МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №2

Специальность ПО11

Выполнил
Зайченко С.В.
студент группы ПО11

Проверил
Крощенко А. А.
ст. преп. кафедры ИИТ

Брест 2025

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

Ход Работы

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__

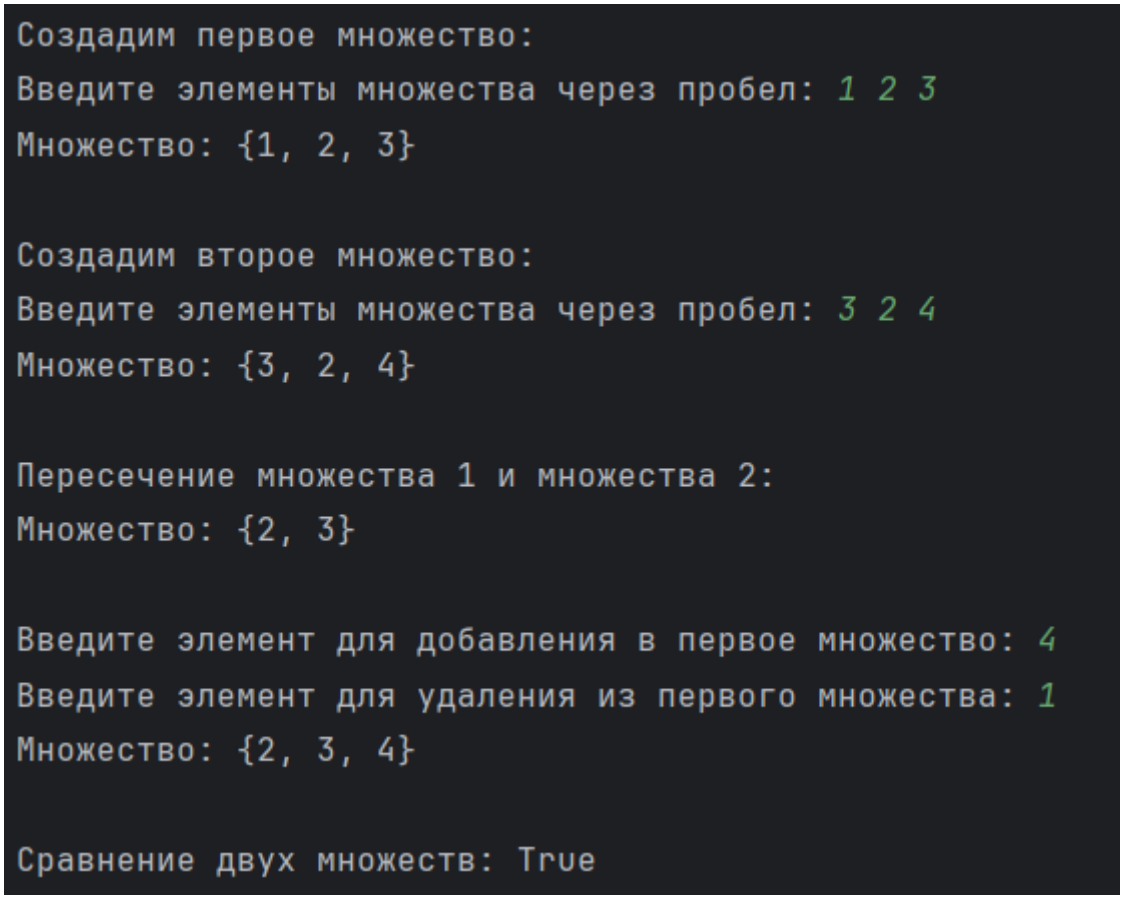
Множество символов переменной мощности – Предусмотреть возможность пересечения двух множеств, вывода на консоль элементов множества, а также метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволять создавать объекты с начальной инициализацией. Реализацию множества осуществить на базе списка. Переопределить метод __eq__, выполняющий сравнение объектов данного типа.
Код программы:

```
class SymbolSet:
  def __init__(self, initial_values=None):
    if initial_values is None:
      self._elements = []
    else:
      self._elements = []
      for val in initial_values:
        self.add(val)

  def add(self, value):
    if value not in self._elements:
      self._elements.append(value)

  def remove(self, value):
    if value in self._elements:
      self._elements.remove(value)

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

  def intersect(self, other):
    intersection = [val for val in self._elements if val in other._elements]
    return SymbolSet(intersection)
```

```
    def __str__(self):
        return "{" + ", ".join(self._elements) + "}"

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

    def display(self):
        print("Множество:", self)

    @property
    def elements(self):
        return self._elements.copy()

    @elements.setter
    def elements(self, values):
        self._elements = []
        for val in values:
            self.add(val)


def create_set_from_input():
    elements = input("Введите элементы множества через пробел: ").split()
    symbol_set = SymbolSet(elements)
    return symbol_set


if __name__ == "__main__":
    print("Создадим первое множество:")
    set1 = create_set_from_input()
    set1.display()

    print("\nСоздадим второе множество:")
    set2 = create_set_from_input()
    set2.display()

    intersected = set1.intersect(set2)
    print("\nПересечение множества 1 и множества 2:")
    intersected.display()

    element_to_add = input("\nВведите элемент для добавления в первое множество: ")
```

```
  set1.add(element_to_add)
  element_to_remove = input("Введите элемент для удаления из первого множества: ")
  set1.remove(element_to_remove)

  set1.display()

  print("\nСравнение двух множеств:", set1 == set2)
```

Рисунки с результатами работы программы:



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 Городской транспорт. На Маршрут назначаются Автобус или Троллейбус. Транспортные средства должны двигаться с определенным для каждого Маршрута интервалом. При поломке на Маршрут должен выходить резервный транспорт или увеличиваться интервал движения.

