# 面向对象程序设计大作业设计报告

张俊华 16030199025

## 一、题目

**销售公司员工管理**

设计一个虚基类Staff（员工），包含编号、姓名和年龄保护数据成员以及相关的成员函数；由Staff派生出销售员类Salesman，包含销售额保护数据成员以及相关的成员函数；再由Staff派生出经理类Manager；然后由Salesman和Manager类派生出销售经理类SalesManager。

设计一个利用文件处理方式实现对公司人员（包括销售员、经理和销售经理）进行管理，具有增加数据、更新数据、查询数据、删除数据以及重组文件的功能。（删除数据在记录中做删除标志，重组文件指在物理上删除有删除标志的记录。）另外要求分别统计这三类员工的人数以及所有员工的总数。

## 实验环境：

Microsoft Visual Studio Community 2017

VisualStudio.15.Release/15.8.6+28010.2041

Microsoft Visual C++ 2017

Windows10 SDK 10.0.17134.0

## 二、编程思想

本程序作业采用纯面向对象的思路进行开发，首先对题目要求进行分析：

本题目主要要求实现以下功能：

* 员工信息的存储
* 员工信息的查询展示
* 用户与程序交互，实现员工信息增加，修改等操作

于是，按照功能类型，对不同功能的代码和数据进行封装，将整体代码设计分为以下三个层次

### 模型层

* 按照题目要求设计 Staff、SalesMan、Manger、SalesManager 四个类，完成员工信息的抽象表示。并定义每个类的功能，如属性的查询，修改，等操作

### 视图层

* 将员工信息进行对用户友好的格式化输出
* 为用户交互创建基本界面（菜单选项等）

### 控制层

* 设计基于文件的数据库，实现数据持续化存储，实现数据库的基本增、删、改、查 操作
* 设计事务逻辑，实现用户信息录入、员工记录修改等操作的基本实现

### 数据持久化的实现

本项目采用 csv （逗号分隔符文件）将数据记录写入磁盘，实现持久化存储。记录与记录之间采用换行符分隔，记录的每个字段之间采用英文逗号, 进行分隔。程序初始化时，将数据从磁盘读入内存中以提高查询和修改的速度，程序结束或手动保存时，将数据写入csv文件中。

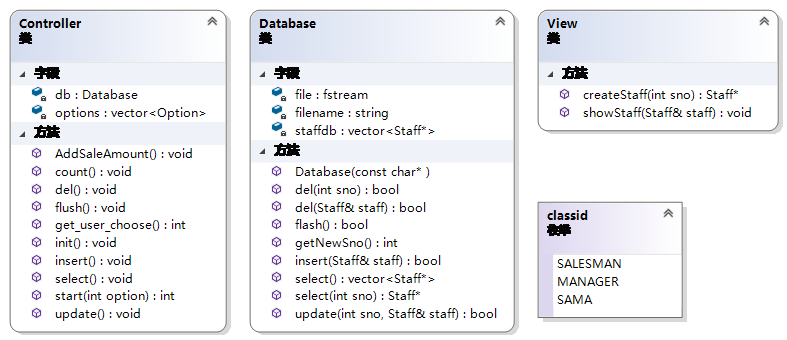
## 三、类的设计

根据上述分析，设计以下类：

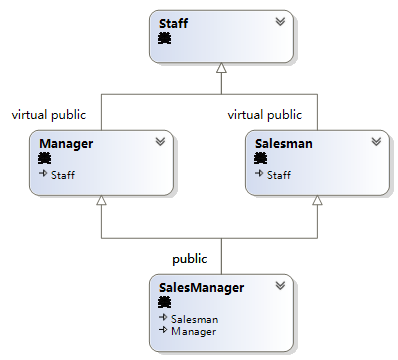
* 模型层
  + Staff ：职员信息基类
  + Salesman：销售人员类
  + Manager：经理类
  + Salesmanager：销售经理类
* 视图层：
  + View ：视图类，实现对员工信息的格式化输出
* 控制层：
  + Controller：控制器，实现交互功能逻辑
  + Database：数据库类，实现数据库基本的增删改查操作，以及职员信息的持久化存储

