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
П. А. Захарчук
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
25.04.2025 г.

Брест 2025

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

Задание 1. 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.

Выполнение:

Код программы:

```
import math

class IsoscelesTriangle:
    def __init__(self, a, b, c):
        if not (a == b or a == c or b == c):
            raise ValueError("Для равнобедренного треугольника две стороны должны быть равны")
        self.a = a
        self.b = b
        self.c = c

    def exists(self):
        return self.a + self.b > self.c and self.a + self.c > self.b and self.b + self.c > self.a

    def perimeter(self):
        if self.exists():
            return self.a + self.b + self.c
        else:
            print("Такого треугольника не существует")
            return None

    def area(self):
        if self.exists():
            p = self.perimeter() / 2
            return math.sqrt(p * (p - self.a) * (p - self.b) * (p - self.c))
        else:
            print("Такого треугольника не существует")
            return None

    def __eq__(self, other):
        if isinstance(other, IsoscelesTriangle):
            return sorted([self.a, self.b, self.c]) == sorted([other.a, other.b, other.c])
        return False

    def __str__(self):
        return f"Равнобедренный треугольник со сторонами: {self.a}, {self.b}, {self.c}"



# Ввод сторон треугольника
side1 = float(input("Введите первую сторону треугольника: "))
side2 = float(input("Введите вторую сторону треугольника: "))
side3 = float(input("Введите третью сторону треугольника: "))

# Проверка на равнобедренность
try:
    triangle = IsoscelesTriangle(side1, side2, side3)
    print(triangle)
    if triangle.exists():
        print("Периметр:", triangle.perimeter())
        print("Площадь:", triangle.area())
```

```
except ValueError as e:
    print(e)
```

Спецификация ввода:
Введите первую сторону треугольника: <1-й элемент>
Введите вторую сторону треугольника: <2-й элемент>
Введите третью сторону треугольника: <3-й элемент>

Пример:
Введите первую сторону треугольника: 5
Введите вторую сторону треугольника: 8
Введите третью сторону треугольника: 8

Спецификация вывода:
Равнобедренный треугольник со сторонами: <1-й элемент>...<n-й элемент>
Периметр: <Периметр>
Площадь: <Площадь>

Пример:
Равнобедренный треугольник со сторонами: 5.0, 8.0, 8.0
Периметр: 21.0
Площадь: 18.99835519196333

Рисунки с результатами работы программы:

```
/home/twinkle/PycharmProjects/pythonProject/venv/bin/python /home/twinkle/PycharmProjects/pythonProject/lab2.py
Введите первую сторону треугольника: 5
Введите вторую сторону треугольника: 8
Введите третью сторону треугольника: 8
Равнобедренный треугольник со сторонами: 5.0, 8.0, 8.0
Периметр: 21.0
Площадь: 18.99835519196333
```

