**“C++程序设计与训练”课程大作业**

**项目报告**

**项目名称：超市商品管理系统**

**姓名：**安亮

**学号：**2013011550

**班级：**自36

**日期：**2014年9月24日

目 录

[1系统功能设计 3](#_Toc364862629)

[2系统总体结构 3](#_Toc364862630)

[3本人工作内容 3](#_Toc364862631)

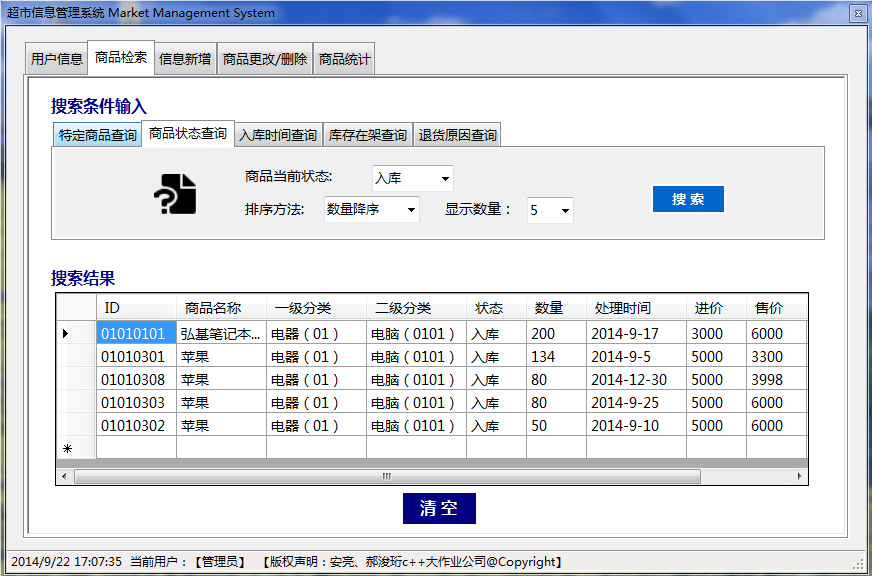
[4项目总结 3](#_Toc364862632)

[5相关问题的说明 4](#_Toc364862633)

# 1系统功能设计

本项目实现日常生活常见的超市商品管理系统。根据现实超市对商品的管理以及客户对功能系统的需要，结合我们的功能界面，我们实现了对超市管理系统的如下功能需求：（基本界面如下图）

**写在前面：**本系统功能实现数据上与给出数据的框架有所不同，原给数据库代表一件特定商品的信息和状态，经过助教老师理解并允许，我们做了如下改进：每条信息代表一次商业操作，包括处理该批货物的状态和处理货物的数目。在生活中更符合营业统计原则，有利于准确统计商品状态转换关系。



**基本功能：（课程作业的基本要求）**

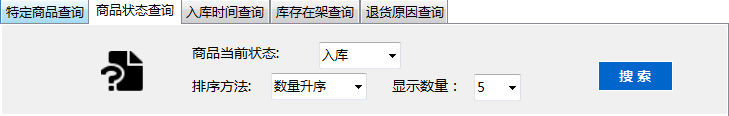
1、“商品查询”

此部分提供多方面的多种查询功能（具体功能如下图显示，完整功能参见如上界面示意图），包括对商品名称、条码，状态和时间的分类及综合排序，退货商品查询等5大项7个功能的实现。

1）特定商品查询：依次在下拉框中选取一级分类，二级分类，数据库加载特定商品名到商品名称中（控件只读Readonly），点击查询显示全部该商品信息。

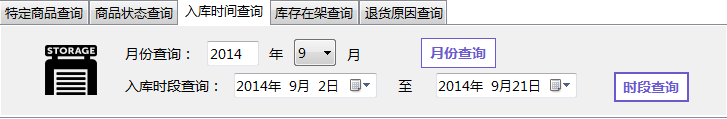
（补图）

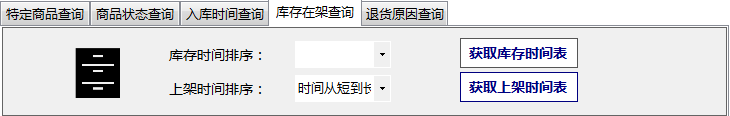
2）商品状态查询：选择商品状态（入库、上架、售出、退回），排序方法和显示数量，全部选择值后才可搜索，按照要求输出相关条目数出。注意：此项得到的不是出于当前状态的数目，而是处理过的相关条目的记录信息，详见数据库相关参数的规定。

