

Отчёт по практической работе 1

Давид Попов (группа П-2-24)

Оглавление

Описание работы.....	1
Структуры данных.....	1
Сохранение и загрузка.....	2
Загрузка программы и авторизация.....	5
Скриншоты работы программы.....	8
Код.....	11

Описание работы

Структуры данных

Я начал с того, что сделал датаклассы для книги и пользовательских типов. Библиотекарь и Читатель наследуются от Пользователя, в котором есть общие поля (имя):

```
Book.py > Book
1  from dataclasses import dataclass
2  from typing import List
3
4  @dataclass
5  class Book:
6      id: int
7      title: str
8      author: str
9      status: str = "available"
```

Figure 1: Датакласс Книга

Я также задал аннотации типов.

Наследуемые и родительские датаклассы:

```

1  ✓ from dataclasses import dataclass
2    from typing import List
3
4    @dataclass
5  ✓ class User:
6      id: int
7      type: str
8      name: str
9
10
11   @dataclass
12  ✓ class Reader(User):
13      books_taken: List[int] = None
14
15  ✓   def __post_init__(self):
16  ✓       if self.books_taken is None:
17           self.books_taken = []
18
19   @dataclass
20  ✓ class Librarian(User):
21       pass

```

Figure 2: Иерархия датаклассов пользователей

В датаклассе Reader если инициализировать член `books_taken` пустым листом в конструкторе, то этот объект будет общий для всех объектов класса. Поэтому я вставил триггер после создания `post_init`, чтобы он уже для нового объекта менял его на пустой лист. Тогда все эти пустые листы будут отдельные для каждого объекта класса.

Сохранение и загрузка

Я сохраняю все книги в файле `books.txt`, а наследников Пользователя в `users.txt` в простом формате:

```
books.txt
1 10
2 1
3 Мастер и Маргарита
4 Михаил Булгаков
5 available
6 2
7 Война и мир
8 Лев Толстой
9 taken
10 3
11 Сон в красном тереме
12 Цао Сюэцинь
13 available
14 4
```

Figure 3: Формат файла книг

Число книг (так же с пользователями) и для каждой книги каждое поле на одной линии. Я добавил поле id и для класса Книга и для класса Пользователь, чтобы проще идентифицировать объекты.

У пользователей-читателей и библиотекарей разный формат сохранения, у читателей также указываются взятые книги (все на одной линии через пробел)

```

users.txt
1  12
2  1
3  reader
4  Андрей Петров
5  2
6  2
7  reader
8  Иван Иванов
9  4 6
10 3
11 reader
12 Чжан Вэй
13 8
14 4
15 reader
16 Ли На
17 10
18 5
19 reader
20 Георгиос Пападопулос
21
22 6

```

Фигура 4: Формат файла пользователей

Я добавил методы сохранения и загрузки `save_books`, `load_books`, `save_users`, `load_users` как методы класса соответственно классов Книги и Пользователя.

Внутреннее представление книг/пользователей — лист, в листе пользователей — объекты классов-наследников хранятся вместе.

```

4  def save_books(filename, books):
5      with open(filename, 'w', encoding='utf-8') as f:
6          f.write(f"{len(books)}\n")
7          for book in books:
8              f.write(f"{book.id}\n")
9              f.write(f"{book.title}\n")
10             f.write(f"{book.author}\n")
11             f.write(f"{book.status}\n")
12

```

Фигура 5: Метод сохранения книг

При загрузке пользователей выбирается, какой объект сделать в зависимости от сохранённого:

```

34
35 def load_users(filename):
36     users = []
37     with open(filename, 'r', encoding='utf-8') as f:
38         count = int(f.readline().strip())
39         for _ in range(count):
40             user_id = int(f.readline().strip())
41             user_type = f.readline().strip()
42             name = f.readline().strip()
43             if user_type == "reader":
44                 books_line = f.readline().strip()
45                 books_taken = [int(bid) for bid in books_line.split()] if books_line else []
46                 users.append(Reader(user_id, user_type, name, books_taken))
47             elif user_type == "librarian":
48                 users.append(Librarian(user_id, user_type, name))
49     return users

