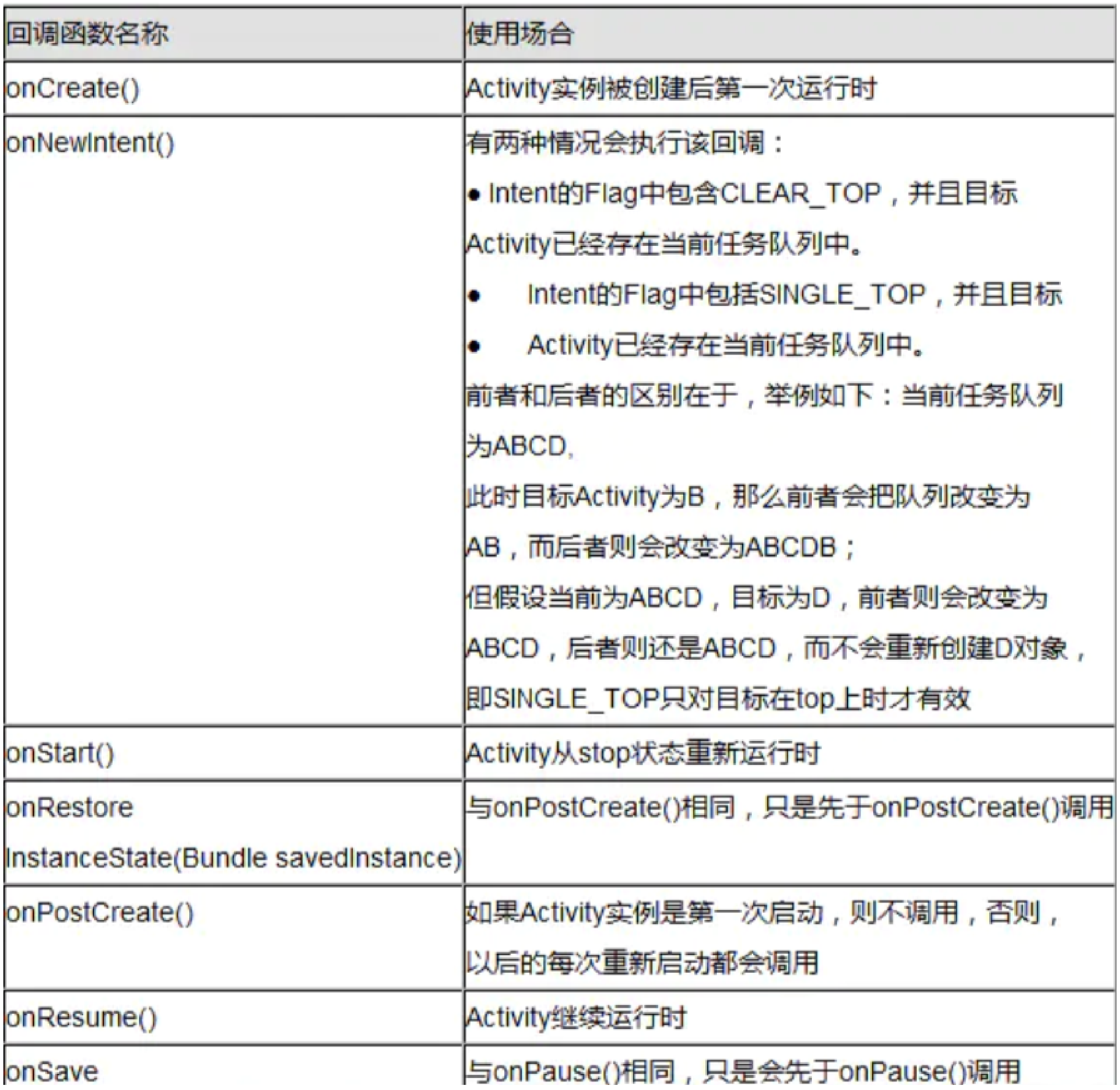


1.2 Activity 注意事项

- Activity 中所有和状态相关的回调函数：



InstanceSate(outState)	
onPause()	Activity暂停时被调用。导致暂停的原因除了onStop()中描述四个原因外，还包括一个，即当用户长按"Home"键出现最近任务列表时，此时正在运行的Activity将被执行onPause()
onCreateDescription()	仅在要停止Activity时调用，先于onStop()
onStop()	一般会导致变为stop状态的原因有以下几个： • 用户按"Back"键后 • 用户正在运行Activity时，按"Home"键 • 程序中调用finish()后 • 用户从A启动B后，A就会变为stop状态
onDestroy()	当Activity被销毁时，销毁的情况包括： • 当用户按下"Back"键后 • 程序中调用finish()后



回调函数名称	使用场合
onCreate()	Activity实例被创建后第一次运行时
onNewIntent()	有两种情况会执行该回调： ● Intent的Flag中包含CLEAR_TOP，并且目标 Activity已经存在当前任务队列中。 ● Intent的Flag中包括SINGLE_TOP，并且目标 ● Activity已经存在当前任务队列中。 前者和后者的区别在于，举例如下：当前任务队列 为ABCD, 此时目标Activity为B，那么前者会把队列改变为 AB，而后者则会改变为ABCDB； 但假设当前为ABCD，目标为D，前者则会改变为 ABCD，后者则还是ABCD，而不会重新创建D对象， 即SINGLE_TOP只对目标在top上时才有效
onStart()	Activity从stop状态重新运行时
onRestore InstanceState(Bundle savedInstance)	与onPostCreate()相同，只是先于onPostCreate()调用
onPostCreate()	如果Activity实例是第一次启动，则不调用，否则， 以后的每次重新启动都会调用
onResume()	Activity继续运行时
onSave	与onPause()相同，只是会先于onPause()调用

- 在这里我会特别提出一个 point，就是异常情况下 activity 被杀死，而后被重新创建的情况。
- 这张图非常重要，可以帮我们解决异常情况下 activity 如何正常回复的问题
- 当系统停止 activity 时，它会调用 onSaveInstanceState()(过程 1)，如果 activity 被销毁了，但是需要创建同样的实例，系统会把过程 1 中的状态数据传给 onCreate() 和 onRestoreInstanceState()，所以我们要在 onSaveInstanceState() 内做保存参数的动作，在 onRestoreInstanceState() 做获取参数的动作。

```java
// Save Activity State
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore); // Save the user's current game state
    savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);
    super.onSaveInstanceState(savedInstanceState); // Always call the superclass so it can save the view hierarchy state
}
```

