МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

По дисциплине “Системное программирование”

Специальность “Программная инженерия”

Выполнил:
А.С. Кот
Студент группы ПО-13
Проверил:
А.Д. Кулик
преп.-стажёр кафедры ИИТ

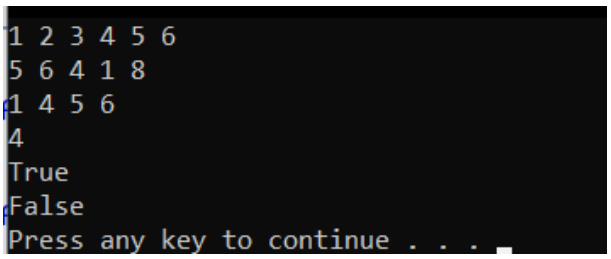
Брест 2026

Цель: закрепить навыки 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.

```
class CharSet:
    def __init__(self, initial=None):
        self._data = []
        if initial:
            for item in initial:
                if item not in self._data:
                    self._data.append(item)
    def add(self, value):
        if value not in self._data:
            self._data.append(value)
    def remove(self, value):
        if value in self._data:
            self._data.remove(value)
    def contains(self, value):
        return value in self._data
    def intersection(self, other):
        result = CharSet()
        for item in self._data:
            if other.contains(item):
                result.add(item)
        return result
    def __eq__(self, other):
        if not isinstance(other, CharSet):
            return False
        if len(self._data) != len(other._data):
            return False
        for item in self._data:
            if item not in other._data:
                return False
        return True
    def __str__(self):
        return " ".join(self._data)
set1 = CharSet(input().split())
set2 = CharSet(input().split())
intersection_set = set1.intersection(set2)
print(intersection_set)
value = input()
print(set1.contains(value))
set1.add(value)
set1.remove(value)
print(set1 == set2)
```



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

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.

```
from abc import ABC, abstractmethod
class Transport(ABC):
    def __init__(self, number):
        self.number = number
        self.route = None
        self.is_working = True
    def assign_route(self, route):
        self.route = route
    def breakdown(self):
        self.is_working = False
        print(f"{self} is broken")
        if self.route:
            self.route.handle_breakdown(self)
    @abstractmethod
    def move(self):
        pass
    def __str__(self):
        return f"{self.__class__.__name__} #{self.number}"
class Bus(Transport):
    def move(self):
        print(f"{self} moves on route {self.route.number}")
class Trolleybus(Transport):
    def move(self):
        print(f"{self} moves on route {self.route.number}")
class Route:
    def __init__(self, number, interval):
        self.number = number
        self.interval = interval
        self.transports = []
        self.reserve = []
    def add_transport(self, transport):
        transport.assign_route(self)
        self.transports.append(transport)
    def add_reserve(self, transport):
        self.reserve.append(transport)
    def handle_breakdown(self, transport):
        print(f"Handling breakdown on route {self.number}")
        if self.reserve:
            reserve_transport = self.reserve.pop(0)
            reserve_transport.assign_route(self)
            self.transports.append(reserve_transport)
```

```
            print(f"Reserve {reserve_transport} assigned to route")
        else:
            self.interval += 5
            print(f"No reserve available. Interval increased to {self.interval}")
    def show_status(self):
        print(f"\nRoute {self.number}")
        print(f"Interval: {self.interval} minutes")
        print("Active transports:")
        for t in self.transports:
            if t.is_working:
                print(f"{t}")
route1 = Route(10, 15)
bus1 = Bus(101)
trolley1 = Trolleybus(201)
reserve_bus = Bus(999)
route1.add_transport(bus1)
route1.add_transport(trolley1)
route1.add_reserve(reserve_bus)
route1.show_status()
bus1.move()
trolley1.move()
bus1.breakdown()
route1.show_status()
trolley1.breakdown()
route1.show_status()
```

```
Route 10
Interval: 15 minutes
Active transports:
Bus #101
Trolleybus #201
Bus #101 moves on route 10
Trolleybus #201 moves on route 10
Bus #101 is broken
Handling breakdown on route 10
Reserve Bus #999 assigned to route

Route 10
Interval: 15 minutes
Active transports:
Trolleybus #201
Bus #999
Trolleybus #201 is broken
Handling breakdown on route 10
No reserve available. Interval increased to 20

Route 10
Interval: 20 minutes
Active transports:
Bus #999
Press any key to continue . . .
```

Вывод: были отработаны навыки объектно-ориентированного программирования Python при решении практических задач.