**汇率转换器**

**测试文档**

**小组成员：张殿海**

**张浩**

目录

[1.引言 1](#_Toc328050575)

[1.1编写目的 1](#_Toc328050576)

[1.2项目背景 1](#_Toc328050577)

[1.3系统简介 1](#_Toc328050578)

[1.4定义 2](#_Toc328050579)

[1.4.1专用术语 2](#_Toc328050580)

[1.4.2测试术语 2](#_Toc328050581)

[1.5参考资料 3](#_Toc328050582)

[2.测试概要 3](#_Toc328050583)

[2.1测试方法 3](#_Toc328050584)

[2.2测试用例设计 3](#_Toc328050585)

[2.3测试环境与配置 4](#_Toc328050586)

[3.测试结果及缺陷分析 4](#_Toc328050587)

[3.1测试执行情况与记录 4](#_Toc328050588)

[3.1.1测试组织 4](#_Toc328050589)

[3.2测试结果 5](#_Toc328050590)

[3.2.1单元测试 5](#_Toc328050591)

[3.2.2集成测试 6](#_Toc328050592)

[3.2.3软件能力评价 7](#_Toc328050593)

[3.3覆盖分析 7](#_Toc328050594)

[3.3.1需求覆盖 7](#_Toc328050595)

[3.3.2测试覆盖 8](#_Toc328050596)

[3.4缺陷的统计与分析 9](#_Toc328050597)

[3.4.1缺陷汇总 9](#_Toc328050598)

[3.4.2缺陷分析 9](#_Toc328050599)

[4.测试结论与建议 9](#_Toc328050600)

[4.1测试结论 9](#_Toc328050601)

[4.2建议 9](#_Toc328050602)

# 1.引言

## 1.1编写目的

本文档是宾馆管理系统的详细说明书。本文档基于概要设计说明书，通过小组成员讨论后,具体化系统的各个功能模块，将逻辑模型、概念模型转化为物理模型，直接指导软件开发、编码测试。

本说明书的预期读者为（1）开发人员，为代码编写和软件测试提供依据；（2）用户，增进用户对系统了解，满足用户需求。

## 1.2项目背景

本系统为汇率转换器，主要实现以下功能：汇率转换，自定义汇率

预期的目标为：

1. 基本信息完善，方便增删改查；
2. 系统运行稳定，安全可靠；
3. 界面简洁友好；
4. 针对检索功能时，实现各种关键基础数据的选择输入，避免不必要的文字输入，减少漏洞产生，提高查询效率；
5. 上述功能完备，可拓展，易维护。

## 1.3系统简介

用例图：



## 1.4定义

### 1.4.1专用术语

|  |  |
| --- | --- |
| 名称 | 解释 |
| Currency（货币） | 货币名称 |
| ExchangeRate（汇率） | 某种货币对美元汇率 |
| User | 用户 |

### 1.4.2测试术语

黑盒测试：黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。在测试中，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。

白盒测试：白盒测试也称结构测试或逻辑驱动测试，它是按照程序内部的结构测试程序，通过测试来检测产品内部动作是否按照设计规格说明书的规定正常进行，检验程序中的每条通路是否都能按预定要求正确工作。 这一方法是把测试对象看作一个打开的盒子，测试人员依据程序内部逻辑结构相关信息，设计或选择测试用例，对程序所有逻辑路径进行测试，通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。

集成测试：集成测试，也叫组装测试或联合测试。在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。程序在某些局部反映不出来的问题，在全局上很可能暴露出来，影响功能的实现。

## 1.5参考资料

|  |  |  |  |
| --- | --- | --- | --- |
| 文档名称 | 版本/日期 | 作者或来源 | 备注 |
| 《软件工程——原理、方法与应用》 |  | 史济民、顾春华、郑红著 |  |
| 《面向对象设计UML实践》 |  | Mark Priestley |  |
| 《Java程序设计实用教程》 | 第二版 | 人民邮电出版社 |  |
| 需求分析说明书 |  |  |  |
| 概要设计说明书 |  |  |  |
| 详细设计说明书 |  |  |  |

# 2.测试概要

## 2.1测试方法

综合应用黑盒测试和白盒测试。

## 2.2测试用例设计

使用以下方法进行测试用例设计：

1. 边界值：大量的错误是发生在输入或输出范围的边界上，针对边界情况设计测试用例，查出更多的错误结果；
2. 等价类划分：用少量代表性的测试数据取得较好的测试结果；
3. 错误推测方法：基于经验和直觉推测程序中所有可能存在的各种错粗，有针对性地设计测试用例；
4. 因果图：利用图解法分析输入的各种组合情况，设计测试用例，检查程序输入输出。

如下测试用例：

“电话”测试用例（有效为7或11位数字字符）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入等价类 | 测试用例说明 | 测试数据 | 期望结果 | 选取理由 |
| 电话 | ①1个数字字符 | 1 | 输入无效 | 仅有一个合法字符 |
| ②6个数字字符  ③10个数字字符 | 123456  1234567890 | 比有效长度恰少一个字符 |
| ④8个数字字符  ⑤11个数字字符 | 12345678  12345678901 | 比有效长度恰多一个字符 |
| ⑥有1个非数字字符 | 123456x | 非法字符最少 |
| ⑦全是非数字字符 | zxcvbnm | 非法字符最多 |
| qwertyuiopa |
| ⑧7个数字字符 | 1234567 | 输入有效 | 类型与长度均有效的输入 |
| ⑨11个数字字符 | 12345678901 |

