如何退出 Activity？如何安全退出已调起多个 Activity 的 Application？
退出 activity 是调用 finish（）函数，如果是多个的话可以通过抛出异常强制退出，这种方法实现但是不能出现异常的对话框怎么实现呢，这个通过让你的 application 实现 unCheckedException 接口，如果实现了这个接口，你就主动捕获了该异常，将不会弹出该异常强制关闭对话框。

用至少两种方式实现一个 Singleton（单例模式）。
一。静态内部类实现：这种方式是 Singleton 类被装载了， instance 不一定被初始化。因为 SingletonHolder 类没有被主动使用，只有显示通过调用 getInstance 方法时，才会显示装载 SingletonHolder 类，从而实例化 instance 。

```
1
2
3
4
5
6
7
8
9
public class Singleton {
    private static class SingletonHolder {
    private static final Singleton INSTANCE = new Singleton();
    }
    private Singleton (){}
    public static final Singleton getInstance() {
    return SingletonHolder.INSTANCE;
    }
}
```

二。双重检查锁：在 JDK1.5 之后，双重检查锁定才能够正常达到单例效果。（因为 volitile 关键字）

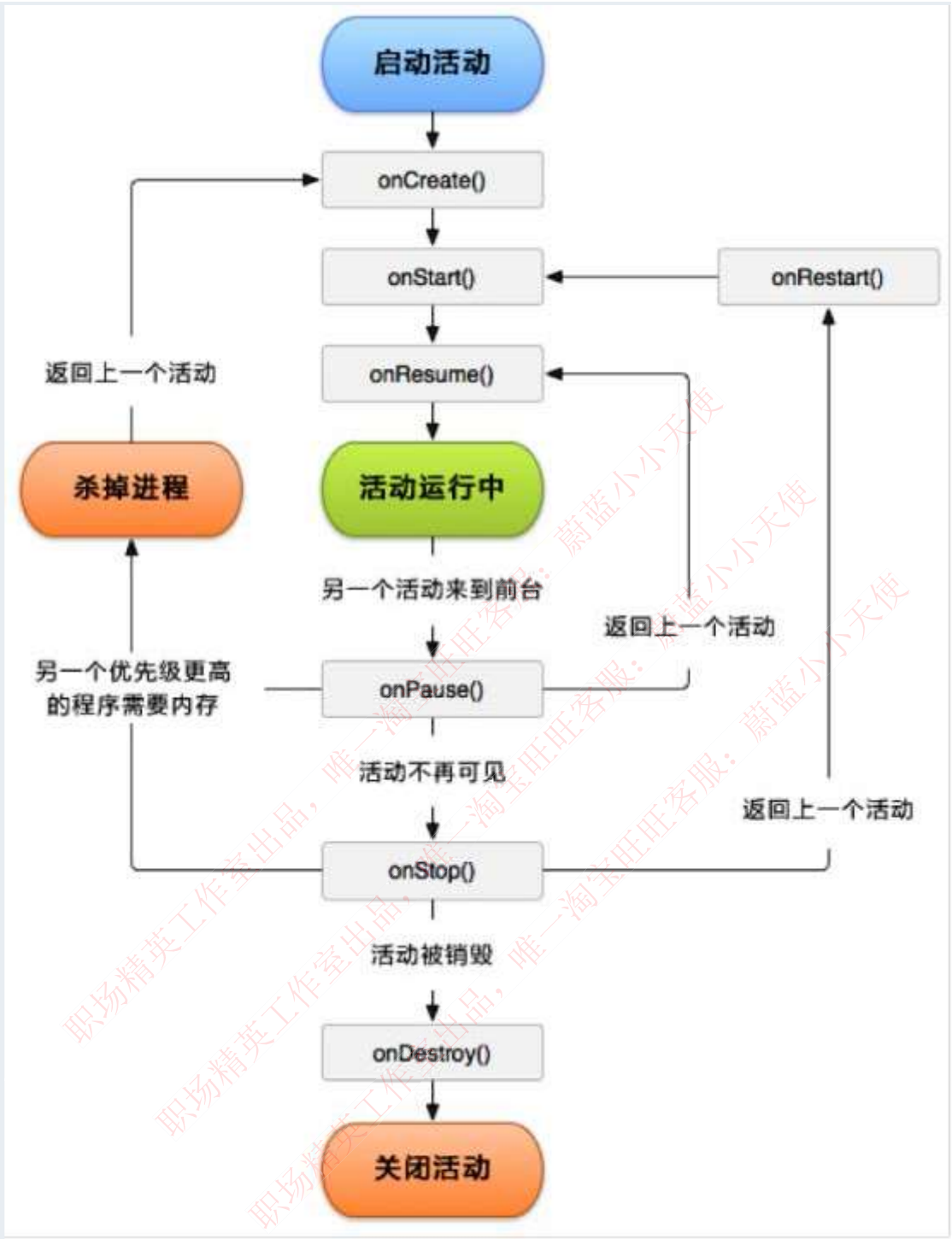
```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
public class Singleton {
    private volatile static Singleton singleton;
    private Singleton (){}
    public static Singleton getSingleton() {
    if (singleton == null) {
        synchronized (Singleton.class) {
        if (singleton == null) {
            singleton = new Singleton();
        }
        }
    }
    return singleton;
    }
}
```

