1、简述 SoundPool 类和 MediaPlayer 的使用有什么要注意的地方？
如果应用程序经常需要播放密集，短促的音效。这时还用 Mediaplayer 就显得不合适。
MediaPlayer 的缺点：资源占用量较高，延迟时间较长不支持多个音频同时播放。
SoundPool 使用音效池来播放一些较短的声音片段，它的优势资源占用量低和反应延迟小。

1）在硬件较差的客户端中，SoundPool 比 MediaPlay 使用延迟更大。
2）避免使用 SoundPool 来播放歌曲获取做游戏的背景音乐。只有那些短促，密集的声音才考虑使用 SoundPool 进行播放。
3）AudioFlinger could not create track, status: -12
SoundPool 即音效池，在创建的时候 maxStream 这个参数代表能够同时播放的最大音效数，这里切忌合理使用，写的太大后会报 AudioFlinger could not create track, status: -12 。。。。一旦报了这个错，你就听不到声音了，呵呵。
4）256 个音效
当调用 load 方法的时候实际就是把音效加载到了 SoundPool 中，此时返回的 streamId 其实就是该音效在 SoundPool 中的 Id，这个 ID 从 0 还是 1 来着（有点记不清了） 递增，不过要注意的是，不要超过 256 这个临界点。也就是说第 257 个声音加载进去后，调用 play 方法其实是播不出来的，说不定还会挤掉一些前面加载好的声音。这个 256 的限制通过查看 SDK 源码基本就能了解清楚，它底层就那么实现的，用一个类似堆栈来存。
5）unload 方法和 release 方法
如果你音效多，也不要指望 unload 方法来清除掉一些音效后再 load 新的进去，虽然 unload 后音效卸载了，但是前面分给它在 SoundPool 里面的 Id 可没有释放掉，也就是说这个时候你 load 新的进去只会在后面继续累加，然后累加多了就超过 256 了，然后就就听不到声音，然后就没有然后了。要想彻底清掉前面的音效请使用 release 方法，它会连内存中占用的资源一起释放掉。
6）其他还有点什么呢，load 需要一点点时间，load 后不要马上 unload，load ---play--unload 的做法并不可取，不要 load 太大的音效，它只会申请 1M 的内存空间。SoundPool 出错后通常会看到 retuen 的值是 0。

下面是 MediaPlayer 和 SoundPool 类的对比特性：
1）soundpool 可以播一些短的反应速度要求高的声音，比如游戏中的爆破声，而 mediaplayer 适合播放长点的。
2）SoundPool 载入音乐文件使用了独立的线程，不会阻塞 UI 主线程的操作。但是这里如果音效文件过大没有载入完成，我们调用 play 方法时可能产生严 重的后果，这里 Android SDK 提供了一个 SoundPool.OnLoadCompleteListener 类来帮助我们了解媒体文件是否载入完成，我们重载 onLoadComplete(SoundPool soundPool, int sampleId, int status) 方法即可获得。
3）从上面的 onLoadComplete 方法可以看出该类有很多参数，比如类似 id，是的 SoundPool 在 load 时可以处理多个媒体一次初始化并放入内存中，这里效率比 MediaPlayer 高了很多。
4）SoundPool 类支持同时播放多个音效，这对于游戏来说是十分必要的，而 MediaPlayer 类是同步执行的只能一个文件一个文件的播放。