МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе No2

Специальность ПО11(о)

Выполнил
И. А. Головач,
студент группы ПО11

Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«1» март 2025 г.

Брест 2025

Вариант 5

Цель работы: закрепить навыки объектно-ориентированного программирования на языке Python.

Задание 1. Реализовать простой класс

Требования к выполнению

• Реализовать пользовательский класс по варианту.

Для каждого класса

• Создать атрибуты (поля) классов

• Создать методы классов

• Добавить необходимые свойства и сеттеры (по необходимости)

• Переопределить магические методы __str__ и __eq__

Множество целых чисел ограниченной мощности – Предусмотреть возможность объединения двух множеств, вывода на консоль элементов множества, а также метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволять создавать объекты с начальной инициализацией. Мощность множества задается при создании объекта. Реализацию множества осуществить на базе списка. Переопределить метод __eq__, выполняющий сравнение объектов данного типа.

Выполнение:

Код программы:

lab2_1.py:

```
class LimitedPowerSet:
    def __init__(self, max_size, initial_elements=None):
        if initial_elements is None:
            initial_elements = []

        try:
            if len(initial_elements) > max_size:
                raise ValueError("Количество начальных элементов превышает максимальную
мощность множества")
        except ValueError as e:
            print(f"Ошибка: {e}")
            raise  # Перебрасываем исключение, чтобы программа могла корректно
обработать его на уровне main()

        self.max_size = max_size
        self.elements = list(set(initial_elements))

    def add(self, value):
        try:
```

```
            if value in self.elements:
                print(f"Элемент {value} уже существует в множестве")
                return

            if len(self.elements) >= self.max_size:
                print(f"Множество достигло максимальной мощности. Элемент не может быть
добавлен")
                return

            self.elements.append(value)
            print(f"Элемент {value} добавлен успешно")
        except Exception as e:
            print(f"Ошибка при добавлении элемента: {e}")

    def remove(self, value):
        try:
            if value in self.elements:
                self.elements.remove(value)
                print(f"Элемент {value} удалён успешно")
            else:
                print(f"Элемент {value} не найден в множестве")
        except Exception as e:
            print(f"Ошибка при удалении элемента: {e}")

    def union(self, other_set):
        try:
            if not isinstance(other_set, LimitedPowerSet):
                raise TypeError("Объединение возможно только с объектами типа
LimitedPowerSet")

            new_max_size = self.max_size + other_set.max_size
            new_elements = list(set(self.elements).union(other_set.elements))
            return LimitedPowerSet(new_max_size, new_elements)
        except Exception as e:
            print(f"Ошибка при объединении множеств: {e}")
            return None

    def contains(self, value):
        try:
            return value in self.elements
        except Exception as e:
            print(f"Ошибка при проверке принадлежности элемента: {e}")
            return False

    def __str__(self):
        try:
            return f"Множество: {self.elements}, Максимальная мощность:
{self.max_size}"
        except Exception as e:
            print(f"Ошибка при формировании строкового представления множества: {e}")
            return "Ошибка: Недоступно для отображения"

    def __eq__(self, other):
        try:
            if not isinstance(other, LimitedPowerSet):
                return False
            return set(self.elements) == set(other.elements)
        except Exception as e:
            print(f"Ошибка при сравнении множеств: {e}")
```

```
            return False


def get_integer(prompt):
    while True:
        try:
            value = input(prompt)
            return int(value)
        except ValueError:
            print("Ошибка: Введите целое число.")


def main():
    print("Добро пожаловать в программу для работы с множествами ограниченной
мощности!")

    try:
        max_size1 = get_integer("Введите максимальную мощность первого множества: ")
        initial_elements1 = input(