### UML 类图：

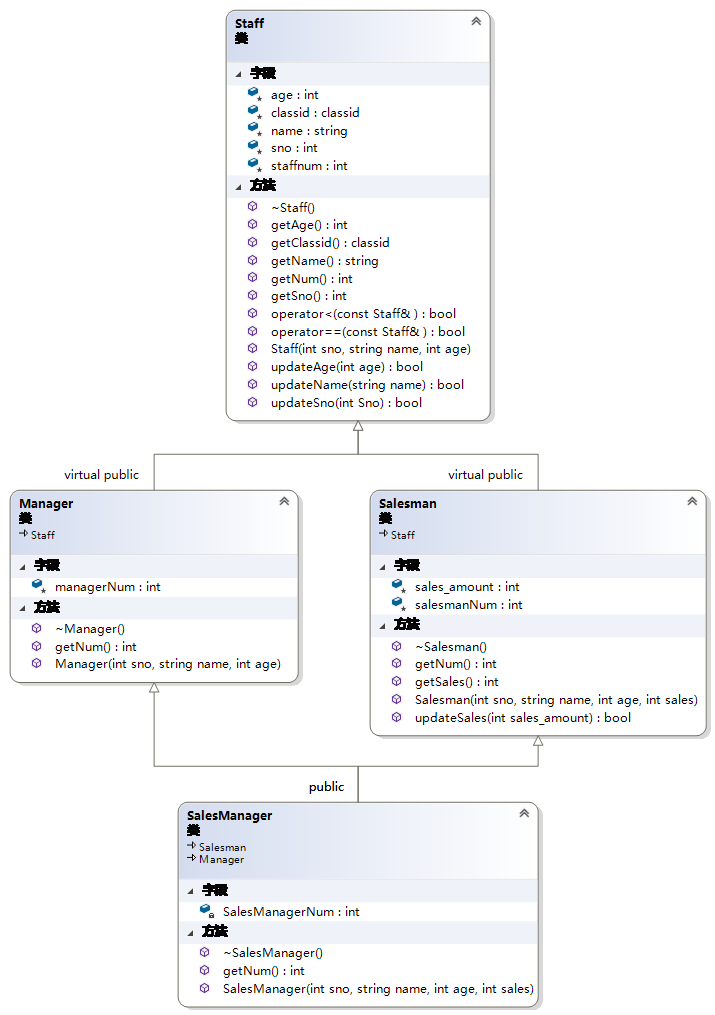
View 、Controller、Database



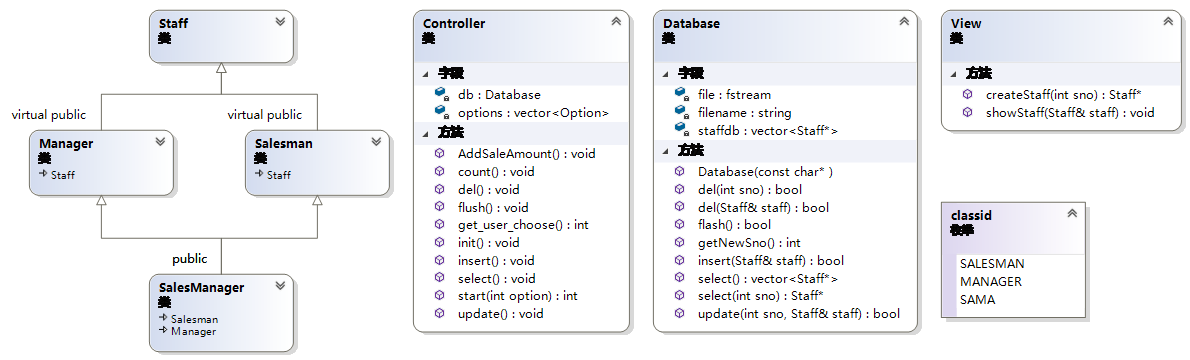
Staff、Manager、Salesman、SalesManager 继承关系