- 获取参数操作：

```java
// onCreate() 方法
@Override
```

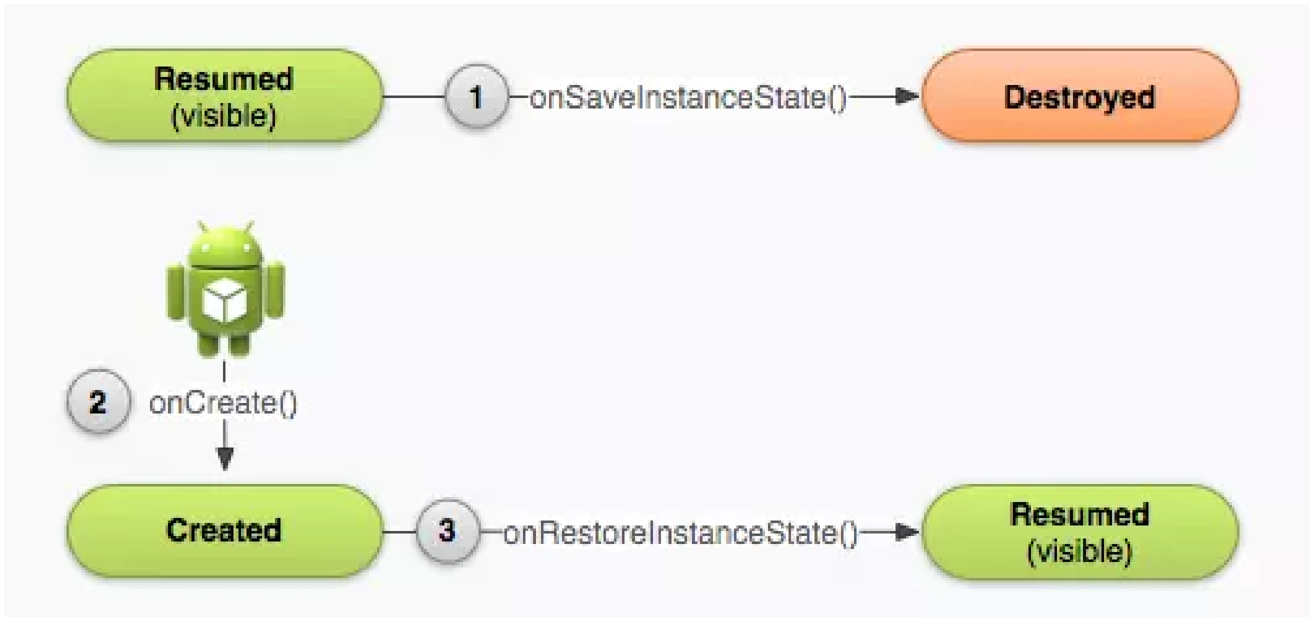


图 1.1: 异常情况下 activity 的重建过程

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Always call the superclass first
    if (savedInstanceState != null) { // Check whether we're recreating a previously destroyed instance
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE); // Restore value of members from saved state
        mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
    } else  // Probably initialize members with default values for a new instance
}
```

- 也可以

```
// onRestoreInstanceState() 方法
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState); // Always call the superclass so it can restore the view hierarchy
    mCurrentScore = savedInstanceState.getInt(STATE_SCORE); // Restore state members from saved instance
    mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
}
```

1.3 一些特殊的方法

1.3.1 onWindowFocusChanged()

- 在 Activity 窗口获得或失去焦点时被调用并且当 Activity 被创建时是在 onResume 之后被调用，当 Activity 被覆盖或者退居后台或者当前 Activity 退出时，它是在 onPause 之后被调用（在这个方法中可以 view 已经绘制完成，可以获取 view 的宽高等属性）

1.3.2 onSaveInstanceState()

- (1) 在 Activity 被覆盖或退居后台之后，系统资源不足将其杀死，此方法会被调用；
- (2) 在用户改变屏幕方向时，此方法会被调用；
- (3) 在当前 Activity 跳转到其他 Activity 或者按 Home 键回到主屏，自身退居后台时，此方法会被调用。
- 第一种情况我们无法保证什么时候发生，系统根据资源紧张程度去调度；
- 第二种是屏幕翻转方向时，系统先销毁当前的 Activity，然后再重建一个新的，调用此方法时，我们可以保存一些临时数据；
- 第三种情况系统调用此方法是为了保存当前窗口各个 View 组件的状态。

- onSaveInstanceState 的调用顺序是在 onstop 之前。（android3.0 之前：在 onPause 之前调用，在 3.0 之后，在 onPause 之后调用）

1.3.3 onRestoreInstanceState()

- 有的人说这个方法和 onSaveInstanceState 是一对，其实不然，
- (1) 在 Activity 被覆盖或退居后台之后，系统资源不足将其杀死，然后用户又回到了此 Activity，此方法会被调用；
- (2) 在用户改变屏幕方向时，重建的过程中，此方法会被调用。
- 我们可以重写此方法，以便可以恢复一些临时数据。
- onRestoreInstanceState 的调用顺序是在 onStart 之后。
- 在当前 Activity 跳转到其他 Activity 或者按 Home 键回到主屏，自身退居后台时：onRestoreInstanceState 不会调用，但是 onSaveInstanceState 会调用，这点就是区别

1.3.4 注意一下两个函数在生命周期中的调用顺序

```
onCreate()
onStart()
onSaveInstanceState()
onResume()
onPause()
onRestoreInstanceState()
onStop()
onDestory()
```

1.4 最后的总结

- 当 Activity 被系统撤销后重新建立时，保存以及恢复数据的函数调用顺序是：
  - onSaveInstanceState(保存数据) —-> onCreate(恢复数据 allstate) —-> onRestoryInstanceState(恢复数据 HierarchyState)
- 如果要取消切换屏幕方法重建 activity，可以配置 configChanges 属性：当支持的最小 sdk 版本大于 android4.0 需要设置这个属性）

```
android:configChanges="keyboardHidden|orientation|screenSize（当支持的最小 sdk 版本大于 android4.0 需要设置这个属性）"
```

1.5 他人总结

- 横竖屏切换时 Activity 的生命周期

1、不设置 Activity 的 android:configChanges 时，切屏会重新调用各个生命周期默认首先销毁当前 activity，然后重新加载。如下图，当横竖屏切换时先执行 onPause/onStop 方法

```
onPause()
onStop()
onCreate()
onStart()
onResume()
```

2、设置 Activity 的 android:configChanges="orientation|keyboardHidden|screenSize" 时，切屏不会重新调用各个生命周期，只会执行 onConfigurationChanged 方法。

- https://juejin.im/post/5a18f58651882531bb6c82e2
- 1. 打开一个全新的 activityA：
  - onCreate() —-> onStart() —-> onResume()

- 2. 从 activity A —-> activity B（全屏）：
  - activity A 先执行: onPause()
  - 然后 activity B 执行: onCreate() —-> onStart() —-> onResume()
  - activity A 再执行: onStop()
- 3. 从 activity A —-> activity B（非全屏）：
  - activity A 先执行: onPause()
  - 然后 activity B 执行: onCreate() —-> onStart() —-> onResume()
  - **activity A** 不会执行 **onStop()**
- 4. activity B（全屏）返回到 activity A：
  - activity B 先执行: onPause()
  - activity A: onRestart —-> onStart() —-> onResume()
  - activity B 再执行: onStop() —-> onDestory()
- 5. activity B（非全屏）返回到 activity A
  - activity B 先执行: onPause()
  - activity A: onResume()
  - activity B 再执行: onStop() —-> onDestory()
- 6. activity B 返回到 activity A：
  - 如果 activityA 已经被销毁，activityA 会重新创建，执行: onCreate() —-> onStart() —-> onResume()
  - activityB 的流程不变
- 7. activity A 按 home 键退居后台：
  - 同 2 的流程: onPause()
- 8. 再从 home 返回到 activity A
  - 同 4 的流程: onRestart —-> onStart() —-> onResume()

1.6 activity 之间的数据传递：6 种方式

1.6.1 使用 Inten 的 putExtra 传递

- 第一个 Activity 中

