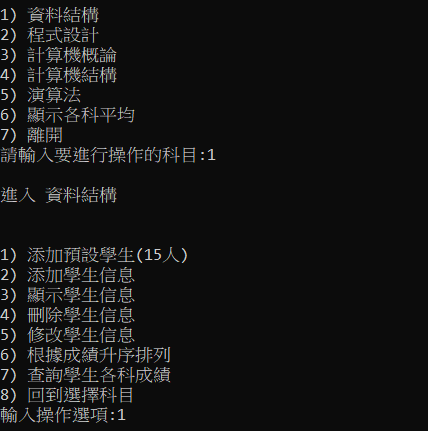


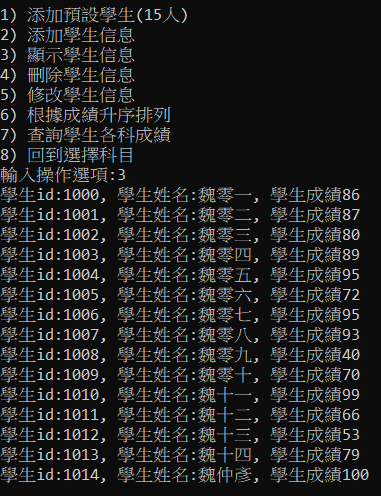
我的StudentScoreManager是純手工打造，不是用鏈結串鍊去完成，而是用物件導向，因為題目沒有規定一定要用鏈結串列或是其他資料結構，而且如果跟同學都一樣就不好玩了XD

利用MVC架構分別管理系統(M: Model 控制學生模型)(V: Veiw 控制介面邏輯)(C: Controller 控制程式處理邏輯)

以下是我的介面，可以輸入需要添加的科目，一開始科目的人數都為 0，需要自行添加，但也可以直接添加預設學生



添加完預設學生後結果如下



注意!!，我的系統同一個可能會拿到不同ID，ID是自增長的，就像下面這樣

1) 添加預設學生(15人)  
2) 添加學生信息  
3) 顯示學生信息  
4) 刪除學生信息  
5) 修改學生信息  
6) 根據成績升序排列  
7) 查詢學生各科成績  
8) 回到選擇科目  
輸入操作選項:8  
  
1) 資料結構  
2) 程式設計  
3) 計算機概論  
4) 計算機結構  
5) 演算法  
6) 顯示各科平均  
7) 離開  
請輸入要進行操作的科目:2  
  
進入 程式設計  
  
  
1) 添加預設學生(15人)  
2) 添加學生信息  
3) 顯示學生信息  
4) 刪除學生信息  
5) 修改學生信息  
6) 根據成績升序排列  
7) 查詢學生各科成績  
8) 回到選擇科目  
輸入操作選項:1  
  
1) 添加預設學生(15人)  
2) 添加學生信息  
3) 顯示學生信息  
4) 刪除學生信息  
5) 修改學生信息  
6) 根據成績升序排列  
7) 查詢學生各科成績  
8) 回到選擇科目  
輸入操作選項:3  
學生id:1015, 學生姓名:魏零一, 學生成績86  
學生id:1016, 學生姓名:魏零二, 學生成績87  
學生id:1017, 學生姓名:魏零三, 學生成績80  
學生id:1018, 學生姓名:魏零四, 學生成績89  
學生id:1019, 學生姓名:魏零五, 學生成績95  
學生id:1020, 學生姓名:魏零六, 學生成績72  
學生id:1021, 學生姓名:魏零七, 學生成績95  
學生id:1022, 學生姓名:魏零八, 學生成績93  
學生id:1023, 學生姓名:魏零九, 學生成績40  
學生id:1024, 學生姓名:魏零十, 學生成績70  
學生id:1025, 學生姓名:魏十一, 學生成績99  
學生id:1026, 學生姓名:魏十二, 學生成績66  
學生id:1027, 學生姓名:魏十三, 學生成績53  
學生id:1028, 學生姓名:魏十四, 學生成績79  
學生id:1029, 學生姓名:魏仲彥, 學生成績100

以下是我的python程式碼實作和講解:

import sys  
  
  
class StudentModel:  
 *"""  
 學生數據模型  
 """* def \_\_init\_\_(self, name="", score=0, subject=0, id=0):  
 self.name = name  
 self.score = score  
 self.subject = subject  
 self.id = id  
  
  
class StudentManagerController:  
 *"""  
 學生管理控制器:住要負責業務邏輯處理  
 """* init\_id = 1000  
  
 @classmethod  
 def \_\_generate\_id(cls, stu):  
 stu.id = cls.init\_id  
 cls.init\_id += 1 # 讓id自增長  
  
 def \_\_init\_\_(self):  
 self.\_\_stu\_list = []  
 self.\_\_class = SubjectManager()  
  