多线程有几种实现方法，都是什么？同步有几种实现方法，都是什么？
参考答案

一、多线程有几种实现方法，都是什么？
方式 1：继承 Thread 类。
A:自定义类 MyThread 继承 Thread 类。
B:MyThread 类里面重写 run()：run Alt+/
C:创建对象
D:启动线程
方式 2：实现 Runnable 接口的步骤：
A:自定义类 MyRunnable 实现 Runnable 接口
B:重写 run()方法
C:创建 MyRunnable 类的对象
D:创建 Thread 类的对象，并把 C 步骤的对象作为构造参数传递
方法 3：实现 Callable 接口，重写 call()方法：不常用
一般使用方法 2
二、同步有几种实现方法，都是什么？
（1）synchronized 关键字：包括 synchronized 方法和 synchronized 块
（2）wait()方法和 notify()方法
（3）Lock 接口及其实现类 ReentrantLock

请描述下 Activity 的生命周期。
参考答案



如上图

1.onCreate

> 在活动第一次被创建时调用，可以在这个方法中完成活动的初始化操作，比如加载布局，绑定事件

2.onStart

> 在活动由不可见变为可见时候调用

3.onResume

> 在活动准备好和用户进行交互时调用，此时活动一定位于返回栈的栈顶，并处于运行状态

4.onPause

> 在系统准备去启动或者恢复另一个活动时候调用。可以在这个方法中将一些消耗 CPU 的资源释放掉，以及保存一些关键数据（但不能做耗时操作，否则影响新的栈顶活动的使用）

5.onStop

> 在活动完全不可见的时候调用
>
> 与 onPause 区别：如果启动一个对话框，那么 onPause 会执行，onStop 不会

6.onDestroy

> 在活动被销毁前调用，之后活动变为销毁状态

7.onRestart

> 这个活动由停止状态变化运行状态之前调用，也就是活动重新被启动了。

我们可以将活动分为三种三种生存周期。

1.完整生存期

> 由 onCreate->onDestroy

2.可见生存期

> 由 onStart->onStop

3.前台生存期

> 由 onResume->onPause

什么是 ANR 和 Force Close？如何避免？

参考答案

ANR: Application Not Responding

产生原因：

1.主线程（UI 线程）响应用户操作事件时间超过 5 秒

2.BroadcastReceiver 超过 10 秒钟任未执行完毕。

避免方法：

Android 应用程序完全运行在一个独立的线程中。任何在主线程中运行的，需要消耗大量时间的操作都会引发 ANR。因此，需要消耗大量时间的操作如访问网络和数据库，都要放到子线程中或者使用异步方式来完成。

Force Close.

产生原因：
程序出现异常，一般像空指针、数组越界、类型转换异常等。
避免方法：
编写程序时要思维缜密，异常出现时可以通过 logcat 查看抛出异常代码出现的位置，然后到程序中进行修改。