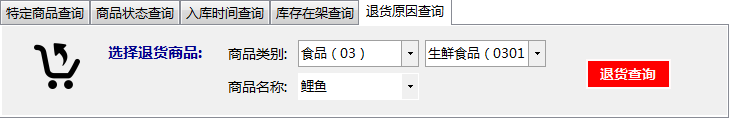
3）商品入库查询：分为两个功能（按时间详尽查询入库操作）：

（1）输入月份，查询在该月份的入库操作记录信息；

（2）选择规定时起止时间，查询该时间段（包括起止天）的入库商品完整信息。

4）库存在架查询：点击对应查询入库、上架的全部商品的时间顺序

5）退货查询：同功能1）选择商品种类，点击查看该商品的全部退货信息。



该部分查询功能分5个选项卡，超出题目要求进行详尽多方面（尤其是入库方面信息）的查询。

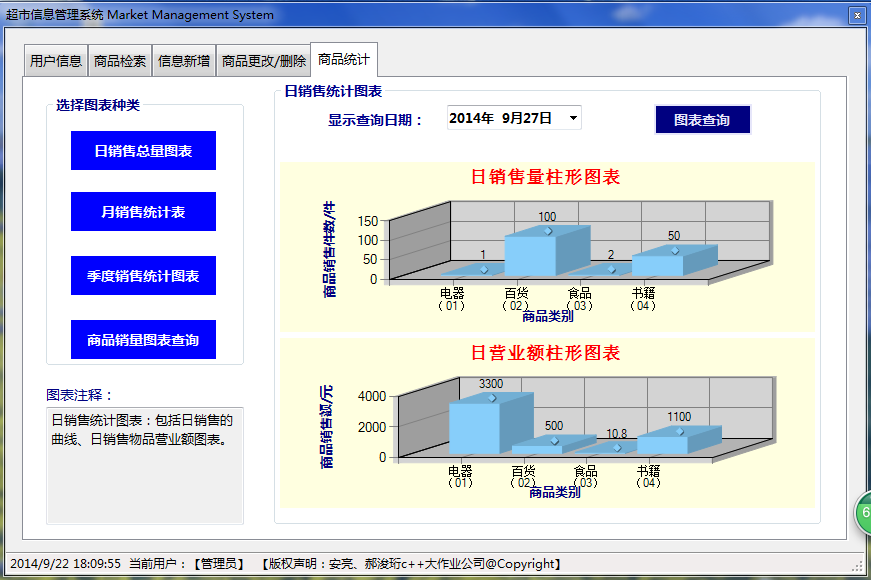
2）“信息新增”

左栏输入一级二级分类可以查询已有的商品供用户提供参考。右侧按照信息依次输入（其中商品条码是自动生成的，不必填写），对四种状态可以添加新的信息，也可以添加新的商品。在除了入库项目之外，其他项目的新建信息都有可行性检查（比如新建上架小于已有库存等问题都不允许操作），商品是否添加成功有弹窗提醒。（截面图如下）

3）“商品更改/删除”

上栏输入一级二级分类或直接输入条码可以查询已有的商品，在下面的列表窗口中供用户提供参考。下侧按照信息依次输入（其中商品条码是自动生成的，不必填写），对四种状态可以更改相关，也可以删除新的商品。更改信息都有可行性检查（比如更改后上架小于已有库存等问题都不允许操作），商品是否添加成功有弹窗提醒。（截面图如下）

4）“商品统计”：主选项按钮实现了4个分区，共9项统计功能，主要是**日期选择查询（根据用户输入获取该日信息）、月（季度）销售额和利润值统计，特定商品（输入特定商品）销售曲线，最大销售额商品和最大利润商品列表及图表**。界面显示如下：

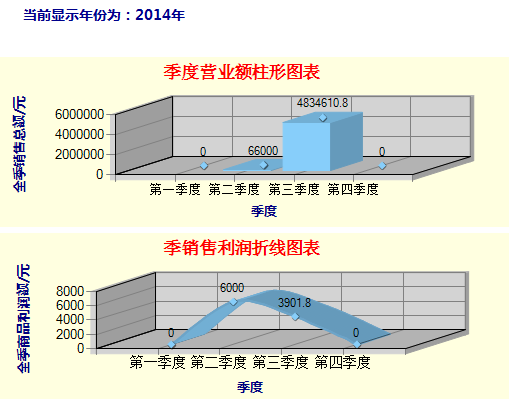
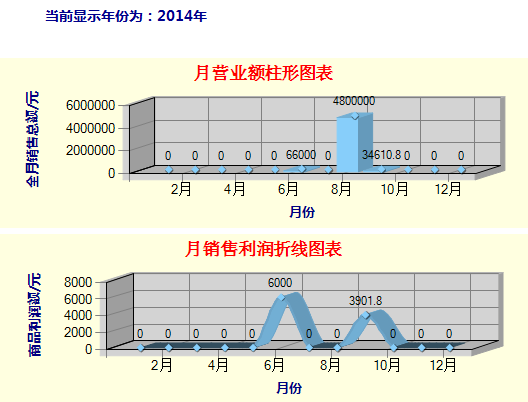


