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
К. А. Головач,
студент группы ПО11

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

Брест 2025

Вариант 6

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___

6) Множество вещественных чисел ограниченной мощности – Предусмотреть возможность пересечения 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:

task_1.py:

```
class LimitedRealSet:
    def __init__(self, max_size, initial_elements=None):
        if initial_elements is None:
            initial_elements = []
        if len(initial_elements) > max_size:
            raise ValueError("Количество начальных элементов превышает максимальную мощность множества")
        self.max_size = max_size
        self.elements = list(set(initial_elements))  # Уникальные элементы

    def add(self, value):
        if value in self.elements:
            print(f"Элемент {value} уже существует в множестве")
            return False
        if len(self.elements) >= self.max_size:
            print("Множество достигло максимальной мощности. Элемент не может быть добавлен")
            return False
        self.elements.append(value)
        print(f"Элемент {value} добавлен успешно")
        return True

    def remove(self, value):
        if value not in self.elements:
            print(f"Элемент {value} не найден в множестве")
            return False
        self.elements.remove(value)
        print(f"Элемент {value} удалён успешно")
        return True

    def intersection(self, other_set):
        if not isinstance(other_set, LimitedRealSet):
            raise TypeError("Пересечение возможно только с объектами типа LimitedRealSet")
        common_elements = list(set(self.elements).intersection(other_set.elements))
        return LimitedRealSet(max(self.max_size, other_set.max_size), common_elements)

    def contains(self, value):
        return value in self.elements

    def __str__(self):
        return f"Множество: {self.elements}, Максимальная мощность: {self.max_size}"

    def __eq__(self, other):
        if not isinstance(other, LimitedRealSet):
            return False
        return set(self.elements) == set(other.elements)


def get_float(prompt):
    while True:
        try:
            return float(input(prompt))
        except ValueError:
            print("Ошибка: Введите вещественное число.")
```

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


def create_set(set_number):
    max_size = get_integer(f"Введите максимальную мощность {set_number} множества: ")
    initial_elements = input(
        f"Введите начальные элементы {set_number} множества через пробел (или оставьте пустым): "
    ).split()
    initial_elements = [float(x) for x in initial_elements] if initial_elements else []
    return LimitedRealSet(max_size, initial_elements)


def add_element(set1, set2):
    set_num = input("В какое множество добавить элемент? (1 или 2): ")
    value = get_float("Введите значение элемента: ")
    target_set = set1 if set_num == "1" else set2 if set_num == "2" else None
    if target_set:
        target_set.add(value)
    else:
        print("Некорректный выбор множества.")


def remove_element(set1, set2):
    set_num = input("Из какого множества удалить элемент? (1 или 2): ")
    value = get_float("Введите значение элемента: ")
    target_set = set1 if set_num == "1" else set2 if set_num == "2" else None
    if target_set:
        target_set.remove(value)
    else:
        print("Некорректный выбор множества.")


def check_element(set1, set2):
    set_num = input("В каком множестве проверить принадлежность? (1 или 2): ")
    value = get_float("Введите значение элемента: ")
    target_set = set1 if set_num == "1" else set2 if set_num == "2" else None
    if target_set:
        print(f"Элемент {value} {'принадлежит' if target_set.contains(value) else 'не принадлежит'} множеству.")
    else:
        print("Некорректный выбор множества.")


def intersect_sets(set1, set2):
    result = set1.intersection(set2)
    print(f"Результат пересечения множеств: {result}")


def compare_sets(set1, set2):
    print("Множества равны." if set1 == set2 else "Множества не равны.")


def show_sets(set1, set2):
    print(f"Первое множество: {set1}")
    print(f"Второе множество: {set2}")
```