⑩漏输“电话”，预期结果：无法后继（错误推测法）

## 2.3测试环境与配置

处理器：Intel(R) Core(TM) i3 CPU 2.27GHz

内存容量：2.00GB

硬盘空间：450G

操作系统：WindowsXP/Win7

JDK：JDK 1.6

数据库：MySOL5.1

# 3.测试结果及缺陷分析

## 3.1测试执行情况与记录

### 3.1.1测试组织

|  |  |  |
| --- | --- | --- |
| **测试机构** | **负责人** | **参与测试人员** |
| 汇率转换器 | 张殿海 | 张浩 |
| 张殿海 |
|  |
|  |

## 3.2测试结果

### 3.2.1单元测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试单元** | **功能** | **输入项** | **期望结果** | **实际结果** | **正确性** | **可操作性** |
| Currency  Converter | 汇率转换 | 待转换货币数量，待转换币种，目标币种 | 将目标货币数量显示在界面上 | 将目标货币数量显示在界面上 | 是 | 优 |
| Modfiy | 自定义汇率 | 汇率，币种 | 汇率写入数据库 | 汇率写入数据库 | 是 | 优 |

### 3.2.2集成测试

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试模块** | **文件相关** | **包含功能** | **输入项** | **期望结果** | **实际结果** | **正确性** | **可操作性** |
| 汇率转换模块 | Main.java | 汇率转换 | 待转换货币数量，待转换币种，目标币种 | 将目标货币数量显示在界面上 | 将目标货币数量显示在界面上 | 是 | 优 |
| 自定义汇率模块 | modify.java | 自定义汇率 | 汇率，币种 | 汇率写入数据库 | 汇率写入数据库 | 是 | 优 |

### 3.2.3软件能力评价

|  |  |  |
| --- | --- | --- |
| **功能及性能** | **要求** | **测试结果** |
| 基本功能 | 实现汇率转换功能 | 功能全部实现，测试通过，  此部分测试结果为优。 |
| 附加功能 | 汇率的增删改查 | 实现了一部分附加功能，此部分测试结果为良。 |
| 容错性 | 具有良好的容错性，输入错误，以及操作错误时给出警告 | 在部分功能上实现了对错误信息的检测和提示，但是仍有一些功能没有对检测错误信息。此部分测试良。 |
| 可操作性 | 操作简单，易于掌握 | 大部分操作简单，但个别功能需要指导说明，此部分测试结果为良。 |
| 效率 | 相应迅速，数据传递迅速 | 满足要求，测试结果优。 |
| 综合 |  | 用户基本需求完成得比较好，附加功能基本达标，但对照小组原有需求还有一定差距。 |

## 3.3覆盖分析

### 3.3.1需求覆盖

|  |  |  |  |
| --- | --- | --- | --- |
| 需求 | 测试类型 | 是否通过 | 备注 |
| 汇率转换 | 集成测试 | Y |  |
| 自定义汇率 | 黑盒测试 | Y |  |

Y表示通过，P表示部分通过，N/A表示不可测试或者用例不适用

### 3.3.2测试覆盖

|  |  |  |  |
| --- | --- | --- | --- |
| 需求 | 用例个数 | 执行总数 | 未/漏测分析和原因 |
| 汇率转换 | 集成测试 | Y |  |
| 自定义汇率 | 黑盒测试 | Y |  |

## 3.4缺陷的统计与分析

### 3.4.1缺陷汇总

1. 无法从网络实时读取汇率。

### 3.4.2缺陷分析

1. 需求定义不明确。需求文档中，存在功能描述不明确，导致开发人员不按照需求进行设计而根据自己的经验来进行设计。
2. 功能性错误。有功能（附加功能）没有实现，导致无法进行需求规定的功能的测试。
3. 页面设计易用性缺陷。操作提示和错误提示信息不完善。
4. 参数的定义相对混乱，导致最后各模块合并的时候出现较大困难。

# 4.测试结论与建议

## 4.1测试结论

1. 测试执行充分，主要是功能性测试。
2. 测试目标完成。获取了当前开发软件的质量水平，指导后继开发。
3. 测试通过。所有基本功能都较好实现。
4. 本次课程设计结束，基本要求达成。但是由于没有实现需求中的所有功能，仍可进行下一阶段项目开发，目标为总结当前设计，继续完善系统。

## 4.2建议

1. 在项目开始的时候应该制定编码标准，数据库标准，需求变更标准，开发和测试人员都严格按照标准进行，可以在后期减少因为开发、测试不一致而导致的问题。
2. 开发人员解决bug的时候，填写bug原因以及解决方式，方便bug的跟踪。
3. 开发人员在开发版本上发现bug，应当通知测试人员，保证发现的bug都能被跟踪。