****

团队项目软件系统设计说明书

题 目： 随手记账本

学 院： 计算机科学与工程学院

专 业： 网络与信息安全

班 级： 2017级网络班

成 员： 杨野

成 员： 汪慧和

成 员： 李婷华

成 员： 王颖奇

教师姓名： 代祖华

目录

[一.引言 3](#_Toc22727)

[1.1编写目的 3](#_Toc17402)

[1.2项目意义 3](#_Toc16847)

[1.3项目研究内容 3](#_Toc3512)

[二.总体设计 3](#_Toc3762)

[2.1软件设计模式设计软件系统总体结构 3](#_Toc32466)

[2.2设计软件系统数据库逻辑结构 5](#_Toc12731)

[2.3软件重用 6](#_Toc14799)

[2.4设计关键类的重点服务 8](#_Toc32534)

[三.账本设计界面 11](#_Toc5868)

[3.1界面的设计与使用 11](#_Toc22494)

[四.帮助设计 15](#_Toc32091)

[4.1 注释设计 15](#_Toc3805)

[4.2方法注释 15](#_Toc25496)

[4.3通用注释 16](#_Toc23570)

[五.维护设计 16](#_Toc5974)

[5.1改正性维护 16](#_Toc7528)

[5.2适应性维护 16](#_Toc12744)

[5.3完善性维护 17](#_Toc22099)

[5.4预防性维护 17](#_Toc861)

# 一.引言

## 1.1编写目的

本软件设计说明书是根据本项目需求的规定并作为后续开发的指南，在需求分析阶段已经对本软件的功能模块进行详细的需求阐述，而本阶段主要在需求分析阶段已经对本软件的功能模块进行详细的需求阐述，而本阶段主要在模块间的接口等，本报告将详细说明本软件的系统设计，供开发人员开发软件时参考，作为软件开发的指南，亦为进行后面的设计和平台实现以及软件测试做准备。

## 1.2项目意义

当前生活节奏加快，人们生活水平不断提高，收入和支出越来越趋于多样化，传统的人工方式记录和管理家庭或自己财务情况的记录方式不仅不便于长期保存，还有一些不可避免的缺点。开发一个能够管理个人收支的软件系统已经很有必要，这样可以减轻人们在个人财务统计和其他财务管理的负担。能够克服传统方法的多个缺点。

## 1.3项目研究内容

本系统属于一个小型的个人记账软件，能够满足个人或者多人的的记账需求，提供用户信息管理，收支管理，统计分析，账目导出等功能。

# 二.总体设计

## 2.1软件设计模式设计软件系统总体结构

记账本是把用户所记录的日常消费、收入、借贷等信息存进统数据库进行管理，用户可以对其查询，修改，删除。可以将记录的内容从数据库导出到个人PC,以便保存。记账软件将这些记录的信息统计，生成统计图表后输出到用户界面，以便用户进行分析。如图2.1.1所示。软件的核心处理设计流程图如图2.1.2所示。