模型层具体类定义



总 UML 图

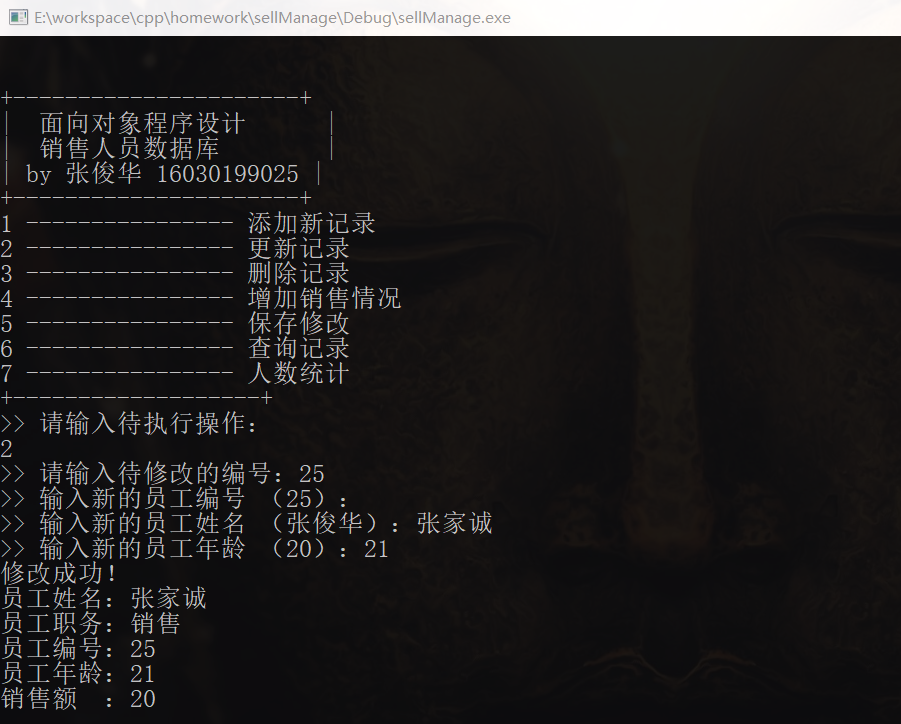


## 四、运行结果

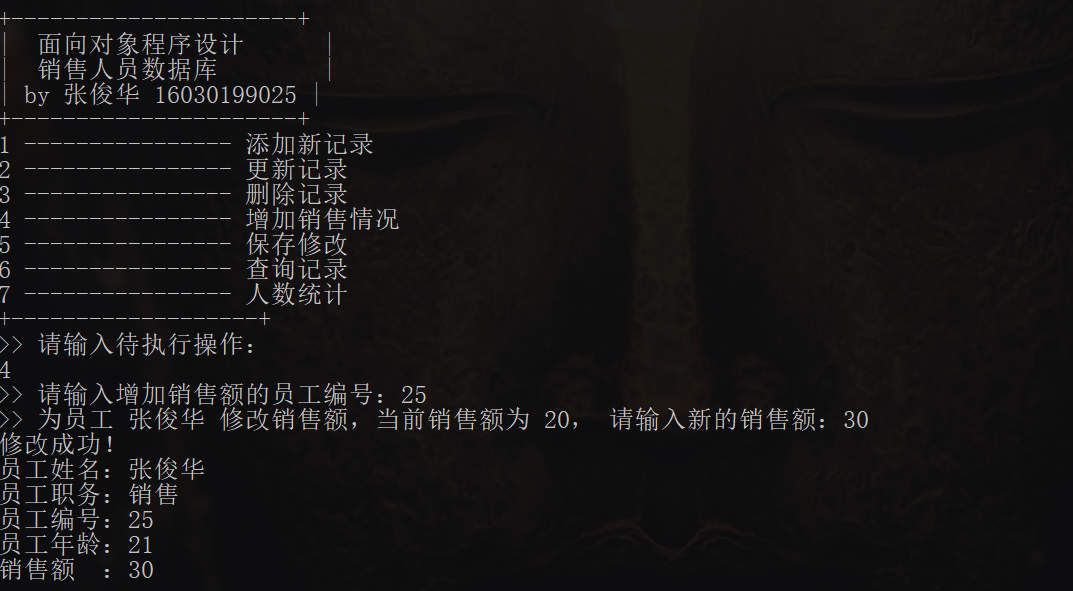
菜单选项：



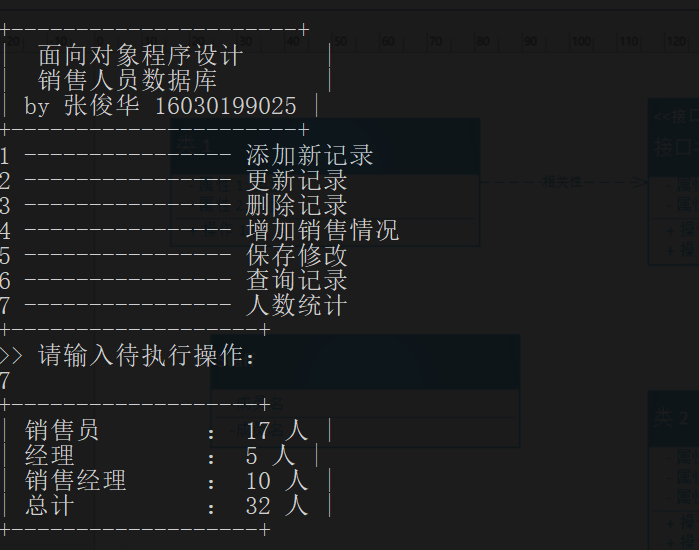
记录修改：



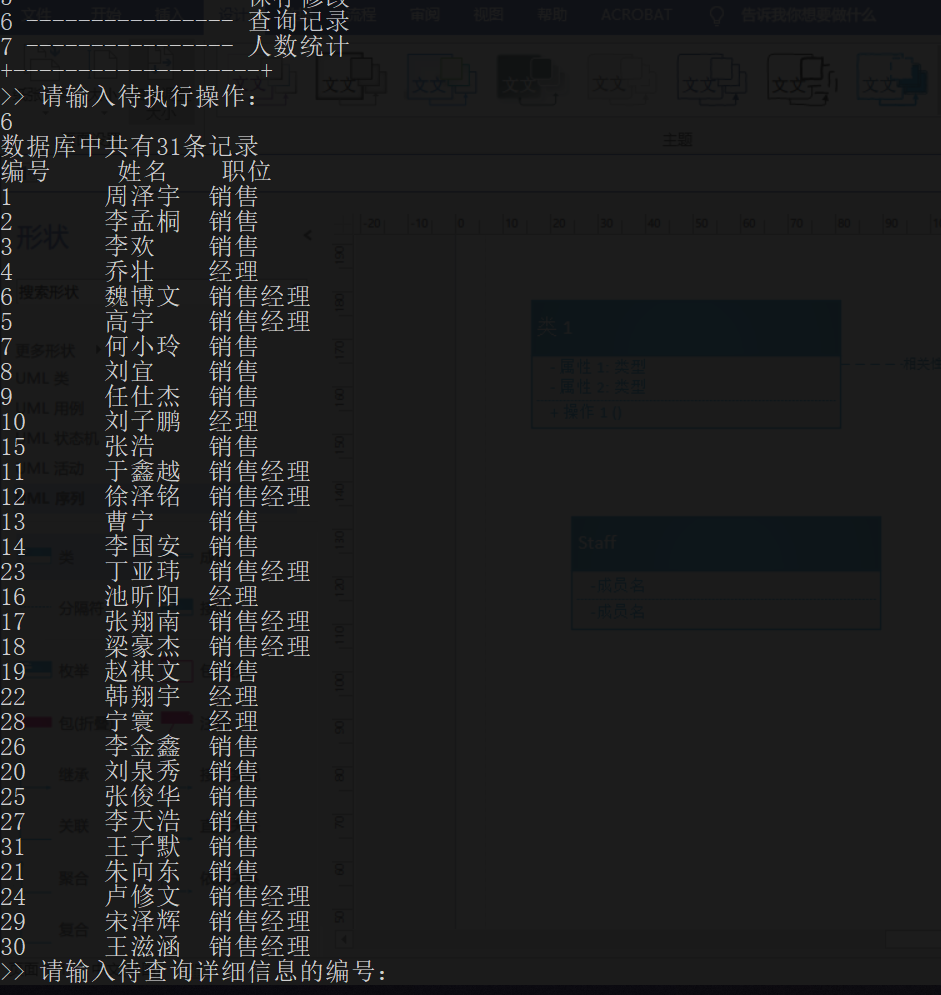
销售情况修改：

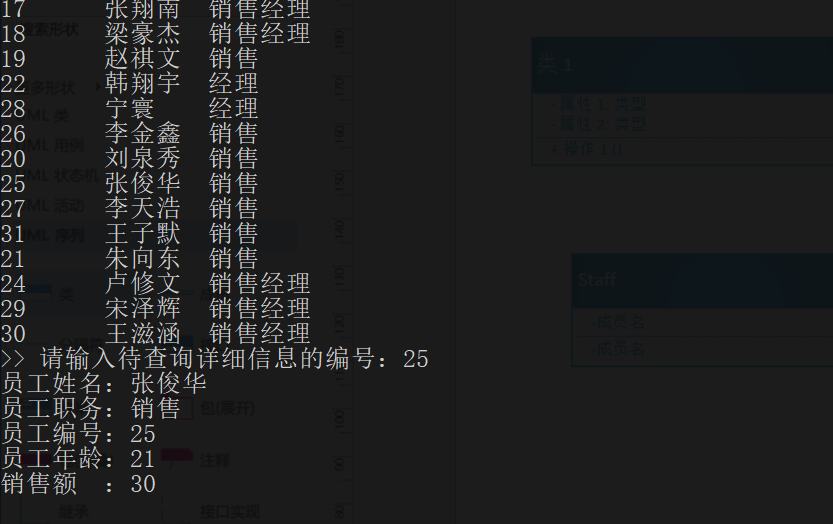


人数统计：

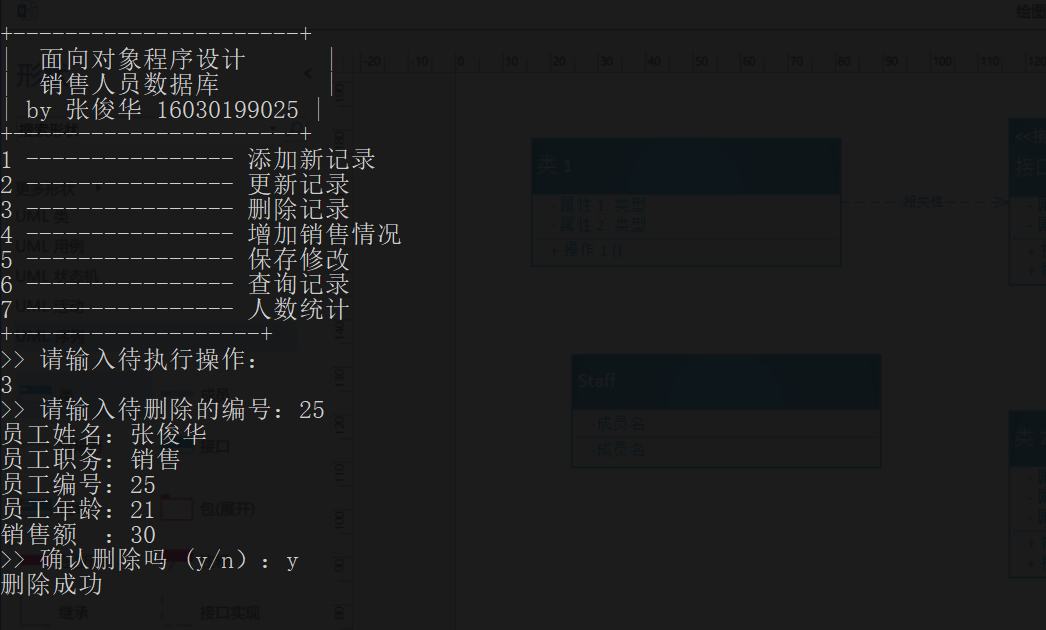


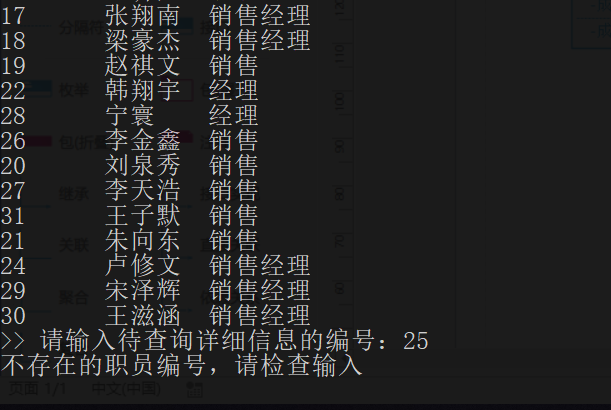
记录查询：



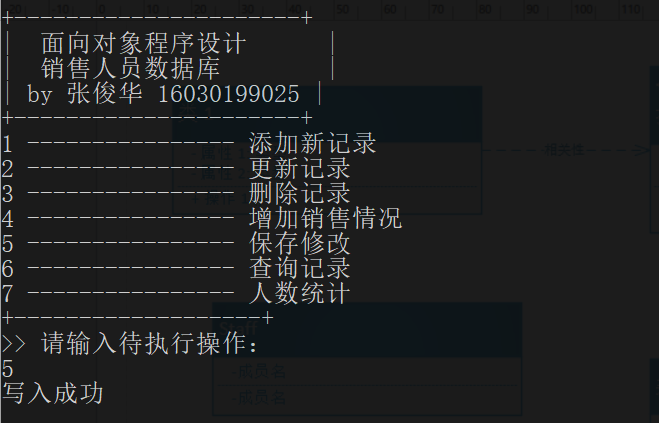


删除记录

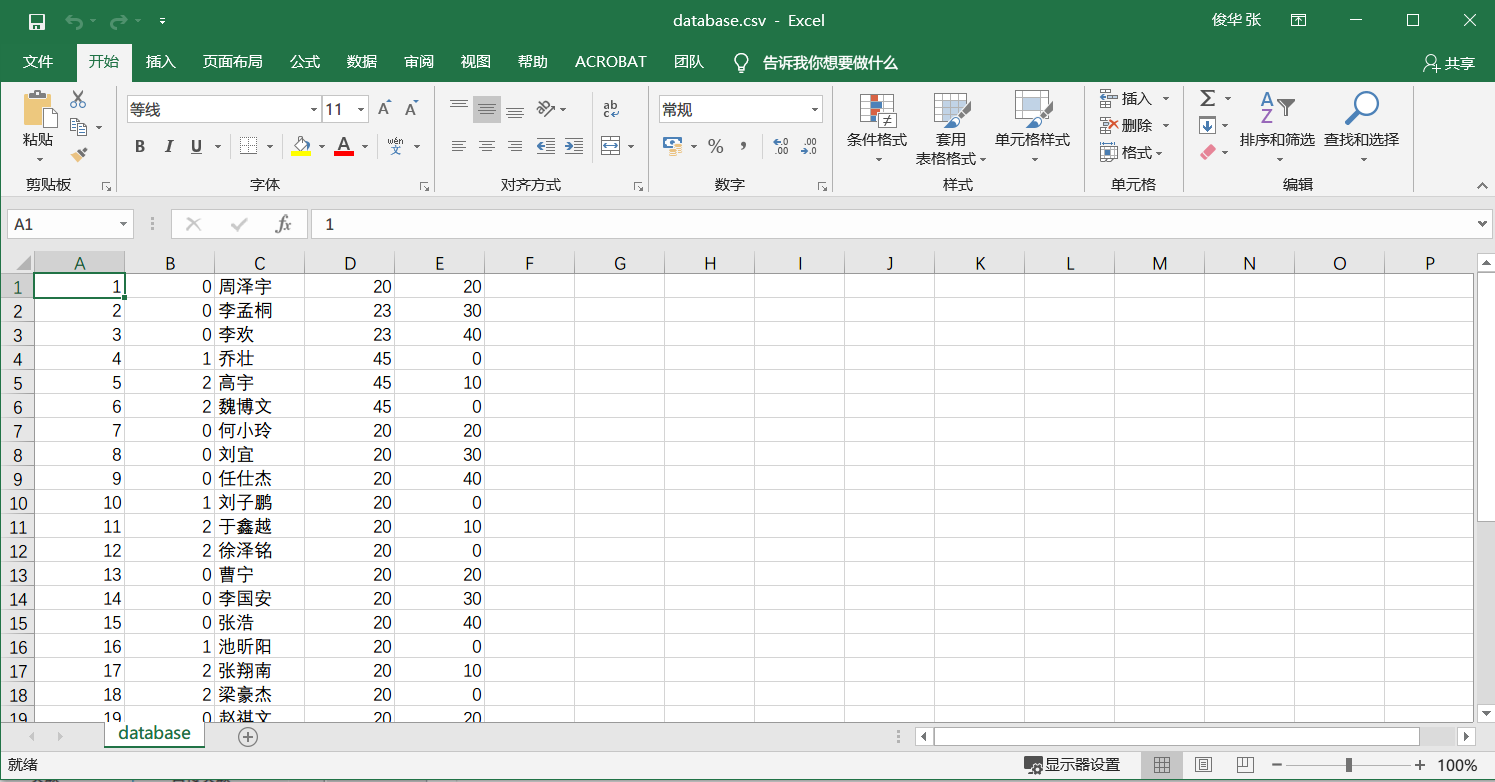




保存修改：



记录在磁盘中的存储结果：



## 五、问题及解决

1. **不同职位的人员，在内存中如何表示？如何能统一对不同职位的人员进行操作？**

* 在 Database （数据库）类中定义 基类指针（Staff\*）的vector 容器，Manager、Salesman、SalesManager实例化之后，将其地址存入 vector 中，之后使用指针对不同的实例进行操作