            "Введите начальные элементы первого множества через пробел (или оставьте
пустым): ").split()
        initial_elements1 = [int(x) for x in initial_elements1] if initial_elements1
else []
        set1 = LimitedPowerSet(max_size1, initial_elements1)
        print(f"Создано первое множество: {set1}\n")
    except Exception as e:
        print(f"Ошибка при создании первого множества: {e}")
        return

    try:
        max_size2 = get_integer("Введите максимальную мощность второго множества: ")
        initial_elements2 = input(
            "Введите начальные элементы второго множества через пробел (или оставьте
пустым): ").split()
        initial_elements2 = [int(x) for x in initial_elements2] if initial_elements2
else []
        set2 = LimitedPowerSet(max_size2, initial_elements2)
        print(f"Создано второе множество: {set2}\n")
    except Exception as e:
        print(f"Ошибка при создании второго множества: {e}")
        return

    while True:
        try:
            print("\nВыберите действие:")
            print("1. Добавить элемент в множество")
            print("2. Удалить элемент из множества")
            print("3. Проверить принадлежность элемента множеству")
            print("4. Объединить множества")
            print("5. Сравнить множества")
            print("6. Вывести информацию о множествах")
            print("7. Выход")

            choice = input("Введите номер действия: ")

            if choice == "1":
                try:
                    set_num = input("В какое множество добавить элемент? (1 или 2): ")
                    value = get_integer("Введите значение элемента: ")
```

```
                if set_num == "1":
                    set1.add(value)
                elif set_num == "2":
                    set2.add(value)
                else:
                    print("Некорректный выбор множества.")
            except Exception as e:
                print(f"Ошибка при добавлении элемента: {e}")

        elif choice == "2":
            try:
                set_num = input("Из какого множества удалить элемент? (1 или 2): ")
                value = get_integer("Введите значение элемента: ")
                if set_num == "1":
                    set1.remove(value)
                elif set_num == "2":
                    set2.remove(value)
                else:
                    print("Некорректный выбор множества.")
            except Exception as e:
                print(f"Ошибка при удалении элемента: {e}")

        elif choice == "3":
            try:
                set_num = input("В каком множестве проверить принадлежность? (1 или
2): ")
                value = get_integer("Введите значение элемента: ")
                if set_num == "1":
                    result = set1.contains(value)
                elif set_num == "2":
                    result = set2.contains(value)
                else:
                    print("Некорректный выбор множества.")
                    continue
                print(f"Элемент {value} {'принадлежит' if result else 'не
принадлежит'} множеству.")
            except Exception as e:
                print(f"Ошибка при проверке принадлежности: {e}")

        elif choice == "4":
            try:
                set3 = set1.union(set2)
                if set3:
                    print(f"Результат объединения множеств: {set3}")
            except Exception as e:
                print(f"Ошибка при объединении множеств: {e}")

        elif choice == "5":
            try:
                if set1 == set2:
                    print("Множества равны.")
                else:
                    print("Множества не равны.")
            except Exception as e:
                print(f"Ошибка при сравнении множеств: {e}")

        elif choice == "6":
            try:
                print(f"Первое множество: {set1}")
```

```
                    print(f"Второе множество: {set2}")
                except Exception as e:
                    print(f"Ошибка при выводе информации о множествах: {e}")

            elif choice == "7":
                print("Программа завершена.")
                break

            else:
                print("Некорректный выбор. Попробуйте снова.")

        except Exception as e:
            print(f"Произошла непредвиденная ошибка: {e}. Продолжаем работу...")


if __name__ == "__main__":
    main()
```

