己写代码感觉无从下手，不知道如何去设计，实现功能 写哪些类，写哪些方法等
我们现在都是面向对象编程，但有些新手还是面向过程的编程方式。面向对象和面向过程的差异是，面向过程解决问题是一条线解决完就完事了，假如下次遇见类似这种问题，还要把之前那个解决的方法逻辑重新写一遍。面向对象会是把问题解决的过程抽象成解决类似问题的一个类，下次遇见这种类型的问题只需要调用当初实现好的方法就可以，效率不言而喻。

当你要写代码的时候，很多人就是看了需求之后直接埋头写代码，写的诗歌什么鬼也不知道，到最后功能是写出来了，但是实现的过程简直就是车祸现场。

当你接到需求的时候：
不要直接写代码！！！ 不要直接写代码！！！ 不要直接写代码！！！

你首先需要先有个思路，为了完成这个功能，我需要怎么做（又回到开头的把大象塞进冰箱的问题），需要哪些步骤，有的功能在目前的项目里有没有，如过这些想清楚了，先把接口类写好，规范好你的步骤，先实现接口之间的调用，看逻辑是否通畅，然后剩下的就是实现接口里的方法了，就这么简单 （手动摊手）此处注意几点

一个方法只做一个事情
一个类只负责一种事情
假如以后有需求变动，怎么样对现在的你来说更灵活
此处推荐两本书《设计模式之禅》、《重构》

自己调试出了问题不知道要怎么去解决
/*

为API生，为框架死，为了bug奋斗一辈子，吃符号亏，上大小写的当，最后死在需求上。
——摘自程序员语录

Bug常年有，你这特别多。 很多Bug的引起都是因为之前写的时候思路不清晰引起的，假如思路清晰，很多bug就已经灭亡在消失的路上了，但是天有不测风云，总有些强壮的Bug还是走到了你面前。

怎么办

我平常都是用的Java 所以这边暂以Java作为例子 Bug主要分为 业务上的Bug 和 程序上的Bug
业务上的主要是输出数据和预期的不一致，显示不正常等，程序上的最严重就是崩溃，其次是内存泄露和性能。

内存泄露和性能问题不在今天讨论的范围，业务上的问题很好处理，基本都是大意引起的，主要面对的就是程序上的崩溃，想处理崩溃，就要理解一们变成语言的异常体系，像Java中，常碰见的异常：

java.lang.NullPointerException(空指针异常)
java.lang.ClassNotFoundException(指定的类不存在)
java.lang.NumberFormatException (字符串转换为数字异常)
java.lang.IndexOutOfBoundsException(数组下标越界异常)
java.lang.IllegalArgumentException (方法的参数错误)
java.lang.IllegalAccessException(没有访问权限)
java.lang.ArithmeticException(数学运算异常)
java.lang.ClassCastException(类型转换异常)
java.lang.FileNotFoundException(文件未找到异常)
java.lang.ArrayStoreException(数组存储异常)
java.lang.NoSuchMethodException(方法不存在异常)
java.lang.NoSuchFiledException(属性不存在异常)
java.lang.EOFException（文件已结束异常）
java.lang.InstantiationException（实例化异常）
java.lang.InterruptedException（线程被终止异常）
java.lang.CloneNotSupportedException（不支持复制异常）
java.lang.OutOfMemoryException（内存溢出异常）
java.lang.NoClassDefFoundException （未找到定义的类）
java.lang.StackOverflowError（堆栈溢出）
知道异常在哪里，然后根据里面的提示信息和错误的堆栈信息，外加上Google和度娘，解决问题，分分钟

> Bjarne Stroustrup（C++之父）说：
> 逻辑应该是清晰的，bug难以隐藏。
> 依赖最少，易于维护。
> 错误处理完全根据一个明确的策略。
> 性能接近最佳，避免代码混乱和无原则的优化。
> 整洁的代码只做一件事。