```
Intent intent = new Intent(this,TwoActivity.class);
intent.putExtra("data",str);
startActivity(intent);
```

- 第二个 Activity

```
Intent intent = getIntent();
String str = intent.getStringExtra("data");
tv.setText(str);
```

1.6.2 使用 Intention 的 Bundle 传递

- 第一个 Activity 中

```
Intent intent = new Intent(MainActivity.this,TwoActivity.class);
Bundle bundle = new Bundle();
bundle.putString("data", str);
intent.putExtra("bun", bundle);
startActivity(intent);
```

- 第二个 Activity

```
Intent intent = getIntent();
Bundle bundle = intent.getBundleExtra("bun");
String str = bundle.getString("data");
tv.setText(str);
```

1.6.3 使用 Activity 销毁时传递数据

- 第一个 Activity 中

```
Intent intent = new Intent(MainActivity.this,TwoActivity.class);
startActivityForResult(intent, 11);
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    String str = data.getStringExtra("data");
    tvOne.setText(str);
}
```

- 第二个 Activity

```
Intent intent = new Intent();
intent.putExtra("data", edtOne.getText().toString().trim());
setResult(3, intent);
finish();
```

1.6.4 SharedPreferences 传递数据

- 第一个 Activity 中

```
SharedPreferences sp = this.getSharedPreferences("info", 1);
Editor edit = sp.edit();
edit.putString("data", str);
edit.commit();
Intent intent = new Intent(MainActivity.this,TwoActivity.class);
startActivity(intent);
```

- 第二个 Activity

```
SharedPreferences sp = this.getSharedPreferences("info", 1);
tv.setText(sp.getString("data", ""));
```

1.6.5 使用序列化对象 Seriazable

- 这里需要建一个工具类

```
import java.io.Serializable;
class DataBean implements Serializable {
    private String name;
    private String sex;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
}
```

- 第一个 Activity 中

```
Intent intent = new Intent(MainActivity.this,TwoActivity.class);
DataBean bean = new DataBean();
bean.setName(" 啦啦");
bean.setSex(" 男");
intent.putExtra("key", bean);
startActivity(intent);
```

- 第二个 Activity

```
Intent intent = getIntent();
//反序列化数据对象
Serializable se = intent.getSerializableExtra("key");
if(se instanceof DataBean){
    //获取到携带数据的 DataBean 对象 db
    DataBean db = (DataBean) se;
    tv.setText(db.getName()+"==="+db.getSex());
}
```

1.6.6 使用静态变量传递数据

- 第一个 Activity 中

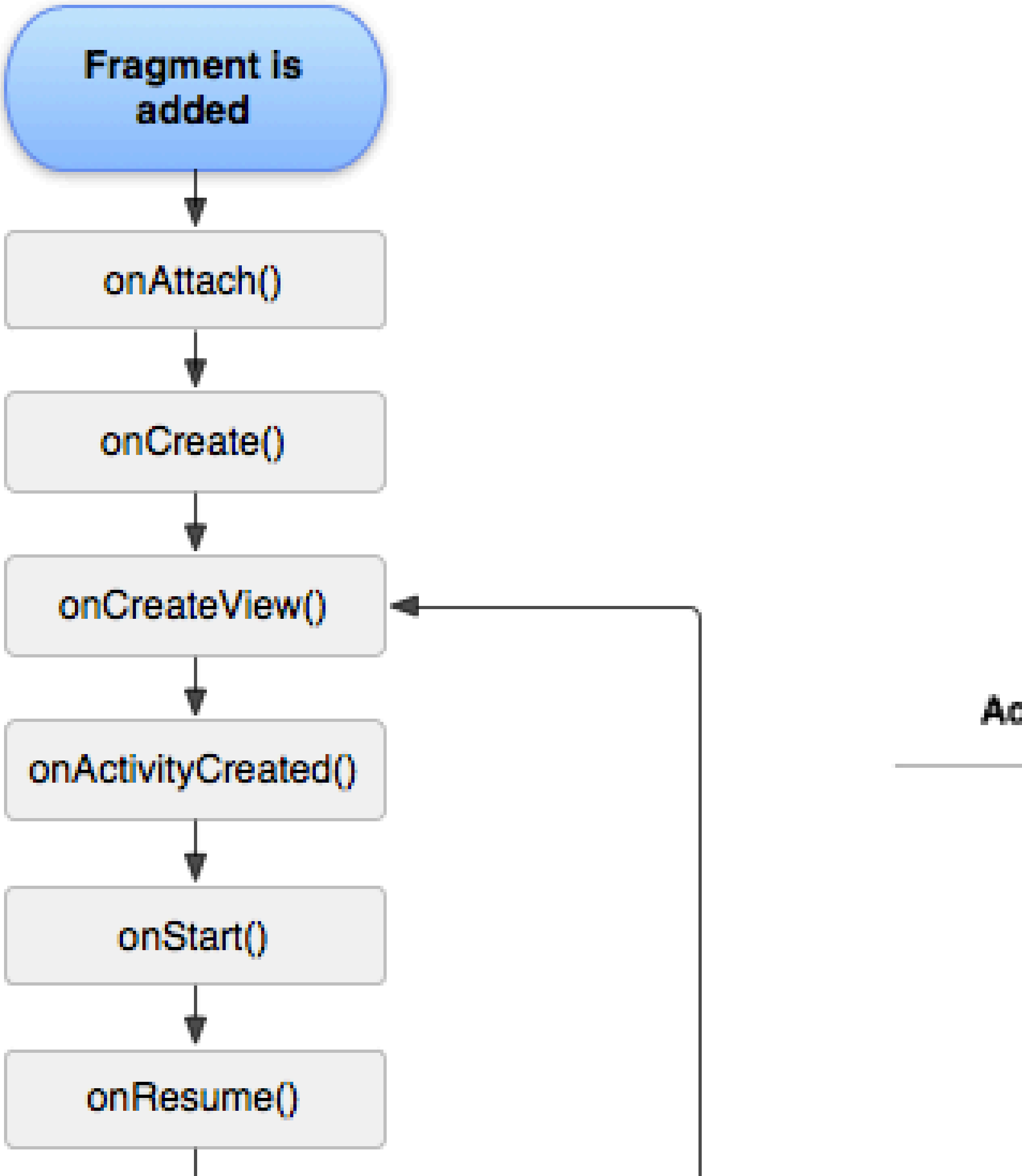
```
Intent intent = new Intent(MainActivity.this,TwoActivity.class);
TwoActivity.name=" 牛逼";
TwoActivity.str=" 你说";
startActivity(intent);
```

- 第二个 Activity

```
protected static String name;
protected static String str;
tv.setText(str+name);
```

Chapter 2

Fragment 生命周期探讨



- Fragment 基本类，生命周期如下：