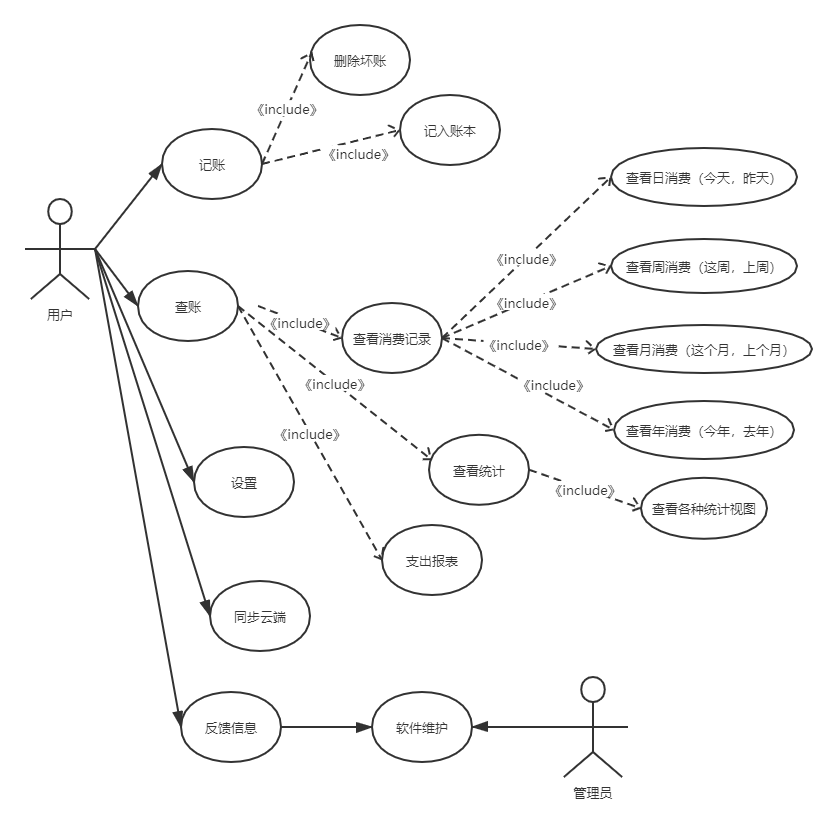


图2.1.1软件设计结构

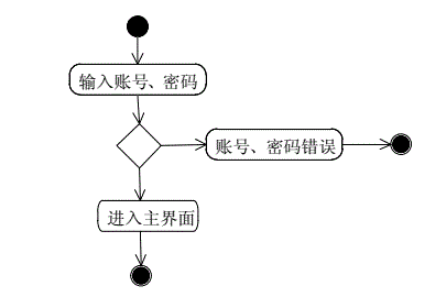


图2.1.2核心处理流程

## 2.2设计软件系统数据库逻辑结构

本系统主要的数据库表如图2.2.1所示：

|  |  |  |
| --- | --- | --- |
| 序号 | 表名 | 说明 |
| 1 | account | 账本表 |
| 2 | user | 用户表 |

图2.2.1数据库表

本系统的数据库关系图如图2.2.2所示：

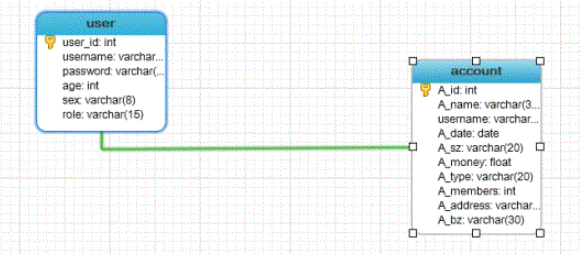


图2.2.2数据库关系图

用户表结构如图2.2.3所示。

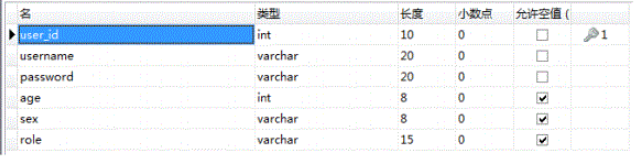


图2.2.3用户图表

账本表结构如图2.2.4所示。

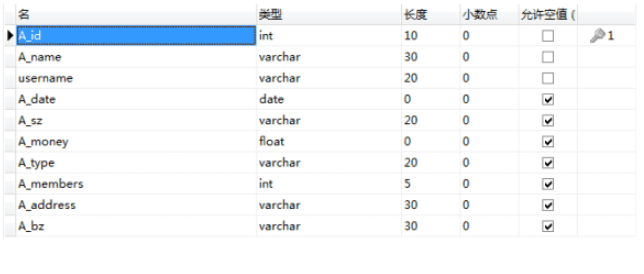


图2.2.4账本表结构

## 2.3软件重用

软件重用，是指在两次或多次不同的软件开发过程中重复使用相同或相似软件元素的过程。软件元素包括程序代码、设计文档、设计过程、需求分析文档甚至领域知识。通常，可重用的元素也称作软构件，可重用的软构件越大，重用的粒度越大。软件重用是软件工程研究的重要课题。它是指在构造新的软件系统的过程中利用已有的软件成分。这里所说的已有的软件成分包括十分广泛的内容，如数据，文档，模式，体系结构，知识，方法和环境等多方面的内容。从广义上说，软件重用包括3个层次：