日销售图表：调查当天出售商品四大类别的销售件数（立体柱形图）和类别的销售营业总额（立体柱形图）。

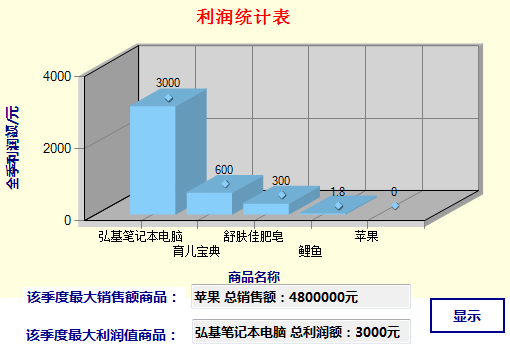
同理有以下种类图表信息生成：

月销售图表：（下左图）显示全年月营业总额柱形变化和利润总额的折线变化。

季销售图表：（下右图）显示全年季度营业总额柱形变化和利润总额的折线变化。







商品销量图表：

功能1：一次选择输入特定商品和季度，显示该季度的销售件数曲线图。

功能2：点击显示，可以得到第N季度（上方已选择）的最大营业额值和最大利润值的对应商品，同时图表显示为所有商品自左向右依次利润递减的商品名称和利润值。

这些功能系统而全面、符合财务信息统计方式地实现了管理人员对超市商品流通状况、商品盈利情况和销售变化趋势的了解。

**改进与新增功能：（创新功能实现）**



功能具体如下介绍：每一部分功能均在新的窗体中尽心，

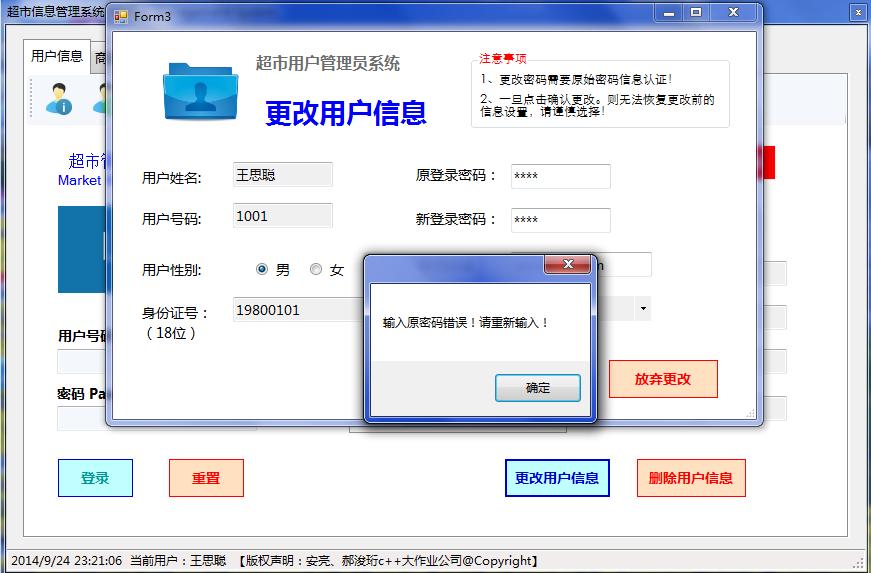
1. 登录功能：输入用户号码和密码（不合理输入、错误输入会有弹窗提醒），登录后会显示登录成功信息和“退出登录”等按键，同时工具栏部分小图标可使用。右侧功能栏会提供当前时间日历和部分非重要信息。登录后示意图如下：特别注意：登录后会自动读取用户的身份，对“管理员”“营业人员”开放不同的权限。



1. 用户信息详细显示：将用户信息（除密码外）的所有信息均显示出来。登录后才可以使用，否则显示空内容。



1. 更改信息：左侧四项为默认的无法更改的字段项，右侧可以更改密码，邮箱和身份验证。对原输入密码和新输入密码的都有相应的判断。



1. 新增信息：在未登录和登陆的条件下，都可以新增用户信息，包括输入两次密码的确认选项等容错性特点。新增信息时默认状态是未登录，注册后仍需要到登录界面登录。



1. 删除信息：删除之前会有提示确认，删除过程先登录后删除数据库信息。

# 2系统总体结构-以封装为核心

表1 小组成员分工说明

|  |  |  |  |
| --- | --- | --- | --- |
| 成员姓名 | 小组成员班级 | 小组成员学号 | 小组成员分工 |
| **安亮** | 自36  (50%) | 2013011550 | 数据库与前端界面项目连接；  数据库基础读写功能的基本实现；  商品新建代码实现；  商品更改/删除功能代码实现；  新建更改信息与已有信息的逻辑判断，限制客户操作数据范围；  类库及相关函数的基本框架；  客户界面UI进一步完善与提升；  界面部分控件的设置； |
| **郝浚珩** | 自36  (50%) | 2013011546 | 数据库逻辑与信息字段设计；  系统功能框架设计；  类库及相关函数的补充完善；  客户界面UI设计与美化；  应用空间的属性、事件设置与用户界面的动态事件实现；  全部商品查询功能与商品信息列表输出功能；  商品信息统计功能代码实现及动态显示绘制图表；  客户输入限制与容错性完善；  报告基本系统模块功能的文字介绍 |

