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
Лесько М.И.
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
05.04.2025 г.

Брест 2025

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

Ход Работы

Задание 1

Равносторонний 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 Triangle:
    def __init__(self, a=1.0, b=1.0, c=1.0):
        """
        Конструктор класса Triangle с проверкой на равносторонность.
        :param a: длина первой стороны
        :param b: длина второй стороны
        :param c: длина третьей стороны
        """
        self.set_sides(a, b, c)

    def set_sides(self, a, b, c):
        """
        Устанавливает стороны треугольника с проверкой на равносторонность.
        :param a: длина первой стороны
        :param b: длина второй стороны
        :param c: длина третьей стороны
        """
        if not math.isclose(a, b, rel_tol=1e-9) or not math.isclose(a, c, rel_tol=1e-9):
            raise ValueError("Треугольник должен быть равносторонним (все стороны равны)")

        if a <= 0 or b <= 0 or c <= 0:
            raise ValueError("Длины сторон должны быть положительными числами")

        self._a = a
        self._b = b
        self._c = c

    @property
    def a(self):
        return self._a

    @property
    def b(self):
        return self._b
```

```
    @property
    def c(self):
        return self._c

    def perimeter(self):
        """Вычисление периметра треугольника"""
        return self._a + self._b + self._c

    def area(self):
        """Вычисление площади треугольника"""
        p = self.perimeter() / 2
        return math.sqrt(p * (p - self._a) * (p - self._b) * (p - self._c))

    def is_valid(self):
        """
        Проверка, соответствует ли треугольник неравенству треугольника
        Для равностороннего треугольника всегда True если стороны положительные
        """
        return (self._a + self._b > self._c and
                self._a + self._c > self._b and
                self._b + self._c > self._a)

    def __str__(self):
        """Строковое представление объекта"""
        return f"Равносторонний треугольник со сторонами: a={self._a}, b={self._b}, c={self._c}"

    def __eq__(self, other):
        """
        Сравнение двух треугольников на равенство сторон
        :param other: другой объект Triangle
        :return: True, если треугольники равны (по сторонам), иначе False
        """
        if not isinstance(other, Triangle):
            return False
        return (math.isclose(self._a, other._a) and
                math.isclose(self._b, other._b) and
                math.isclose(self._c, other._c))

def validate_input(prompt):
    """Функция для валидации ввода чисел"""
    while True:
        try:
            value = float(input(prompt))
            if value <= 0:
                print("Длина стороны должна быть положительной. Попробуйте снова.")
                continue
            return value
        except ValueError:
            print("Пожалуйста, введите корректное число.")
```

```
def main():
    print("Введите длины сторон равностороннего треугольника:")
    while True:
        try:
            a = validate_input("Введите длину стороны a: ")
            b = validate_input("Введите длину стороны b: ")
            c = validate_input("Введите длину стороны c: ")

            triangle = Triangle(a, b, c)
            break
        except ValueError as e:
            print(f"Ошибка: {e}")
            print("Пожалуйста, введите равные длины для всех трех сторон.\n")

    print("\nИнформация о треугольнике:")
    print(triangle)
    print(f"Периметр: {triangle.perimeter()}")
    print(f"Площадь: {triangle.area():.2f}")
    print(f"Треугольник существует: {'да' if triangle.is_valid() else 'нет'}")
    print("\nСоздадим второй треугольник для сравнения:")
    while True:
        try:
            a2 = validate_input("Введите длину стороны a для второго треугольника: ")
            b2 = validate_input("Введите длину стороны b для второго треугольника: ")
            c2 = validate_input("Введите длину стороны c для второго треугольника: ")

            triangle2 = Triangle(a2, b2, c2)
            break
        except ValueError as e:
            print(f"Ошибка: {e}")
            print("Пожалуйста, введите равные длины для всех трех сторон.\n")

    print(f"\nСравнение треугольников: {'равны' if triangle == triangle2 else 'не равны'}")

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

```
Введите длину стороны a: 3
Введите длину стороны b: 3
Введите длину стороны c: 3

Информация о треугольнике:
Равносторонний треугольник со сторонами: a=3.0, b=3.0, c=3.0
Периметр: 9.0
Площадь: 3.90
Треугольник существует: да
```

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.
```

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