(1)产品重用，如代码数据软件模式体系结构等的重用；

(2)知识重用，如方法、标准、经验、领域知识、软件过程等的重用

(3)环境重用，如系统程序、软件工具、开发环境等的重用。

### 2.3.1实现层的软件重用

实现层的软件重用是指已有的程序代码的重用。它包括以下3个方面：

(1)代码提取

　　软件开发人员对已有的程序代码，经理解、选择后，把适合于重用的代码据取出来，并进行必要的修改。使其适用于当前的上下文。这种可重用对象是最低程度的抽象，软件开发人员不但必须知道有关的规格说明。还必须清楚它的实现部分。一般来说，因为在理解修改及测试方面的代价。使用这种重用对象并不能有效地提高软件开发效率，提高软件的质量。

　　(2)源代码部件

　　源代码部件是专门编写经过严格测试后。专用于重用目的代码段。高级程序语言程序的一些程序单元，如过程，函数，包，类等。通常用于完成一个完整的功能，有良好的输入／输出界而，可以作为可重用的源代码部件。源代码部件的抽象程度比代码提取高，软件开发人员只需知道源代码部件的抽象规格说明部分即可。不必考虑它的实现部分。因为源代码部件是为重用目的专门编写的，其界面清晰。抽象规格说明明确，故使用方便而且质量高。目前使用得十分广泛。例如，在数值计算领域中的数学函数库。作为软件可重用对象，使用得十分成功。许多商品化的软件开发环境均有以重用为目的源代码部件库(函数库)。用标准函数的形式向用户提供大量可重用的源代码部件。但是，这种方法的不足之外在于源代码部件仍然是基于代码这一较低层次的抽象上，与实际问题领域有较大的距离。

　　(3)软件逻辑结构

　　也称为软件模式，是可重用源代码部件的形式化扩充。这一层次的可重用对象着重于对[数据结构](https://wiki.mbalib.com/wiki/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84" \o "数据结构)、逻辑结构和过程的抽象。软件逻辑结构主要描述软件的各个成分(包括模块，数据结构等)以及它们之问的关系(如调用与被调用，参数传递关系等)，它们可以作为一个整体被重用。对于基于形式化规格说明的软件逻辑结构，其可重用对象的选择、理解。特化均较为容易，也便于使用自动工具支持重用过程。由于软件人[员工](https://wiki.mbalib.com/wiki/%E5%91%98%E5%B7%A5" \o "员工)作于较高层次的抽象，能明显地提高[软件生产率](https://wiki.mbalib.com/wiki/%E8%BD%AF%E4%BB%B6%E7%94%9F%E4%BA%A7%E7%8E%87" \o "软件生产率)。但是，目前还缺乏描述软件逻辑结构的较好的形式化规格说明方法。

### 2.3.2设计层的软件重用

与编码阶段相比，软件设计阶段在[软件生命周期](https://wiki.mbalib.com/wiki/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F" \o "软件生命周期)中占有更重要的地位。重用已有的软件系统的[设计信息](https://wiki.mbalib.com/wiki/%E8%AE%BE%E8%AE%A1%E4%BF%A1%E6%81%AF" \o "设计信息)，可以更好地降低[软件开发成本](https://wiki.mbalib.com/wiki/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E6%88%90%E6%9C%AC" \o "软件开发成本)，提高软件生产卑。设计层的软件重用包括以下向个方面：

(1)甚高级语言

甚高级语言(VHLL]也称为可执行的规格说明语言。它允许开发者使用比高级语言更高层次抽象的规格说明描述用户需求。产生可执行系统。作为一种语言，甚高级语言与高级语官一样，具有独自的语法语义，可通过编译，从需求规格说明自动影射到可执行的代码。甚商级语言通常以集合论，谓词逻辑等高层次的数学抽象作为基础。