```

Фигура 6: Метод загрузки пользователей

Загрузка программы и авторизация

Для хранения текущего состояния программы я сделал класс MenuState

```

Book.py M    User.py M    users.txt U
MenuState.py > MenuState > _init_
1  class MenuState:
2      def __init__(self):
3          self.books = []
4          self.users = []
5          self.current_user = None

```

При загрузке я спрашиваю имя пользователя, если такого пользователя нет, то программа abortируется. Если есть, то сохраняется в current_user. И далее от лица этого объекта вызывается метод меню, этот вызов осуществляется полиморфно: в методе main() написано просто current_user.menu()

```

7
8 def main():
9     state = MenuState()
10
11     # generate_test_data()
12
13     load_data(state)
14     login(state)
15     state.current_user.menu(state)
16     save_data()
17     print("Данные сохранены. До свидания!")
18     sys.exit(0)
19

```

Метод показа меню объявлен как абстрактный в абстрактном базовом классе Пользователь:

```

5
6 @dataclass
7 class User(ABC):
8     id: int
9     type: str
10    name: str
11
12    @abstractmethod
13    def menu(self, state):
14        pass
15

```

Этот класс не позволит создавать экземпляры, он нужен только для создания иерархии, и в нашей программе у него два наследника — Пользователь и Библиотекарь, которые реализуют абстрактный метод menu, и этот метод вызывается полиморфно из главного меню.

Само реализовано в цикле аналогично тому как мы делали раньше.

Код пунктов меню прямо в функции меню, например:

```

elif choice == "2":
    for book in state.books:
        if book.status == "available":
            print(f"{book.id}: {book.title}")

    try:
        book_id = int(input("ID книги: "))
        for book in state.books:
            if book.id == book_id and book.status == "available":
                book.status = "taken"
                self.books_taken.append(book_id)
                print(f"Книга '{book.title}' взята!")
                break
    except:
        print("Ошибка!")

```

Это опция взятия книги. Выводим список доступных книг, и спрашиваем id, обернули в ловлю исключения, если пользователь введёт ерунду.

Если книга есть и в наличии, то берём её и изменяем состояние state.

Аналогично реализованы остальные пункты.

Каждый из методов меню выходит когда пользователь выбирает последний пункт, и в главном модуле изменения сохраняются

```

201         print(f"Пользователь {name} ({new_user.type}) зарегистрирован")
202     elif choice == "4":
203         Librarian.show_users_brief(state.users)
204
205     elif choice == "5":
206         Librarian.show_books_brief(state.books)
207
208     elif choice == "6":
209         break
210
211

```

Какие-то методы реализованы как методы класса и вызываются от класса как здесь show_users_brief.

Скриншоты работы программы

Запуск, меню библиотекаря

```
PS C:\projects\oaip-python\prac1-26> py main.py
Добро пожаловать в библиотеку!
Введите ваше имя: Ван Мин
Добро пожаловать, Ван Мин!
Меню библиотекаря:
1) Добавить книгу
2) Удалить книгу
3) Зарегистрировать пользователя
4) Список пользователей
5) Список всех книг
6) Выход
Выбор: 4
1: Андрей Петров (reader)
2: Иван Иванов (reader)
3: Чжан Вэй (reader)
4: Ли На (reader)
5: Георгиос Пападопулос (reader)
6: Мария Константины (reader)
7: Аик Ованисян (reader)
8: Анаит Саркисян (reader)
9: Мухаммад Ахмед (reader)
10: Фатима Али (reader)
101: Светлана Сидорова (librarian)
102: Ван Мин (librarian)
Меню библиотекаря:
1) Добавить книгу
2) Удалить книгу
3) Зарегистрировать пользователя
4) Список пользователей
5) Список всех книг
6) Выход
Выбор: |
```

Попытка удалить взятую книгу

```
4) Список пользователей
5) Список всех книг
6) Выход
Выбор: 2
1: Мастер и Маргарита (available)
2: Война и мир (taken)
3: Сон в красном тереме (available)
4: Путешествие на Запад (taken)
5: Одиссея (taken)
6: Илиада (taken)
7: Мхитар Спарпет (available)
8: Давид Сасунский (taken)
9: Тысяча и одна ночь (available)
10: Калила и Димна (available)
ID книги для удаления: 2
Книга не свободна. Нельзя удалить
Меню библиотекаря:
```