【**语言与平台及其他注意事项】**

C#语言，Access数据库。以Visual Studio 2012/2013为开发平台，可以运行于win7,win8,win8.1操作系统。请将market.mdb文件放在E盘根目录下，即E:\market.mdb。

【**命名规则】**

在这个工程中，方法（函数）采用camelCasing命名方法，私有字段（变量）也采用camelCasing命名法，而公有的属性（在此工程中是私有变量的代言人）采用PascalCasing命名方法。类名一般采用PascalCasing命名法，有的特殊，但也会保证整齐美观容易辨识。临时变量用camelCasing命名或者直接全部小写。

**【核心逻辑\*数学模型】【重要】**

本工程在逻辑设计上有点特殊，因此导致每种商品的类的结构与样例数据中不一样。我们的数学模型更加符合市场的状态，且易于商品统计。

结构上，有如下特点：

1. 我们的每条数据记录表示的不是一个商品，而是一批商品，用(Item)item.Num数量属性表示该批商品的数量，在数据库中体现为字段“批次处理商品数目”。
2. “状态”相应地也不再是一件商品的状态，而是一批商品的操作。ID是这次操作的唯一标示。条形码仅仅与商品的名称有一一对应的关系。
3. 不再有“入库时间”、“售出时间”等说法，而是统一成了“状态处理时间”，即如果该批次商品的状态（操作）为“入库”的话，这个“状态处理时间”就是入库时间，以此类推。
4. 同一种商品（名称相同，也即条形码相同）的进价和售价始终相同且不变。

有了这样的结构，就有如下逻辑：

1. 不再也无法跟踪每一件商品的状态改变，数据库里存储的一条条操作记录也没法改变“状态”，不然会出现逻辑问题。
2. “新增”这个操作是在数据库中以新的ID增加了一条操作记录。比如，同一种商品（即具有相同条形码的商品）的入库和上架操作是各自新增一条记录，而不是改变原有记录的“状态”。这样不会消除用户的入库信息，方便统计。
3. 始终会有同种商品的入库总数>=上架总数>=售出总数>=退回总数。这样符合客观实际。
4. 上架操作的商品必须已经有库存，也就是该商品的“入库”状态必须有记录，且这些记录的“批次处理商品数目”的和必须大于等于这次操作准备上架的商品的数目。同样的，售出时必须有足够上架，退回时必须有足够售出，逻辑和上架操作相仿。
5. 删除记录时必须判断删除后是否依旧符合第3条逻辑（黄色部分）。
6. 更改信息只能更改数目，此时需要加入一个判断，判断改动是否符合第3条逻辑。

**【工程的层次】**

程序的结构可以分为三层：

其中，数据库的结构依赖于对商品的内存存储结构的设计，而逻辑的实现则依赖于界面的需求。因此，整个工程的实现可以分为5个部分：

1. Access数据库market.mdb，用来存储数据，是这个工程的底层。
2. MDBmanip.dll模块，封装了对数据库的操作，为界面逻辑的编写提供统一的易用的接口。这部分连接了数据库和逻辑层。
3. ItemInfo.dll模块，封装了商品的所有的类。这些类的对象是从界面上的数据到数据库中的数据的转换中介。从数据库中读出的数据将以对象数组的方式提供给需要数据的函数，经函数处理转换为界面上的输出数据；从界面上读出的数据将以对象的形式提供给操作数据库的函数（封装在MDBmanip.dll中的函数），由它们将数据写入数据库。这个模块是核心也是桥梁。
4. 界面背后的逻辑结构。即通过单击按钮、下拉选择组合框元素、改变文本框内容而触发的相应操作的逻辑。这部分连接了逻辑层和界面。这部分负责工程功能的具体实现。
5. 界面，用户所能见到的逻辑表现，也是系统功能的体现。是整个工程的前端。

【market.mdb说明】

这部分用来存储数据。包含两个表：BaseInfo和UserInfo 。BaseInfo用来存储商品的信息，UserInfo用来存储用户的信息。它们的结构与商品的类的定义和用户的类的定义直接相关。

比较如下：

|  |  |
| --- | --- |
| BaseInfo中的字段 | Computer等类的属性（字段的对外表现） |
| ID【短文本】【主键】 | ID【string】 |
| 条形码【短文本】 | StripCode【string】 |
| 一级分类【短文本】 | FirstClass【string】 |
| 二级分类【短文本】 | SecondClass【string】 |
| 商品名称【短文本】 | ItemName【string】 |
| 状态【短文本】 | State【string】 |
| 状态处理时间【时间】 | StateDate【Date】 |
| 状态处理季度【整形数字】 | Season【int】 |
| 商品进价（元）【双精度数字】 | PurchasePrice【double】 |
| 商品售价（元）【双精度数字】 | SalePrice【double】 |
| 批次处理商品数目【整形数字】 | Num【int】 |
| 退回原因【短文本】 | ReturnReason【string】 |
| 附加信息1【短文本】 | 继承于Item类的四大类的新增信息 |
| 附加信息2【短文本】 | 继承于四大类的13最远派生类的新增信息 |

