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

Е. А. Германович

студент группы ПО11

Проверил

А. А. Крощенко

ст. преп. кафедры ИИТ,

22.02.2025 г.

Брест 2025

Цель работы: закрепить базовые знания языка программирования Python при решении практических задач

**Задание 1. Прямоугольник, заданный длинами двух сторон – Предусмотреть**

**возможность определения площади и периметра, а также логические методы,**

**определяющие, является ли прямоугольник квадратом и существует ли такой**

**прямоугольник. Конструктор должен позволять создавать объекты с**

**начальной инициализацией. Переопределить метод \_\_eq\_\_, выполняющий**

**сравнение объектов данного типа.**Выполнение:

**Код программы:**

class Rectangle:

def \_\_init\_\_(self, width: float, height: float):

if width < 0 or height < 0:

raise ValueError("Ошибка: Ширина и высота прямоугольника должны быть положительными числами.")

self.\_width = width

self.\_height = height

@property

def width(self) -> float:

return self.\_width

@property

def height(self) -> float:

return self.\_height

def area(self) -> float:

return self.\_width \* self.\_height

def perimeter(self) -> float:

return 2 \* (self.\_width + self.\_height)

def is\_square(self) -> bool:

return self.\_width == self.\_height

def is\_valid(self) -> bool:

return self.\_width > 0 and self.\_height > 0

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

if isinstance(other, Rectangle):

return self.\_width == other.width and self.\_height == other.height

return False

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

return f"Rectangle(width={self.\_width}, height={self.\_height})"

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

try:

width = float(input("Введите ширину прямоугольника: "))

height = float(input("Введите высоту прямоугольника: "))

rect = Rectangle(width, height)

print(rect)

print(f"Площадь: {rect.area()}")

print(f"Периметр: {rect.perimeter()}")

print(f"Является квадратом: {rect.is\_square()}")

print(f"Существует: {rect.is\_valid()}")

rect2 = Rectangle(4, 5)

print(f"Сравнение с Rectangle(4, 5): {rect == rect2}")

except ValueError as e:

print(e)

**Спецификация ввода:**

Введите ширину прямоугольника: <1-й элемент>  
Введите длину прямоугольника: <2-й элемент>

**Пример:**

Введите ширину прямоугольника: 10

Введите длину прямоугольника: 10

**Спецификация вывода:**

Rectangle<(width, height)>

Площадь: <Площадь>

Периметр: <Периметр>

Является квадратом: <Булево значение>

Существует: <Булево значение>

Сравнение с Rectangle(4, 5): <Булево значение>

**Пример:**

Rectangle(width=10.0, height=10.0)

Площадь: 100.0

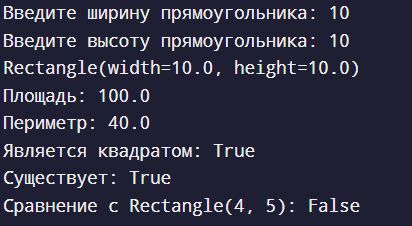
Периметр: 40.0

Является квадратом: True

Существует: True

Сравнение с Rectangle(4, 5): False

**Рисунки с результатами работы программы:**

****

**Задание 2. Система Вступительные экзамены. Абитуриент регистрируется на**

**Факультет, сдает Экзамены. Преподаватель выставляет Оценку. Система**

**подсчитывает средний балл и определяет Абитуриентов, зачисленных в**

**учебное заведение.**Выполнение:

**Код программы:**

class Applicant:

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

self.name = name

self.faculty = None

self.grades = []

def register\_faculty(self, faculty):

self.faculty = faculty

faculty.add\_applicant(self)

def add\_grade(self, grade):

self.grades.append(grade)

def average\_score(self) -> float:

if not self.grades:

return 0

return sum(g.value for g in self.grades) / len(self.grades)

def \_\_str\_\_(self):

return f"Абитуриент: {self.name}, Факультет: {self.faculty.name if self.faculty else 'Не зарегистрирован'}"

class Faculty:

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

self.name = name

self.applicants = []

def add\_applicant(self, applicant):

self.applicants.append(applicant)

def \_\_str\_\_(self):

return f"Факультет: {self.name}"

class Exam:

