<https://github.com/JakeWharton/timber>

懒人的Log工具类，调用Log输出方法时不需要指定TAG，会自动使用当前类名作为TAG。

这个库为常轻巧，只有一个java文件：Timber.java，Timber类和两个内部类：Tree、DebugTree

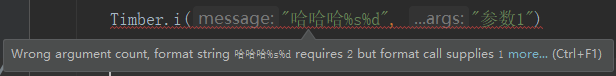
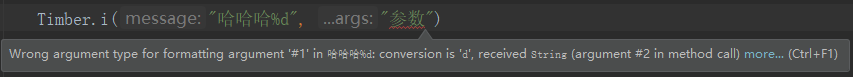
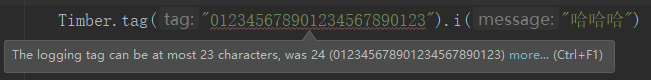
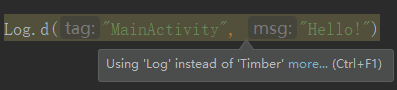
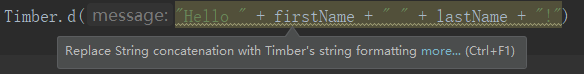
消息长度最大为4000，如果大于这个长度，则分割后再分别输出

# 使用步骤

1. 添加依赖：implementation 'com.jakewharton.timber:timber:4.7.1'
2. 初始化，Timber中的log输出方法最终是使用Tree类进行输出的，所以需要指定Timber输出Log时使用的Tree实现类是哪一个，目前有一个现有的实现类：DebugTree，它实现为自动使用类名做为tag  
   在Application的onCreate中 ：Timber.plant(Timber.DebugTree())
3. 输出Log：
   1. 正常输出：  
      Timber.i("哈哈哈") // MainActivity: 哈哈哈
   2. 格式化输出：  
      Timber.i("我叫%s，我今年%d岁", "小明", 28) // MainActivity: 我叫小明，我今年28岁
   3. 异常输出  
      Timber.i(exception)
   4. 消息带异常输出  
      Timber.i(exception, "别怕，这是自己抛的异常") // 先输出参数二，换行输出异常  
      Timber.i(exception, "自己抛的异常%s%s", "哈哈", "呵呵")
   5. 自定义TAG  
      Timber.tag("MyTag").i("哈哈哈") // MyTag: 哈哈哈

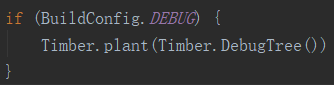
v、d、i、w、e级别的输出都是一样的。

Timber还内嵌了方法参数的检测规则，当传入的参数不正确时，直接在AS中报错，如下：

1. 参数数量不对  
   
2. 参数类型不对  
   
3. Android TAG的最大长度不能超过23位，这个限制在Api 24中删除  
   
4. 提示使用Timber代替系统Log  
   
5. 提示使用格式化方法  
     
   即应该用：Timber.d("Hello%s%s!", firstName, lastName)

# 实现原理与扩展

查看其源码即可知其原理，大概如下：

* 通过创建一个异常，从异常中获取堆栈信息，从而取出调用方法的类名作为TAG
* Timber翻译过来意思为“木材”
* Timber.plant(Tree)，plant tree翻译过来意思为“种树”，可以多次调用这个方法即可种多棵树，还有一个重载的方法可以接收可变参数的Tree。 所有种进去的树，保存在一个叫FOREST（森林）的集合中，并且把这个集合复制到一个叫forestAsArray的数组中
* Tree的功能就是输出log，所以如果种了多个树，则同一个log就会输出多次，Timber中保存了一个Tree的代理实现，它会把log的输出全部转发给森林中的每一个Tree对象进行输出。
* DebugTree在release发布版本也是会输出log的，所以如果要控制Log在release版本中不输出，则可以判断仅仅在Debug版本才种树就可以了，如下：  
  
* Tree类中只有一个抽象方法：protected abstract void log(int priority,String tag, String message,Throwable t)，log如何输出全靠这个方法实现，所以如果这里不实现，则不会有log输出。另外Tree中有一个boolean isLoggable(String tag, int priority)方法，默认返回true，我们可以覆盖这个方法，然后根据tag或者根据priotity来控制是否可以输出log。扩展Tree类时，可以继承DebugTree，这样输出Log的方法就不用我们管了，我们只需要重写isLoggable来控制什么情况可以或者不可以输出log就够了，比如：如果是Debug版本，则可以输出i级别或以下级别，如果是release版本，则可以输出e级别
* Timber可以和这个库配合使用：<https://github.com/JakeWharton/pidcat/>
* 可以学习一下Timber中的检测方法参数是否合法的Link检查是怎么实现的。在它里面有一个link.jar