# 《健康管理系统》软件设计说明书

## 软件系统总体结构

该系统主要分为两个大的模块，用户模块和管理员模块；结构图如下图1.1。

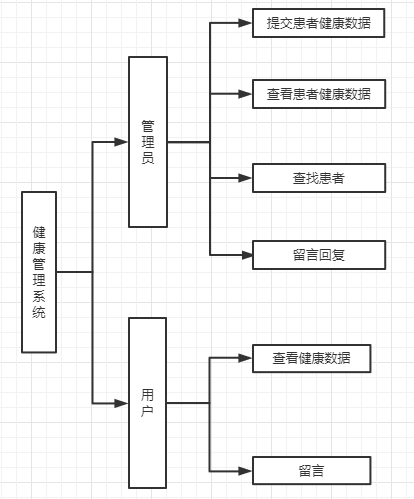


图1.1 健康管理系统总体结构图

## 软件系统数据库逻辑结构

### 2.1 本系统的E-R图

系统主要分为两个大的模块，一个是用户模块，另一个是管理员模块。用户主要有两个功能，一个是用户查看自己的健康数据，另一个是用户留言功能。管理员主要有四个功能，分别是提交患者健康数据、查看患者健康数据、查找患者以及留言回复。此系统的E-R图,如下所示:

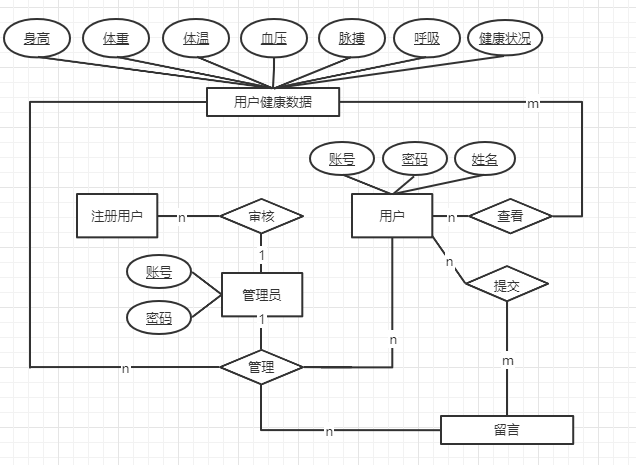


图2.1 系统的E-R图

### 2.2数据库表设计

数据库在一个信息管理系统中占有非常重要的地位，数据库结构设计的好坏将直接对应用系统的效率以及实现的效果产生影响。合理的数据库结构设计可以提高数据存储的效率，保证数据的完整性和一致性，同时合理的数据库结构设计也有利于程序的实现。

设计数据库系统时应首先了解用户各个方面的需求，包括现有的以及将来可能增加的需求，用户的需求具体体现在各种信息的提供、保存、更新和查询，这就要求数据库结构能充分满足各种信息的输入和输出。收集基本数据、数据结构以及数据处理的流程，组成一份详尽的数据字典，为后面的具体设计打下基础。

如下所示的数据项和数据结构：

1. 用户表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 账号 | Userid | char | 20 |
| 密码 | Password | char | 20 |
| 姓名 | Username | char | 20 |

表2.1 user用户表

1. 管理员表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 账号 | id | char | 20 |
| 密码 | Password | char | 20 |

表2.2 管理员表

1. 健康数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 存储代码 | 类型 | 长度 |
| 身高 | Height | Int | 5 |
| 体重 | Weight | Int | 5 |
| 体温 | Temper | Int | 5 |
| 血压 | Pressure | Int | 5 |
| 脉搏 | Pulse | Int | 5 |
| 呼吸 | Breath | char | 20 |
| 健康状况 | Health | Char | 20 |

表2.3 健康数据表

## 3．软件重用方案

#### 3.1软件重用的定义

重用，也叫再用或复用，是指同一事物不做修改或稍加改动就多次重复使用。

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、[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B/1928697" \t "_blank)、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](https://baike.baidu.com/item/%E8%BD%AF%E6%9E%84%E4%BB%B6" \t "_blank)，可重用的软构件越大，重用的粒度越大。

#### 3.2软件重用的层次

（1）知识重用（例如，[软件工程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B" \t "_blank)知识的重用）。

