Activity缓存方法。
有a、b两个Activity，当从a进入b之后一段时间，可能系统会把a回收，这时候按back，执行的不是a的onRestart而是onCreate方法，a被重新创建一次，这是a中的临时数据和状态可能就丢失了。
可以用Activity中的onSaveInstanceState()回调方法保存临时数据和状态，这个方法一定会在活动被回收之前调用。方法中有一个Bundle参数，putString()、putInt()等方法需要传入两个参数，一个键一个值。数据保存之后会在onCreate中恢复，onCreate也有一个Bundle类型的参数。
示例代码：

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //这里，当Acivity第一次被创建的时候为空
        //所以我们需要判断一下
        if( savedInstanceState != null ){
            savedInstanceState.getString("anAnt");
        }
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        outState.putString("anAnt","Android");

    }
```

一、onSaveInstanceState (Bundle outState)
当某个activity变得“容易”被系统销毁时，该activity的onSaveInstanceState就会被执行，除非该activity是被用户主动销毁的，例如当用户按BACK键的时候。
注意上面的双引号，何为“容易”？言下之意就是该activity还没有被销毁，而仅仅是一种可能性。这种可能性有哪些？通过重写一个activity的所有生命周期的onXXX方法，包括onSaveInstanceState和onRestoreInstanceState方法，我们可以清楚地知道当某个activity（假定为activity A）显示在当前task的最上层时，其onSaveInstanceState方法会在什么时候被执行，有这么几种情况：

1、当用户按下HOME键时。
这是显而易见的，系统不知道你按下HOME后要运行多少其他的程序，自然也不知道activity A是否会被销毁，故系统会调用onSaveInstanceState，让用户有机会保存某些非永久性的数据。以下几种情况的分析都遵循该原则
2、长按HOME键，选择运行其他的程序时。
3、按下电源按键（关闭屏幕显示）时。
4、从activity A中启动一个新的activity时。
5、屏幕方向切换时，例如从竖屏切换到横屏时。（如果不指定configchange属性） 在屏幕切换之前，系统会销毁activity A，在屏幕切换之后系统又会自动地创建activity A，所以onSaveInstanceState一定会被执行
总而言之，onSaveInstanceState的调用遵循一个重要原则，即当系统“未经你许可”时销毁了你的activity，则onSaveInstanceState会被系统调用，这是系统的责任，因为它必须要提供一个机会让你保存你的数据（当然你不保存那就随便你了）。另外，需要注意的几点：
1.布局中的每一个View默认实现了onSaveInstanceState()方法，这样的话，这个UI的任何改变都会自动地存储和在activity重新创建的时候自动地恢复。但是这种情况只有在你为这个UI提供了唯一的ID之后才起作用，如果没有提供ID，app将不会存储它的状态。
2.由于默认的onSaveInstanceState()方法的实现帮助UI存储它的状态，所以如果你需要覆盖这个方法去存储额外的状态信息，你应该在执行任何代码之前都调用父类的onSaveInstanceState()方法
（super.onSaveInstanceState()）。 既然有现成的可用，那么我们到底还要不要自己实现onSaveInstanceState()?这得看情况了，如果你自己的派生类中有变量影响到UI，或你程序的行为，当然就要把这个变量也保存了，那么就需要自己实现，否则就不需要。
3.由于onSaveInstanceState()方法调用的不确定性，你应该只使用这个方法去记录activity的瞬间状态（UI的状态）。不应该用这个方法去存储持久化数据。当用户离开这个activity的时候应该在onPause()方法中存储持久化数据（例如应该被存储到数据库中的数据）。
4.onSaveInstanceState()如果被调用，这个方法会在onStop()前被触发，但系统并不保证是否在onPause()之前或者之后触发。
二、onRestoreInstanceState (Bundle outState)

至于onRestoreInstanceState方法，需要注意的是，onSaveInstanceState方法和onRestoreInstanceState方法“不一定”是成对的被调用的，（本人注：我昨晚调试时就发现原来不一定成对被调用的！）onRestoreInstanceState被调用的前提是，activity A“确实”被系统销毁了，而如果仅仅是停留在有这种可能性的情况下，则该方法不会被调用，例如，当正在显示activity A的时候，用户按下HOME键回到主界面，然后用户紧接着又返回到activity A，这种情况下activity A一般不会因为内存的原因被系统销毁，故activity A的onRestoreInstanceState方法不会被执行另外，onRestoreInstanceState的bundle参数也会传递到onCreate方法中，你也可以选择在onCreate方法中做数据还原。 还有onRestoreInstanceState在onstart之后执行。 至于这两个函数的使用，给出示范代码（留意自定义代码在调用super的前或后）：

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
        savedInstanceState.putBoolean("MyBoolean", true);
        savedInstanceState.putDouble("myDouble", 1.9);
        savedInstanceState.putInt("MyInt", 1);
        savedInstanceState.putString("MyString", "Welcome back to
Android");
        // etc.
        super.onSaveInstanceState(savedInstanceState);
}

@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);

        boolean myBoolean =
savedInstanceState.getBoolean("MyBoolean");
        double myDouble = savedInstanceState.getDouble("myDouble");
        int myInt = savedInstanceState.getInt("MyInt");
        String myString = savedInstanceState.getString("MyString");
}
```