Рисунки с результатами работы программы lab2_1.py:

```
C:\Users\ilyag\AppData\Local\Programs\Python\Python313\python.exe D:\labs_university\SSP\lab1\spp_po11\reports\Golovach_Ilya\5\lab2\src\lab2_1.py
Добро пожаловать в программу для работы с множествами ограниченной мощности!
Введите максимальную мощность первого множества: 4
Введите начальные элементы первого множества через пробел (или оставьте пустым): 1 2 5 4
Создано первое множество: Множество: [1, 2, 4, 5], Максимальная мощность: 4

Введите максимальную мощность второго множества: 6
Введите начальные элементы второго множества через пробел (или оставьте пустым): 6
Создано второе множество: Множество: [6], Максимальная мощность: 6


Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 6
Первое множество: Множество: [1, 2, 4, 5], Максимальная мощность: 4
Второе множество: Множество: [6], Максимальная мощность: 6

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 1
В какое множество добавить элемент? (1 или 2): 2
Введите значение элемента: 3
Элемент 3 добавлен успешно

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
```

```
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 2
Из какого множества удалить элемент? (1 или 2): 1
Введите значение элемента: 1
Элемент 1 удалён успешно

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 3
В каком множестве проверить принадлежность? (1 или 2): 1
Введите значение элемента: 1
Элемент 1 не принадлежит множеству.

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 5

Множества не равны.

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: Некорректный выбор. Попробуйте снова.
```

```
Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 6
Первое множество: Множество: [2, 4, 5], Максимальная мощность: 4
Второе множество: Множество: [6, 3], Максимальная мощность: 6

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 4
Результат объединения множеств: Множество: [2, 3, 4, 5, 6], Максимальная мощность: 10

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 5
Множества не равны.

Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Объединить множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
Введите номер действия: 7
Программа завершена.
```

Задание 2. Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы

Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

Выполнение:

Код программы:

