软件工程课程设计

**仓库管理系统技术说明**

**目录**

1. **需求分析**
2. **系统设计**
3. **程序实现**
4. **测试**
5. **需求分析**
6. 设计目的

熟悉软件开发流程，将学和用联系起来。

1. 运行环境

Windows10系统

QT Creator编译器

C++语言

1. 系统功能需求

仓库管理系统

进货

出货

增加库存

新增商品

减少库存

删除商品

查询

显示库存

查找商品

退出系统

清空

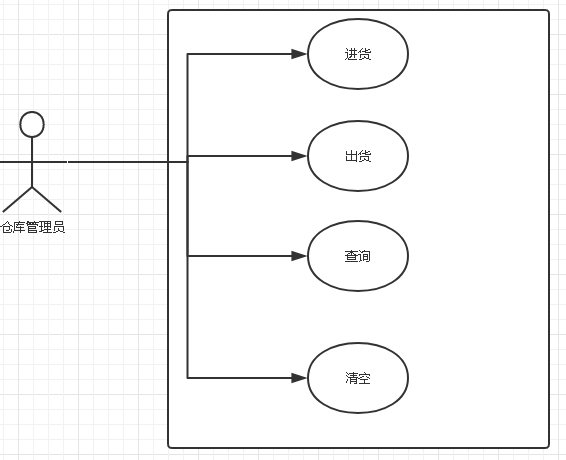
约束说明：

|  |  |  |
| --- | --- | --- |
| **功能** | **概述** | **约束** |
| 进货 | 商品编号 | 少于100个字符；不为空；只能是整数； |
| 商品名称 | 少于100个字符；不为空 |
| 商品数量 | 只能是正整数；不能是空；最大值999999999 |
| 出货 | 商品编号 | 和进货中相应约束相同 |
| 商品数量 |
| 查询 | 商品编号 | 任何一个可以为空；且输入格式没有限制 |
| 商品名称 |
| 清空 | 询问是否确定清空 | 只有用户点击确认后才能清空仓库 |
| 退出系统 | 询问是否确定退出 | 只有用户点击确认后才能退出系统 |

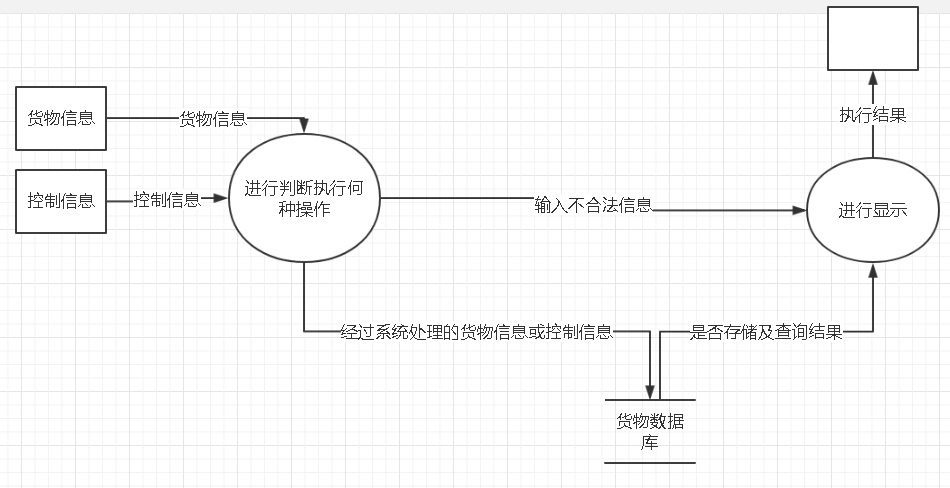
逻辑说明：

* + 1. 在出货时，如果所有货物已经全部出货，要提示用户“所有货物已出货”，并删除该货物在数据库中的记录；如果出货数量大于库存，则出货失败。
    2. 上面的查询包含两个子功能，显示库存和查找商品，表格中说明的是查找商品的约束条件，显示库存不需用用户输入其他信息。
    3. 每类货物有一个ID，ID相同被视为同一类
    4. 查询功能两个查询条件是或的关系，即只要输入ID或者name就可以查询，如果两个都输入那么将会输入ID匹配以及name匹配的所有货物。
    5. 仓库容量暂定1500