Задание 2. Система Аэрофлот. Администратор формирует летную Бригаду (пилоты, штурман, радист, стюардессы) на Рейс. Каждый Рейс выполняется Самолетом с определенной вместимостью и дальностью полета. Рейс может быть отменен из-за погодных условий в Аэропорту отлета или назначения. Аэропорт назначения может быть изменен в полете из-за технических неисправностей, о которых сообщил командир.
Выполнение:
Код программы:

```
from abc import ABC, abstractmethod

# Абстрактный класс для людей
class Person(ABC):
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"{self.name}, возраст: {self.age}"

# Абстрактный класс для членов экипажа
```

```python
class CrewMember(Person):
    def __init__(self, name, age, experience):
        super().__init__(name, age)
        self.experience = experience  # опыт в годах

    @abstractmethod
    def do_job(self):
        pass  # каждая роль выполняет свою работу

# Конкретные роли членов экипажа
class Pilot(CrewMember):
    def do_job(self):
        return f"{self.name} управляет самолетом"

    def __str__(self):
        return f"Пилот {self.name}, опыт: {self.experience} лет"

class Navigator(CrewMember):
    def do_job(self):
        return f"{self.name} прокладывает маршрут"

    def __str__(self):
        return f"Штурман {self.name}, опыт: {self.experience} лет"

class RadioOperator(CrewMember):
    def do_job(self):
        return f"{self.name} поддерживает связь"

    def __str__(self):
        return f"Радист {self.name}, опыт: {self.experience} лет"

class FlightAttendant(CrewMember):
    def do_job(self):
        return f"{self.name} обслуживает пассажиров"

    def __str__(self):
        return f"Стюардесса {self.name}, опыт: {self.experience} лет"

# Класс самолета
class Airplane:
    def __init__(self, model, capacity, max_range):
        self.model = model
        self.capacity = capacity  # вместимость (пассажиры)
        self.max_range = max_range  # дальность полета в км
        self.has_technical_issue = False  # флаг неисправности

    def report_issue(self):
        self.has_technical_issue = True
        print(f"Самолет {self.model} сообщил о технической неисправности")

    def __str__(self):
        return f"Самолет {self.model}, вместимость: {self.capacity}, дальность: {self.max_range} км"

# Класс аэропорта
class Airport:
    def __init__(self, name, is_good_weather=True):
        self.name = name
        self.is_good_weather = is_good_weather  # хорошая ли погода

    def set_weather(self, is_good):
        self.is_good_weather = is_good
        print(f"Погода в аэропорту {self.name}: {'хорошая' if is_good else 'плохая'}")
```

```
    def __str__(self):
        return f"Аэропорт {self.name}"

# Класс летной бригады (агрегация членов экипажа)
class FlightCrew:
    def __init__(self):
        self.members = []  # список членов экипажа

    def add_member(self, member):
        self.members.append(member)
        print(f"Добавлен в бригаду: {member}")

    def show_crew(self):
        print("Состав бригады:")
        for member in self.members:
            print(f"- {member}")

    def do_jobs(self):
        print("Бригада выполняет свои обязанности:")
        for member in self.members:
            print(f"- {member.do_job()}")

# Класс рейса
class Flight:
    def __init__(self, flight_number, airplane, crew, departure_airport, destination_airport):
        self.flight_number = flight_number
        self.airplane = airplane  # ассоциация с самолетом
        self.crew = crew  # ассоциация с бригадой
        self.departure_airport = departure_airport  # ассоциация с аэропортом
        self.destination_airport = destination_airport  # ассоциация с аэропортом
        self.is_canceled = False

    def check_flight_status(self):
        if self.is_canceled:
            print(f"Рейс {self.flight_number} отменен")
            return False
        if not self.departure_airport.is_good_weather:
            print(f"Рейс {self.flight_number} не может вылететь: плохая погода в
{self.departure_airport}")
            return False
        if not self.destination_airport.is_good_weather:
            print(f"Рейс {self.flight_number} не может приземлиться: плохая погода в
{self.destination_airport}")
            return False
        if self.airplane.has_technical_issue:
            print(f"Рейс {self.flight_number} имеет технические проблемы")
