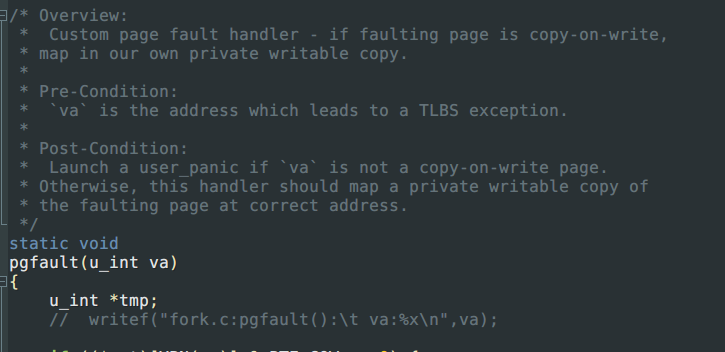
OO第三单元总结

第三单元主要强调了面向对象编程的抽象和规格，本文总结了规格化设计的发展历史，阐述了规格化设计方法的核心要素。总结了撰写规格的方法与我们现有作业中规格的问题。

# 规格化设计的发展历史

其实规格化设计这个概念自从编程语言诞生以来就一直存在着，因为编写程序本来就是一个对现实事物进行抽象的过程。最初的编程语言都是面向过程的语言，所以当时的规格就是对于某个语句的执行效果的说明。比如某个函数的作用有一个Overview，写明这个函数的合法输入和输出之类的。那我们所上的课程来举例子，就是类似操作系统实验中的代码，如下图。



使用注释来解释说明函数的功能也许就是最早的关于程序规格化的体现，这种功能性的注释不仅起到了“备忘”的作用，更能够方便他人理解你写的程序。毫不夸张地说，如果没有这些重要的规格说明，只凭我自己是无法完全理解实验代码的功能的，因为你所要顾及的不仅仅是这一个独立的函数，也许还会引用别的文件下面的函数，这就会使得使用者更加困扰。这个问题在大型工程中尤为严重。

引入了面向对象思想之后，程序的抽象程度再一次提高。拥有封装属性的对象相较于原本的基本数据类型来说更加难以理解，这就进一步提高了我们撰写程序规格的要求。在多人合作完成的大型工程中，统一的规格标准不但有利于团队之间的沟通，还能够显著方式程序出现比较低级的错误，进而可以显著地提高工作效率。这也是为什么编程规格在如今如此受欢迎的原因。

# 规格化设计方法的核心要素

规格化设计的核心要素是要求所撰写的规格满足限制性、一般性和清晰性。具体体现在规格上就是要求：

1. 限制性体现在每个类都要对应地有其Requires，每个类都要满足RepOK才能正常工作。不同的类不能够通用其限制条件，也就是说每个类都有单独的规格体现类自身的特性，这就是规格的限制性。
2. 一个好的规格必须要有好的一般性来确保没有任何或者只有很少的可接受程序被排除。
3. 清晰性指的是规格的描述语言要足够清晰，不能够有模棱两可的语句出现。

# 过程规格和类规格的撰写要点

过程规格和类规格的撰写个人感觉都比较简单，但是还是要注意上面提到的那三点原则。

首先，撰写过程规格的时候要注意写清楚方法使用的前置条件和后置条件，这个很关键，我看过的大多数作业也都做得很好，就是有些人会漏写Requires的部分，这个需要注意。特别需要注意的是Modifies这一项。这一项说明了这个方法的执行过程最多能够修改哪些对象变量。这个很多同学都会漏掉一些比较细节的对象，但是出于个人原因还是没从这上面扣过分。记得之前PPT上面曾经写过，System.println.out都会修改System里面的一个变量，这种系统方法的调用所modify的东西今后还是要小心。还有同学理解错了Requires的意义，这种错误当然就另当别论了。

之后，轮到写类规格了。类规格定义了类中数据的有效性，确保在满足前置条件的情况下，后置条件一定会得到满足。同时，类规格将类与抽象函数紧密地结合在了一起。Overview本身要求涵盖类整体内容的概述。我看到很多同学的Overview从头到尾都在介绍类中的变量，但是却没有介绍这个类的核心功能，这显然是有问题的。还有人把抽象函数和不变式都写到了过程规格里面，其实这也是不对的，因为用户并不关心这一部分内容。这些只要由RepOK方法来确保就可以了。有的同学RepOK干脆写的和return true一样，这样的RepOK也是没有意义的。

最后就是最近一次作业里面对于子类规格的撰写了。这一点除了要满足类规格的普遍标准之外，还需要注意子类的RepOK同时也需要满足父类的RepOK，并且使用子类替换父类要求能够保持程序行为的不变形。父类的承诺，子类不能减弱；父类的要求，子类不能加强。

# Bug及逻辑叙述总结

这三次作业没有被查出Bug，对于逻辑方面，个人确实有描述的不到位的地方，比如第一次的Overview只写了类的功能而没有提及类里面的变量，Modifies没有写全等等。但是由于同学手下留情并没有扣分，今后还要注意才行。