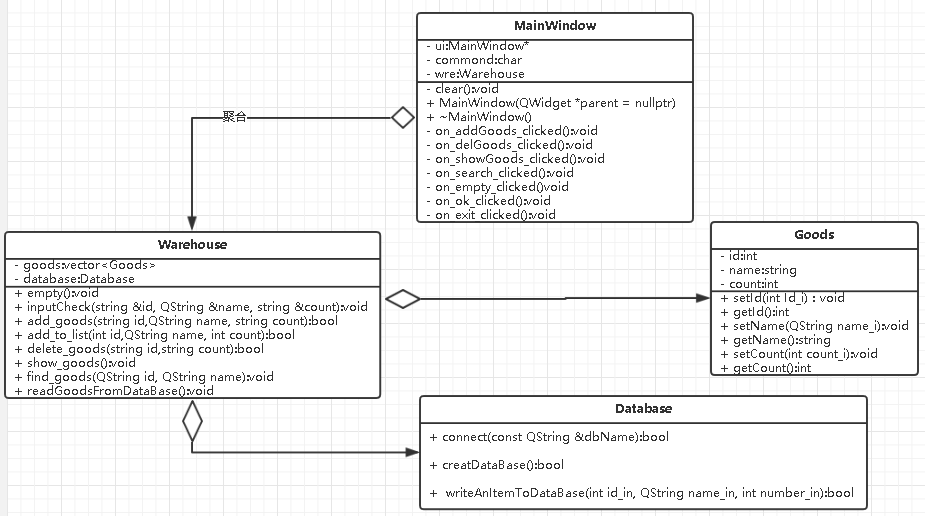
1. 用例图



1. 数据流图



1. **系统设计**
   1. 类图



* 1. MVC模式说明

由于系统比较简单，只用到了四个类，类之间的关系也很简单。

* + 1. MainWindow负责用户界面，即view，获取用户输入或将执行结果显示给用户，在仓库管理系统中，它主要是获取用户指令是进货、出货、查询还是清空仓库，并根据用户不同的输入来改变界面以适合相应的操作。然后如果是进货、出货或者查询，还需要用户输入货物的相关信息。等这些信息都录入完毕，它就向下层发送相关指令，进行处理。最后等下层处理完后，会返回处理结果，MainWindown在负责把处理结果显示给用户。
    2. Warehouse负责逻辑处理，即control, 根据用户的输入，进行判断，并将需要进行数据库操作的信号传给下层，同时负责把下层处理结果传递给上层。
    3. Database负责数据库操作，即modle,接收从上层传来的数据，并按照指令更新数据库或返回相应数据。
    4. Goods类是表示货物的实体类。是为了方便表示而存在。
  1. 异常处理
     1. 在程序中使用try，throw,catch异常处理方法。在遇到系统无法处理的异常情况时，会直接抛出异常，显示给用户，部分异常抛出信息中有对用户操作的建议，比如“物品Id和name不匹配”。
     2. 同时为了避免用户输入导致系统奔溃，收到用户输入后在Warehouse类中进行检查其是否符合上表中的约束，在保证安全后在传给下层。
  2. 安全性

由于时间有限，在系统实现时，这方面并没有进行过多的考虑，在后续进行时可从以下方面考虑。

* + 1. 设计登录页面，限制访问，不能任何人都可以访问。
    2. 对数据库内的数据进行加密后存放在本地磁盘。并将解密密钥和用户口令关联起来，只有用户正常登录，系统通过用户口令和相关加密算法来查看数据。如果不通过系统，直接打开数据库文件，显示已加密数据。
    3. 为保证绝对安全，对数据库文件设置访问权限。

1. **程序实现**
   1. 学习google编程风格，并尽量达到要求
      1. 头文件以.h结尾
      2. 所有头文件都应该使用 #define 来防止头文件被多重包含
      3. 尽可能地避免使用前置声明。使用 #include 包含需要的头文件即可
      4. 将函数变量尽可能置于最小作用域内, 并在变量声明时进行初始化.
      5. 如果你的类型需要, 就让它们支持拷贝 / 移动. 否则, 就把隐式产生的拷贝和移动函数禁用.

……………….

