数据科学导论 -HW 1 报告

孙育泉 10234900421
2025.09.17

I 思路

本次要实现一个小型的宿舍管理程序，支持添加、查询、删除、显示学生的信息.

为此我们首先考虑将学生的信息封装成一个类，然后为了方便地对学生的数据进行操作，考虑封装一个用来处理增删改逻辑的管理系统，最后，为了方便使用 CLI 进行交互，我们考虑封装一个用于交互的 “前端”.

II 具体实现

II.1 `Student` 类

要存储的信息有:

- 性别
- 学号
- 姓名
- 宿舍号
- 电话

构建字典的时候，我们使用学号作为键. 由于 性别只有男女，我们使用枚举的方式来构建.

py
```py
class Gender(Enum):
    MALE = 1
    FEMALE = 0

    def __str__(self):
        return "男" if self == Gender.MALE else "女"

class Student:
    def __init__(
        self, stu_id: int, name: str, gender: Gender, dorm_id: int, telephone: str
    ):
        self.stu_id = stu_id
        self.name = name
        self.gender = gender
        self.dorm_id = dorm_id
        self.telephone = telephone
```

II.2 处理逻辑 `StudentManager`

注意这部分只负责对数据进行操作，而不涉及和外部的交互，实现的功能有:

- 添加学生
- 移除学生
- 通过学号查找学生
- 列举所有学生

py
```py
class StudentManager:
    def __init__(self):
        self._students: Dict[int, Student] = {}
```

```
4
5       def add_student(self, student: Student) -> bool:
6           if student.stu_id in self._students:
7               return False
8           self._students[student.stu_id] = student
9           return True
10
11      def remove_student(self, student_id: int) -> bool:
12          if student_id in self._students:
13              del self._students[student_id]
14              return True
15          return False
16
17      def find_student(self, student_id: int) -> Optional[Student]:
18          return self._students.get(student_id)
19
20      def list_all_students(self) -> List[Student]:
21          return list(self._students.values())
```

II.3 交互逻辑 `StudentCLI`

主要有以下几个组件构成:

- 清屏：使用的是系统的 shell 命令行
- 由于很多信息都需要读入整数，因此直接定义一个用于读取整数的函数 `prompt_for_int()`
- 美化输出的列表的函数 `display_students_table()`
- 将每个功能的逻辑都封装成函数，然后在执行 `run()` 的时候直接调用即可

py

```
1  class StudentCLI:
2      def __init__(self):
3          self.manager = StudentManager()
4
5      def clear_screen(self):
6          os.system('cls' if os.name == 'nt' else 'clear')
7
8      def prompt_for_int(self, prompt: str) -> int:
9          while True:
10             val_str = input(prompt)
11             if val_str.isdecimal():
12                 return int(val_str)
13             print("无效输入！请输入一个有效的数字。")
14
15     def display_students_table(self, students: List[Student]):
16         if not students:
17             print("未找到任何学生信息。")
18             return
19
20         # 定义列宽
21         id_w, name_w, gender_w, dorm_w, phone_w = 8, 20, 8, 8, 15
22
23         # 表头
24         header_line = "-" * (id_w + name_w + gender_w + dorm_w + phone_w + 11)
25         print(header_line)
26         print(f"| {'学号':<{id_w}} | {'姓名':<{name_w}} | {'性别':<{gender_w}} | {'宿舍
   号':<{dorm_w}} | {'电话':<{phone_w}} |")
27         print(header_line)
28
29         # 数据
30         for student in students:
```

```
            print(f"| {student.stu_id:<{id_w}} | {student.name:<{name_w}} | {str(student.gender):<{gender_w}} | {student.dorm_id:<{dorm_w}} | {student.telephone:<{phone_w}} |")

        print(header_line)

    def add_student(self):
        print("--- 添加新学生 ---")

        while True:
            stu_id = self.prompt_for_int("请输入学生学号: ")
            if not self.manager.find_student(stu_id):
                break
            print(f"错误: 学号 {stu_id} 已存在. 请输入一个不同的学号. ")

        name = input("请输入学生姓名: ")

        while True:
            gender_val = self.prompt_for_int("请输入性别 (0 代表女性, 1 代表男性): ")
            if gender_val in [0, 1]:
                gender = Gender(gender_val)
                break
            print("无效输入! 请输入 0 或 1. ")

        dorm_id = self.prompt_for_int("请输入学生宿舍号: ")

        while True:
            telephone = input("请输入学生电话: ")
            if telephone.isdecimal():
                break
            print("无效输入! 请输入一个有效的电话号码. ")

        new_student = Student(stu_id, name, gender, dorm_id, telephone)
        self.manager.add_student(new_student)

        print("\n学生添加成功! ")

    def remove_student(self):
        print("--- 删除学生 ---")
        student_id = self.prompt_for_int("请输入要删除的学生的学号: ")
        if self.manager.remove_student(student_id):
            print("学生删除成功! ")
        else:
            print("未找到该学生! ")

    def find_student(self):
        print("--- 查找学生 ---")
        student_id = self.prompt_for_int("请输入要查找的学生的学号: ")
        student = self.manager.find_student(student_id)
        if student:
            self.display_students_table([student])
        else:
            print("未找到该学生! ")

    def list_all_students(self):
        print("--- 所有学生列表 ---")
        all_students = self.manager.list_all_students()
        self.display_students_table(all_students)
```

```
87
88      def run(self):
89          while True:
90              print("\n--- 学生管理系统 ---")
91              print("1. 添加学生")
92              print("2. 删除学生")
93              print("3. 查找学生")
94              print("4. 列出所有学生")
95              print("5. 退出")
96              choice = input("请输入您的选择: ")
97
98              self.clear_screen()
99              if choice == "1":
100                 self.add_student()
101             elif choice == "2":
102                 self.remove_student()
103             elif choice == "3":
104                 self.find_student()
105             elif choice == "4":
106                 self.list_all_students()
107             elif choice == "5":
108                 print("正在退出程序. 再见! ")
109                 break
110             else:
111                 print("无效选择! 请重试. ")
112
113             input("\n按回车键返回主菜单...")
114             self.clear_screen()
```

II.4 主函数部分

直接调用即可.

py

```
1 if __name__ == "__main__":
2     cli = StudentCLI()
3     cli.run()
```

> **Remark**
>
> 完整代码见 "./hw1.py" 文件.