# 项目名称

随机摇号

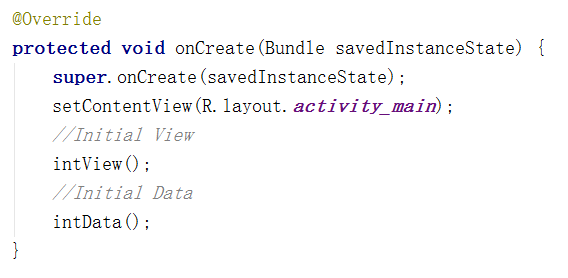
# 项目简介

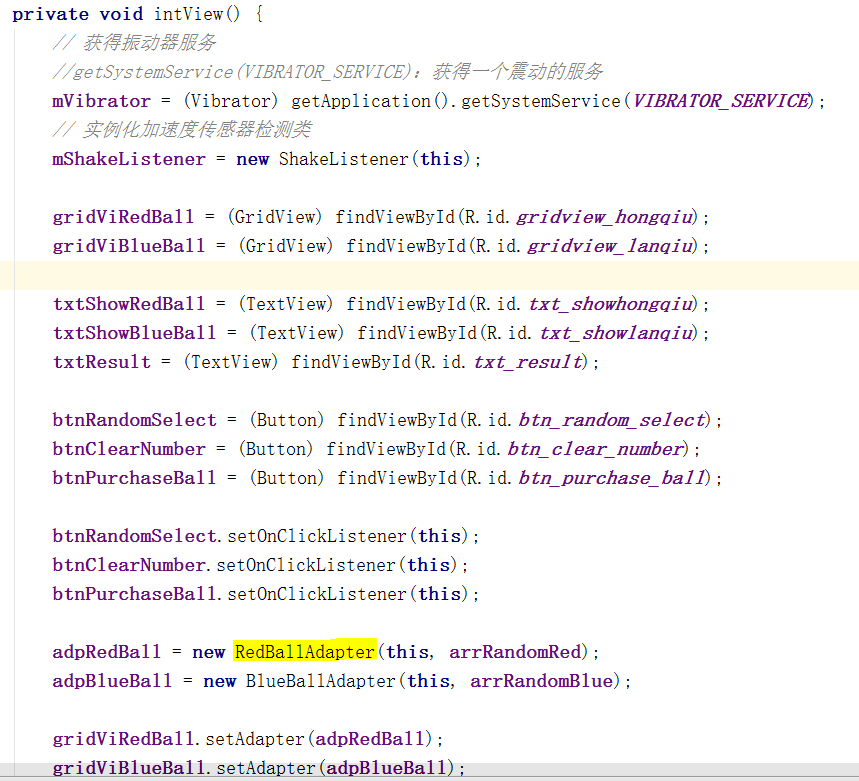
晃动手机，可随机选号，分别选出6个红球，一个蓝球。也可以单击随机选号按钮，选出6个红球，一个篮球。