利用越高级语言，系统的设计信息作为可重用软件成分重用，而且甚高级语言比高级语言更接近于用户语言，表达自然、筒结。因此能有效地减少开发人员的设计乃至编码阶段的工作量，提高软件的开发效率。但其缺点是，现有的甚高级语言执行效率低，达不到实用的程度。但是，在硬件效率大幅度提高的基础上，甚高级语言的效率可以逐渐达到可以被接受的程度。另外，使用广谱怕让高级语言(即混合高级语亩和甚高缎语言的一种语言)的优化功能，还可以用高级语言成份对甚高级语言成份进行优化，提高甚高级语言的运行效率。因此，从长远观点来看，甚高级语言是实现高层次软件重用的重要途径。

(2)应用生成器

应用生成器类似于程序语言的编译器，它将输入的程序规格说明自动转换成可执行的程序。应用生成器已广泛用来生成数据库统计、分析等多种用途的程序。由于应用生成器直接来源于特定应用领域中的非常高层的抽象，因此十分接近原始的系统需求。

使用应用生成器进行重用，实质上重用了整个系统的[设计过程](https://wiki.mbalib.com/wiki/%E8%AE%BE%E8%AE%A1%E8%BF%87%E7%A8%8B" \o "设计过程)。实现的算法和数据结构都是自动选择的。如果要在某一应用领域编写出多个类似的软件系统，或者在软件件生命期中系统要修改甚至重写多次，则使用应用生成器是十分有效的。因为一个应用生成器往往针对特定的应用领域，因此它的实现效率也比较高。YACC是生成器中最著名、最成功的实例之一，它是一个语法分析器生成器，根据开发人员给出的语法表示，自动生成对应的语法分析器的源代码。应用生成器是较为广泛应用和成功软件的可重用对象。不足的是，应用生成器只针对特定应用领域，开以出通用的生成器十分困难，从而限制了它的使用范围。

(3)变换系统

利用变换系统开发软件的过程是：首先用高层规格说明语言描述软件系统的需求，然后根据变换规则，将规格说明转化为可运行系统。在变换过程中，可以有人的干预。因此变换过程一般可以看作是一种交互的编译过程。变换系统根据变换规则把一个程序变换成另一程序，后者在语义上与前者相同，但在性能上要优于前者。利用变换系统，可以逐步得到运行行为越来越好的程序。

利用变换系统可以实现以下重用：(1)原型重用。在变换系统中，原型可作为系统的初始规格说明，通过不断变换和完善，成为实用的可执行系统。(2)开发经历重用。使用变换系统，开发人员的修改可以只在需求层次上进行，对于某些局部改动，原先的开发经历，即变换序列的大部分可被重用，而不须[重新设计](https://wiki.mbalib.com/wiki/%E9%87%8D%E6%96%B0%E8%AE%BE%E8%AE%A1" \o "重新设计)。(3)变换规则。变换规则由匹配模式，应用条件和替换模式构成，大部分变换规则可被重用于多种不同的情况。

变换系统是一种较为成功的可重用对象。

### 2.3.3体系结构层的软件重用

　　最有效的软件重用是在软件体系结构层的重用。

　　软件体系统结构重用是指将软件的框架组织、全局结构等作为一个整体加以重用。与软件逻辑结构相比，软件体系结构更着重于系统与各子系统，各子系统之间的相互关系而非数据结构和算法。与应用生成器相比，均是重用系统设计，但应用生成器一般只适用于特定应用领域，隐含重用体系结构的信息，而可重用软件体系结构则通常是显式重用软件体系结构，并可以通过集成其它体系结构。建立新的更高层次的体系结构。软件体系结构的抽象直接来源于应用领域，可以用领域语言描述。从领域语言描述到实现可以全部通过自动映射来实现，开发者可通过选择特定的体系结构来适应不同应用的需求。

　　软件体系结构的重用吸取了其它软件可重用对象的优点，是目前最理想的可重用软件对象。建立一个完备的软件体系结构库，以及用于支持管理体系结构构件的软件开发环境，形成一种新的基于软件重用的软件开发范型，将对今后的软件开发产生重要的影响。