```
void onAttach(Context context)
void onCreate(Bundle savedInstanceState)
View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
void onActivityCreated(Bundle savedInstanceState)
void onStart()
void onResume()
void onPause()
void onStop()
void onDestroyView()
void onDestroy()
void onDetach()
```

2.1 首先需要提出的一些 points:

- Fragment 是直接从 Object 继承的，而 Activity 是 Context 的子类。因此我们可以得出结论：Fragment 不是 Activity 的扩展。但是与 Activity 一样，在我们使用 Fragment 的时候我们总会扩展 Fragment(或者是她的子类)，并可以通过子类更改她的行为。
- 使用 Fragment 时，必要构建一个无参构造函数，系统会默认带。但一但写有参构造函数，就必要构建无参构造函数。一般来说我们传参数给 Fragment，会通过 bundle，而不会用构造方法传，代码如下：

```
public static MyFragment newInstance(int index){
    MyFragment mf = new MyFragment();
    Bundle args = new Bundle();
    args.putInt("index",index);
    mf.setArguments(args);
    return mf;
}
```

2.2 生命周期

- onAttach()：onAttach() 回调将在 Fragment 与其 Activity 关联之后调用。需要使用 Activity 的引用或者使用 Activity 作为其他操作的上下文，将在此回调方法中实现。
  - 将 Fragment 附加到 Activity 以后，就无法再次调用 setArguments()--除了在最开始，无法向初始化参数添加内容。
- onCreate(Bundle savedInstanceState)：此时的 Fragment 的 onCreate() 回调时，该 fragmet 还没有获得 Activity 的 onCreate() 已完成的通知，所以不能将依赖于 Activity 视图层次结构存在性的代码放入此回调方法中。在 onCreate() 回调方法中，我们应该尽量避免耗时操作。此时的 bundle 就可以获取到 activity 传来的参数

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Bundle args = getArguments();
        if (args != null) {
            mLabel = args.getCharSequence("label", mLabel);
        }
    }
```

- onCreateView()

```
onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
```

  - 其中的 Bundle 为状态包与上面的 bundle 不一样。
  - 不要将视图层次结构附加到传入的 ViewGroup 父元素中，该关联会自动完成。如果在此回调中将碎片的视图层次结构附加到父元素，很可能会出现异常。
  - 这句话什么意思呢？就是不要把初始化的 view 视图主动添加到 container 里面，以为这会系统自带，所以 inflate 函数的第三个参数必须填 false，而且不能出现 container.addView(v) 的操作。

```
View v = inflater.inflate(R.layout.hello_world, container, false);
```

- onActivityCreated()
  - onActivityCreated() 回调会在 Activity 完成其 onCreate() 回调之后调用。在调用 onActivityCreated() 之前，Activity 的视图层次结构已经准备好了，这是在用户看到用户界面之前你可对用户界面执行的最后调整的地方。
  - 如果 Activity 和她的 Fragment 是从保存的状态重新创建的，此回调尤其重要，也可以在这里确保此 Activity 的其他所有 Fragment 已经附加到该 Activity 中了
- Fragment 与 Activity 相同生命周期调用：接下来的 onStart(), onResume(), onPause(), onStop() 回调方法将和 Activity 的回调方法进行绑定，也就是说与 Activity 中对应的生命周期相同，因此不做过多介绍。
- onDestroyView(): 该回调方法在视图层次结构与 Fragment 分离之后调用。
- onDestroy()：不再使用 Fragment 时调用。(备注：Fragment 仍然附加到 Activity 并任然可以找到，但是不能执行其他操作)
- onDetach()：Fragment 生命周期最后回调函数，调用后，Fragment 不再与 Activity 绑定，释放资源。

2.3 Fragment 每个生命周期方法的意义、作用

- onAttach()
  - 执行该方法时，Fragment 与 Activity 已经完成绑定，该方法有一个 Activity 类型的参数，代表绑定的 Activity，这时候你可以执行诸如 mActivity = activity 的操作。
- onCreate()
  - 初始化 Fragment。可通过参数 savedInstanceState 获取之前保存的值。
- onCreateView()
  - 初始化 Fragment 的布局。加载布局和 findViewById 的操作通常在此函数内完成，但是不建议执行耗时的操作，比如读取数据库数据列表。
- onActivityCreated()
  - 执行该方法时，与 Fragment 绑定的 Activity 的 onCreate 方法已经执行完成并返回，在该方法内可以进行与 Activity 交互的 UI 操作，所以在该方法之前 Activity 的 onCreate 方法并未执行完成，如果提前进行交互操作，会引发空指针异常。
- onStart()
  - 执行该方法时，Fragment 由不可见变为可见状态。
- onResume()
  - 执行该方法时，Fragment 处于活动状态，用户可与之交互。
- onPause()
  - 执行该方法时，Fragment 处于暂停状态，但依然可见，用户不能与之交互。
- onSaveInstanceState()
  - 保存当前 Fragment 的状态。该方法会自动保存 Fragment 的状态，比如 EditText 键入的文本，即使 Fragment 被回收又重新创建，一样能恢复 EditText 之前键入的文本。
- onStop()
  - 执行该方法时，Fragment 完全不可见。
- onDestroyView()
  - 销毁与 Fragment 有关的视图，但未与 Activity 解除绑定，依然可以通过 onCreateView 方法重新创建视图。通常在 ViewPager+Fragment 的方式下会调用此方法。
- onDestroy()

  - 销毁 Fragment。通常按 Back 键退出或者 Fragment 被回收时调用此方法。

- onDetach()
  - 解除与 Activity 的绑定。在 onDestroy 方法之后调用。

- setUserVisibleHint()
  - 设置 Fragment 可见或者不可见时会调用此方法。在该方法里面可以通过调用 getUserVisibleHint() 获得 Fragment 的状态是可见还是不可见的，如果可见则进行懒加载操作。

2.4 Fragment 生命周期执行流程

- 1、Fragment 创建
  - setUserVisibleHint() —-> onAttach() —-> onCreate() —-> onCreateView() —-> onActivityCreated() —-> onStart() —-> onResume()
- 2、Fragment 变为不可见状态（锁屏、回到桌面、被 Activity 完全覆盖）
  - onPause() —-> onSaveInstanceState() —-> onStop()
- 3、Fragment 变为部分可见状态（打开 Dialog 样式的 Activity）
  - onPause() —-> onSaveInstanceState()
- 4、Fragment 由不可见变为活动状态
  - onStart() —-> OnResume()
- 5、Fragment 由部分可见变为活动状态
  - onResume()
- 6、Fragment 退出
  - onPause() —-> onStop() —-> onDestroyView() —-> onDestroy() —-> onDetach()
  - （注意退出不会调用 onSaveInstanceState 方法，因为是人为退出，没有必要再保存数据）
- 7、Fragment 被回收又重新创建
  - 被回收执行: onPause() —-> onSaveInstanceState() —-> onStop() —-> onDestroyView() —-> onDestroy() —-> onDetach()
  - 重新创建执行：onAttach() —-> onCreate() —-> onCreateView() —-> onActivityCreated() —-> onStart() —-> onResume() —-> setUserVisibleHint()
- 横竖屏切换
  - 与 Fragment 被回收又重新创建一样。

2.5 onHiddenChanged 的回调时机

- 当使用 add()+show()，hide() 跳转新的 Fragment 时，旧的 Fragment 回调 onHiddenChanged()，不会回调 onStop() 等生命周期方法，而新的 Fragment 在创建时是不会回调 onHiddenChanged()，这点要切记。