# 知识点

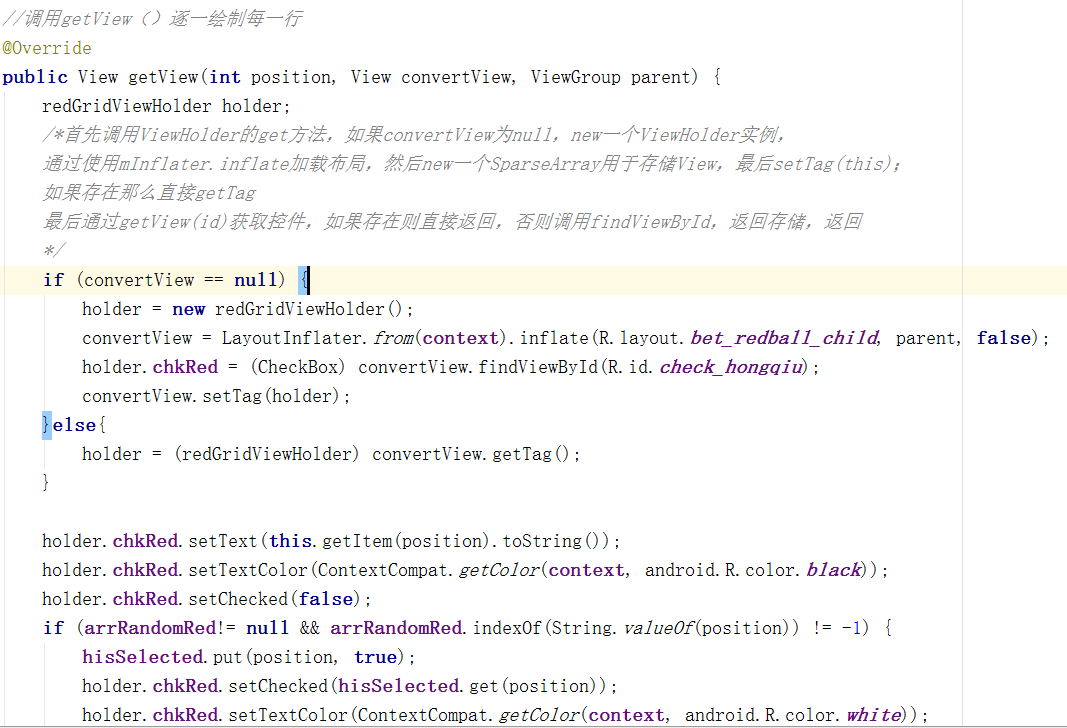
1. 自定义适配器
2. Android重力监听器
3. 排列组合算法
4. 随机选号算法