## 2.4设计关键类的重点服务

账本导出模块设计关键的重点服务。顺序图设计如图2.4.1所示。

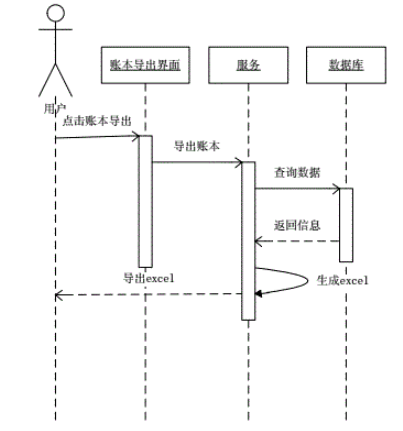


图2.4.1顺序图设计

核心处理流程设计如图2.4.2所示。

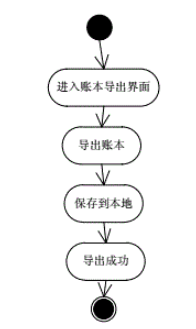


图2.4.2核心处理流程设计

用户信息管理模块设计关键的重点服务。顺序图设计如图2.4.3所示。

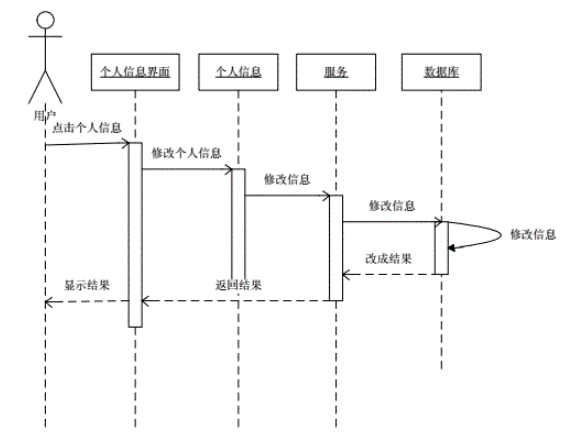


图2.4.3顺序图设计

核心处理流程设计如图2.4.4所示。

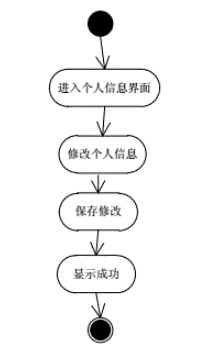


图2.4.4核心处理流程设计

# 三.账本设计界面

我们设计界面的理念是简洁，清晰，方便操作。让用户看就知道该如何操作本软件，从哪里可以找到需要的信息。另外，软件让用户能有一-种更好的使用体验，我们软件提供了提示的功能。

## 3.1界面的设计与使用

使用本软件可实现轻松记账，并使用密码来保护您的账本。账本密码会在您第一次使用本软件的时候被设置如图3.1.1所示。在记账的时候，您可以通过左滑显示数字框来为您的记录增加备注。如果您需要查阅账本，可以通过下滑或点击标题栏来进入账本页面如图3.1.2所示。

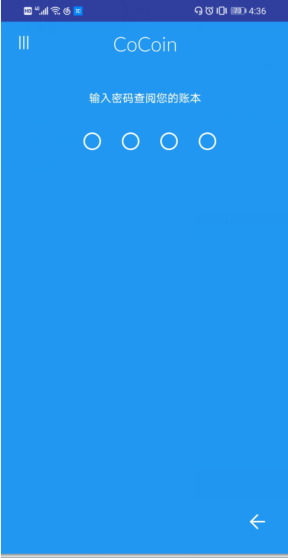
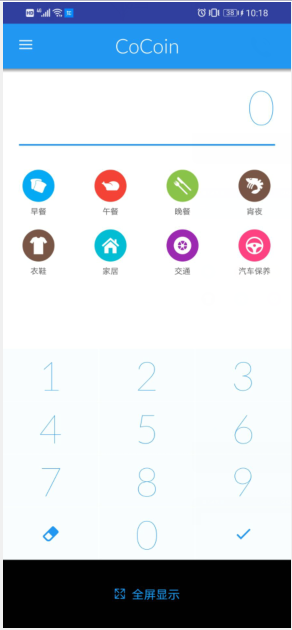
 