其中有几个概念需要解释：

一级分类：包含四种，电器（01），百货（02），食品（03），书籍（04）。

二级分类：目前包含13种，对应情况如下：

|  |  |  |
| --- | --- | --- |
| 一级分类 | 二级分类 | 对应的类 |
| 电器（01） | 电脑（0101）  家用电器（0102）  移动电子设备（0103） | Computer  Appliance  Mobile |
| 百货（02） | 日用品（0201）  玩具（0202）  体育用品（0203） | Daily  Toy  Sports |
| 食品（03） | 生鲜食品（0301）  果蔬食品（0302）  干货（0303） | Fresh  FruitVegetable  DryFood |
| 书籍（04） | 育儿（0401）  美容（0402）  文学（0403）  历史（0404） | Parenting  Cosmetology  Literature  History |

状态：包括“入库”，“上架”，“售出”，“退回”四种状态。

状态处理时间：表示该状态发生的时间，比如状态是“入库”，则表示入库时间；状态是“退回”，则表示退回时间。

状态处理季度：对应于“状态处理时间”的一个概念。表示这个时间的季度。

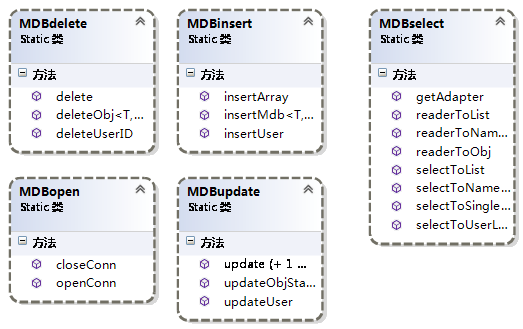
|  |  |
| --- | --- |
| UserInfo中的字段【短文本】 | User类里的属性【string】 |
| 用户姓名 | Name |
| 用户号码【主键】 | Number |
| 用户性别 | Gender |
| 身份证号 | ID |
| 密码 | Password |
| 邮箱 | Mail |
| 身份验证 | Character |
| 登陆 | Login |

用户号码是一个四位的数字组成的字符串，用作用户的唯一标识。

操作market.mdb的函数封装在了MDBmanip.dll中，一般情况下，为了工程有一个清晰的、易于调试的层次结构，不允许跳过MDBmanip中的函数而直接在界面控件对应的代码里直接操作数据库。但由于代码是两个人合作，而且有些地方在写的过程中不太注意，就会出现极个别特殊情况。

【MDBmanip.dll说明】

这个模块用来操作数据库，相当于数据库的对外接口。类图如下：



这里面定义了5个静态类，每个类里封装了相应的操作。作用是使面向数据库的操作集成化，方便使用和修改。同一类的操作封装在同一个类里，方便查找和使用。

【ItemInfo.dll说明】

这部分是商品类的定义部分。

由于类图太大，看不清，所以随报告附上一份原图。具体每个函数的实现方法参见安亮同学的“本人工作内容”。类图剪切截图参见安亮的“本人工作内容”。

【界面逻辑】

依托于界面上的功能和控件，借助ItemInfo和MDBmanip两个命名空间，完成对market.mdb的操作。

【界面】

系统功能的体现，也是所有底层和逻辑所服务的对象。详情见报告第一部分“系统功能设计”。

# 3本人工作内容

这是进行复杂软件开发的第三步，即详细设计。此部分需要**针对自己的工作内容**说清楚具体的模块是如何设计和实现的，类是如何具体实现的，类中的重点方法和算法是如何设计和实现的，界面是如何设计的，容错功能是如何设计的，典型功能的逻辑处理流程以及各种设计思路等等，可能需要用到UML对象图、UML状态图、UML序列图和流程图等图形化工具。详细设计是编写代码前的最后一步设计工作，因而需要在需求分析和概要设计的基础上，说清楚**所有需要在编码前明确的设计事项**。

（此部分的子标题和结构自行拟定。此部分内容每人是不一样的，不可共用。）

【我的核心思想】

封装。完成了两个最核心dll的95%的工作。

界面上，完成了两个实现核心逻辑的界面——新增和更改删除。这两个的逻辑最为精细复杂。

【概况】

我的主要工作由五部分组成：

1. ItemInfo类库（ItemInfo.dll模块）的编写
2. MDBmanip类库（DataBaseManip.dll模块）的编写
3. “商品新增”和“商品更改/删除”页面功能实现代码的编写
4. 用户界面部分代码
5. 数据库market.mdb结构的设计