Lab2_2.py:

```
from datetime import datetime

# Базовый класс для всех людей
class Person:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return f"{self.__class__.__name__}: {self.name}"


# Класс Читатель
class Reader(Person):
    def __init__(self, name):
        super().__init__(name)
        self.orders = []  # Список заказов читателя
        self.is_blacklisted = False  # Флаг "черного списка"

    def place_order(self, book, catalog):
        """Оформить заказ на книгу."""
        if self.is_blacklisted:
            print(f"Ошибка: Читатель {self.name} находится в черном списке. Заказ
невозможен.")
            return None

        order = Order(book, self)
        if catalog.search_book(book):
            self.orders.append(order)
            print(f"Читатель {self.name} оформил заказ на книгу '{book.title}'.")
            return order
        else:
            print(f"Ошибка: Книга '{book.title}' отсутствует в каталоге.")
            return None

    def return_book(self, order):
        """Вернуть книгу."""
        if order in self.orders:
            self.orders.remove(order)
```

```
            order.return_book()
            print(f"Читатель {self.name} вернул книгу '{order.book.title}'.")
        else:
            print(f"Ошибка: У читателя {self.name} нет заказа на книгу
'{order.book.title}'.")

    def blacklist(self):
        """Добавить читателя в черный список."""
        self.is_blacklisted = True
        print(f"Читатель {self.name} добавлен в черный список.")


# Класс Книга
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def __str__(self):
        return f"'{self.title}' by {self.author}"


# Класс Заказ
class Order:
    def __init__(self, book, reader):
        self.book = book
        self.reader = reader
        self.issue_date = datetime.now()
        self.return_date = None

    def return_book(self):
        """Пометить книгу как возвращенную."""
        self.return_date = datetime.now()
        print(f"Книга '{self.book.title}' возвращена читателем {self.reader.name}.")


# Класс Каталог
class Catalog:
    def __init__(self):
        self.books = []  # Список книг в каталоге

    def add_book(self, book):
        """Добавить книгу в каталог."""
        self.books.append(book)
        print(f"Книга '{book.title}' добавлена в каталог.")

    def search_book(self, book):
        """Поиск книги в каталоге."""
        return book in self.books


# Класс Библиотекарь
class Librarian(Person):
    def issue_book(self, order):
        """Выдать книгу читателю."""
        print(f"Библиотекарь {self.name} выдал книгу '{order.book.title}' читателю
{order.reader.name}.")
```

```
# Интерфейс для управления
class IManageable:
    def manage_blacklist(self, reader):
        pass


# Класс Администратор
class Administrator(Person, IManageable):
    def __init__(self, name):
        super().__init__(name)
        self.blacklist = []

    def manage_blacklist(self, reader):
        """Добавить читателя в черный список."""
        if not reader.is_blacklisted:
            reader.blacklist()
            self.blacklist.append(reader)
            print(f"Администратор {self.name} добавил читателя {reader.name} в черный
список.")
        else:
            print(f"Ошибка: Читатель {reader.name} уже находится в черном списке.")


# Вспомогательная функция для безопасного ввода данных
def safe_input(prompt, error_message="Некорректный ввод. Попробуйте снова.",
validator=lambda x: x):
    while True:
        try:
            value = input(prompt)
            if validator(value):
                return value
            else:
                print(error_message)
        except Exception as e:
            print(f"Ошибка: {e}. Попробуйте снова.")


# Основная программа
def main():
    print("Добро пожаловать в систему библиотеки!")

    # Создание объектов
    librarian = Librarian("Мария Петровна")
    admin = Administrator("Александр Сидоров")
    catalog = Catalog()

    readers = {}
    books = {}

    while True:
        print("\nВыберите действие:")
        print("1. Добавить читателя")
        print("2. Добавить книгу в каталог")
        print("3. Оформить заказ на книгу")
        print("4. Вернуть книгу")
        print("5. Добавить читателя в черный список")
        print("6. Вывести информацию о читателях и книгах")
        print("7. Выход")
```

```
choice = safe_input(
    "Введите номер действия: ",
    error_message="Ошибка: Введите число от 1 до 7.",
    validator=lambda x: x.isdigit() and 1 <= int(x) <= 7
)

if choice == "1":
    try:
        name = safe_input("Введите имя читателя: ")
        if name in readers:
            print(f"Ошибка: Читатель с именем '{name}' уже существует.")
        else:
            readers[name] = Reader(name)
            print(f"Читатель '{name}' успешно добавлен.")
    except Exception as e:
        print(f"Ошибка при добавлении читателя: {e}")

elif choice == "2":
    try:
        title = safe_input("Введите название книги: ")
        author = safe_input("Введите автора книги: ")
        book = Book(title, author)
        catalog.add_book(book)
        books[title] = book
        print(f"Книга '{title}' успешно добавлена в каталог.")
    except Exception as e:
        print(f"Ошибка при добавлении книги: {e}")

elif choice == "3":
    try:
        reader_name = safe_input("Введите имя читателя: ")
        if reader_name not in readers:
            print(f"Ошибка: Читатель с именем '{reader_name}' не найден.")
            continue

        book_title = safe_input("Введите название книги: ")
        if book_title not in books:
            print(f"Ошибка: Книга с названием '{book_title}' не найдена.")
            continue

        reader = readers[reader_name]
        book = books[book_title]
        order = reader.place_order(book, catalog)
        if order:
            librarian.issue_book(order)
    except Exception as e:
        print(f"Ошибка при оформлении заказа: {e}")

elif choice == "4":
    try:
        reader_name = safe_input("Введите имя читателя: ")
        if reader_name not in readers:
            print(f"Ошибка: Читатель с именем '{reader_name}' не найден.")
            continue

        book_title = safe_input("Введите название книги: ")
        if book_title not in books:
            print(f"Ошибка: Книга с названием '{book_title}' не найдена.")
            continue
```

```
                reader = readers[reader_name]
                book = books[book_title]
                for order in reader.orders:
                    if order.book == book:
                        reader.return_book(order)
                        break
                else:
                    print(f"Ошибка: У читателя '{reader_name}' нет заказа на книгу
'{book_title}'.")
            except Exception as e:
                print(f"Ошибка при возврате книги: {e}")

        elif choice == "5":
            try:
                reader_name = safe_input("Введите имя читателя: ")
                if reader_name not in readers:
                    print(f"Ошибка: Читатель с именем '{reader_name}' не найден.")
                    continue

                reader = readers[reader_name]
                admin.manage_blacklist(reader)
            except Exception as e:
                print(f"Ошибка при добавлении в черный список: {e}")

        elif choice == "6":
            try:
                print("\nСписок читателей:")
                for name, reader in readers.items():
                    status = " (в черном списке)" if reader.is_blacklisted else ""
                    print(f"- {name}{status}")

                print("\nСписок книг в каталоге:")
                for title, book in books.items():
                    print(f"- {book}")
            except Exception as e:
                print(f"Ошибка при выводе информации: {e}")

        elif choice == "7":
            print("Программа завершена.")
            break


if __name__ == "__main__":
    main()
```