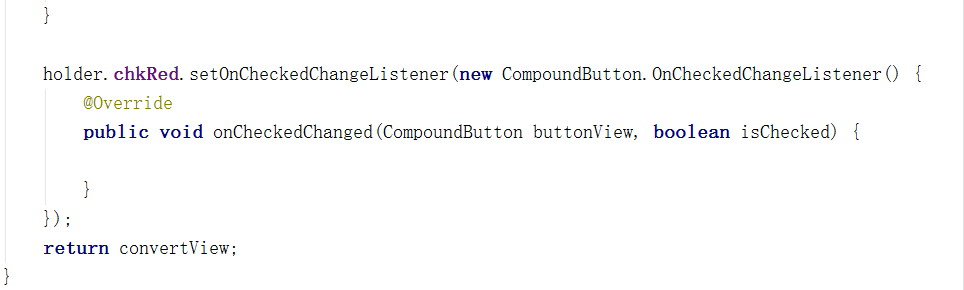
# 代码分析





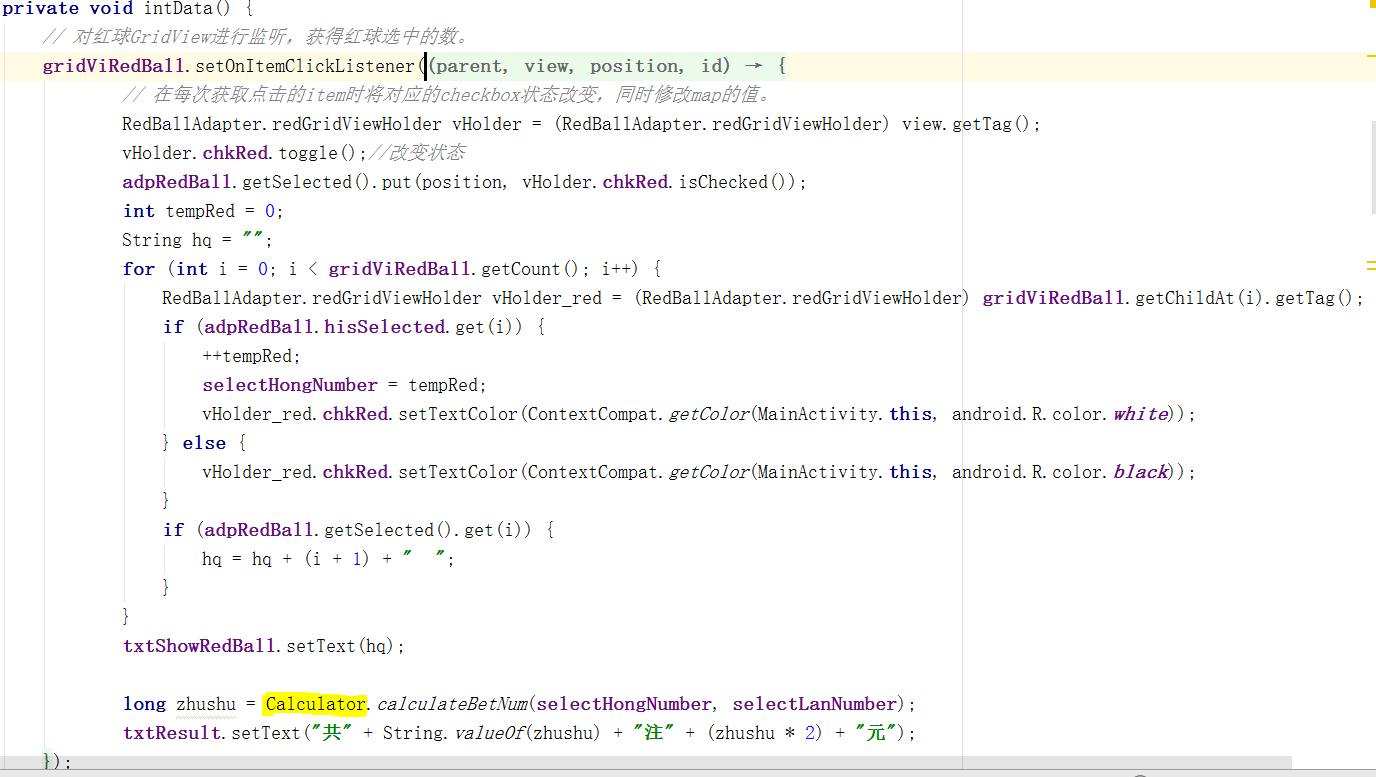
图中高亮部分的主要代码如下：





分析：

对于Adapter一般都继承BaseAdapter复写几个方法，getView里面使用ViewHolder模式。首先分析下ViewHolder的作用，通过convertView.setTag与convertView进行绑定，然后当convertView复用时，直接从与之对于的ViewHolder(getTag)中拿到convertView布局中的控件，省去了findViewById的时间。也就是说，实际上们每个convertView会绑定一个ViewHolder对象，这个viewHolder主要用于帮convertView存储布局中的控件。那么我们只要写出一个通用的ViewHolder，然后对于任意的convertView，提供一个对象让其setTag即可；既然是通用，那么我们这个ViewHolder就不可能含有各种控件的成员变量了，因为每个Item的布局是不同的，最好的方式是什么呢？提供一个容器，专门存每个Item布局中的所有控件，而且还要能够查找出来；既然需要查找，那么ListView肯定是不行了，需要一个键值对进行保存，键为控件的Id，值为控件的引用，相信大家立刻就能想到Map。





第一个高亮部分的分析：



这是红球和篮球的排列组合算法:

1. 红球多于6个，蓝球为1个：C(m，6)
2. 红球为6个，篮球多于1个：C(6,6)\*c(n,1)
3. 红球多于6个, 篮球多于1个: C(m，6) \*c(n,1)

第二个高亮部分的分析：

当加速度感应器感受到重力变化，就去通知onShakeListener调用onShake(),震动的同时随机选号。

**一、加速度感应器**

在这里需要设置一个速度阈值和时间间隔，控制一定的时间间隔才能触发第二次震动，还有感受速度太小也不去onShake（）。

1)得到sensorManager。

sensorManager=(SensorManager) mContext.getSystemService(Context.SENSOR\_SERVICE);