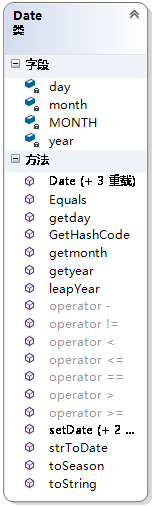
【详细分析】

1. ItemInfo类库

类库包含8个文件：

1. BasicType\_date.cs

这个文件只定义了一个类Date，类中包含的内容如下：



其中，重载了六个比较运算符，为了在排序的时候可以直接对Date对象进行大小比较。为了重载==和!=（C#中规定比较运算符的重载必须成对），又必须重载了Equals成员函数和GetHashCode成员函数。其中，GetHashCode成员函数的算法来自于一个学习网页<http://stackoverflow.com/questions/263400/what-is-the-best-algorithm-for-an-overridden-system-object-gethashcode>

由于我写代码的疏忽，没有设计与year、month、day相对应的属性来完成对这些私有数据成员的赋值和取值，所以队友郝浚珩为我添加了+getyear():int、+getmonth():int、+getday():int三个函数来取出三个变量的值。

<<readonly>>+MONTH[13]存放每个月对应的天数

为了方便一些功能的实现，定义了3个静态函数：

<<static>>+strToDate(string):Date

实现从一个“\*\*\*\*#\*\*#\*\*”形式的字符串中读出日期的函数。#可以是任意的非数字字符或汉字。

<<static>>+leapYear(int):bool

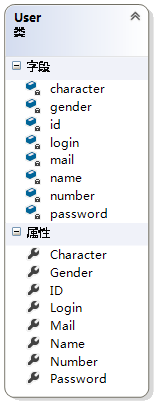
判断实参year是否是闰年

<<static>>+toSeason(Date):int

根据输入的日期判断其季度，返回值为1、2、3、4代表四个季度

1. User.cs

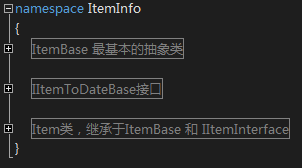
在这个文件中定义了一个类User，内容如下：



所有的字段都是字符串，且是私有的。所有属性也是字符串，都是公有的。它们与数据库结构的对应关系参见前面第二部分的分析。

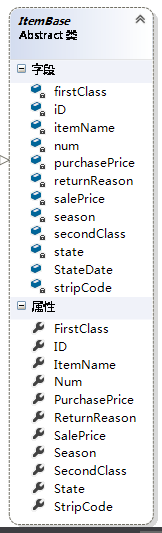
1. Item.cs

包含3个内容：



从图中可以看出3个类，他们之间的关系是：Item类继承于抽象基类ItemBase和接口IItemToDateBase。

【ItemBase】中只定义了私有字段和公有属性。如图：



有个特例：+StateDate:Date为公有字段。这一点完全可以改用和其他一样的属性实现，只是写报告时时间不允许我再作改动。

【注：对于属性的说明】我定义的属性暂时只是简单的对对应私有字段的读写操作。使用属性是为了保证私有字段能够通过一种比较简洁且可读性强的方式读写。显然这比专门写get、set函数要高效多了。实现样例：

private string iD;

public string ID

{

get{ return iD;}

set{iD=value;}

}

则对ID赋值相当于对iD赋值，对ID读出相当于对iD读出。后面类似的东西就不再赘述。

定义【IItemToDateBase】接口，里面声明了两个抽象函数。



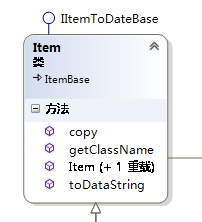
这两个虚函数在13个密封的最远派生类里进行了重写override。功能是：

getClassName：返回类名字符串。本来是有用的，但代码后来改动过，这个函数就没有使用。

toDataString：返回可用于数据库insert语句的字符串。这么写是为了方便封装操作数据库的函数。

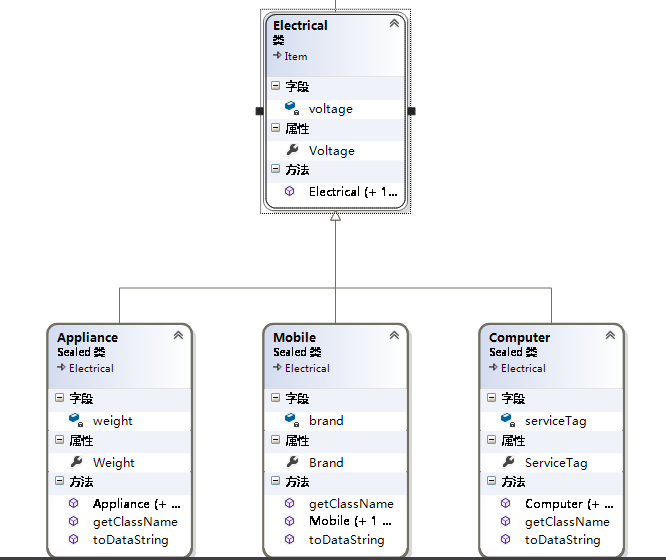
【注：为什么使用接口】其实完全可以直接将虚函数定义在抽象类里，但是为了将这些与本来的商品没有任何关系的方法单独拿出来，就使用了接口。这也是为了提高程序的面向对象的特性。