Удаление свободной книги

```
1: Мастер и Маргарита (available)
2: Война и мир (taken)
3: Сон в красном тереме (available)
4: Путешествие на Запад (taken)
5: Одиссея (taken)
6: Илиада (taken)
7: Мхитар Спарпет (available)
8: Давид Сасунский (taken)
9: Тысяча и одна ночь (available)
10: Калила и Димна (available)
ID книги для удаления: 7
Удалить 'Мхитар Спарпет'? (y/n): y
Книга удалена!
Меню библиотекаря:
1) Добавить книгу
2) Удалить книгу
3) Зарегистрировать пользователя
4) Список пользователей
5) Список всех книг
6) Выход
Выбор: 5
1: Мастер и Маргарита (available)
2: Война и мир (taken)
3: Сон в красном тереме (available)
4: Путешествие на Запад (taken)
5: Одиссея (taken)
6: Илиада (taken)
8: Давид Сасунский (taken)
9: Тысяча и одна ночь (available)
10: Калила и Димна (available)
Меню библиотекаря:
```

Создание пользователя

```

PS C:\projects\oaip-python\prac1-26> py .\main.py
Добро пожаловать в библиотеку!
Введите ваше имя: Ван Мин
Добро пожаловать, Ван Мин!
Меню библиотекаря:
1) Добавить книгу
2) Удалить книгу
3) Зарегистрировать пользователя
4) Список пользователей
5) Список всех книг
6) Выход
Выбор: 3
Имя нового пользователя: Бобб Мидда
Тип пользователя:
1) Читатель
2) Библиотекарь
Выбор: 1
Пользователь Бобб Мидда (reader) зарегистрирован!
Меню библиотекаря:
1) Добавить книгу
2) Удалить книгу
3) Зарегистрировать пользователя
4) Список пользователей
5) Список всех книг
6) Выход
Выбор: |

```

Меню пользователя

```

1) Доступные книги
2) Взять книгу
3) Вернуть книгу
4) Мои книги
5) Выход
Выбор: 1
1: Мастер и Маргарита (Михаил Булгаков)
3: Сон в красном тереме (Цао Сюэцинь)
7: Мхитар Спарпет (Мурацан)
9: Тысяча и одна ночь (Неизвестный)
10: Калила и Димна (Ибн аль-Мукаффа)
11: bicsha (bubb)
12: 123 (456)

Меню пользователя:
1) Доступные книги
2) Взять книгу
3) Вернуть книгу
4) Мои книги
5) Выход
Выбор: 4
У вас нет книг!

Меню пользователя:
1) Доступные книги
2) Взять книгу
3) Вернуть книгу
4) Мои книги
5) Выход
Выбор: |

```

Взятие книги

```
5) Выход
Выбор: 2
1: Мастер и Маргарита
3: Сон в красном тереме
7: Мхитар Спарпет
9: Тысяча и одна ночь
10: Калила и Димна
11: Ёсссча
12: 123
ID книги: 3
Книга 'Сон в красном тереме' взята!

Меню пользователя:
1) Доступные книги
2) Взять книгу
3) Вернуть книгу
4) Мои книги
5) Выход
Выбор: 4
3: Сон в красном тереме (Цао Сюэцинь)

Меню пользователя:
1) Доступные книги
2) Взять книгу
3) Вернуть книгу
4) Мои книги
5) Выход
Выбор: |
```