战争游戏的至关重要环节就要到来了，这次的结果将决定王国的生死存亡，小 B 负责首都的防卫工作。首都位于一个四面环山的盆地中，周围的 n 个小山构成一个环，作为预警措施，小 B 计划在每个小山上设置一个观察哨，日夜不停的瞭望周围发生的情况。 一旦发生外地入侵事件，山顶上的岗哨将点燃烽烟，若两个岗哨所在的山峰之间没有更高的山峰遮挡且两者之间有相连通路，则岗哨可以观察到另一个山峰上的烽烟是否点燃。由于小山处于环上，任意两个小山之间存在两个不同的连接通路。满足上述不遮挡的条件下，一座山峰上岗哨点燃的烽烟至少可以通过一条通路被另一端观察到。对于任意相邻的岗哨，一端的岗哨一定可以发现一端点燃的烽烟。 小 B 设计的这种保卫方案的一个重要特性是能够观测到对方烽烟的岗哨对的数量，她希望你能够帮她解决这个问题。

//AC 代码：

```
#include<stdio.h>
#include<vector>
using namespace std;
struct node{
	long val,cnt;
	node(long val):val(val),cnt(1){}
};
int main(){
	int N,i,x,Maxi;
	//freopen("input.txt","r",stdin);
	while(scanf("%d",&N)!=EOF){
		vector<long> d(N);
		for(i=0;i<N;i++) scanf("%ld",&d[i]);
		vector<node> v;
		node tmp(d[0]);
		long Max=-1;
		for(i=1;i<N;i++)
			if(d[i]==d[i-1]) tmp.cnt++;
			else{
				v.push_back(tmp);
				if(Max<tmp.val){
					Max=tmp.val;
					Maxi=v.size()-1;
				}
				tmp.val=d[i];
				tmp.cnt=1;
			}
```

```
            v.push_back(tmp);
            if(Max<tmp.val){
                Max=tmp.val;
                Maxi=v.size()-1;
            }
            int n=0;
            long cnt=0;
            vector<node> stack;
            for(i=Maxi;n!=v.size();n++,i=(i+1)%v.size()){
                while(stack.size()&&v[i].val>stack[stack.size()-1].val){
                    node &m=stack[stack.size()-1];
                    cnt+=m.cnt+m.cnt*(m.cnt-1)/2;
                    stack.pop_back();
                    if(stack.size()) cnt+=m.cnt;
                }
                if(stack.size()){
                    if(v[i].val==stack[stack.size()-1].val)
                        stack[stack.size()-1].cnt+=v[i].cnt;
                    else
                        stack.push_back(v[i]);
                }else
                    stack.push_back(v[i]);
            }
            while(stack.size()){
                node &m=stack[stack.size()-1];
                cnt+=m.cnt*(m.cnt-1)/2;
                stack.pop_back();
                if(stack.size()) cnt+=2*m.cnt;
                if(stack.size()==1&&stack[stack.size()-1].cnt==1) cnt-=m.cnt;
            }
            printf("%ld\n",cnt);
        }
}
```

//这是左程云老师的思路　我动手实现了一下　应该是最优解　时间复杂度 O(N)　用单调栈

尽管是一个 CS 专业的学生，小 B 的数学基础很好并对数值计算有着特别的兴趣，喜欢用计算机程序来解决数学问题，现在，她正在玩一个数值变换的游戏。她发现计算机中经常用不同的进制表示一个数，如十进制数 123 表达为 16 进制时只包含两位数 7、11（B），用八进制表示为三位数 1、7、3，按不同进制表达时，各个位数的和也不同，如上述例子中十六进制和八进制中各位数的和分别是 18 和 11,。 小 B 感兴趣的是，一个数 A 如果按 2 到 A-1 进制表达时，各个位数之和的均值是多少？她希望你能帮她解决这个问题？ 所有的计算均基于十进制进行，结果也用十进制表示为不可约简的分数形式。

不仅考察进制换算，还考察最大公约数

```
def f1(n): # 进制换算，计数
    ret = 0
    for i in range(2,n): # 一次除以 2 至 n-1，相当于换算成 2 至 n-1 进制
        t = n
        while 1:
            ret += t % i # 将结果加入返回值
            t = t // i
            if t == 0:
                break
    return ret
def f2(n, m): # 求最大公约数
    ret = 1
    while m > 0:
        t = n % m
        n = m
        m = t
    ret = n
    return ret
if __name__ == '__main__':
    while 1:
        try:
            n = int(raw_input())
        except:
            break
        r = f1(n) # r 是总的进制换算相加的数
        l = len(range(2, n)) # 被除数 2 至（n-1）
        s = f2(r, l) # 最大公约数
        r , l = r//s, l//s # 结果除以最大公约数
        print str(r) + '/' + str(l)
```