Код программы:

```
from abc import ABC, abstractmethod

class Transport(ABC):
  def __init__(self, vehicle_type, interval):
```

```
        self.vehicle_type = vehicle_type
        self.interval = interval

    @abstractmethod
    def start(self):
        pass

    def breakdown(self):
        print(f"{self.vehicle_type} сломался. Необходимо вызвать резервный транспорт.")

    def change_interval(self, new_interval):
        self.interval = new_interval
        print(f"Интервал движения на маршруте увеличен до {new_interval} минут.")

class Bus(Transport):
    def __init__(self, interval):
        super().__init__("Автобус", interval)

    def start(self):
        print(f"{self.vehicle_type} начал движение с интервалом {self.interval} минут.")

class Trolleybus(Transport):
    def __init__(self, interval):
        super().__init__("Троллейбус", interval)

    def start(self):
        print(f"{self.vehicle_type} начал движение с интервалом {self.interval} минут.")

class ReserveTransport:
    def __init__(self, vehicle_type):
        self.vehicle_type = vehicle_type

    def start(self):
        print(f"Резервный {self.vehicle_type} вышел на маршрут.")

class Route:
    def __init__(self, route_name, transport):
        self.route_name = route_name
        self.transport = transport
        self.reserve_transport = None

    def start_route(self):
        print(f"Маршрут {self.route_name} начинается.")
```

```
        self.transport.start()

    def breakdown(self):
        print(f"На маршруте {self.route_name} произошла поломка.")
        if self.reserve_transport:
            self.reserve_transport.start()
        else:
            self.transport.change_interval(self.transport.interval + 5)  # Увеличиваем интервал
на 5 минут

    def assign_reserve_transport(self, reserve_transport):
        self.reserve_transport = reserve_transport
        print(f"Резервный транспорт {reserve_transport.vehicle_type} назначен на маршрут
{self.route_name}.")

class CityTransportSystem:
    def __init__(self):
        self.routes = []

    def add_route(self, route):
        self.routes.append(route)

    def start_all_routes(self):
        for route in self.routes:
            route.start_route()

def create_transport():
    vehicle_type = input("Введите тип транспортного средства (автобус/троллейбус):
").strip().lower()
    interval = int(input("Введите интервал движения (в минутах): "))
    if vehicle_type == "автобус":
        return Bus(interval)
    elif vehicle_type == "троллейбус":
        return Trolleybus(interval)
    else:
        print("Неверный тип транспортного средства. Создаем автобус по умолчанию.")
        return Bus(interval)

def create_reserve_transport():
    vehicle_type = input("Введите тип резервного транспортного средства
(автобус/троллейбус): ").strip().lower()
    if vehicle_type == "автобус":
        return ReserveTransport("Автобус")
```

```
    elif vehicle_type == "троллейбус":
        return ReserveTransport("Троллейбус")
    else:
        print("Неверный тип резервного транспортного средства. Создаем автобус по
умолчанию.")
        return ReserveTransport("Автобус")

def create_route():
    route_name = input("Введите название маршрута: ")
    transport = create_transport()
    route = Route(route_name, transport)
    return route

def main():
    city_transport_system = CityTransportSystem()

    num_routes = int(input("Введите количество маршрутов: "))
    for _ in range(num_routes):
        route = create_route()
        reserve_transport = create_reserve_transport()
        route.assign_reserve_transport(reserve_transport)
        city_transport_system.add_route(route)

    city_transport_system.start_all_routes()

    simulate_breakdown = input("\nХотите смоделировать поломку на маршруте?
(да/нет): ").strip().lower()
    if simulate_breakdown == "да":
        route_name = input("Введите название маршрута для поломки: ")
        for route in city_transport_system.routes:
            if route.route_name == route_name:
                route.breakdown()

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

Рисунки с результатами работы программы:

```
Введите количество маршрутов: 2
Введите название маршрута: кобрин
Введите тип транспортного средства (автобус/троллейбус): автобус
Введите интервал движения (в минутах): 23
Введите тип резервного транспортного средства (автобус/троллейбус): троллейбус
Резервный транспорт Троллейбус назначен на маршрут кобрин.
Введите название маршрута: кобрин2
Введите тип транспортного средства (автобус/троллейбус): троллейбус
Введите интервал движения (в минутах): 283
Введите тип резервного транспортного средства (автобус/троллейбус): автобус
Резервный транспорт Автобус назначен на маршрут кобрин2.
Маршрут кобрин начинается.
Автобус начал движение с интервалом 23 минут.
Маршрут кобрин2 начинается.
Троллейбус начал движение с интервалом 283 минут.

Хотите смоделировать поломку на маршруте? (да/нет): да
Введите название маршрута для поломки: кобрин
На маршруте кобрин произошла поломка.
Резервный Троллейбус вышел на маршрут.
```

Вывод: Закрепил навыки объектно-ориентированного программирования на языке Python