* 1. 在实现时尽可能让程序高效运行。
  2. 在每个函数开始的位置说明其用途，方便后续维护
  3. 尽量减少各个模块之间的耦合，降低系统复杂性，提升可维护性。

1. **测试**
   1. 黑盒测试

根据需求中的约束说明进行测试：

* + 1. 进货模块的等价类划分：

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 商品编号 | a.少于100个字符且不为空且只能是整数； | A．大于100个字符；  B. 为空；  C．包含非数字字符； |
| 商品名称 | b. 少于100个字符且不为空 | D．大于100个字符；  E. 为空; |
| 商品数量 | c.只能是正整数且不能是空且最大值999999999 | F. 负数  G．小数  H. 为空  I. 最大值超过999999999 |
| 逻辑测试 | d.相同编号相同名称;  e.不同编号相同名称;  f.不同编号不同名称； | J. 相同编号不同名称； |

测试用例的设计：

|  |  |  |
| --- | --- | --- |
| **测试数据** | **测试范围** | **期望结果** |
| 编号：123  名称：apple  数量：2 | 有效等价类a；  有效等价类b;  有效等价类c; | 有效 |
| 编号：  名称：apple  数量：2 | 无效等价类A; | 无效 |
| 编号：111…（大于100个1）  名称：apple  数量：2 | 无效等价类B; | 无效 |
| 编号：abc123  名称：apple  数量：2 | 无效等价类C; | 无效 |
| 编号：124  名称：aaa…(大于100个a)  数量：2 | 无效等价类D | 无效 |
| 编号：124  名称：  数量：2 | 无效等价类E | 无效 |
| 编号：125  名称：banana  数量：-2 | 无效等价类F | 无效 |
| 编号：125  名称：banana  数量：0.2 | 无效等价类G | 无效 |
| 编号：125  名称：banana  数量: | 无效等价类H | 无效 |
| 编号：125  名称：banana  数量:100000000 | 无效等价类I | 无效 |
| 编号：123  名称：apple  数量：3 | 有效等价类d | 有效 |
| 编号：126  名称：apple  数量：4 | 有效等价类e | 有效 |
| 编号：127  名称：onion  数量：4 | 有效等价类f | 有效 |
| 编号：123  名称：potato  数量：4 | 无效等价类I | 无效 |

* + 1. 出货模块的等价类划分：

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 商品编号 | a.少于100个字符且不为空且只能是整数 | A．大于100个字符；  B. 为空；  C．包含非数字字符； |
| 商品数量 | c.只能是正整数且不能是空且最大值999999999 | D. 负数  E．小数  F. 为空  G. 最大值超过999999999 |
| 逻辑测试 | d.存在的编号数量小于等于库存 | H. 不存在的编号  I. 存在的编号数量大于库存 |

测试用例的设计：

商品编号和商品数量的测试用例可以直接使用进货模块的数据。

下面的测试用例是在假设进货模块测试完成后数据库中已经存在下面的记录的前提下进行的。

编号：123

名称：apple

数量：2

|  |  |  |
| --- | --- | --- |
| **测试数据** | **测试范围** | **期望结果** |
| 编号：123  数量：1 | 有效等价类d | 有效 |
| 编号：123456  数量：1 | 无效等价类H | 无效 |
| 编号：123  数量：3 | 无效等价类I | 无效 |

* + 1. 查找商品模块的等价类划分：

|  |  |  |
| --- | --- | --- |
| **输入条件** | **有效等价类** | **无效等价类** |
| 逻辑测试 | 1. 存在的编号不存在的名称 2. 不存在的编号存在的名称 3. 存在的编号存在的名称 | 1. 不存在的编号不存在的名称 |

测试用例的设计：

下面的测试用例是在假设进货模块中所有的有效测试用例都已经在数据库中的前提下进行的。

|  |  |  |
| --- | --- | --- |
| **测试数据** | **测试范围** | **期望结果** |
| 编号：123  名称：zzzzzz | 有效等价类a | 有效 |
| 编号：999999  名称：apple | 有效等价类b | 有效 |
| 编号：127  名称：apple | 有效等价类c | 有效 |
| 编号：99999999  名称：zzzzzzzzz | 无效等价类A | 无效 |

* 1. 白盒测试

由于程序比较简单，并且自己写的程序自己来测试，所以在黑盒测试基本能够完成对所有情况的测试，有些测试，比如仓库容量小于1500目前人工测试工作量巨大，没有完成。

* 1. 系统测试

在完成上面各个模块的测试工作后，还需一步系统测试，这里主要测的是数据的持久化，退出系统重新进入，看上次存储的数据是否还在。

系统UI：

