Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №2d
По дисциплине: “СПП”

Выполнил:
Студент 3 курса
Группы ПО-12
Мартынюк В. В.
Проверила:
Кулик А. Д.

Брест 2026

Цель работы: Закрепление навыков объектно-ориентированного программирования на языке Python.

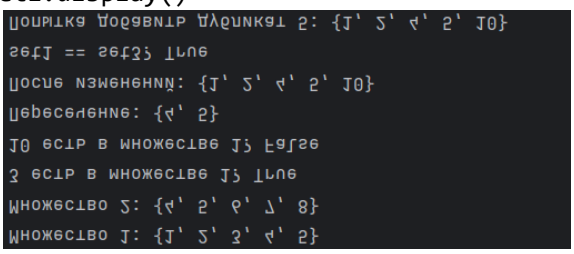
Задание 1

10) 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.

Решение:

```
class VariableSet:
    def __init__(self, initial_elements=None):
        self.elements = []

        if initial_elements:
            for item in initial_elements:
                self.add(item)

    def add(self, element):
        if element not in self.elements:
            self.elements.append(element)

    def remove(self, element):
        if element in self.elements:
            self.elements.remove(element)
        else:
            print(f"Элемент {element} не найден в
множестве")

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

    def intersection(self, other_set):
        result = VariableSet()

        for element in self.elements:
            if other_set.contains(element):
                result.add(element)

        return result

    def display(self):
        elements_str = ', '.join(str(e) for e in
self.elements)
        print(f"{{{elements_str}}}")

    def __eq__(self, other):
```

```
        if len(self.elements) !=
len(other.elements):
            return False

        for element in self.elements:
            if not other.contains(element):
                return False

        return True

    def __len__(self):
        return len(self.elements)


if __name__ == "__main__":
    set1 = VariableSet([1, 2, 3, 4, 5])
    set2 = VariableSet([4, 5, 6, 7, 8])

    print("Множество 1:", end=" ")
    set1.display()

    print("Множество 2:", end=" ")
    set2.display()

    print(f"3 есть в множестве 1?
{set1.contains(3)}")
    print(f"10 есть в множестве 1?
{set1.contains(10)}")

    intersection_set = set1.intersection(set2)
    print("Пересечение:", end=" ")
    intersection_set.display()

    set1.add(10)
    set1.remove(3)
    print("После изменений:", end=" ")
    set1.display()

    set3 = VariableSet([1, 2, 4, 5, 10])
    print(f"set1 == set3? {set1 == set3}")

    set1.add(5)
    print("Попытка добавить дубликат 5:", end="")
    set1.display()
```



Задание 2

10) 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.

Решение:

```
class Transport:
    def __init__(self, number, typ):
        self.number = number
        self.typ = typ
        self.works = True
        self.route = None

    def __str__(self):
        return f"{self.typ} {self.number}"


class Route:
    def __init__(self, num, name, interval):
        self.num = num
        self.name = name
        self.interval = interval
        self.vehicles = []
        self.reserve = []

    def add_vehicle(self, vehicle):
        vehicle.route = self
        self.vehicles.append(vehicle)
        print(f"{vehicle} назначен на маршрут {self.num}")

    def add_reserve(self, vehicle):
        self.reserve.append(vehicle)
        print(f"{vehicle} в резерве маршрута {self.num}")

    def breakdown(self, vehicle):
        if vehicle in self.vehicles:
            self.vehicles.remove(vehicle)
            print(f"{vehicle} сломался!")

            if self.reserve:
                new = self.reserve.pop(0)
                self.add_vehicle(new)
                print(f"Резервный {new} вышел на линию")
            else:
                self.interval = self.interval * 2
                print(f"Резерва нет! Интервал увеличен до {self.interval} мин")

    def info(self):
        print(f"\nМаршрут {self.num}: {self.name}")
        print(f"Интервал: {self.interval} мин")
        print(f"На линии: {[str(v) for v in self.vehicles]}")
        print(f"В резерве: {[str(v) for v in self.reserve]}")
```

```
if __name__ == "__main__":
    route_5 = Route(5, "Центр - Северный", 10)

    bus1 = Transport("A001AA", "автобус")
    bus2 = Transport("A002AA", "автобус")
    trolley = Transport("T001TT", "троллейбус")
    reserve = Transport("A003AA", "автобус")

    route_5.add_vehicle(bus1)
    route_5.add_vehicle(bus2)
    route_5.add_vehicle(trolley)
    route_5.add_reserve(reserve)

    route_5.info()

    print("\nАвария")
    route_5.breakdown(bus1)

    route_5.info()
```

```
автобус A001AA назначен на маршрут 5
автобус A002AA назначен на маршрут 5
троллейбус T001TT назначен на маршрут 5
автобус A003AA в резерве маршрута 5

Маршрут 5: Центр - Северный
Интервал: 10 мин
На линии: ['автобус A001AA', 'автобус A002AA', 'троллейбус T001TT']
В резерве: ['автобус A003AA']

Авария
автобус A001AA сломался!
автобус A003AA назначен на маршрут 5
Резервный автобус A003AA вышел на линию

Маршрут 5: Центр - Северный
Интервал: 10 мин
На линии: ['автобус A002AA', 'троллейбус T001TT', 'автобус A003AA']
В резерве: []
```

Вывод: В результате выполнения лабораторной работы были закреплены знания объектно-ориентированного программирования Python при решении задач.