（2）方法和标准的重用（例如，面向对象方法或国家制定的软件开发规范的重用）。

（3）软件成分的重用。

#### 3.3软件重用的方案

第一步：将功能移出类实例方法

由于类继承机制缺乏精确性，因此对于代码重用来说它并不是一种最理想的机制。也就是说，如果要重用某个类的单个方法，就必须继承该类的其他方法以及数据成员。这种累赘不必要地将要重用此方法的代码复杂化了。继承类对其父类的依赖性引入了额外的复杂性：对父类的更改会影响子类；当更改父类或子类中的任一方时，很难记住覆盖了哪些方法（或者没有覆盖哪些方法）；而且是否应该调用相应的父类方法也不明朗。执行单一概念性任务的任何方法都应该是独立的，并应将其作为要重用的首选方法。要实现这一点，我们必须返回到过程式编程，将代码移出类实例方法并将其移入全局可见的过程中。为了提高这类过程的可重用性，应该像编写静态实用方法那样编写这类方法：每个过程只使用其自身的输入参数和对其他全局可见过程的调用完成其工作，而且不应该使用任何非局部变量。这种外部依赖性的减弱降低了使用该过程的复杂性，从而可促进在别处对它的重用。当然，即便那些不计划重用的代码也会从这种结构中受益，因为它的结构总是相当清晰。在 Java 中，方法不能脱离类而存在。但是可以采取相关步骤，使方法成为单个类的、公共可见的静态方法。

第二步：将非基本数据类型的输入参数类型转换为接口类型

通过接口参数类型而非通过类继承利用多态性，这是在面向对象编程方法中实现可重用性的真正基础。可重用性是通过编写接口，而不是通过编写类来实现的。如果一个方法的所有参数均为一些已知接口的引用，而这些接口又是由您从未听过的一些类实现的，那么该方法可对编写代码时还不存在的类的对象进行操作。从技术上讲，可重用的是方法，而不是传递给该方法的对象。一旦某个功能块可作为一个全局可见的独立过程，就可以通过将它的每个类级输入参数类型转换为接口类型，从而进一步提高它的可重用性。这样，实现该接口类型的任何类的对象都符合该参数的要求，而不仅仅是符合原始类的要求。这样，该过程便潜在地可用于更多的对象类型。

第三步：选择耦合性较小的输入参数接口类型

在执行第二步时，应该选择能充分描述过程对参数的要求且累赘最少的任何接口。参数对象要实现的接口越小，任一特定类能实现该接口的机会就越大。因而其对象可用作该参数的类的数量也就越多。

#### 3.4软件重用的好处

使用软件重用技术可以减少软件开发活动中大量的重复性工作，这样就能提高软件生产率，降低开发成本，缩短开发周期。同时，由于[软构件](https://baike.baidu.com/item/%E8%BD%AF%E6%9E%84%E4%BB%B6" \t "_blank)大都经过严格的质量认证，并在实际运行环境中得到校验，因此，重用软构件有助于改善[软件质量](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E8%B4%A8%E9%87%8F" \t "_blank)。此外，大量使用软构件，软件的灵活性和标准化程度也可望得到提高。

## 4．设计关键类的重点服务

在本次设计中，我们的系统主要分为两个大的模块，一个是用户模块，另一个是管理员模块。

#### 4.1用户模块

首先应该创建一个UserAddDialog类，用于用户的注册，随后创建一个LoginDialog类，作为登录界面，在用户填写完自己账号和密码，并选择对应的用户选项，点击登录按钮，进行登录；登录成功后将跳转到用户界面，控制跳转对应的界面应该由一个Perspective类通过权限划分来控制；跳转后的用户界面主要有两个功能，一个是用户查看自己的健康数据，另一个是用户留言功能，这两个功能分别由UserData类和UserLWord类来管理。

#### 4.2管理员模块

在设计管理员模块时，我们可以重用用户设计时的部分类，例如注册以及登录界面我们可以直接使用用户模块中创建过的UserAddDialog类和LoginDialog类，用于管理管理员的注册以及登录，同样的在管理员填写完自己账号和密码，并选择对应的用户选项，点击登录按钮，进行登录；登录成功后将跳转到用户界面，控制跳转对应的界面还是由Perspective类通过权限划分来控制；跳转后的管理员界面主要有四个功能，分别是提交患者健康数据、查看患者健康数据、查找患者以及留言回复，我们可以分别创建四个类来管理这四项功能。初步设想是这样，后续在系统实际设计阶段还会做进一步完善和修改。