```
def handle_set_operation(choice, set1, set2):
    actions = {
        "1": lambda: add_element(set1, set2),
        "2": lambda: remove_element(set1, set2),
        "3": lambda: check_element(set1, set2),
        "4": lambda: intersect_sets(set1, set2),
        "5": lambda: compare_sets(set1, set2),
        "6": lambda: show_sets(set1, set2),
    }
    action = actions.get(choice, lambda: print("Некорректный выбор. Попробуйте снова."))
    action()


def print_menu():
    print("""
Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Пересечь множества
5. Сравнить множества
6. Вывести информацию о множествах
7. Выход
""")


def main():
    print("Добро пожаловать в программу для работы с множествами вещественных чисел ограниченной
мощности!")
    try:
        set1 = create_set("первого")
        print(f"Создано первое множество: {set1}\n")
        set2 = create_set("второго")
        print(f"Создано второе множество: {set2}\n")
    except ValueError as e:
        print(f"Ошибка при создании множества: {e}")
        return
    while True:
        print_menu()
        choice = input("Введите номер действия: ")
        if choice == "7":
            print("Программа завершена.")
            break
        handle_set_operation(choice, set1, set2)


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

Рисунки с результатами работы программы lab2_1.py:

```
C:\Users\kirja\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\kirja\OneDrive\Pa
Добро пожаловать в программу для работы с множествами вещественных чисел ограниченной мощност
Введите максимальную мощность первого множества: 5
Введите начальные элементы первого множества через пробел (или оставьте пустым): 1 2 3 4 5
Создано первое множество: Множество: [1.0, 2.0, 3.0, 4.0, 5.0], Максимальная мощность: 5

Введите максимальную мощность второго множества: 5
Введите начальные элементы второго множества через пробел (или оставьте пустым): 5 6 7 8 9
Создано второе множество: Множество: [5.0, 6.0, 7.0, 8.0, 9.0], Максимальная мощность: 5


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

Введите номер действия:
```

```
Введите номер действия: 1
В какое множество добавить элемент? (1 или 2): 1
Введите значение элемента: 6
Элемент 6.0 добавлен успешно

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

Введите номер действия: 6
Первое множество: Множество: [1.0, 2.0, 3.0, 4.0, 6.0], Максимальная мощность: 5
Второе множество: Множество: [5.0, 6.0, 7.0, 8.0, 9.0], Максимальная мощность: 5

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

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

Введите номер действия: 2
Из какого множества удалить элемент? (1 или 2): 1
Введите значение элемента: 6
Элемент 6.0 удалён успешно

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

Введите номер действия: 6
Первое множество: Множество: [1.0, 2.0, 3.0, 4.0], Максимальная мощность: 5
Второе множество: Множество: [5.0, 6.0, 7.0, 8.0, 9.0], Максимальная мощность: 5
```

```
Выберите действие:
1. Добавить элемент в множество
2. Удалить элемент из множества
3. Проверить принадлежность элемента множеству
4. Пересечь 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: 9
Элемент 9.0 не принадлежит множеству.

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

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

Введите номер действия: 4
Результат пересечения множеств: Множество: [], Максимальная мощность: 5

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

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

6) Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от

услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

Выполнение:

Код программы:

task_2.py:

```
class Subscriber:
    def __init__(self, name, phone_number):
        self.name = name
        self.phone_number = phone_number
        self.services = []
        self.is_active = True
        self.balance = 0.0

    def pay_bill(self, amount):
        if amount <= 0:
            print("Ошибка: Сумма платежа должна быть положительной.")
            return
        self.balance += amount
        print(f"Абонент {self.name} пополнил баланс на {amount}. Текущий баланс: {self.balance}")

    def request_service_change(self, admin, new_services):
        if not self.is_active:
            print(f"Ошибка: Абонент {self.name} временно отключен.")
            return
        admin.change_services(self, new_services)

    def request_number_change(self, admin, new_number):
        if not self.is_active:
            print(f"Ошибка: Абонент {self.name} временно отключен.")
            return
        admin.change_number(self, new_number)

    def __str__(self):
        status = "Активен" if self.is_active else "Отключен"
        return f"Абонент: {self.name}, Номер: {self.phone_number}, Баланс: {self.balance}, Статус: {status}"


class Admin:
    def change_number(self, subscriber, new_number):
        subscriber.phone_number = new_number
        print(f"Администратор изменил номер абонента {subscriber.name} на {new_number}")

    def change_services(self, subscriber, new_services):
        subscriber.services = new_services
        print(f"Администратор обновил услуги абонента {subscriber.name}: {', '.join(new_services)}")

    def suspend_subscriber(self, subscriber):
        subscriber.is_active = False
        print(f"Администратор временно отключил абонента {subscriber.name}")


class PhoneStation:
    def __init__(self):
        self.subscribers = {}
```

```
    def add_subscriber(self, name, phone_number):
        if name in self.subscribers:
            print(f"Ошибка: Абонент с именем '{name}' уже существует.")
            return
        self.subscribers[name] = Subscriber(name, phone_number)
        print(f"Абонент '{name}' успешно добавлен.")