2.6 FragmentPagerAdapter+ViewPager 的注意事项

- 1、使用 FragmentPagerAdapter+ViewPager 时，切换回上一个 Fragment 页面时（已经初始化完毕），不会回调任何生命周期方法以及 onHiddenChanged()，只有 setUserVisibleHint(boolean isVisibleToUser) 会被回调，所以如果你想进行一些懒加载，需要在这里处理。
- 2、在给 ViewPager 绑定 FragmentPagerAdapter 时，new FragmentPagerAdapter(fragmentManager) 的 FragmentManager，一定要保证正确，如果 ViewPager 是 Activity 内的控件，则传递 getSupportFragmentManager()，如果是 Fragment 的控件中，则应该传递 getChildFragmentManager()。只要记住 ViewPager 内的 Fragments 是当前组件的子 Fragment 这个原则即可。
- 3、你不需要考虑在"内存重启"的情况下，去恢复的 Fragments 的问题，因为 FragmentPagerAdapter 已经帮我们处理啦。

2.7 setUserVisibleHint() 不调用的问题

- 通常情况下都是因为 PagerAdapter 不是 FragmentPagerAdapter 造成的，FragmentPagerAdapter 内部实现了对 setUserVisibleHint() 方法的调用，所以需要懒加载的结构最好使用 FragmentPagerAdapter +Fragment 的结构，少用 PagerAdapter。

2.8 Fragment 注意事项

- 在使用 Fragment 时，我发现了一个金矿，那就是 setRetainInstance() 方法, 此方法可以有效地提高系统的运行效率，对流畅性要求较高的应用可以适当采用此方法进行设置。
- Fragment 有一个非常强大的功能--就是可以在 Activity 重新创建时可以不完全销毁 Fragment，以便 Fragment 可以恢复。在 onCreate() 方法中调用 setRetainInstance(true/false) 方法是最佳位置。当 Fragment 恢复时的生命周期如上图所示，注意图中的红色箭头。当在 onCreate() 方法中调用了 setRetainInstance(true) 后，Fragment 恢复时会跳过 onCreate() 和 onDestroy() 方法，因此不能在 onCreate() 中放置一些初始化逻辑.

Chapter 3

Android Fragment 生命周期图

- http://www.cnblogs.com/purediy/p/3276545.html
- Fragment 与 Activity 生命周期对比图：

3.1 生命周期分析

3.1.1 当一个 fragment 被创建的时候 (它会经历以下状态)

- onAttach()
- onCreate()
- onCreateView()
- onActivityCreated()

3.1.2 当这个 fragment 对用户可见的时候

- onStart()
- onResume()

3.1.3 当这个 fragment 进入“后台模式”的时候

- onPause()
- onStop()

3.1.4 当这个 fragment 被销毁了（或者持有它的 activity 被销毁了）

- onPause()
- onStop()
- onDestroyView()
- onDestroy() // 本来漏掉类这个回调，感谢 xiangxue336 提出。
- onDetach()

3.1.5 就像 activity 一样，在以下的状态中，可以使用 Bundle 对象保存一个 fragment 的对象。

- onCreate()
- onCreateView()
- onActivityCreated()

3.1.6 **fragments** 的大部分状态都和 **activity** 很相似，但 **fragment** 有一些新的状态。

- onAttached() – 当 fragment 被加入到 activity 时调用（在这个方法中可以获得所在的 activity）。
- onCreateView() – 当 activity 要得到 fragment 的 layout 时，调用此方法，fragment 在其中创建自己的 layout(界面)。
- onActivityCreated() – 当 activity 的 onCreated() 方法返回后调用此方法
- onDestroyView() – 当 fragment 中的视图被移除的时候，调用这个方法。
- onDetach() – 当 fragment 和 activity 分离的时候，调用这个方法。
- Notes:
  - 一旦 activity 进入 resumed 状态（也就是 running 状态），你就可以自由地添加和删除 fragment 了。
  - 因此，只有当 activity 在 resumed 状态时，fragment 的生命周期才能独立的运转，其它时候是依赖于 activity 的生命周期变化的。

Chapter 4

activity lifecycle

4.0.1 Activity 横竖屏切换生命周期变化

1. 新建一个 Activity，并把各个生命周期打印出来 onCreate, 创建 activity 时调用。设置在该方法中，还以 Bundle 中可以提出用于创建该 Activity 所需的信息。onStart, activity 变为在屏幕上对用户可见时，即获得焦点时，会调用。onResume, activity 开始与用户交互时调用（无论是启动还是重新启动一个活动，该方法总是被调用的）onSaveInstanceState onPause, activity 被暂停或收回 cpu 和其他资源时调用，该方法用于保存活动状态的 onStop, activity 被停止并转为不可见阶段及后续的生命周期事件时，即失去焦点时调用 onDestroy, activity 被完全从系统内存中移除时调用，该方法被调用可能是因为有人直接调用 finish() 方法或者系统决定停止该活动以释放资源。onRestoreInstanceState, Android 在横竖排切换时候，将主动销毁 activity 和重新创建一个新的 activity 出来，在此过程中，onRestoreInstanceState 就要被回调 onConfigurationChanged, 配置指定属性后, 屏幕方向发生变化后回调此函数.

2. 运行 Activity，得到如下信息

```
onCreate  -->
onStart  -->
onResume  -->
```

3. 按 crtl+f12 切换成横屏时

```
onPause  -->
onStop  -->
onDestroy  -->
onCreate  -->
onStart  -->
onRestoreInstanceState  -->
onResume  -->
```

4. 再按 crtl+f12 切换成竖屏时，发现又打印了相同的 log

```
onPause  -->
onStop  -->
onDestroy  -->
onCreate  -->
onStart  -->
onRestoreInstanceState  -->
onResume  -->
```

5. 修改 AndroidManifest.xml 把该 Activity 添加

```
android:configChanges="orientation",
```

执行步骤 3(切换成横屏时)

```
onPause  -->
onStop  -->
onDestroy  -->
onCreate  -->
onStart  -->
onRestoreInstanceState  -->
onResume  -->
```

6. 再执行步骤 4(切换竖屏)，发现再打印相同信息

```
onPause  -->
onStop  -->
onDestroy  -->
onCreate  -->
onStart  -->
onRestoreInstanceState  -->
onResume  -->
```

4.0.2 Why do developers often put app initialization code in the Application class?

- The Application class is instantiated before any other class when the process for the application is created.

Chapter 5

fragmnet

5.0.1 What are Retained Fragments?

- By default, Fragments are destroyed and recreated along with their parent Activity’s when a configuration change occurs.
- Calling setRetainInstance(true) allows us to bypass this destroy-and-recreate cycle, signaling the system to retain the current instance of the fragment when the activity is recreated.

5.0.2 How would you communicate between two Fragments?

All Fragment-to-Fragment communication is done either through a shared ViewModel or through the associated Activity. Two Fragments should never communicate directly.