            return False
        return True

    def cancel_flight(self):
        self.is_canceled = True
        print(f"Рейс {self.flight_number} отменен")

    def change_destination(self, new_airport):
        if self.airplane.has_technical_issue:
            self.destination_airport = new_airport
            print(f"Рейс {self.flight_number} перенаправлен в {new_airport}")
        else:
            print("Нет причин менять аэропорт назначения")

    def __str__(self):
```

```
        return (f"Рейс {self.flight_number}: {self.departure_airport} -> {self.destination_airport}, "
                f"самолет: {self.airplane.model}, статус: {'отменен' if self.is_canceled else 'активен'}")

# Класс администратора
class Administrator:
    def __init__(self, name):
        self.name = name

    def create_crew(self):
        return FlightCrew()  # создаем новую бригаду

    def create_flight(self, flight_number, airplane, crew, departure_airport, destination_airport):
        flight = Flight(flight_number, airplane, crew, departure_airport, destination_airport)
        print(f"Администратор {self.name} создал рейс: {flight}")
        return flight

# Функция для безопасного ввода целых чисел
def get_int_input(prompt):
    while True:
        try:
            return int(input(prompt))
        except ValueError:
            print("Пожалуйста, введите целое число.")

# Функция для безопасного ввода чисел с плавающей запятой
def get_float_input(prompt):
    while True:
        try:
            return float(input(prompt))
        except ValueError:
            print("Пожалуйста, введите число.")

# Основная функция с пользовательским вводом
def main():
    # Ввод данных администратора
    admin_name = input("Введите имя администратора: ")
    admin = Administrator(admin_name)

    # Ввод данных аэропортов
    airports = []
    num_airports = get_int_input("Сколько аэропортов вы хотите создать (минимум 2)? ")
    while num_airports < 2:
        print("Требуется минимум 2 аэропорта.")
        num_airports = get_int_input("Сколько аэропортов вы хотите создать (минимум 2)? ")

    for i in range(num_airports):
        name = input(f"Введите название аэропорта {i + 1}: ")
        weather_input = input("Хорошая погода в этом аэропорту? (да/нет): ").lower()
        is_good_weather = weather_input == "да"
        airports.append(Airport(name, is_good_weather))

    # Ввод данных самолета
    plane_model = input("Введите модель самолета: ")
    plane_capacity = get_int_input("Введите вместимость самолета (пассажиры): ")
    plane_range = get_int_input("Введите дальность полета самолета (км): ")
    plane = Airplane(plane_model, plane_capacity, plane_range)

    # Создание бригады
    crew = admin.create_crew()
    num_crew = get_int_input("Сколько членов экипажа вы хотите добавить? ")
    for i in range(num_crew):
        print(f"\nДобавление члена экипажа {i + 1}")
```

```
        name = input("Введите имя: ")
        age = get_int_input("Введите возраст: ")
        experience = get_float_input("Введите опыт (в годах): ")
        print("Выберите роль:")
        print("1. Пилот")
        print("2. Штурман")
        print("3. Радист")
        print("4. Стюардесса")
        role = get_int_input("Введите номер роли (1-4): ")
        while role not in [1, 2, 3, 4]:
            print("Неверный выбор роли.")
            role = get_int_input("Введите номер роли (1-4): ")

        if role == 1:
            crew.add_member(Pilot(name, age, experience))
        elif role == 2:
            crew.add_member(Navigator(name, age, experience))
        elif role == 3:
            crew.add_member(RadioOperator(name, age, experience))
        elif role == 4:
            crew.add_member(FlightAttendant(name, age, experience))

    # Показываем бригаду
    crew.show_crew()

    # Ввод данных рейса
    flight_number = input("Введите номер рейса: ")
    print("Выберите аэропорт вылета:")
    for i, airport in enumerate(airports):
        print(f"{i + 1}. {airport}")
    dep_idx = get_int_input("Введите номер аэропорта вылета: ") - 1
    while dep_idx < 0 or dep_idx >= len(airports):
        print("Неверный выбор аэропорта.")
        dep_idx = get_int_input("Введите номер аэропорта вылета: ") - 1

    print("Выберите аэропорт назначения:")
    for i, airport in enumerate(airports):
        print(f"{i + 1}. {airport}")