Код

```
Book.py
from dataclasses import dataclass
from typing import List

@dataclass
class Book:
    id: int
    title: str
    author: str
    status: str = "available"

def save_books(filename, books):
    with open(filename, 'w', encoding='utf-8') as f:
        f.write(f"{len(books)}\n")
        for book in books:
            f.write(f"{book.id}\n")
            f.write(f"{book.title}\n")
            f.write(f"{book.author}\n")
            f.write(f"{book.status}\n")

def load_books(filename):
    books = []
    with open(filename, 'r', encoding='utf-8') as f:
        count = int(f.readline().strip())
        for _ in range(count):
            book_id = int(f.readline().strip())
            title = f.readline().strip()
            author = f.readline().strip()
            status = f.readline().strip()
```

```

        books.append(Book(book_id, title, author, status))
    return books
MenuState.py
class MenuState:
    def __init__(self):
        self.books = []
        self.users = []
        self.current_user = None
User.py
from abc import ABC, abstractmethod
from dataclasses import dataclass
from typing import List
from Book import Book

@dataclass
class User(ABC):
    id: int
    type: str
    name: str

    @abstractmethod
    def menu(self, state):
        pass

    def save_users(filename, users):
        with open(filename, 'w', encoding='utf-8') as f:
            f.write(f"{len(users)}\n")
            for user in users:
                f.write(f"{user.id}\n")
                f.write(f"{user.type}\n")
                f.write(f"{user.name}\n")
                if user.type == "reader":
                    f.write(" ".join(str(bid) for bid in user.books_taken) + "\n")

    def load_users(filename):
        users = []
        with open(filename, 'r', encoding='utf-8') as f:
            count = int(f.readline().strip())
            for _ in range(count):
                user_id = int(f.readline().strip())
                user_type = f.readline().strip()
                name = f.readline().strip()
                if user_type == "reader":
                    books_line = f.readline().strip()
                    books_taken = [int(bid) for bid in books_line.split()] if books_line else []
                    users.append(Reader(user_id, user_type, name, books_taken))
                elif user_type == "librarian":
                    users.append(Librarian(user_id, user_type, name))
        return users

    def find_user_by_name(name, users):
        for user in users:
            if user.name == name:
                return user
        return None

@dataclass
class Reader(User):
    books_taken: List[int] = None

    def __post_init__(self):
        if self.books_taken is None:
            self.books_taken = []

    def menu(self, state):
        while True:
            print("\nМеню пользователя:")
            print("1) Доступные книги")
            print("2) Взять книгу")
            print("3) Вернуть книгу")
            print("4) Мои книги")
            print("5) Выход")

            choice = input("Выбор: ")

            if choice == "1":
                for book in state.books:
                    if book.status == "available":
                        print(f"{book.id}: {book.title} ({book.author})")

```

```

elif choice == "2":
    for book in state.books:
        if book.status == "available":
            print(f"{book.id}: {book.title}")

    try:
        book_id = int(input("ID книги: "))
        for book in state.books:
            if book.id == book_id and book.status == "available":
                book.status = "taken"
                self.books_taken.append(book_id)
                print(f"Книга '{book.title}' взята!")
                break
    except:
        print("Ошибка!")

elif choice == "3":
    if not self.books_taken:
        print("У вас нет книг!")
        continue

    for book_id in self.books_taken:
        for book in state.books:
            if book.id == book_id:
                print(f"{book.id}: {book.title}")

    try:
        book_id = int(input("ID книги для возврата: "))
        if book_id in self.books_taken:
            self.books_taken.remove(book_id)
            for book in state.books:
                if book.id == book_id:
                    book.status = "available"
                    print(f"Книга '{book.title}' возвращена!")
                    break
    except:
        print("Ошибка!")

elif choice == "4":
    if not self.books_taken:
        print("У вас нет книг!")
        continue

    for book_id in self.books_taken:
        for book in state.books:
            if book.id == book_id:
                print(f"{book.id}: {book.title} ({book.author})")

elif choice == "5":
    break

```

@dataclass

class Librarian(User):

```

    def show_books_brief(books):
        for book in books:
            print(f"{book.id}: {book.title} ({book.status})")

    def show_users_brief(users):
        for user in users:
            print(f"{user.id}: {user.name} ({user.type})")

    def menu(self, state):
        while True:
            print("Меню библиотекаря:")
            print("1) Добавить книгу")
            print("2) Удалить книгу")
            print("3) Зарегистрировать пользователя")
            print("4) Список пользователей")
            print("5) Список всех книг")
            print("6) Выход")

            choice = input("Выбор: ")

            if choice == "1":
                new_id = max([b.id for b in state.books], default=0) + 1
                title = input("Название: ")
                author = input("Автор: ")

                print(f"\nID: {new_id}")

