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
Д. М. Андросюк
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
01.03.2025 г.

Брест 2025

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

Ход Работы

Задание 1

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__

Равнобедренный треугольник, заданный длинами сторон – Предусмотреть возможность определения площади и периметра, а также логический метод, определяющий существует или такой треугольник. Конструктор должен позволять создавать объекты с начальной инициализацией. Переопределить метод __eq__, выполняющий сравнение объектов данного типа.

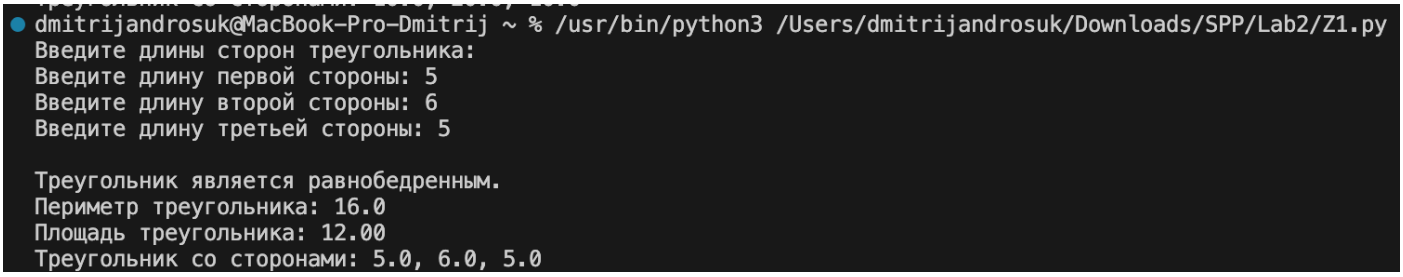
Код программы:

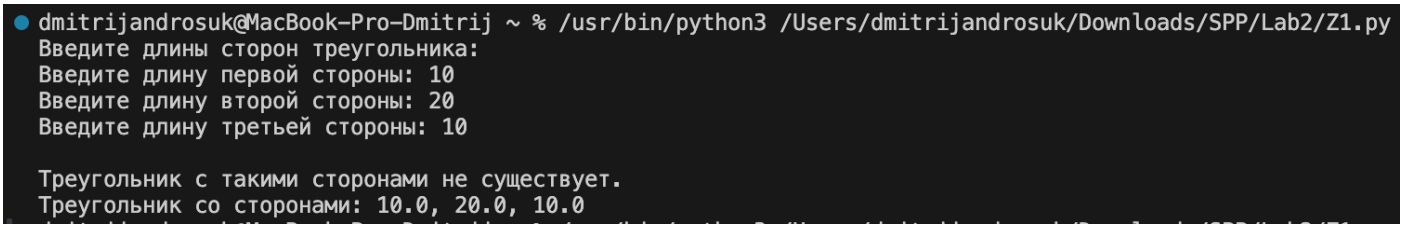
```
import math

class IsoscelesTriangle:
    def __init__(self, side_a, side_b, side_c):

        self.side_a = side_a
        self.side_b = side_b
        self.side_c = side_c

    def is_isosceles(self):
        return (self.side_a == self.side_b) or (self.side_a == self.side_c) or (self.side_b == self.side_c)

    def exists(self):
        return (self.side_a + self.side_b > self.side_c) and \
               (self.side_a + self.side_c > self.side_b) and \
               (self.side_b + self.side_c > self.side_a)

    def perimeter(self):
        return self.side_a + self.side_b + self.side_c

    def area(self):
        if not self.exists():
            return 0
        s = self.perimeter() / 2
        return math.sqrt(s * (s - self.side_a) * (s - self.side_b) * (s - self.side_c))

    def __str__(self):
        return f"Треугольник со сторонами: {self.side_a}, {self.side_b}, {self.side_c}"

    def __eq__(self, other):
        if not isinstance(other, IsoscelesTriangle):
            return False
```

```
        return sorted([self.side_a, self.side_b, self.side_c]) == sorted([other.side_a, other.side_b, other.side_c])

    @property
    def side_a(self):
        return self._side_a

    @side_a.setter
    def side_a(self, value):
        if value <= 0:
            raise ValueError("Длина стороны должна быть положительной")
        self._side_a = value

    @property
    def side_b(self):
        return self._side_b

    @side_b.setter
    def side_b(self, value):
        if value <= 0:
            raise ValueError("Длина стороны должна быть положительной")
        self._side_b = value

    @property
    def side_c(self):
        return self._side_c

    @side_c.setter
    def side_c(self, value):
        if value <= 0:
            raise ValueError("Длина стороны должна быть положительной")
        self._side_c = value


def input_positive_number(prompt):
    while True:
        try:
            value = float(input(prompt))
            if value <= 0:
                print("Число должно быть положительным. Попробуйте снова.")
            else:
                return value
        except ValueError:
            print("Ошибка: введите число.")


if __name__ == "__main__":
    print("Введите длины сторон треугольника:")
    side_a = input_positive_number("Введите длину первой стороны: ")
    side_b = input_positive_number("Введите длину второй стороны: ")
    side_c = input_positive_number("Введите длину третьей стороны: ")

    triangle = IsoscelesTriangle(side_a, side_b, side_c)

    if not triangle.exists():
```

```
            print("\nТреугольник с такими сторонами не существует.")
        else:
            if triangle.is_isosceles():
                print("\nТреугольник является равнобедренным.")
                print(f"Периметр треугольника: {triangle.perimeter()}")
                print(f"Площадь треугольника: {triangle.area():.2f}")
            else:
                print("\nТреугольник не является равнобедренным.")
        print(triangle)
```