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

self.name = name

self.grades = []

def add\_grade(self, grade):

self.grades.append(grade)

def \_\_str\_\_(self):

return f"Экзамен: {self.name}"

class Grade:

def \_\_init\_\_(self, value: float, applicant: Applicant, exam: Exam):

self.value = value

self.applicant = applicant

self.exam = exam

exam.add\_grade(self)

applicant.add\_grade(self)

def \_\_str\_\_(self):

return f"Оценка: {self.value}, Абитуриент: {self.applicant.name}, Экзамен: {self.exam.name}"

class Teacher:

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

self.name = name

def assign\_grade(self, value: float, applicant: Applicant, exam: Exam):

Grade(value, applicant, exam)

def \_\_str\_\_(self):

return f"Преподаватель: {self.name}"

class AdmissionSystem:

def \_\_init\_\_(self):

self.applicants = []

self.faculties = []

def add\_applicant(self, applicant):

self.applicants.append(applicant)

def add\_faculty(self, faculty):

self.faculties.append(faculty)

def admit\_applicants(self, threshold: float):

admitted = []

for applicant in self.applicants:

if applicant.average\_score() >= threshold:

admitted.append(applicant)

return admitted

def \_\_str\_\_(self):

return f"Система зачислений: {len(self.applicants)} абитуриентов, {len(self.faculties)} факультетов"

def get\_user\_input():

system = AdmissionSystem()

faculties = []

while True:

faculty\_name = input("Введите название факультета (или 'стоп' для завершения): ")

if faculty\_name.lower() == "стоп":

break

faculty = Faculty(faculty\_name)

system.add\_faculty(faculty)

faculties.append(faculty)

print(f"Факультет '{faculty\_name}' добавлен.")

applicants = []

while True:

applicant\_name = input("Введите имя абитуриента (или 'стоп' для завершения): ")

if applicant\_name.lower() == "стоп":

break

applicant = Applicant(applicant\_name)

print("Доступные факультеты:")

for i, faculty in enumerate(faculties):

print(f"{i + 1}. {faculty.name}")

faculty\_index = int(input("Выберите факультет (номер): ")) - 1

applicant.register\_faculty(faculties[faculty\_index])

system.add\_applicant(applicant)

applicants.append(applicant)

print(f"Абитуриент '{applicant\_name}' зарегистрирован на факультет '{faculties[faculty\_index].name}'.")

exams = []

while True:

exam\_name = input("Введите название экзамена (или 'стоп' для завершения): ")

if exam\_name.lower() == "стоп":

break

exam = Exam(exam\_name)

exams.append(exam)

print(f"Экзамен '{exam\_name}' добавлен.")

teachers = []

while True:

teacher\_name = input("Введите имя преподавателя (или 'стоп' для завершения): ")

if teacher\_name.lower() == "стоп":

break

teacher = Teacher(teacher\_name)

teachers.append(teacher)

print(f"Преподаватель '{teacher\_name}' добавлен.")

while True:

print("Выберите абитуриента:")

for i, applicant in enumerate(applicants):

print(f"{i + 1}. {applicant.name}")

applicant\_index = int(input("Номер абитуриента (или '0' для завершения): ")) - 1

if applicant\_index == -1:

break

print("Выберите экзамен:")

for i, exam in enumerate(exams):

print(f"{i + 1}. {exam.name}")

exam\_index = int(input("Номер экзамена: ")) - 1

print("Выберите преподавателя:")

for i, teacher in enumerate(teachers):

print(f"{i + 1}. {teacher.name}")

teacher\_index = int(input("Номер преподавателя: ")) - 1

grade\_value = float(input("Введите оценку: "))

teachers[teacher\_index].assign\_grade(grade\_value, applicants[applicant\_index], exams[exam\_index])

print(f"Оценка {grade\_value} выставлена абитуриенту {applicants[applicant\_index].name} за экзамен {exams[exam\_index].name}.")

return system

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

system = get\_user\_input()

threshold = float(input("Введите порог среднего балла для зачисления: "))

admitted = system.admit\_applicants(threshold)

print("\nСписок зачисленных:")

for a in admitted:

print(f"{a.name} (Средний балл: {a.average\_score()})")  
  
**Спецификация ввода:**

Введите название факультета (или 'стоп' для завершения):<Название факультета>  
 Введите имя абитуриента (или 'стоп' для завершения): <Имя>  
 Выберите факультет (номер): <Номер созданного факультета>