图3.1.1 图3.1.2

在账本中，有对今天、昨天、这周、这个月、上个月、今年和去年的记录。您可以通过点击饼状图的饼块或者柱状图的柱形来查看详细支出信息如图3.1.3所示。本软件的所图表都可以通过点击来获取详细信息，您也可以通过点击图表旁的选择按钮来对某些不易点击得到的地方进行查阅。点击或选择图表中的某一部分会有弹窗提示相关信息如图3.1.4所示，点击查看可以在弹出的对话框中看到列表，列表中即为每条记录的信息。通过点击列表中的某一条记录，您可以看到更为详细的情况。如果您希望通过更多的视图来查阅自己的支出情况，右滑打开菜单即可如图3.1.5所示。

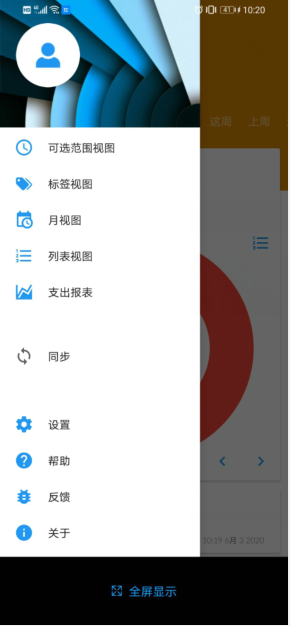
 

图3.1.3 图3.1.4  
 在自定义时间范围视图中，您可以查阅到某时间段内的所有支出情况。点击右下角的按钮即可设定时间范围如图3.1.6所示。

在标签视图中，您可以查阅到自己在不同标签上的支出情况。右滑打开菜单，您可以看到本软件的所有标签，点击标签可以跳到该标签的支出情况。在月视图中，您可以查阅到自己在不同月份的支出情况。您可以直观看到自己在某一-月份的总支出、总记录数、在不同标签上的支出和占比、具体支出的详细信息如图3.1.7所示和3.1.8所示。右滑打开菜单，您可以看到自己每月的支出概况，同样的，点击某一月份将会跳到该月份的详细信息。  
 在列表视图中，您可以进行筛选、查询、更改、删除记录的操作。左滑某一记录进行编辑，右滑进行删除。右滑打开菜单，您可以设置支出范围、时间范围、标签来筛选特定的支出记录。点击右上角的搜索按钮，可以通过备注来搜索特定的记录。  
 在报表视图中，您可以生成自己的年度支出报表或月支出报告。通过点击右下角按钮即可以选择报表的时间范围。

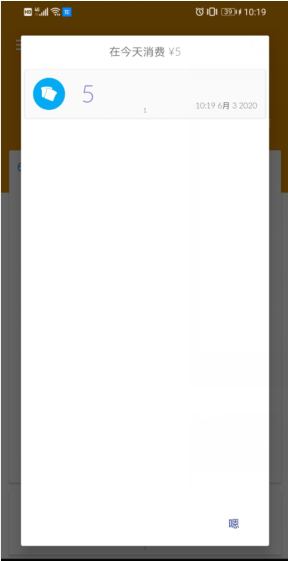
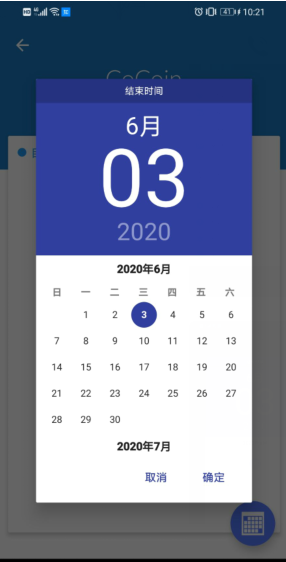
 

图3.1.5 图3.1.6  
  
 当您登录了本软件之后，您可以通过同步功能进行记录的备份。  
 您可以对自己的账本进行个性化设置。比如，您可以更改账本的名字、对标签进行排序、设置是否提醒每月支出的限额，如果您选择颜色提醒，当每月支出总和达到警告值时，记录的界面将会变色以示提醒如图3.1.9所示和图3.1.10所示。