1. **如何能够区分不同职位的人员？如何能调用 Salesman、SalesManager 子类中的成员函数？**

* 在 Staff 类中额外定义 enum 类型的成员变量 classid ，用来存储实例的具体类型，在业务逻辑中判断 classid 值，再将基类指针强制类型转换到其真实子类指针，调用其成员函数完成对应操作。

1. **在实现中发现，由于 SalesMan、Manger、SalesManager 采用了虚继承方式，如采用基类指针强制类型转换到子类指针，会无法通过编译**

* 查阅相关资料发现，可以通过动态类型转换，
* dynamic\_cast <Salesman\*> (&staff);
* 在运行时完成指针类型的转换

1. **如何实现不同职位人员的统计？**

* 两个思路
  + 遍历整个数据库中的所有对象实例，维护三个不同的计数值完成计数
  + 在每个子类中定义静态成员变量，将其初值初始化为 0 ，在构造函数中对成员变量进行增一操作
* 本实现采用了第二种方法，使用类中的静态成员变量进行技术统计，避免了每次遍历所有实例。提高了查询效率。

9898

## 六、源代码

### 头文件定义：

* class.h
* #pragma once  
  #include<string>  
  enum classid {  
   SALESMAN, MANAGER, SAMA  
  };  
  class Staff {  
  protected:  
   int sno;  
   std::string name;  
   int age;  
   classid classid;  
   static int staffnum;  
  public:  
   Staff(int sno, std::string name, int age);  
   ~Staff();  
   enum classid getClassid() const;  
   virtual int getSno() const;  
   virtual std::string getName() const;  
   virtual int getAge() const;  
   bool operator==(const Staff &);  
   bool operator < (const Staff &)const;  
    
   virtual bool updateSno(int Sno);  
   virtual bool updateName(std::string name);  
   virtual bool updateAge(int age);  
   static int getNum();  
    