Введите название экзамена (или 'стоп' для завершения): <Название предмета>  
 Введите имя преподавателя (или 'стоп' для завершения): <Имя преподавателя>

Номер абитуриента <Номер абитуриента>

Номер экзамена <Номер экзамена>

Номер преподавателя <Номер преподавателя>

Введите оценку: <Оценка>

Номер абитуриента <Номер абитуриента>

Введите порог среднего балла для зачисления:<Оценка порога>

**Пример:**

Введите название факультета (или 'стоп' для завершения): ЭИС

Факультет 'ЭИС' добавлен.

Введите название факультета (или 'стоп' для завершения): стоп

Введите имя абитуриента (или 'стоп' для завершения): Германович Егор

Доступные факультеты:

1. ЭИС

Выберите факультет (номер): 1

Абитуриент 'Германович Егор' зарегистрирован на факультет 'ЭИС'.

Введите имя абитуриента (или 'стоп' для завершения): стоп

Введите название экзамена (или 'стоп' для завершения): СПП

Экзамен 'СПП' добавлен.

Введите название экзамена (или 'стоп' для завершения): стоп

Введите имя преподавателя (или 'стоп' для завершения): Крощенко

Преподаватель 'Крощенко' добавлен.

Введите имя преподавателя (или 'стоп' для завершения): стоп

Выберите абитуриента:

1. Германович Егор

Номер абитуриента (или '0' для завершения): 1

Выберите экзамен:

1. СПП

Номер экзамена: 1

Выберите преподавателя:

1. Крощенко

Номер преподавателя: 1

Введите оценку: 9

Оценка 9.0 выставлена абитуриенту Германович Егор за экзамен СПП.

Выберите абитуриента:

1. Германович Егор

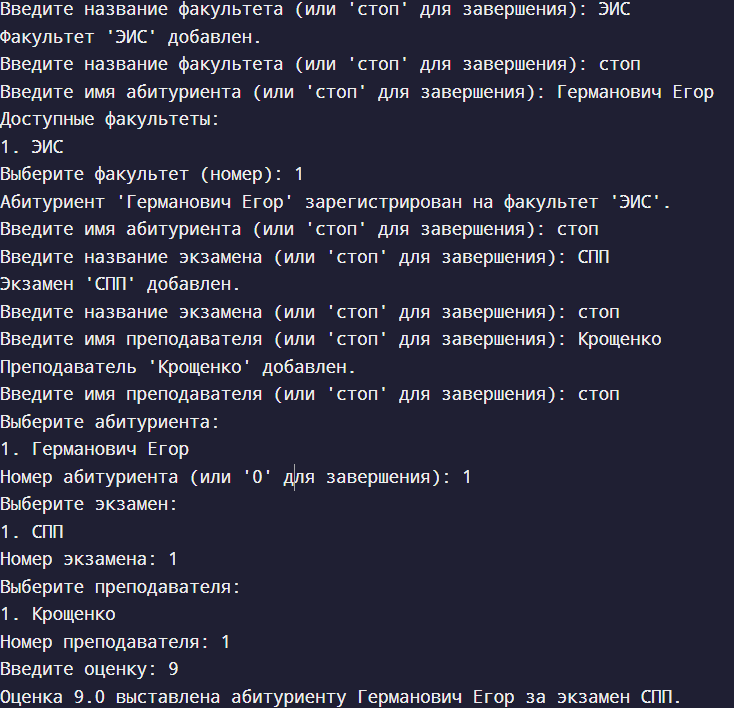
Номер абитуриента (или '0' для завершения): 0

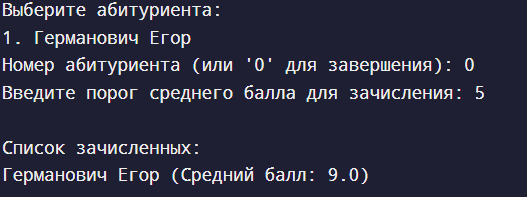
Введите порог среднего балла для зачисления: 5

Список зачисленных:

Германович Егор (Средний балл: 9.0)

**Рисунки с результатами работы программы:**

****

****

**Вывод:** закрепил базовые знания Python при решении практических задач