【Item】类的实现



此处最重要的是三点：

1. 写了构造函数，其中有一个是利用字符串数组进行初始化。在代码里可以详细看到其实现方式。
2. 给出了接口里的抽象函数的实现，变成了虚函数。
3. 实现了复制函数copy。由于C#不允许重载“=”，所以采取这种方式完成复制。用于排序时的交换。
4. Electrical.cs



（1）里面定义了四个类，一个为四大基类之一Electrical，继承与Item类。在继承的时候新增了字段voltage电压和对应的属性Voltage。

另外三个类为继承Electrical而来的，分别在继承的时候增加了一个新的字段和属性。其中字段信息如下：

-weight:double

-brand:string

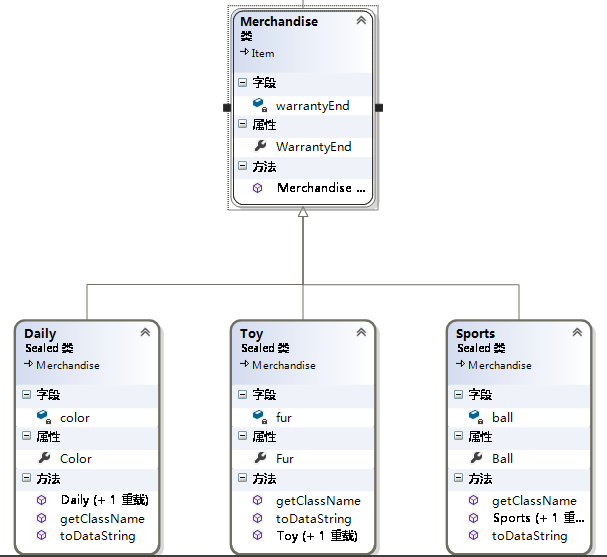
-serviceTag:string

（2）重载了IItemToDateBase接口中的两个函数。

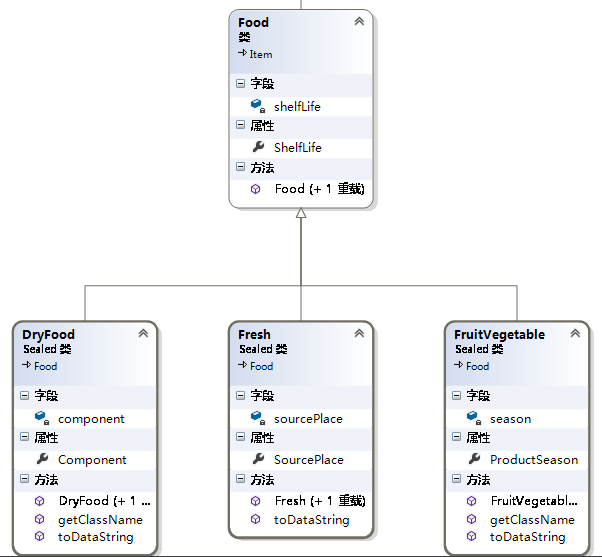
（3）构造函数，还是从字符串数组里初始化信息。

【注：以下3个文件很相似，不再赘述】

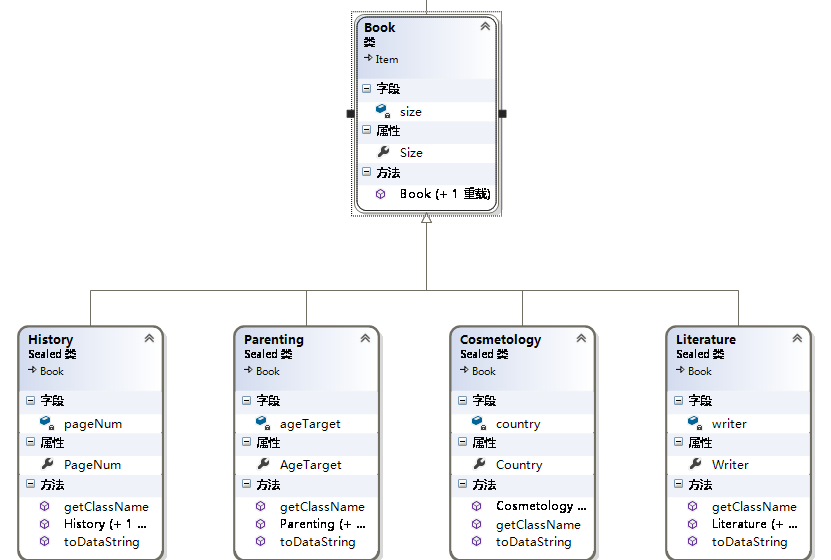
1. Merchandise.cs



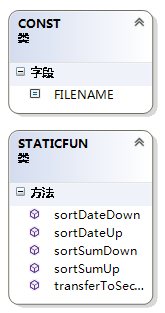
1. Food.cs



1. Book.cs

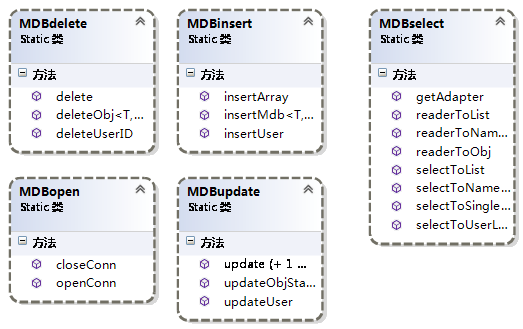


1. CONST.cs



（1）CONST里面定义了常量FILENAME，极其重要，为market.mdb的默认路径，所有函数将会从这个路径搜索market.mdb。默认放在E盘根目录下。

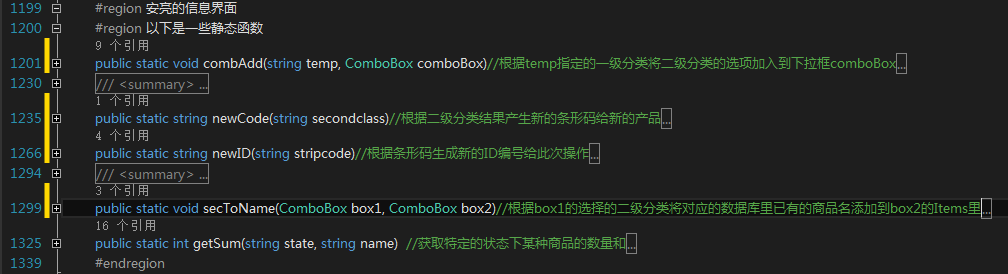
1. MDBmanip类库



1. 在代码的注释里有功能和设计思路的详细说明。函数的名称就体现了它们的功能。
2. Select的返回值为ArrayList型动态数组。
3. insertMDB函数和deleteObj函数使用了泛型约束<T,U>where T:U where U:Item ，这样就限定了T为我们的13个最远派生类。设计泛型是为了适应不同的类的对象。但是，只有insertMDB函数的泛型起到了作用，而deleteObj其实并未真正发生作用，原因是时间不足，没能实现更加优秀的功能。