 @property  
 def stu\_list(self):  
 return self.\_\_stu\_list  
  
 @property  
 def sub\_data(self):  
 return self.\_\_class  
  
 def add\_student(self, stu, sub):  
 *"""  
 添加學生訊息  
 :param stu: 需要添加的學生對象  
 :param sub: 需要添加的科目  
 """* StudentManagerController.\_\_generate\_id(stu) # 加入學生id  
 self.\_\_class.add\_subject\_list(sub, stu) # 加入學生到dict裡面  
  
 def add\_base\_stu(self, sub):  
 *"""  
 添加基本學生訊息  
 :param sub: 需要添加的科目  
 """* std\_name = ["魏零一", "魏零二", "魏零三", "魏零四", "魏零五", "魏零六", "魏零七", "魏零八", "魏零九", "魏零十", "魏十一", "魏十二", "魏十三", "魏十四",  
 "魏仲彥"]  
 std\_score = [86, 87, 80, 89, 95, 72, 95, 93, 40, 70, 99, 66, 53, 79, 100]  
  
 for stu in self.\_\_class.subject\_dict[sub]:  
 for s in std\_name:  
 if stu.name == s:  
 return True  
  
 for i in range(len(std\_name)):  
 self.add\_student(StudentModel(std\_name[i], std\_score[i]), sub)  
  
 def remove\_student(self, stu\_id, sub):  
 *"""  
 移除學生信息  
 :param stu\_id: 需要移除的學生編號  
 :param sub: 需要添加的科目  
 :return: 移除是否成功  
 """* for item in self.\_\_class.subject\_dict[sub]:  
 if item.id == stu\_id:  
 self.\_\_class.subject\_dict[sub].remove(item)  
 return True  
 return False # 告知使用者沒有刪到任何人  
  
 def update\_student(self, new\_stu, sub):  
 *"""  
 修改學生信息(需要id)  
 :param new\_stu: 需要修改的學生  
 :param sub: 需要添加的科目  
 :return: 是否修改成功  
 """* for item in self.\_\_class.subject\_dict[sub]:  
 if item.id == new\_stu.id:  
 item.name = new\_stu.name  
 item.score = new\_stu.score  
 return True  
 return False  
  
 def order\_by\_score(self, sub):  
 *"""  
 根據成績升序排列  
 :param sub: 需要添加的科目  
 """* for i in range(len(self.\_\_class.subject\_dict[sub]) - 1):  
 for y in range(i + 1, len(self.\_\_class.subject\_dict[sub])):  
 if self.\_\_class.subject\_dict[sub][i].score > self.\_\_class.subject\_dict[sub][y].score:  
 self.\_\_class.subject\_dict[sub][i], self.\_\_class.subject\_dict[sub][y] = \  
 self.\_\_class.subject\_dict[sub][y], self.\_\_class.subject\_dict[sub][i]  
  
 def order\_by\_id(self, sub):  
 *"""  
 根據序號升序排列  
 :param sub: 需要添加的科目  
 """* for i in range(len(self.\_\_class.subject\_dict[sub]) - 1):  
 for y in range(i + 1, len(self.\_\_class.subject\_dict[sub])):  
 if self.\_\_class.subject\_dict[sub][i].id > self.\_\_class.subject\_dict[sub][y].id:  
 self.\_\_class.subject\_dict[sub][i], self.\_\_class.subject\_dict[sub][y] = \  
 self.\_\_class.subject\_dict[sub][y], self.\_\_class.subject\_dict[sub][i]  
  
 def clear\_stu\_data(self):  
 self.\_\_stu\_list = []  
  
 def query\_score(self, std\_nam):  
 *"""  
 用名字查詢科目成績  
 :return: 字典，{科目:分數}  
 """* flag = False  
 q\_dict = {}  
 for k, v in self.\_\_class.subject\_dict.items():  
 for s in v:  
 if std\_nam == s.name:  
 q\_dict[k] = s.score  
 flag = True  
 if flag:  
 return q\_dict  
 else:  
 return False # 查無此人  
  
  
class StudentManageView:  
 *"""  
 學生管理視圖:住要負責介面邏輯處理  
 """* def \_\_init\_\_(self):  
 self.\_\_controller = StudentManagerController()  
  
 def \_\_diplay\_menu(self):  
 print()  
 print("1) 添加預設學生(15人)")  
 print("2) 添加學生信息")  
 print("3) 顯示學生信息")  
 print("4) 刪除學生信息")  
 print("5) 修改學生信息")  
 print("6) 根據成績升序排列")  
 print("7) 查詢學生各科成績")  
 print("8) 回到選擇科目")  
  