- The recommended way to communicate between fragments is to create a shared ViewModel object. Both fragments can access the ViewModel through their containing Activity. The Fragments can update data within the ViewModel and if the data is exposed using LiveData the new state will be pushed to the other fragment as long as it is observing the LiveData from the ViewModel.

```java
public class SharedViewModel extends ViewModel {
    private final MutableLiveData <Item> selected = new MutableLiveData < Item > ();
    public void select(Item item) {
        selected.setValue(item);
    }
    public LiveData <Item> getSelected() {
        return selected;
    }
}
public class MasterFragment extends Fragment {
    private SharedViewModel model;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        model = ViewModelProviders.of(getActivity()).get(SharedViewModel.class);
        itemSelector.setOnClickListener(item -> {
                model.select(item);
            });
    }
}
public class DetailFragment extends Fragment {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SharedViewModel model = ViewModelProviders.of(getActivity()).get(SharedViewModel.class);
        model.getSelected().observe(this, {
                item ->
                    // Update the UI.
                    // model.select(item); // 这行充当占位符，复制的上面的
                    });
    }
}
```

- Another way is to define an interface in your Fragment A, and let your Activity implement that Interface. Now you can call the interface method in your Fragment, and your Activity will receive the event. Now in your activity, you can call your second Fragment to update the textview with the received value.

5.1 transaction stack backstack

5.1.1 What is the chief purpose of line five in this code snippet?

```
override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_post_create)
^^Iif (savedInstanceState != null) return
^^Ival fragment = CreatePostFragment()
^^I^^IsupportFragmentManager
^^I^^I.beginTransaction()
^^I^^I.add(R.id. fragment_container, fragment)
^^I^^I.commit()
}
```

- to make sure that the activity creates a new fragment each time it is restored from a previous state

Chapter 6

View

6.1 What should you use to display a large, scrolling list of elements?

- Recycler View

6.2 Given the fragment below, how would you get access to a TextView with an ID of text$_{home}$ contained in the layout file of a Fragment class?

```
    private lateinit var textView: TextView
    override fun onCreateView(...): View? {
        val root = inflator.inflator(R>layout.fragment_home, container, false)
        textView = ??
        return root
    }
// root.findViewById(R.id.text_home)
```

Chapter 7

Intent

7.1 You have created a NextActivity class that relies on a string containing some data that pass inside the intent Which code snippet allows you to launch your activity?

```
Intent(this, NextActivity::class.java).apply {
    putExtra(EXTRA_NEXT, "some data")
}.also { intent ->
    startActivity(intent)
}
```

7.2 You have created an AboutActivity class that displays details about your app. Which code snippet allows you to launch your activity?

```
Intent(this, AboutActivity::class.java).also {
    intent -> startActivity(intent)
}
```

7.3 Which definition will prevent other apps from accessing your Activity class via an intent?

```
<activity android:name=".ExampleActivity" />
```

- Intent filters are used to make activities accessible to other apps using intents. So we have to choose option which have no intent filter to make sure it is not accessible by intent

7.4 You want to allow users to take pictures in your app. Which is not an advantage of creating an appropriate intent, instead of requesting the camera permission directly?

- Users can select their favorite photo apps to take pictures.
- You do not have to make a permission request in your app to take a picture.
- You do not have to design the UI. The app that handles the camera intent will provide the UI.
- You have full control over the user experience. The app that handles the camera intent will respect your design choices. (ANSWER)

7.5 onActivityResult()

- When will an activity's onActivityResult()be called?
  - when calling finish() in the target activity

7.6 startActivityWithResult(): You want to open the default Dialer app on a device. What is wrong with this code?

```
val dialerIntent = Intent()
val et = findViewById(R.id.some_edit_text)
dialerIntent.action = Intent.ACTION_DIAL
dialerIntent.data = Uri.parse("tel:" + et.getText()?.toString())
startActivity(dialerIntent) // <--
```

- startActivityWithResult() should be used instead of startActivity() when using Intent.ACTION$_{DIAL}$.

Chapter 8

Data

8.1 storage

8.1.1 To persist a small collection of key-value data, what should you use?

- SharedPereferences

8.1.2 What allows you to properly restore a user's state when an activity is restarted?

- the onSaveInstance()method
- persistent storage
- ViewModel objects
- all of these answers (Refrence) (ANSWER)

8.1.3 To preserve on-device memory, how might you determine that the user's device has limited storage capabilities?

```
ActivityManager.isLowRamDevice() ;
```

- Use the ActivityManager.isLowRamDevice() method to find out whether a device defines itself as "low RAM."

8.2 How would you retrieve the value of a user's email from SharedPreferences while ensuring that the returned value is not null?

```
  getDefaultSharedPreferances(this).getString(EMAIL,"")
```

- In Method "getDefaultSharedPrefarances(this).getString()" Second parameter is passed so that it can be returned, in case key doesn't exist. So we need to pass an empty string to be returned in case key doesn't exist.