4. getAdapter函数是为了返回数据适配器填充临时表格，交给界面“商品更改/删除”中的DataGridView来输出。只有这一个DataGridView是我输出的，其他界面上的DataGridView都是队友郝浚珩通过传ArrayList实现的。
5. “商品新增”和“商品更改/删除”页面

这两部分最麻烦的地方在于保证逻辑正确，即满足前面入库总数>=上架总数>=售出总数>=退回总数。 在form1.cs文件里，我的代码被一个有我的名字的region块包括了起来，很容易辨认。由于第一部分已经对功能做了说明，这里就作简要的说明。

1. 写了五个静态函数来集成一些使用频率比较高的代码



1. “信息新增”
2. 左侧，商品类别的选择分两步，第一个框选择一级分类，第二个框选择二级分类，将已有的满足二级分类的商品名称输出到下面的文本框里。
3. 入库，选择商品类别，然后输入商品名称。如果想输入已有商品的名称，则可以先在左边进行查找。按下“确认提交”按钮，先判断是否信息填写完整，之后判断商品是否为已有商品，如果是，则条形码、价格等数据全部采用默认值。
4. “商品更改/删除”

【注】（由于写大作业的报告时候较晚，难以展开说明，请查看源代码，并亲身体验一下两个页面的设计）

1. 用户界面少量代码

修补了三个按钮的功能。

1. market.mdb

这部分详见报告的第二部分。那里详细体现了market.mdb的组织结构

【容错性】

1. 在我写的两个界面里，“确认提交”或者“更改\*\*条目”按钮都可以对输入文本框信息不完整的情况加以判断，并通过消息框来提醒。
2. 插入的时候，通过各层类和最远派生类的构造函数完成了对部分数据输入为空时的判断和操作，此时，将使用默认值。

# 4项目总结

总结：此次我来写逻辑和数据库部分，工作量相当大。但队友也做出了特别大贡献，独自设计了新的逻辑，并且设计了界面，完成了界面逻辑这个大头的大部分代码，十分给力。

# 5相关问题的说明

如果想运行源代码，请先编译ItemInfo，生成ItemInfo.dll，然后将该dll添加进

DataBaseManip的引用里，然后再编译，会生成DataBaseManip.dll。之后将两个dll包含进主程序（界面程序）的引用里，之后才能运行。

注意数据库market.mdb务必放在 E盘根目录下，使得market.mdb文件的路径为：E:\market.mdb

如有任何问题，请拨打13021174210，安亮同学，本人随时等候“面试答辩”。