  };  
    
  class Salesman : virtual public Staff {  
  protected:  
   int sales\_amount;  
   static int salesmanNum;  
  public:  
   Salesman(int sno, std::string name, int age, int sales = 0);  
   ~Salesman();  
   bool updateSales(int sales\_amount);  
   int getSales();  
   static int getNum();  
  };  
    
  class Manager : virtual public Staff {  
  protected:  
   static int managerNum;  
  public:  
   Manager(int sno, std::string name, int age) ;  
   ~Manager();  
   static int getNum();  
  };  
    
  class SalesManager : public Salesman,public Manager {  
   static int SalesManagerNum;  
  public:  
   SalesManager(int sno, std::string name, int age, int sales = 0);  
   ~SalesManager();  
   static int getNum();  
  };
* controller.h
* #pragma once  
  #include "class.h"  
  #include <fstream>  
  #include <vector>  
  using namespace std;  
  /\*  
   数据库对象  
  \*/  
  class Database{  
  private:  
   string filename;  
   std::fstream file;  
   vector<Staff \*> staffdb;  
  public:  
   bool insert(Staff &staff);  
    
   bool update(int sno, Staff &staff);  
    
   bool del(int sno);  
    
   bool del(Staff &staff);  
    
   Staff \* select(int sno);  
   vector<Staff \*> select();  
    
   int getNewSno();  
    
   bool flash();  
    
   Database(const char \*);  
  };  
    
  typedef struct Option {  
   int id;  
   void(\*fun)();  
   string name;  
  } Option;  
    
  class Controller {  
  private:  
   static Database db;  
   static vector<Option> options;  
  public:  
   static void init();  
   static int get\_user\_choose();  
   static int start(int option);  
   static void insert();  
   static void update();  
   static void del();  
   static void flush();  
   static void AddSaleAmount();  
   static void select();  
   static void count();  
  };
* view.h
* #pragma once  
  #include "class.h"  
  class View {  
  public:  
   static void showStaff(Staff &staff);  
    
   static Staff \* createStaff(int sno = -1);  
  };

### 具体实现

* main.cpp
* #include "class.h"   
  #include "view.h"   
  #include "controller.h"   
  int main() {   
   Controller::init();   
   while (true)   
   {   
   Controller::start(Controller::get\_user\_choose());   
   }   
  };
* class.cpp
* #pragma once  
  #include "class.h"  
  int Staff::staffnum = 0;  
  int Salesman::salesmanNum = 0;  
  int Manager::managerNum = 0;  
  int SalesManager::SalesManagerNum = 0;  
  Staff::Staff(int sno, std::string name, int age) {  
   this->sno = sno;  
   this->name = name;  
   this->age = age;  
   staffnum++;  
  }  
    
  Staff::~Staff()  
  {  
   staffnum--;  
  }  
    
  classid Staff::getClassid() const  
  {  
   return this->classid;  
  }  
    
  int Staff::getSno() const  
  {  
   return sno;  
  }  
    
  std::string Staff::getName() const  
  {  
   return name;  
  }  
    
  int Staff::getAge() const  
  {  
   return age;  
  }  
    
  bool Staff::operator==(const Staff & s)  
  {  
   if (this->sno == s.sno && this->name == s.name) {  
   return true;  
   }  
   else return false;  
  }  
    
  bool Staff::operator<(const Staff &s) const  
  {  
   return sno < s.sno;  
  }  
    
  bool Staff::updateSno(int Sno)  
  {  
   this->sno = sno;  
   return true;  
  }  
    
  bool Staff::updateName(std::string name)  
  {  
   this->name = name;  
   return true;  
  }  
    
  bool Staff::updateAge(int age)  
  {  
   this->age = age;  
   return true;  
  }  
    
  int Staff::getNum()  
  {  
   return staffnum;  
  }  
    
  Salesman::Salesman(int sno, std::string name, int age, int sales) :Staff(sno, name, age) {  
   sales\_amount = sales;  
   salesmanNum++;  
   classid = SALESMAN;  
  }  
    
  Salesman::~Salesman()  
  {  
   salesmanNum--;  
  }  
    
  bool Salesman::updateSales(int sales\_amount)  
  {  
   this->sales\_amount = sales\_amount;  
   return true;  
  }  
    
  int Salesman::getSales()  
  {  
   return sales\_amount;  
  }  
    
  int Salesman::getNum()  
  {  
   return salesmanNum;  
  }  
    
  Manager::Manager(int sno, std::string name, int age) : Staff(sno, name, age) {  
   classid = MANAGER;  
   managerNum++;  
  }  
  Manager::~Manager()  
  {  
   managerNum--;  
  }  
  int Manager::getNum()  
  {  
   return managerNum;  
  }  
  ;  
    