 def \_\_select\_menu(self, sub):  
 item = input("輸入操作選項:")  
 if item == "1":  
 if (self.\_\_controller.add\_base\_stu(sub)):  
 print("以添加過預設學生")  
 return False  
 if item == "2":  
 self.\_\_input\_student(sub)  
 return False  
 elif item == "3":  
 self.\_\_output\_student(sub)  
 return False  
 elif item == "4":  
 self.\_\_delete\_student(sub)  
 return False  
 elif item == "5":  
 self.\_\_modify\_student(sub)  
 return False  
 elif item == "6":  
 self.\_\_output\_student\_order\_by\_score(sub)  
 return False  
 elif item == "7":  
 self.query\_stu\_score()  
 return False  
 elif item == "8":  
 self.\_\_controller.clear\_stu\_data()  
 return True  
 else:  
 print("輸入錯誤")  
 return False  
  
 def main(self):  
 while True:  
 self.\_\_diplay\_subject()  
 sub\_name = self.\_\_select\_subject()  
 if sub\_name: # True --> != 0 False == 0  
 print("\n進入 " + sub\_name + "\n")  
 while True:  
 self.\_\_diplay\_menu()  
 if self.\_\_select\_menu(sub\_name):  
 break  
  
 def \_\_input\_student(self, sub):  
 name = input("請輸入學生姓名:")  
 score = int(input("請輸入學生成績:"))  
 stu = StudentModel(name, score)  
 self.\_\_controller.add\_student(stu, sub)  
  
 def \_\_output\_student(self, sub):  
 sdu\_dict = self.\_\_controller.sub\_data.subject\_dict  
 for item in sdu\_dict[sub]:  
 print("學生id:%d, 學生姓名:%s, 學生成績%d" %  
 (item.id, item.name, item.score))  
  
 def \_\_delete\_student(self, sub):  
 stu\_id = int(input("請輸入要刪除的學生編號:"))  
 if self.\_\_controller.remove\_student(stu\_id, sub):  
 print("刪除成功")  
 else:  
 print("刪除失敗")  
  
 def \_\_modify\_student(self, sub):  
 stu = StudentModel()  
 stu.id = int(input("請輸入要修改的學生編號:"))  
 stu.name = input("請輸入學生姓名:")  
 stu.score = int(input("請輸入學生成績:"))  
  
 if self.\_\_controller.update\_student(stu, sub):  
 print("修改成功")  
 else:  
 print("修改失敗")  
  
 def \_\_output\_student\_order\_by\_score(self, sub):  
 self.\_\_controller.order\_by\_score(sub)  
 self.\_\_output\_student(sub)  
 self.\_\_controller.order\_by\_id(sub)  
  
 def query\_stu\_score(self):  
 name = input("輸入要查詢對象的姓名: ")  
 query\_dict = self.\_\_controller.query\_score(name)  
 sum = 0  
 if not query\_dict:  
 print("查無此人")  
 return 0  
 for k, v in query\_dict.items():  
 sum += v  
 print("科目: '" + str(k) + "' 的分數是: " + str(v))  
 print("個人總分: " + str(sum))  
 print("平均分數: " + str(sum / len(query\_dict)))  
  
 # 下面是選擇科目  
 def \_\_diplay\_subject(self):  
 print()  
 print("1) 資料結構")  
 print("2) 程式設計")  
 print("3) 計算機概論")  
 print("4) 計算機結構")  
 print("5) 演算法")  
 print("6) 顯示各科平均")  
 print("7) 離開")  
  
 def \_\_select\_subject(self):  
 item = input("請輸入要進行操作的科目:")  
 if item == "1":  
 return "資料結構"  
 elif item == "2":  
 return "程式設計"  
 elif item == "3":  
 return "計算機概論"  
 elif item == "4":  
 return "計算機結構"  
 elif item == "5":  
 return "演算法"  
 elif item == "6":  
 self.show\_all\_class\_score()  
 elif item == "7":  
 sys.exit(0)  
 else:  
 print("輸入錯誤")  
 return False  
  
 def show\_all\_class\_score(self):  
 average = self.\_\_controller.sub\_data.show\_all\_average()  
 print()  
 for k, v in average.items():  
 print(str(k) + " 平均分數是: " + str(v))  
 print()  
  
  
class SubjectManager:  
 def \_\_init\_\_(self):  
 self.\_\_subject\_dict = {  
 "資料結構": [],  
 "程式設計": [],  
 "計算機概論": [],  
 "計算機結構": [],  
 "演算法": [],  
 }  
  
 @property  
 def subject\_dict(self):  
 return self.\_\_subject\_dict  
  
 def add\_subject\_list(self, sub\_nam, stu):  
 self.\_\_subject\_dict[sub\_nam].append(stu)  
  
 def show\_all\_average(self):  
 query\_dict = {}  
 sum\_num = 0  
 for k, v in self.\_\_subject\_dict.items():  
 for s in v:  
 sum\_num += s.score  
 query\_dict[k] = sum\_num / (len(self.\_\_subject\_dict[k]) + 1)  
 sum\_num = 0  
 return query\_dict  
  
  
view = StudentManageView()  
view.main()