    def process_payment(self, name, amount):
        if name not in self.subscribers:
            print(f"Ошибка: Абонент '{name}' не найден.")
            return
        self.subscribers[name].pay_bill(amount)

    def request_service_change(self, name, admin, new_services):
        if name not in self.subscribers:
            print(f"Ошибка: Абонент '{name}' не найден.")
            return
        self.subscribers[name].request_service_change(admin, new_services)

    def request_number_change(self, name, admin, new_number):
        if name not in self.subscribers:
            print(f"Ошибка: Абонент '{name}' не найден.")
            return
        self.subscribers[name].request_number_change(admin, new_number)

    def suspend_subscriber(self, name, admin):
        if name not in self.subscribers:
            print(f"Ошибка: Абонент '{name}' не найден.")
            return
        admin.suspend_subscriber(self.subscribers[name])

    def show_info(self):
        print("\nСписок абонентов:")
        for subscriber in self.subscribers.values():
            print(subscriber)


def main():
    print("Добро пожаловать в систему телефонной станции!")
    station = PhoneStation()
    admin = Admin()

    actions = {
        "1": lambda: station.add_subscriber(
            input("Введите имя абонента: "),
            input("Введите номер телефона: ")
        ),
        "2": lambda: station.process_payment(
            input("Введите имя абонента: "),
            float(input("Введите сумму платежа: "))
        ),
        "3": lambda: station.request_service_change(
            input("Введите имя абонента: "),
            admin,
            input("Введите новые услуги через запятую: ").split(",")
        ),
        "4": lambda: station.request_number_change(
            input("Введите имя абонента: "),
            admin,
```

```
            input("Введите новый номер телефона: ")
        ),
        "5": lambda: station.suspend_subscriber(
            input("Введите имя абонента: "),
            admin
        ),
        "6": lambda: station.show_info(),
        "7": lambda: exit("Программа завершена."),
    }

    while True:
        print("""
Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход
""")
        choice = input("Введите номер действия: ")
        action = actions.get(choice, lambda: print("Некорректный выбор. Попробуйте снова."))
        action()


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

Рисунки с результатами работы программы lab2_2.py:

```
Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход

Введите номер действия: 1
Введите имя абонента: Кирилл
Введите номер телефона: +375
Абонент 'Кирилл' успешно добавлен.

Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход
```

```
Введите номер действия: 6

Список абонентов:
Абонент: Кирилл, Номер: +375, Баланс: 0.0, Статус: Активен

Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход

Введите номер действия:
```

```
Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход

Введите номер действия: 2
Введите имя абонента: Кирилл
Введите сумму платежа: 1000
Абонент Кирилл пополнил баланс на 1000.0. Текущий баланс: 1000.0

Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход
```

```
Ошибка: Абонент "123" не найден.

Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход

Введите номер действия: 4
Введите имя абонента: Кирилл
Введите новый номер телефона: 213
Администратор изменил номер абонента Кирилл на 213

Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход
```

```
Введите номер действия: 5
Введите имя абонента: Кирилл
Администратор временно отключил абонента Кирилл

Выберите действие:
1. Добавить абонента
2. Оплатить счет
3. Изменить услуги
4. Изменить номер
5. Отключить абонента
6. Вывести информацию об абонентах
7. Выход

Введите номер действия: 6

Список абонентов:
Абонент: Кирилл, Номер: 213, Баланс: 1000.0, Статус: Отключен
```

Вывод закрепил навыки объектно-ориентированного программирования на языке Python.