  SalesManager::SalesManager(int sno, std::string name, int age, int sales) : Salesman(sno, name, age), Manager(sno, name, age), Staff(sno, name, age) {  
   classid = SAMA;  
   this->sales\_amount = sales;  
   salesmanNum--;  
   managerNum--;  
   SalesManagerNum++;  
  }  
  SalesManager::~SalesManager()  
  {  
   SalesManagerNum--;  
  }  
  int SalesManager::getNum()  
  {  
   return SalesManagerNum;  
  }  
  ;
* view.cpp
* #pragma once   
  #include "view.h"   
  #include "iostream"   
  #include "class.h"   
  using namespace std;   
  void View::showStaff(Staff &staff)   
  {   
   cout << "员工姓名：" << staff.getName() << endl;   
   cout << "员工职务：";   
   switch (staff.getClassid())   
   {   
   case SALESMAN:   
   cout << "销售";   
   break;   
   case MANAGER:   
   cout << "经理";   
   break;   
   case SAMA:   
   cout << "销售经理";   
   default:   
   break;   
   }   
     
   cout << endl;   
     
   cout << "员工编号：" << staff.getSno() << endl;   
   cout << "员工年龄：" << staff.getAge() << endl;   
   Salesman \*sman;   
   switch (staff.getClassid())   
   {   
   case SALESMAN:   
   sman = dynamic\_cast <Salesman\*> (&staff);   
   cout << "销售额\t：" << sman->getSales() << endl;   
   break;   
   case SAMA:   
   sman = dynamic\_cast <SalesManager\*> (&staff);   
   cout << "销售额\t：" << sman->getSales() << endl;   
   break;   
   default:   
   break;   
   }   
  }   
     
     
  Staff \* View::createStaff(int sno)   
  {   
   Staff \*staff;   
   int classid;   
   string name;   
   int age;   
   if (sno == -1) {   
   cout << "请输入职员编号:";   
   cin >> sno;   
   }   
   cout << "请输入职员类型 （0 ，1，2）:";   
   cin >> classid;   
   cout << "请输入职员姓名：" << endl;   
   cin >> name;   
   cout << "请输入职员年龄：";   
   cin >> age;   
   switch (classid)   
   {   
   case 0:   
   staff = new Salesman(sno, name, age);   
   break;   
   case 1:   
   staff = new Manager(sno, name, age);   
   break;   
   case 2:   
   staff = new SalesManager(sno, name, age);   
   break;   
   default:   
   staff = new Salesman(sno, name, age);   
   cout << "请检查输入内容是否有误" << endl ;   
   }   
     
   //showStaff(\*staff);   
   return staff;   
  }
* controller.cpp
* #pragma once   
  #include "class.h"   
  #include "controller.h"   
  #include "view.h"   
  #include <vector>   
  #include <cstdio>   
  #include <iostream>   
  #include <algorithm>   
  using namespace std;   
     
  Database Controller::db = Database("database.csv");   
  vector<Option> Controller::options;   
  bool Database::insert(Staff & staff)   
  {   
   for (Staff \* it : staffdb) {   
   if ((\*it).getSno() == staff.getSno()) {   
   return false;   
   }   
   }   
   staffdb.push\_back(&staff);   
   return true;   
  }   
     
  bool Database::update(int sno, Staff & staff)   
  {   
   for (auto it = staffdb.begin(); it != staffdb.end(); it++) {   
   if ((\*\*it).getSno() == sno) {   
   (\*it) = &staff;   
   return true;   
   }   
   }   
   return true;   
  }   
     
  bool Database::del(int sno)   
  {   
   for (auto it = staffdb.begin(); it != staffdb.end(); it++) {   
   if ((\*\*it).getSno() == sno) {   
   staffdb.erase(it);   
   return true;   
   }   
   }   
   return false;   
  }   
     
  bool Database::del(Staff & staff)   
  {   
   for (auto it = staffdb.begin(); it != staffdb.end(); it++) {   
   if ((\*\*it).getSno() == staff.getSno()&& (\*\*it).getName() == staff.getName()) {   
   staffdb.erase(it);   
   return true;   
   }   
   }   
   return false;   
  }   
     
  Staff \* Database::select(int sno)   
  {   
   for (auto staff : staffdb) {   
   if ((\*staff).getSno() == sno) {   
   return staff;   
   }   
   }   
   cerr << "不存在的职员编号，请检查输入" << endl;   
  }   
     
  vector<Staff\*> Database::select()   
  {   
   return staffdb;   
  }   
     
  int Database::getNewSno()   
  {   
   sort(staffdb.begin(), staffdb.end());   
   return (\*staffdb[staffdb.size()-1]).getSno() + 1;   
  }   
     
  bool Database::flash()   
  {   
   ofstream outputfile(filename);   
   for (auto staff : staffdb) {   
   outputfile << (\*staff).getSno() << ",";   
   switch ((\*staff).getClassid())   
   {   
   case SALESMAN:   
   outputfile << 0;   
   break;   
   case MANAGER:   
   outputfile << 1;   
   break;   
   case SAMA:   
   outputfile << 2;   
   break;   
   default:   
   return false;   
   }   
   outputfile << ",";   
   outputfile << (\*staff).getName() << ",";   
   outputfile << (\*staff).getAge() << ",";   
   switch ((\*staff).getClassid())   
   {   
   case SALESMAN:   
   outputfile << dynamic\_cast <Salesman\*>(staff)->getSales()<<",";   
   break;   
   case MANAGER:   
   outputfile << 0 << ",";   
   break;   
   case SAMA:   
   outputfile << dynamic\_cast <SalesManager\*>(staff)->getSales()<<",";   
   break;   
   default:   
   return false;   
   }   
   outputfile << endl;   
     
