Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №2

По дисциплине «Современные платформы программирования»

Выполнила:

Студентка 3 курса

Группы ПО-11

Лозицкая М.А.

Проверил:

Козик И.Д.

Брест 2025

**Цель работы:** закрепить навыки объектно-ориентированного программирования на языке Python

**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\_\_

Вариант 3

Прямоугольный 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 RightTriangle:

def \_\_init\_\_(self, a: float, b: float, c: float):

"""Конструктор с начальной инициализацией длин сторон"""

self.a = a

self.b = b

self.c = c

@property

def a(self) -> float:

return self.\_a

@a.setter

def a(self, value: float):

if value <= 0:

raise ValueError("Длина стороны должна быть положительной")

self.\_a = value

@property

def b(self) -> float:

return self.\_b

@b.setter

def b(self, value: float):

if value <= 0:

raise ValueError("Длина стороны должна быть положительной")

self.\_b = value

@property

def c(self) -> float:

return self.\_c

@c.setter

def c(self, value: float):

if value <= 0:

raise ValueError("Длина стороны должна быть положительной")

self.\_c = value

def is\_valid(self) -> bool:

"""Проверяет, существует ли треугольник с такими сторонами"""

sides = sorted([self.a, self.b, self.c])

return math.isclose(sides[0] \*\* 2 + sides[1] \*\* 2, sides[2] \*\* 2) and \

(sides[0] + sides[1] > sides[2])

def area(self) -> float:

"""Вычисляет площадь треугольника"""

if not self.is\_valid():

raise ValueError("Треугольник с такими сторонами не существует")

sides = sorted([self.a, self.b, self.c])

return 0.5 \* sides[0] \* sides[1]

def perimeter(self) -> float:

"""Вычисляет периметр треугольника"""

if not self.is\_valid():

raise ValueError("Треугольник с такими сторонами не существует")

return self.a + self.b + self.c

def \_\_str\_\_(self) -> str:

"""Строковое представление объекта"""

return f"Прямоугольный треугольник со сторонами: {self.a}, {self.b}, {self.c}"

def \_\_eq\_\_(self, other) -> bool:

"""Сравнение двух треугольников на равенство (с учетом перестановки сторон)"""

if not isinstance(other, RightTriangle):

return False

return sorted([self.a, self.b, self.c]) == sorted([other.a, other.b, other.c])

def input\_side(prompt: str) -> float:

"""Функция для ввода длины стороны с проверкой"""

while True:

try:

value = float(input(prompt))

if value <= 0:

print("Ошибка: длина стороны должна быть положительной. Попробуйте снова.")

continue

return value

except ValueError:

print("Ошибка: введите числовое значение. Попробуйте снова.")

def main():

"""Основная функция программы"""

print("Программа для работы с прямоугольными треугольниками")

triangles = [] # Для хранения только валидных треугольников

while True:

print("\n" + "=" \* 50)

print("Введите длины сторон треугольника:")

a = input\_side("Сторона a: ")

b = input\_side("Сторона b: ")

c = input\_side("Сторона c: ")

try:

triangle = RightTriangle(a, b, c)

if not triangle.is\_valid():

print("\nОшибка: треугольник с такими сторонами не существует!")

print("Причина:", end=" ")

sides = sorted([a, b, c])

if not math.isclose(sides[0] \*\* 2 + sides[1] \*\* 2, sides[2] \*\* 2):

print("не выполняется теорема Пифагора")

else:

print("не выполняется неравенство треугольника")

# Проверяем, есть ли валидные треугольники для сравнения

if triangles:

print("\nСравнение невозможно - текущий треугольник невалиден")

continue

# Только для валидных треугольников:

triangles.append(triangle)

print("\n" + str(triangle))

print(f"Площадь: {triangle.area():.2f}")

print(f"Периметр: {triangle.perimeter():.2f}")

# Сравнение только если есть с чем сравнивать (кроме текущего)

if len(triangles) > 1:

print("\nСравнение с ранее введенными треугольниками:")

for i, prev\_triangle in enumerate(triangles[:-1], 1):

if triangle == prev\_triangle:

print(f" Совпадает с треугольником #{i}")

else:

print(f" Не совпадает с треугольником #{i}")

except ValueError as e:

print(f"\nОшибка при создании треугольника: {e}")

if triangles:

print("Сравнение невозможно - текущий треугольник не создан")

continue

# Запрос на продолжение

while True:

choice = input("\nХотите продолжить? (да/нет): ").strip().lower()

if choice in ['нет', 'н', 'no', 'n']:

print("\nПрограмма завершена. Всего создано треугольников:", len(triangles))

return

elif choice in ['да', 'д', 'yes', 'y']:

break

else:

print("Неизвестная команда. Пожалуйста, введите 'да' или 'нет'")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Результат работы программы:

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

**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.

Вариант 2

Система Платежи. Клиент имеет Счет в банке и Кредитную Карту (КК). Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет. Администратор может заблокировать КК за превышение кредита.

Код программы:  
from abc import ABC, abstractmethod

from datetime import datetime

from typing import Optional, List

class PaymentSystem(ABC):

"""Абстрактный класс платежной системы"""

@abstractmethod

def process\_payment(self, amount: float) -> bool:

pass

class BankAccount(PaymentSystem):

"""Реализация банковского счета"""

def \_\_init\_\_(self, account\_number: str, balance: float = 0.0):

self.account\_number = account\_number

self.balance = balance

self.is\_active = True

def process\_payment(self, amount: float) -> bool:

if not self.is\_active or self.balance < amount:

return False

self.balance -= amount

return True

class CreditCard(PaymentSystem):

"""Реализация кредитной карты"""

def \_\_init\_\_(self, card\_number: str, credit\_limit: float = 10000.0):

self.card\_number = card\_number

self.credit\_limit = credit\_limit

self.used\_credit = 0.0

def process\_payment(self, amount: float) -> bool:

if self.used\_credit + amount > self.credit\_limit:

return False

self.used\_credit += amount

return True

class Order:

"""Класс для управления заказами"""

def \_\_init\_\_(self, order\_id: str, amount: float):

self.order\_id = order\_id

self.amount = amount

self.is\_paid = False

def pay(self, payment\_system: PaymentSystem) -> bool:

if payment\_system.process\_payment(self.amount):

self.is\_paid = True

return True

return False

class Client:

"""Класс клиента с платежными методами"""

def \_\_init\_\_(self, name: str):

self.name = name

self.accounts: List[BankAccount] = []

self.credit\_cards: List[CreditCard] = []

def add\_account(self, account: BankAccount) -> None:

self.accounts.append(account)

def add\_credit\_card(self, card: CreditCard) -> None:

self.credit\_cards.append(card)

def make\_payment(self, order: Order, payment\_method: PaymentSystem) -> bool:

return order.pay(payment\_method)

Результат работы программы:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**Вывод:** закрепила навыки объектно-ориентированного программирования на языке Python