    dest_idx = get_int_input("Введите номер аэропорта назначения: ") - 1
    while dest_idx < 0 or dest_idx >= len(airports) or dest_idx == dep_idx:
        print("Неверный выбор аэропорта. Аэропорт назначения должен отличаться от аэропорта
вылета.")
        dest_idx = get_int_input("Введите номер аэропорта назначения: ") - 1

    flight = admin.create_flight(flight_number, plane, crew, airports[dep_idx], airports[dest_idx])

    # Проверяем статус рейса
    print("\nПроверка статуса рейса:")
    if flight.check_flight_status():
        print("Рейс готов к выполнению!")
        crew.do_jobs()

    # Симуляция проблем
    simulate_issues = input("\nХотите симулировать проблемы (погода/неисправность)? (да/нет):
").lower()
    if simulate_issues == "да":
        # Симуляция плохой погоды
        print("\nВыберите аэропорт для изменения погоды:")
        for i, airport in enumerate(airports):
            print(f"{i + 1}. {airport}")
        weather_idx = get_int_input("Введите номер аэропорта: ") - 1
        while weather_idx < 0 or weather_idx >= len(airports):
            print("Неверный выбор аэропорта.")
```

```
            weather_idx = get_int_input("Введите номер аэропорта: ") - 1
        airports[weather_idx].set_weather(False)

        if not flight.check_flight_status():
            flight.cancel_flight()

        # Симуляция технической неисправности
        print("\nСимуляция технической неисправности:")
        airports[weather_idx].set_weather(True)  # возвращаем хорошую погоду
        plane.report_issue()
        if not flight.check_flight_status():
            print("Выберите новый аэропорт назначения:")
            for i, airport in enumerate(airports):
                print(f"{i + 1}. {airport}")
            new_dest_idx = get_int_input("Введите номер аэропорта: ") - 1
            while new_dest_idx < 0 or new_dest_idx >= len(airports) or new_dest_idx == dep_idx:
                print("Неверный выбор аэропорта.")
                new_dest_idx = get_int_input("Введите номер аэропорта: ") - 1
            flight.change_destination(airports[new_dest_idx])

    # Проверяем финальный статус
    print(f"\nИтоговый статус: {flight}")


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

Спецификация ввода:

Введите имя администратора: <Имя>
Сколько аэропортов вы хотите создать (минимум 2)? <Количество>
Введите название аэропорта 1: <Название аэропорта>
Хорошая погода в этом аэропорту? (да/нет): <Выбор ввод строки>
Введите название аэропорта 2: <Название аэропорта>
Хорошая погода в этом аэропорту? (да/нет): <Выбор ввод строки>
Введите модель самолета: <Модель самолета>
Введите вместимость самолета (пассажиры): <Кол-во пассажиров>
Введите дальность полета самолета (км): <Дальность полета>
Сколько членов экипажа вы хотите добавить? <Количество>
Введите имя: <Имя>
Введите возраст: <Возраст>
Введите опыт (в годах): <Опыт>
Введите номер роли (1-4): <Выбор ввод номера>
Введите номер рейса: <Номер>
Введите номер аэропорта вылета: <Выбор ввод номера>
Введите номер аэропорта назначения: <Выбор ввод номера>
Хотите симулировать проблемы (погода/неисправность)? (да/нет): <Выбор ввод строки>
Выберите аэропорт для изменения погоды:
Введите номер аэропорта: <Выбор ввод номера>
Симуляция технической неисправности:
Выберите новый аэропорт назначения:
Введите номер аэропорта: <Выбор ввод номера>

Пример:

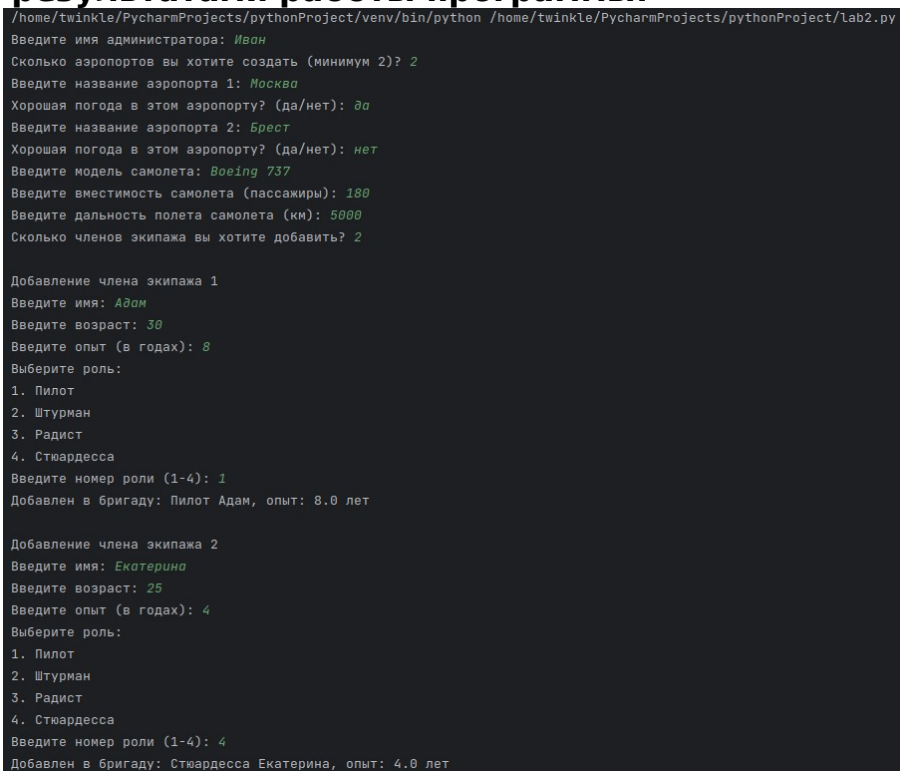
Введите имя администратора: Иван
Сколько аэропортов вы хотите создать (минимум 2)? 2
Введите название аэропорта 1: Москва
Хорошая погода в этом аэропорту? (да/нет): да
Введите название аэропорта 2: Брест
Хорошая погода в этом аэропорту? (да/нет): нет
Введите модель самолета: Boeing 737
Введите вместимость самолета (пассажиры): 180
Введите дальность полета самолета (км): 5000
Сколько членов экипажа вы хотите добавить? 2

Добавление члена экипажа 1
Введите имя: Адам
Введите возраст: 30
Введите опыт (в годах): 8
Выберите роль:
1. Пилот
2. Штурман
3. Радист
4. Стюардесса
Введите номер роли (1-4): 1
Добавлен в бригаду: Пилот Адам, опыт: 8.0 лет

Введите номер рейса: 23
Введите номер аэропорта вылета: 1
Введите номер аэропорта назначения: 2

Хотите симулировать проблемы (погода/неисправность)? (да/нет): да
Введите номер аэропорта: 1
Симуляция технической неисправности:
Введите номер аэропорта: 2

Рисунки с результатами работы программы:



```
Состав бригады:
- Пилот Адам, опыт: 8.0 лет
- Стюардесса Екатерина, опыт: 4.0 лет
Введите номер рейса: 23
Выберите аэропорт вылета:
1. Аэропорт Москва
2. Аэропорт Брест
Введите номер аэропорта вылета: 1
Выберите аэропорт назначения:
1. Аэропорт Москва
2. Аэропорт Брест
Введите номер аэропорта назначения: 2
Администратор Иван создал рейс: Рейс 23: Аэропорт Москва -> Аэропорт Брест, самолет: Boeing 737, статус: активен

Проверка статуса рейса:
Рейс 23 не может приземлиться: плохая погода в Аэропорт Брест

Хотите симулировать проблемы (погода/неисправность)? (да/нет): да

Выберите аэропорт для изменения погоды:
1. Аэропорт Москва
2. Аэропорт Брест
Введите номер аэропорта: 1
Погода в аэропорту Москва: плохая
Рейс 23 не может вылететь: плохая погода в Аэропорт Москва
Рейс 23 отменен

Симуляция технической неисправности:
Погода в аэропорту Москва: хорошая
Самолет Boeing 737 сообщил о технической неисправности
Рейс 23 отменен
Выберите новый аэропорт назначения:
1. Аэропорт Москва
2. Аэропорт Брест
Введите номер аэропорта: 2
Рейс 23 перенаправлен в Аэропорт Брест

Итоговый статус: Рейс 23: Аэропорт Москва -> Аэропорт Брест, самолет: Boeing 737, статус: отменен
```

Вывод: закрепил базовые знания Python при решении практических задач.