2）得到sensor

sensor=sensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER);

3）注册感应监听

sensorManager.registerListener(this, sensor, SensorManager.SENSOR\_DELAY\_NORMAL);



**二、控制震动**

得到Vibrator对象

vibrator=(Vibrator) getBaseContext().getSystemService(Context.VIBRATOR\_SERVICE);

震动方法有两种

vibrator.vibrate(500);震动多少毫秒

或者设置一个震动周期的数组

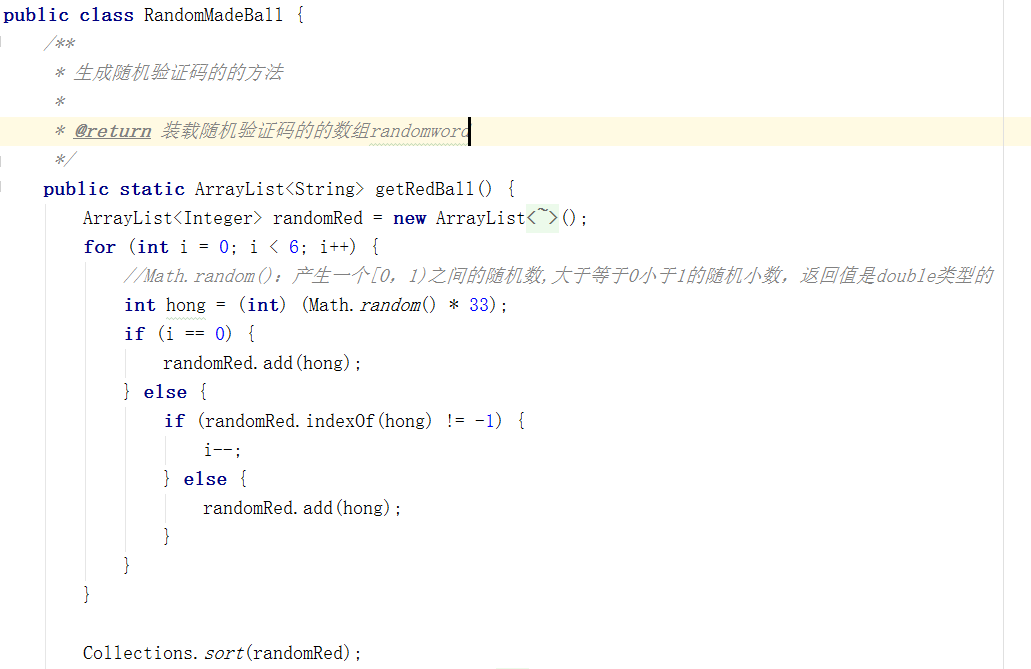
vibrator.vibrate(new long[]{100,200,100,200}, -1); //停止，震动，停止，震动