   }   
   outputfile.flush();   
   outputfile.close();   
   return true;   
  }   
     
  Database::Database(const char \* filename)   
  {   
   this->filename = filename;   
   file.open(filename, ios::out | ios::in);   
   // staffdb = new vector<Staff>;   
   char line[1000];   
   while (file.peek() != EOF)   
   {   
   Staff \*staff;   
   char name[100];   
   int age;   
   int sno;   
   int classid;   
   int sales;   
   file.getline(line, 1000);   
   sscanf\_s(line, "%d,%d,%[^,],%d,%d", &sno, &classid, name,sizeof(name), &age, &sales);   
   switch (classid)   
   {   
   case 0:   
   staff = new Salesman(sno, name, age,sales);   
   // TODO: get sno   
   break;   
   case 1:   
   staff = new Manager(sno, name, age);   
   break;   
   case 2:   
   staff = new SalesManager(sno, name, age,sales);   
   break;   
   default:   
   return;   
   }   
   staffdb.push\_back(staff);   
   }   
   file.close();   
  }   
     
  void Controller::init()   
  {   
   options.push\_back({ 1,insert,"添加新记录" });   
   options.push\_back({ 2,update,"更新记录" });   
   options.push\_back({ 3,del,"删除记录" });   
   options.push\_back({ 4,AddSaleAmount,"增加销售情况" });   
   options.push\_back({ 5,flush,"保存修改" });   
   options.push\_back({ 6,select,"查询记录" });   
   options.push\_back({ 7,count,"人数统计" });   
     
   cout << "数据库初始化中..." << endl;   
   //db = Database("database.csv");   
   cout << "初始化成功" << endl;   
  }   
     
  int Controller::get\_user\_choose()   
  {   
   cout << endl << endl << endl;   
   printf("+----------------------+\n");   
   printf("| 面向对象程序设计 |\n");   
   printf("| 销售人员数据库 |\n");   
   printf("| by 张俊华 16030199025 |\n", SalesManager::getNum());   
   printf("+----------------------+\n");   
   int choose;   
   for (auto option : options) {   
   cout << option.id << " ---------------- " << option.name << endl;   
   }   
   printf("+-------------------+\n");   
   printf(">> 请输入待执行操作：\n");   
   cin >> choose;   
   return choose;   
  }   
     
  int Controller::start(int option)   
  {   
   for (auto o : options) {   
   if (o.id == option) {   
   (o.fun)();   
   }   
   }   
   return 0;   
  }   
     
  void Controller::insert()   
  {   
   cout << "==============" << endl;   
   auto n = View::createStaff(db.getNewSno());   
   db.insert(\*n);   
     
  }   
     
  void Controller::update()   
  {   
   printf(">> 请输入待修改的编号：");   
   int sno;   
   cin >> sno;   
   auto staff = db.select(sno);   
   cin.get();   
   printf(">> 输入新的员工编号 （%d）：", staff->getSno());   
   int newsno = sno;   
   if (cin.peek() != '\n') {   
   cin >> newsno;   
   }   
   cin.get();   
   string newname = staff->getName();   
   printf(">> 输入新的员工姓名 （%s）：", staff->getName().c\_str());   
   if (cin.peek() != '\n') {   
   cin >> newname;   
   }   
   cin.get();   
   int newage = staff->getAge();   
   printf(">> 输入新的员工年龄 （%d）：", staff->getAge());   
   if (cin.peek() != '\n') {   
   cin >> newage;   
   }   
   cin.get();   
   staff->updateName(newname);   
   staff->updateAge(newage);   
   staff->updateSno(newsno);   
   //if (db.update(sno, staff)) {   
   cout << "修改成功！" << endl;   
   //}   
   View::showStaff(\*db.select(newsno));   
  }   
     
  void Controller::del()   
  {   
   printf(">> 请输入待删除的编号：");   
   int sno;   
   cin >> sno;   
   auto staff = db.select(sno);   
   View::showStaff(\*staff);   
   printf(">> 确认删除吗 (y/n）：");   
   char c;   
   cin >> c;   
   if (c == 'y') {   
   db.del(\*staff);   
   cout << "删除成功" << endl;   
   }   
  }   
     
  void Controller::flush()   
  {   
   db.flash();   
   cout << "写入成功" << endl;   
  }   
     
  void Controller::AddSaleAmount()   
  {   
   printf(">> 请输入增加销售额的员工编号：");   
   int sno;   
   int newsales;   
   cin >> sno;   
   auto staff = db.select(sno);   
   Salesman \*sman = NULL;   
   switch (staff->getClassid()) {   
   case SALESMAN:   
   sman = dynamic\_cast <Salesman\*>(staff);   
   break;   
   case MANAGER:   
   printf("选择的员工为经理，无销售额属性!\n\n");   
   return;   
   case SAMA:   
   sman = dynamic\_cast <SalesManager\*>(staff);   
   break;   
   }   
   printf(">> 为员工 %s 修改销售额，当前销售额为 %d， 请输入新的销售额：", sman->getName().c\_str(), sman->getSales());   
   cin >> newsales;   
   sman->updateSales(newsales);   
   printf("修改成功！\n");   
   View::showStaff(\*sman);   
     
  }   
     
  void Controller::select()   
  {   
   cout << "数据库中共有" << db.select().size() << "条记录" << endl;   
   printf("编号 \t 姓名 \t 职位\n");   
   for (auto p : db.select()) {   
   cout << p->getSno() << "\t" << p->getName() << "\t";   
   switch (p->getClassid()) {   
   case SALESMAN:   
   cout << "销售" << endl;   
   break;   
   case MANAGER:   
   cout << "经理" << endl;   
   break;   
   case SAMA:   
   cout << "销售经理" << endl;   
   break;   
   }   
   }   
   printf(">> 请输入待查询详细信息的编号：");   
   int sno;   
   cin >> sno;   
   auto staff = db.select(sno);   
   View::showStaff(\*staff);   
  }   
     
  void Controller::count()   
  {   
   printf("+-------------------+\n");   
   printf("| 销售员 \t： %d 人 |\n", Salesman::getNum());   
   printf("| 经理 \t： %d 人 |\n", Manager::getNum());   
   printf("| 销售经理\t： %d 人 |\n", SalesManager::getNum());   
   printf("| 总计 \t： %d 人 |\n", Staff::getNum());   
   printf("+-------------------+\n");   
  }