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 Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

Код программы:

```
from abc import ABC, abstractmethod
from typing import List

class Person(ABC):
    def __init__(self, name: str, person_id: int):
        self.name = name
        self.person_id = person_id

    @abstractmethod
    def display_info(self):
        pass

class Student(Person):
    def __init__(self, name: str, student_id: int):
        super().__init__(name, student_id)
        self.courses = []
```

```
    def enroll(self, course):
        self.courses.append(course)
        course.add_student(self)

    def display_info(self):
        print(f"Студент: {self.name}, ID: {self.person_id}")

class Teacher(Person):
    def __init__(self, name: str, teacher_id: int):
        super().__init__(name, teacher_id)
        self.courses = []

    def create_course(self, course_name: str):
        course = Course(course_name, self)
        self.courses.append(course)
        return course

    def assign_grade(self, student, course, grade_value: int):
        grade = Grade(student, course, grade_value)
        Archive.add_grade(grade)

    def display_info(self):
        print(f"Преподаватель: {self.name}, ID: {self.person_id}")

class Course:
    def __init__(self, name: str, teacher: Teacher):
        self.name = name
        self.teacher = teacher
        self.students = []

    def add_student(self, student: Student):
        self.students.append(student)

    def display_info(self):
        print(f"Курс: {self.name}, Преподаватель: {self.teacher.name}")
        print("Записанные студенты:")
        for student in self.students:
            print(f" - {student.name}")

class Grade:
    def __init__(self, student: Student, course: Course, grade_value: int):
        self.student = student
        self.course = course
        self.grade_value = grade_value

    def display_info(self):
        print(f"Оценка: {self.grade_value}, Студент: {self.student.name}, Курс: {self.course.name}")

class Archive:
    _grades = []

    @classmethod
    def add_grade(cls, grade: Grade):
        cls._grades.append(grade)
```

```
    @classmethod
    def display_grades(cls):
        print("Архив оценок:")
        for grade in cls._grades:
            grade.display_info()

def create_teacher():
    name = input("Введите имя преподавателя: ")
    teacher_id = int(input("Введите ID преподавателя: "))
    return Teacher(name, teacher_id)

def create_student():
    name = input("Введите имя студента: ")
    student_id = int(input("Введите ID студента: "))
    return Student(name, student_id)

def create_course(teacher):
    course_name = input("Введите название курса: ")
    return teacher.create_course(course_name)

def enroll_student(students, courses):
    if not students:
        print("Нет доступных студентов.")
        return
    if not courses:
        print("Нет доступных курсов.")
        return

    print("Выберите студента:")
    for i, student in enumerate(students):
        print(f"{i + 1}. {student.name} (ID: {student.person_id})")
    student_index = int(input("Введите номер студента: ")) - 1

    print("Выберите курс:")
    for i, course in enumerate(courses):
        print(f"{i + 1}. {course.name} (Преподаватель: {course.teacher.name})")
    course_index = int(input("Введите номер курса: ")) - 1

    students[student_index].enroll(courses[course_index])
    print(f"Студент {students[student_index].name} записан на курс {courses[course_index].name}.")

def assign_grade(students, courses):
    if not students:
        print("Нет доступных студентов.")
        return
    if not courses:
        print("Нет доступных курсов.")
        return

    print("Выберите студента:")
    for i, student in enumerate(students):
        print(f"{i + 1}. {student.name} (ID: {student.person_id})")
    student_index = int(input("Введите номер студента: ")) - 1

    print("Выберите курс:")
```

```
    for i, course in enumerate(courses):
        print(f"{i + 1}. {course.name} (Преподаватель: {course.teacher.name})")
    course_index = int(input("Введите номер курса: ")) - 1

    grade_value = int(input("Введите оценку: "))
    courses[course_index].teacher.assign_grade(students[student_index], courses[course_index],
grade_value)
    print(f"Оценка {grade_value} выставлена студенту {students[student_index].name} за курс
{courses[course_index].name}.")

def main():
    students = []
    courses = []
    teacher = None

    while True:
        print("\n--- Меню ---")
        print("1. Создать преподавателя")
        print("2. Создать студента")
        print("3. Создать курс")
        print("4. Записать студента на курс")
        print("5. Выставить оценку студенту")
        print("6. Показать курсы")
        print("7. Показать архив оценок")
        print("8. Выйти")
        choice = input("Введите ваш выбор: ")

        if choice == "1":
            teacher = create_teacher()
            print(f"Преподаватель {teacher.name} создан.")
        elif choice == "2":
            student = create_student()
            students.append(student)
            print(f"Студент {student.name} создан.")
        elif choice == "3":
            if teacher:
                course = create_course(teacher)
                courses.append(course)
                print(f"Курс {course.name} создан.")
            else:
                print("Сначала создайте преподавателя.")
        elif choice == "4":
            enroll_student(students, courses)
        elif choice == "5":
            assign_grade(students, courses)
        elif choice == "6":
            if courses:
                for course in courses:
                    course.display_info()
            else:
                print("Нет доступных курсов.")
        elif choice == "7":
            Archive.display_grades()
        elif choice == "8":
            print("Выход...")
```

```
            break
        else:
            print("Неверный выбор. Попробуйте снова.")

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

Рисунки с результатами работы программы:

```
--- Меню ---
1. Создать преподавателя
2. Создать студента
3. Создать курс
4. Записать студента на курс
5. Выставить оценку студенту
6. Показать курсы
7. Показать архив оценок
8. Выйти
Введите ваш выбор: 2
Введите имя студента: Dima
Введите ID студента: 1
Студент Dima создан.

--- Меню ---
1. Создать преподавателя
2. Создать студента
3. Создать курс
4. Записать студента на курс
5. Выставить оценку студенту
6. Показать курсы
7. Показать архив оценок
8. Выйти
Введите ваш выбор: 3
Сначала создайте преподавателя.

--- Меню ---
1. Создать преподавателя
2. Создать студента
3. Создать курс
4. Записать студента на курс
5. Выставить оценку студенту
6. Показать курсы
7. Показать архив оценок
8. Выйти
Введите ваш выбор: 1
Введите имя преподавателя: Vasya
Введите ID преподавателя: 3
Преподаватель Vasya создан.

--- Меню ---
1. Создать преподавателя
2. Создать студента
3. Создать курс
4. Записать студента на курс
5. Выставить оценку студенту
6. Показать курсы
7. Показать архив оценок
8. Выйти
Введите ваш выбор: 3
Введите название курса: Mathem
Курс Mathem создан.
```

```
--- Меню ---
1. Создать преподавателя
2. Создать студента
3. Создать курс
4. Записать студента на курс
5. Выставить оценку студенту
6. Показать курсы
7. Показать архив оценок
8. Выйти
Введите ваш выбор: 6
Курс: Mathem, Преподаватель: Vasya
Записанные студенты:

--- Меню ---
1. Создать преподавателя
2. Создать студента
3. Создать курс
4. Записать студента на курс
5. Выставить оценку студенту
6. Показать курсы
7. Показать архив оценок
8. Выйти
Введите ваш выбор: 5
Выберите студента:
1. Dima (ID: 1)
Введите номер студента: 1
Выберите курс:
1. Mathem (Преподаватель: Vasya)
Введите номер курса: 1
Введите оценку: 6
Оценка 6 выставлена студенту Dima за курс Mathem.
```

Вывод: Закрепил навыки объектно-ориентированного программирования на языке Python.