-1表示重复次数为不重复



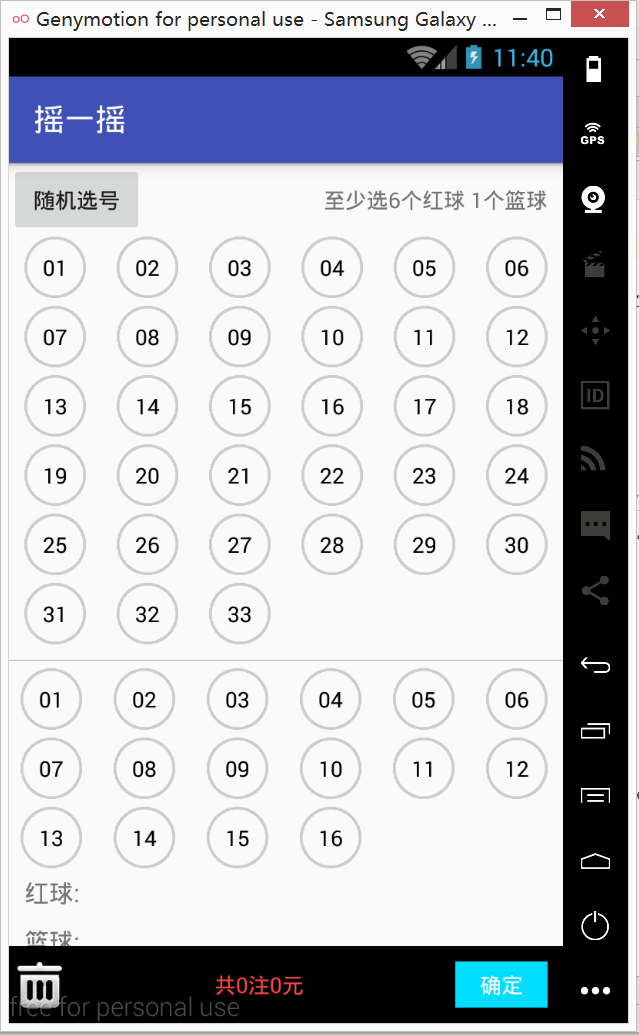
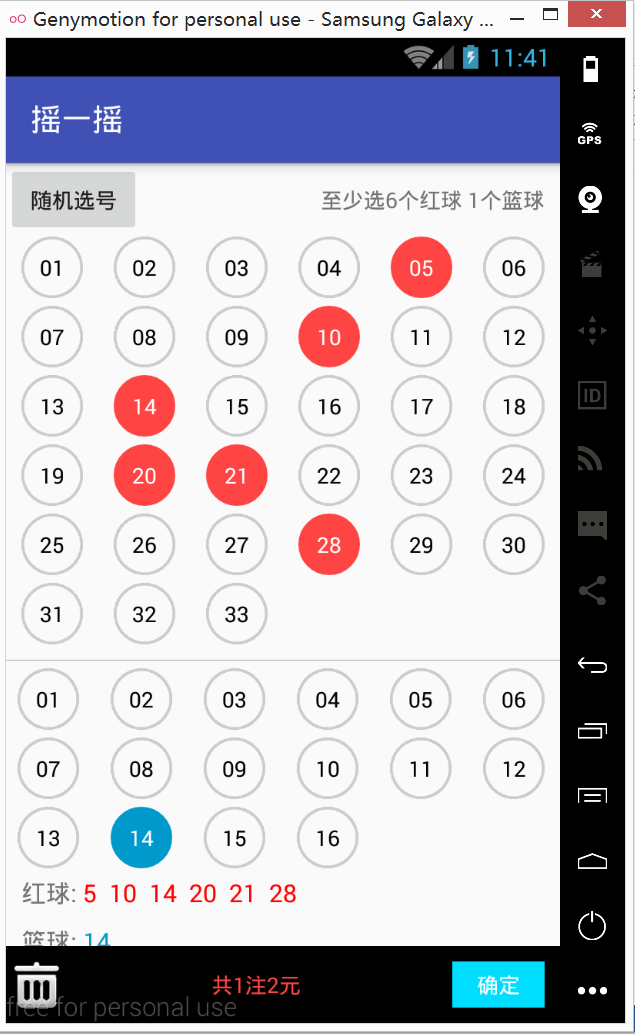
第三个高亮部分的分析：





这个是产生随机选号的算法。

# 效果图

# Github地址

<https://github.com/xiangshuwei/myyaoyiyao>

# 总结与体会

通过本学期的安卓课程，我学会了基本的安卓开发知识和开发过程，并且将这些知识点使用在答辩设计中，加深了我对本学期所学知识点的理解和体会，与此同时，在韩老师的带领下学会了将知识融汇贯通，谢谢韩老师。

# 参考文献

适配器：<http://blog.csdn.net/k316378085/article/details/51880123>

Android重力监听感应：<http://blog.csdn.net/huweigoodboy/article/details/39880779>

[new Handler().postDelayed() 延迟intent跳转](http://blog.csdn.net/x605940745/article/details/19401549) ：

http://blog.csdn.net/x605940745/article/details/19401549