```

```

        print(f"Название: {title}")
        print(f"Автор: {author}")
        print(f"Статус: available")

        confirm = input("Подтвердить (y/n): ")
        if confirm.lower() == 'y':
            state.books.append(Book(new_id, title, author, "available"))
            print("Книга добавлена!")

    elif choice == "2":
        Librarian.show_books_brief(state.books)
        try:
            book_id = int(input("ID книги для удаления: "))
            for i, book in enumerate(state.books):
                if book.id == book_id:
                    if book.status != 'available':
                        print('Книга не свободна. Нельзя удалить')
                        break
                    confirm = input(f"Удалить '{book.title}'? (y/n): ")
                    if confirm.lower() == 'y':
                        del state.books[i]
                        print("Книга удалена!")
                        break
        except:
            print("Ошибка!")

    elif choice == "3":
        name = input("Имя нового пользователя: ")
        if User.find_user_by_name(name, state.users):
            print("Пользователь уже существует!")
            continue

        print("Тип пользователя:")
        print("1) Читатель")
        print("2) Библиотекарь")

        type_choice = input("Выбор: ")

        new_id = max([u.id for u in state.users], default=0) + 1

        if type_choice == "1":
            new_user = Reader(new_id, "reader", name, [])
        elif type_choice == "2":
            new_user = Librarian(new_id, "librarian", name)
        else:
            print("Некорректный выбор!")
            continue

        state.users.append(new_user)
        print(f"Пользователь {name} ({new_user.type}) зарегистрирован!")
    elif choice == "4":
        Librarian.show_users_brief(state.users)

    elif choice == "5":
        Librarian.show_books_brief(state.books)

    elif choice == "6":
        break

main.py
from Book import Book
from User import Reader, Librarian, User
import sys
from dataclasses import dataclass
from MenuState import MenuState

def main():
    state = MenuState()

    # generate_test_data()

    load_data(state)
    login(state)
    state.current_user.menu(state)
    save_data(state)
    print("Данные сохранены. До свидания!")
    sys.exit(0)

def load_data(state):
    state.books = Book.load_books("books.txt")

```

```

state.users = User.load_users("users.txt")

def save_data(state):
    Book.save_books("books.txt", state.books)
    User.save_users("users.txt", state.users)

def login(state):
    print("Добро пожаловать в библиотеку!")

    if not state.users:
        print(f"Первый запуск. Регистрация как первый библиотекарь.")
        name = input("Введите ваше имя: ")
        new_id = 1
        new_user = Librarian(new_id, "librarian", name)
        state.users.append(new_user)
        current_user = new_user
        print(f"Библиотекарь {name} зарегистрирован и вошел в систему!")
    else:
        name = input("Введите ваше имя: ")
        user = User.find_user_by_name(name, state.users)
        if not user:
            print(f"Пользователь {name} не найден!")
            sys.exit(1)
        state.current_user = user
        print(f"Добро пожаловать, {name}!")

def generate_test_data():
    books = [
        Book(1, "Мастер и Маргарита", "Михаил Булгаков", "available"),
        Book(2, "Война и мир", "Лев Толстой", "taken"),
        Book(3, "Сон в красном тереме", "Цао Сюэцинь", "available"),
        Book(4, "Путешествие на Запад", "У Чэньцзэнь", "taken"),
        Book(5, "Одиссея", "Гомер", "available"),
        Book(6, "Илиада", "Гомер", "taken"),
        Book(7, "Мхитар Спарпет", "Мурацан", "available"),
        Book(8, "Давид Сасунский", "Ованес Туманян", "taken"),
        Book(9, "Тысяча и одна ночь", "Неизвестный", "available"),
        Book(10, "Калила и Димна", "Ибн аль-Мукаффа", "taken")
    ]

    users = []

    readers = [
        Reader(1, "reader", "Андрей Петров", [2]),
        Reader(2, "reader", "Иван Иванов", [4, 6]),
        Reader(3, "reader", "Чжан Вэй", [8]),
        Reader(4, "reader", "Ли На", [10]),
        Reader(5, "reader", "Георгиос Пападопулос", []),
        Reader(6, "reader", "Мария Константину", []),
        Reader(7, "reader", "Айк Ованисян", []),
        Reader(8, "reader", "Анаит Саркисян", []),
        Reader(9, "reader", "Мухаммад Ахмед", []),
        Reader(10, "reader", "Фатима Али", [])
    ]

    librarians = [
        Librarian(101, "librarian", "Светлана Сидорова"),
        Librarian(102, "librarian", "Ван Мин")
    ]

    users.extend(readers)
    users.extend(librarians)

    Book.save_books("books.txt", books)
    User.save_users("users.txt", users)

main()

```