Рисунки с результатами работы программы lab2_2.py:

```
C:\Users\ilyag\AppData\Local\Programs\Python\Python313\python.exe D:\labs_university\SSP\lab1\spp_po11\reports\Golovach_Ilya\5\lab2\src\lab2_2.py
Добро пожаловать в систему библиотеки!

Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 1
Введите имя читателя: Сергей
Читатель 'Сергей' успешно добавлен.

Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 2
Введите название книги: Война и Мир
Введите автора книги: Лев Толстой
Книга 'Война и Мир' добавлена в каталог.
Книга 'Война и Мир' успешно добавлена в каталог.

Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 6

Список читателей:
- Сергей

Список книг в каталоге:
- 'Война и Мир' by Лев Толстой
```

```
Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 3
Введите имя читателя: Сергей
Введите название книги: Лев Толстой
Ошибка: Книга с названием 'Лев Толстой' не найдена.

Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 3
Введите имя читателя: Сергей
Введите название книги: Война и Мир
Читатель Сергей оформил заказ на книгу 'Война и Мир'.
Библиотекарь Мария Петровна выдал книгу 'Война и Мир' читателю Сергей.

Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 6

Список читателей:
- Сергей

Список книг в каталоге:
- 'Война и Мир' by Лев Толстой
```

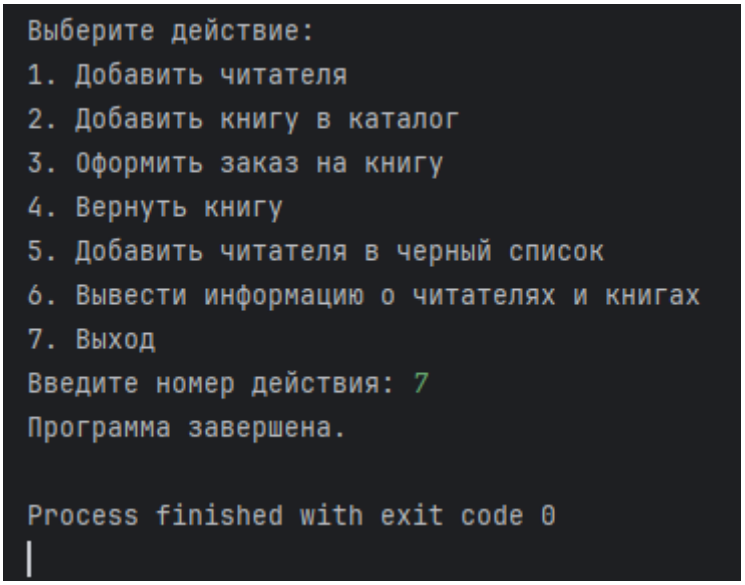
```
Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 4
Введите имя читателя: Сергей
Введите название книги: Война и Мир
Книга 'Война и Мир' возвращена читателем Сергей.
Читатель Сергей вернул книгу 'Война и Мир'.

Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 5
Введите имя читателя: Сергей
Читатель Сергей добавлен в черный список.
Администратор Александр Сидоров добавил читателя Сергей в черный список.

Выберите действие:
1. Добавить читателя
2. Добавить книгу в каталог
3. Оформить заказ на книгу
4. Вернуть книгу
5. Добавить читателя в черный список
6. Вывести информацию о читателях и книгах
7. Выход
Введите номер действия: 6

Список читателей:
- Сергей (в черном списке)

Список книг в каталоге:
- 'Война и Мир' by Лев Толстой
```



Вывод закрепил навыки объектно-ориентированного программирования на языке Python.