Chapter 9

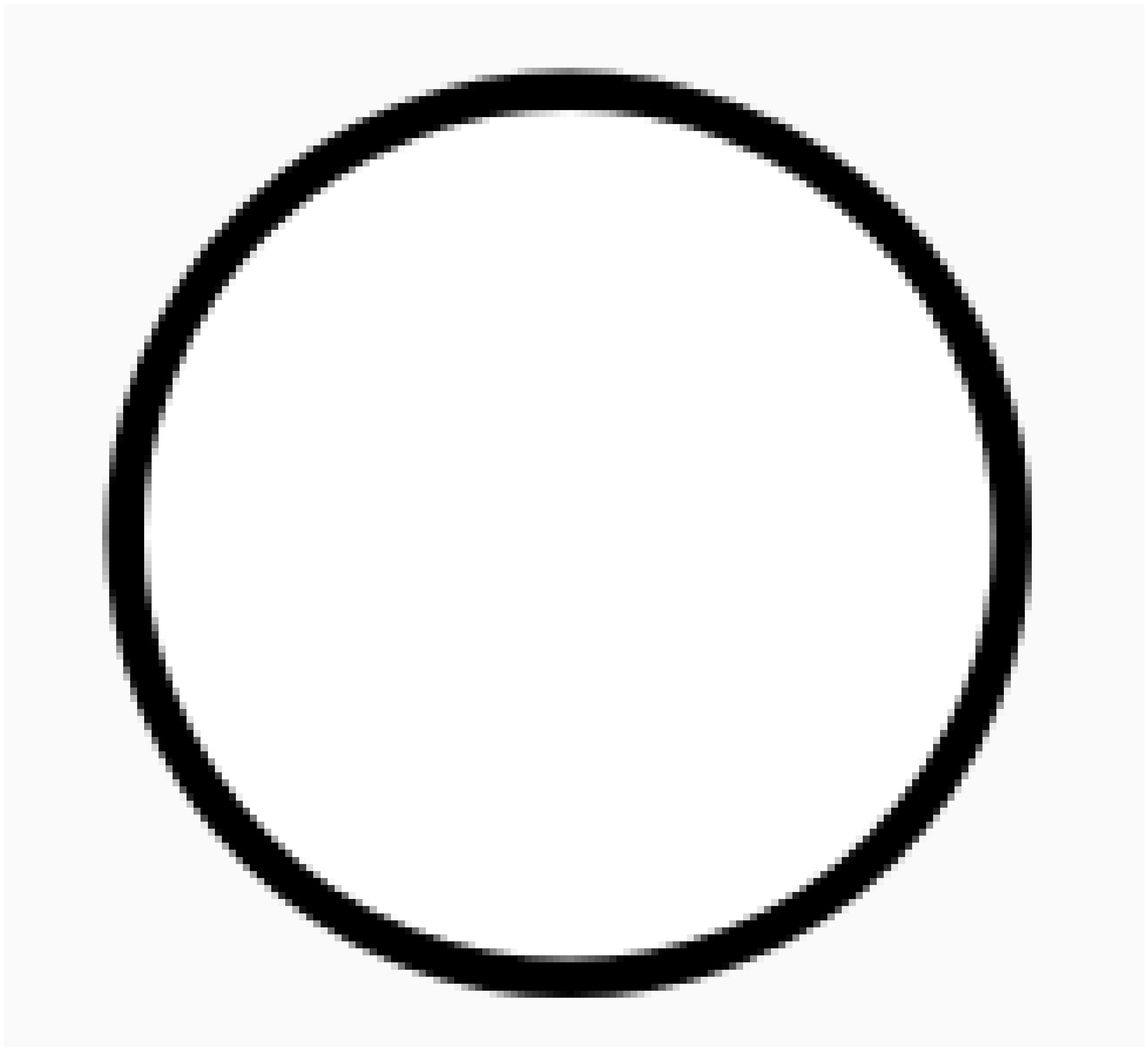
xml resource files

9.1 layout

9.1.1 Which layout is best for large, complex hierarchies?

- ConstraintLayout

9.1.2 Which drawable definition allows you to achieve the shape below?



```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="oval">
<stroke
    android:width="4dp"
    android:color="@android:color/black" />
<solid android:color="@android:color/white" />
</shape>
```

9.1.3 Which image best corresponds to the following LinearLayout?

```
  <LinearLayout
      android:layout_width="match_parent"
^^I  android:layout_height="match_parent"
^^I  android:orientation="horizontal"
^^I  android:gravity="center">
^^I<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
^^I<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
  </LinearLayout>
```

- gravity="center" 是描述的竖直方向上位于中间，而水平方向上同样也是位于中间，每个键的宽度由自身内容的宽度决定

- gravity: todo

9.1.4 Which code snippet would achieve the layout displayed below?

```
<androidx.constraintlayout.widget.ConstraintLayout
^^I...>

^^I<TextView
^^I^^Iandroid:id="@+id/text_dashboard"
^^I^^Iandroid:layout_width="match_parent"
^^I^^Iandroid:layout_height="wrap_content"
^^I^^Iandroid:layout_marginStart="8dp"
^^I^^Iandroid:layout_marginEnd="8dp"
^^I^^Iandroid:textAlignment="center"
^^I^^Iandroid:text="Dashboard"
^^I^^Iapp:layout_constraintEnd_toEndOf="parent"
^^I^^Iapp:layout_constraintStart_toStartOf="parent"
^^I^^Iapp:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

- 左右各留 8dp, 文字中间对齐；文本框高度由自身高度决定；文本框宽度拉伸 match$_{parent}$;

9.1.5 实现矩形，右下角白色，左上角黑色的渐变效果

```
  <shape xmlns:android-"http://schemas.android.com/apk/res/android"
  android:shape-"rectangle">
  <gradient
^^I   android:startColor-"@android:color/white"
^^I   android:endColor-"@android:color/black"
^^I   android:angle-"135"/>
^^I</shape>
```

9.1.6 Given the following dimens.xml file, how would you define an ImageView with medium spacing at the bottom?

```
<?xml version=1.0 encoding="utf-8"?>
<resources>
    <dimen name="spacing_medium">8dp</dimen>
    <dimen name="spacing_large">12dp</dimen>
</resources>
```

```
<ImageView
   android:id=@+id/image_map_pin"
   android:layout_width="wrap_content"
   android:layout_heignt="wrap_content"
   android:layout_marginBottom="@dimen/spacing_medium"
   android:src=@drawable/map_pin />
```

9.1.7 You want to provide a different drawable for devices that are in landscape mode and whose language is set to French. which directory is named correctly?

- drawable-fr-land

9.1.8 What folder should you use for your app's launcher icons?

- /mipmap

9.2 permissions

9.2.1 写外部存储权限

```
<uses-permission
     android:name="android.permission.WRITE_EXTERNAL_STORAGE"
     android:maxSdkVersion="18" />
```

9.2.2 When would you use the ActivityCompat.shouldShowRequestPermissionRationale() function?

```
ActivityCompat.shouldShowRequestPermissionRationale();
```

- when a user has previously denied the request for a given permission and selected "Don't ask again," but you need the permission for your app to function

9.2.3 Why might you need to include the following permission to your app?

```
android.permission.ACCESS_NETWORK_STATE
```

- to monitor the network state of the devices so that you don't attempt to make network calls when the network is unavailable

Chapter 10

annotation

10.1 @VisibleForTesting:

- to denote that a class, methos, or field has its visibility relaxed to make code testable

Chapter 11

apk

11.1 basic

11.1.1 When would you use a product flavour in your build setup?

when you want to provide different version of your app with custom configuration and resources

11.1.2 To shrink your code in release builds, what tool does Android Studio use?

- R8

11.2 build configuration

11.2.1 Which statement, in build.gradle file, correctly denotes that the corresponding module is an Android library module?

```
apply plugin: 'com.android.library'
```

11.2.2 You would like to enable analytics tracking only in release builds. How can you create a new field in the generated BuildConfig class to store that value?

```
buildTypes {
^^Idebug {
^^I^^IbuildConfigField 'boolean', 'ENABLE_ANALYTICS', 'false'
^^I}
^^Irelease {
^^I^^IbuildConfigField 'boolean', 'ENABLE_ANALYTICS', 'true'
^^I}
}
```

11.2.3 To optimize your APK size, what image codec should you use?

- WebP (Reference)

11.2.4 Given an APK named app-internal-debug.apk produced from the build process, which statement is likely to be true?

- This APK is created from the debug build type and internal product flavor.

11.3 build errors

11.3.1 When attempting to build your project, what might the following error indicate?

```
Conversion to Dalvik format filed: Unable to execute dex: method ID not in [0, 0xffff]: 65536
```

- You have exceeded the total number of methods that can be referenced within a single DEX file.

Chapter 12

testing

12.1 Why do you use the AndroidJUnitRunner when running UI tests?

- Notice: AndroidJUnitRunner lets us run JUnit3/4-style tests on Android Devices
  - The test runner facilitates loading your test package and the app under test onto a device or emulator, runs the test, and reports the results.

12.2 Given the test class below, which code snippet would be a correct assertion?

```
assertNotNull(resultAdd)
```

Chapter 13

debugging

13.1 network

13.1.1 You have built code to make a network call and tested that it works in your development environment. However, when you publish it to the Play console, the networking call fails to work. What will NOT help you troubleshoot this issue?

- checking whether ProGuard -keepclassmembers have been added to the network data transfer objects (DTOs) in question
- checking for exceptions in the server logs or server console
- checking that the network data transfer object has @SerizlizedName applied to its member properties
- using the profiler tools in Android Studio to detect anomalies in CPU, memory, and network usage (ANSWER: this does not help)

Chapter 14

Frameworks

14.1 Retrofit

14.1.1 You need to remove an Event based on it;s id from your API, Which code snippet defines that request in Retrofit?

- @DELETE("events/{id}") fun deleteEvent(@Path("id") id: Long): Call

14.1.2 You need to retrieve a list of photos from an API. Which code snippet defines an HTML GET request in Retrofit?