图3.1.7 图3.1.8

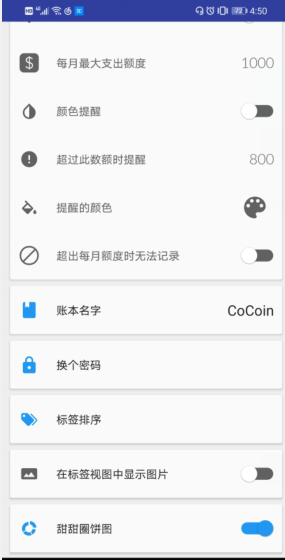
 

图3.1.9 图3.1.10

# 四.帮助设计

## 4.1 注释设计

- 包

- 公有类与接口

- 公有的和受保护的构造器及方法

- 公有的和受保护的域

应该为上面几部分编写注释，注释应该放置在所描述特性的前面。注释以/\*\* 开始，并以\*/结束。

## 4.2方法注释

每一个方法注释必须放在所描述的方法之前。除了通用标记之外， 还可以使用下面的标记：

@param 变量描述,这个标记将对当前方法的“param”（参数）部分添加一个条目。这个描述可以占据多行，并可以使用 HTML 标记。一个方法的所有 @param 标记必须放在一起。

@return 描述,这个标记将对当前方法添加“return”（返回）部分。这个描述可以跨越多行，并可以使用 HTML 标记。

@throws 类描述,这个标记将添加一个注释，用于表示这个方法有可能抛出异常。

## 4.3通用注释

下面的标记可以用在类文档的注释中：

@author 姓名 这个标记将产生一个 “author” (作者）条目。可以使用多个 @author 标记，每个 @author 标记对应一个作者

@version 文本这个标记将产生一个“version”（版本）条目。这里的文本可以是对当前版本的任何描述。

下面的标记可以用于所有的文档注释中：

@since 文本 这个标记将产生一个“since” （始于）条目。这里的 text 可以是对引人特性的版本描述。例如： @since version 1.7.10

@deprecated这个标记将对类、方法或变量添加一个不再使用的注释。文本中给出了取代的建议。通过 @see 和@link标记，可以使用超级链接， 链接到 javadoc 文档的相关部分或外部文档。

@see 引用这个标记将在“see also” 部分增加一个超级链接。它可以用于类中，也可以用于方法中。

@Override为了避免发生类型错误， 可以使用 @Override 对覆盖超类的方法进行标记，就表示这是重写父类的方法

# 五.维护设计

### 5.1改正性维护

是指改正在系统开发阶段已发生而系统测试阶段尚未发现的错误。在实际使用过程中不影响系统的正常运行，其维护工作可随时进行:而有的错误非常重要，甚至影响整个系统的正常运行，其维护工作必须制定计划，进行修改，并且要进行复查和控制。

### 5.2适应性维护

是指使用软件适应信息技术变化和管理需求变化而进行的修改。由于目前计算机硬件价格的不断下降.各类系统软件屡出不穷，人们常常为改善系统硬件环境和运行环境而产生系统更新换代的需求;企业的外部市场环境和管理需求的不断变化也使得各级管理人员不断提出新的信息需求。这些因素都将导致适应性维护工作的产生。进行这方面的维护工作也要像系统开发一样，有计划、有步骤地进行。

### 5.3完善性维护

这是为扩充功能和改善性能而进行的修改，主要是指对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征。这些功能对完善系统功能是非常必要的。另外，还包括对处理效率和编写程序的改进，这将关系到系统开发质量的重要方面。这方面的维护除了要有计划、有步骤地完成外.还要注统开发质量的重要方面。这方面的维护除了要有计划、有步骤地完成外。还要注意将相关的文档资料加入到前面相应的文档中去。

### 5.4预防性维护

预防性维护为消除设备失效和生产计划外中断的原因而制定的措施，作为制造过程设计的一项输出。从预防医学的立场出发，对设备的异状进行早期发现和早期治疗。预防性维护是软件产品交付后进行的修改，以在软件产品中的潜在错误成为实际错误前，检测和更正他们。