```
@GET("photo") fun listPhotos() : Call<List>
```

Chapter 15

需要分类出去的

15.0.1 What are the permission protection levels in Android?

- Normal —A lower-risk permission that gives requesting applications access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user’s explicit approval.

- Dangerous —A higher-risk permission. Any dangerous permissions requested by an application may be displayed to the user and require confirmation before proceeding, or some other approach may be taken to avoid the user automatically allowing the use of such facilities.

- Signature —A permission that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user’s explicit approval.

- SignatureOrSystem —A permission that the system grants only to applications that are in the Android system image or that are signed with the same certificate as the application that declared the permission.

15.0.2 What is Android Data Binding?

The Data Binding Library is a support library that allows you to bind UI components in your layouts to data sources in your app using a declarative format rather than programmatically.

Layouts are often defined in activities with code that calls UI framework methods. For example, the code below calls findViewById() to find a TextView widget and bind it to the userName property of the viewModel variable:

```
TextView textView = findViewById(R.id.sample_text);
textView.setText(viewModel.getUserName());
```

The following example shows how to use the Data Binding Library to assign text to the widget directly in the layout file. This removes the need to call any of the Java code shown above.

```
<TextView
    android:text="@{viewmodel.userName}" />
```

- The pros of using Android Data Binding:
  - Reduces boilerplate code which in turns brings
  - Less coupling
  - Stronger readability
  - Powerful, easy to implement custom attribute and custom view
  - Even faster than findViewById - The binding does a single pass on the View hierarchy, extracting the Views with IDs. This mechanism can be faster than calling findViewById for several Views.

15.0.3 What is the ViewHolder pattern? Why should we use it?

Every time when the adapter calls getView() method, the findViewById() method is also called. This is a very intensive work for the mobile CPU and so affects the performance of the application and the battery consumption increases. ViewHolder is a design pattern which can be applied as a way around repeated use of findViewById().

A ViewHolder holds the reference to the id of the view resource and calls to the resource will not be required after you "find" them: Thus performance of the application increases.

```java
private static class ViewHolder {
    final TextView text;
    final TextView timestamp;
    final ImageView icon;
    final ProgressBar progress;

    ViewHolder(TextView text, TextView timestamp, ImageView icon, ProgressBar progress) {
        this.text = text;
        this.timestamp = timestamp;
        this.icon = icon;
        this.progress = progress;
    }
}
public View getView(int position, View convertView, ViewGroup parent) {
    View view = convertView;
    if (view == null) {
        view = // inflate new view
        ViewHolder holder = createViewHolderFrom(view);
        view.setTag(holder);
    }
    ViewHolder holder = view.getTag();
    // TODO: set correct data for this list item
    // holder.icon.setImageDrawable(...)
    // holder.text.setText(...)
    // holder.timestamp.setText(...)
    // holder.progress.setProgress(...)
    return view;
}
private ViewHolder createViewHolderFrom(View view) {
    ImageView icon = (ImageView) view.findViewById(R.id.listitem_image);
    TextView text = (TextView) view.findViewById(R.id.listitem_text);
    TextView timestamp = (TextView) view.findViewById(R.id.listitem_timestamp);
    ProgressBar progress = (ProgressBar) view.findViewById(R.id.progress_spinner);
    return new ViewHolder(text, timestamp, icon, progress);
}
```

- View.setTag(Object) allows you to tell the View to hold an arbitrary object. If we use it to hold an instance of our ViewHolder after we do our findViewById(int) calls, then we can use View.getTag() on recycled views to avoid having to make the calls again and again.

15.0.4 What is the difference between Handler vs AsyncTask vs Thread?

Mid Top 113 Android Interview Questions Android 113 Answer The Handler class can be used to register to a thread and provides a simple channel to send data to this thread. A Handler allows you communicate back with the UI thread from other background thread. The AsyncTask class encapsulates the creation of a background process and the synchronization with the main thread. It also supports reporting progress of the running tasks. And a Thread is basically the core element of multithreading which a developer can use with the following disadvantage: Handle synchronization with the main thread if you post back results to the user interface No default for canceling the thread No default thread pooling No default for handling configuration changes in Android Having Tech or Coding Interview? Check 113 Android Interview Questions Source: stackoverflow.com

15.0.5 What is the difference between compileSdkVersion and targetSdkVersion?

Mid Top 113 Android Interview Questions Android 113 Answer The compileSdkVersion is the version of the API the app is compiled against. This means you can use Android API features included in that version of the API (as well as all previous versions, obviously). If you try and use API 16 features but set compileSdkVersion to 15, you will get a compilation error. If you set compileSdkVersion to 16 you can still run the app on a API 15 device as long as your app's execution paths do not attempt to invoke any APIs specific to API 16.

The targetSdkVersion has nothing to do with how your app is compiled or what APIs you can utilize. The targetS-dkVersion is supposed to indicate that you have tested your app on (presumably up to and including) the version you specify. This is more like a certification or sign off you are giving the Android OS as a hint to how it should handle your app in terms of OS features.

15.0.6 What is the difference between a Bundle and an Intent?

Mid Top 113 Android Interview Questions Android 113 Answer A Bundle is a collection of key-value pairs. However, an Intent is much more. It contains information about an operation that should be performed. This new operation is defined by the action it can be used for, and the data it should show/edit/add. The system uses this information for finding a suitable app component (activity/broadcast/service) for the requested action. Think of the Intent as a Bundle that also contains information on who should receive the contained data, and how it should be presented.

15.0.7 What are the wake locks available in android?

A - PARTIAL$_{WAKELOCK}$ B - SCREEN$_{DIMWAKELOCK}$ C - SCREEN$_{BRIGHTWAKELOCK}$ D - FULL$_{WAKELOCK}$ E - FULL$_{WAKELOCK}$ Answer : E Explanation When CPU is on mode, PARTIAL$_{WAKELOCK}$ will be active.

When CPU + bright Screen low is on mode, SCREEN$_{DIMWAKELOCK}$ will be active.

When CPU + bright Screen High is on mode,SCREEN$_{BRIGHTWAKELOCK}$ will be active.

When CPU, Screen, bright Screen High is on mode, FULL$_{WAKELOCK}$ will be active.

Chapter 16

项目中用到的小点

16.1 api level 28, androidx 之前的最后一个版本

16.2 android api level 30 androidx 中项目一定需要修改的条款: androidx.fragment 还没有弄通

16.2.1 gradles.propertiess

16.2.2 project build.gragle: gragle versions

16.2.3 app module build.gradle

- 什么是 Jetifier? 例如，要使用 androidx 打包的依赖项创建新项目，此新项目需要在 gradle.properties 文件中添加以下行：

16.2.4 JavaVersion.VERSION$_{18}$

```
java version 8
 compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
```

16.2.5 references

```
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Menu;
import android.view.MenuItem;
import androidx.fragment.app.Fragment;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;
```

16.2.6 xml 中也还有一些注意事项

```
<com.me.generalprac.CustomTitleView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
<include layout="@layout/custom_title"/>
```