Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Наследование и полиморфизм в языке Python»

ОТЧЕТ
по лабораторной работе №4.3
дисциплины
«Основы программной инженерии»

Выполнил:
Мизин Глеб Егорович
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Цель работы: приобретение навыков по созданию иерархии классов при написании программ с помощью языка программирования Python версии 3.x.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class Rational:
    def __init__(self, a=0, b=1):
        a = int(a)
        b = int(b)
        if b == 0:
            raise ValueError()
        self.__numerator = abs(a)
        self.__denominator = abs(b)
        self.__reduce()
        # Сокращение дроби

    def __reduce(self):
        # Функция для нахождения наибольшего общего делителя
        def gcd(a, b):
            if a == 0:
                return b
            elif b == 0:
                return a
            elif a >= b:
                return gcd(a % b, b)
            else:
                return gcd(a, b % a)

        c = gcd(self.__numerator, self.__denominator)
        self.__numerator //= c
        self.__denominator //= c
```

Рисунок 1 – Пример 1

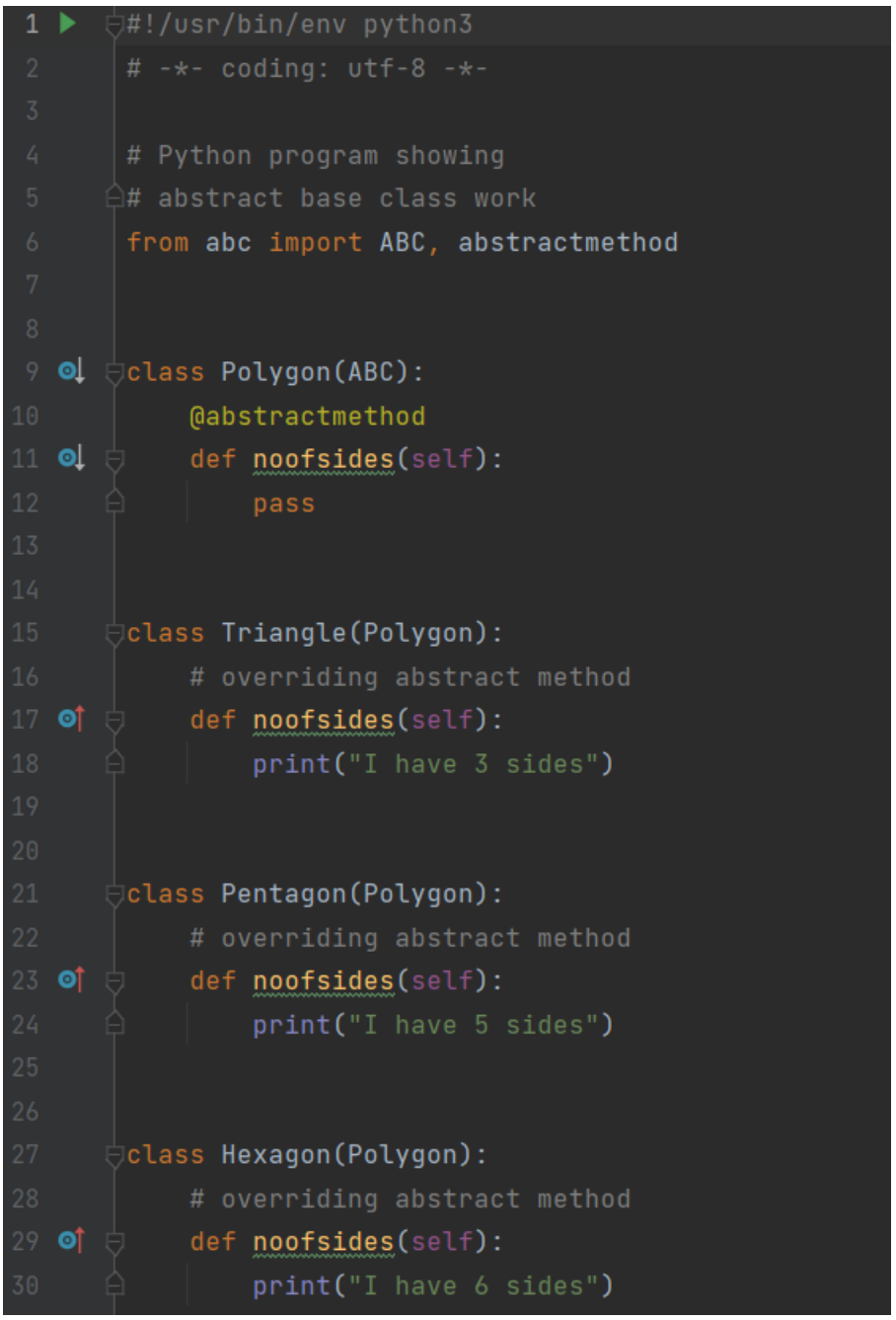

Рисунок 2 – Пример 2

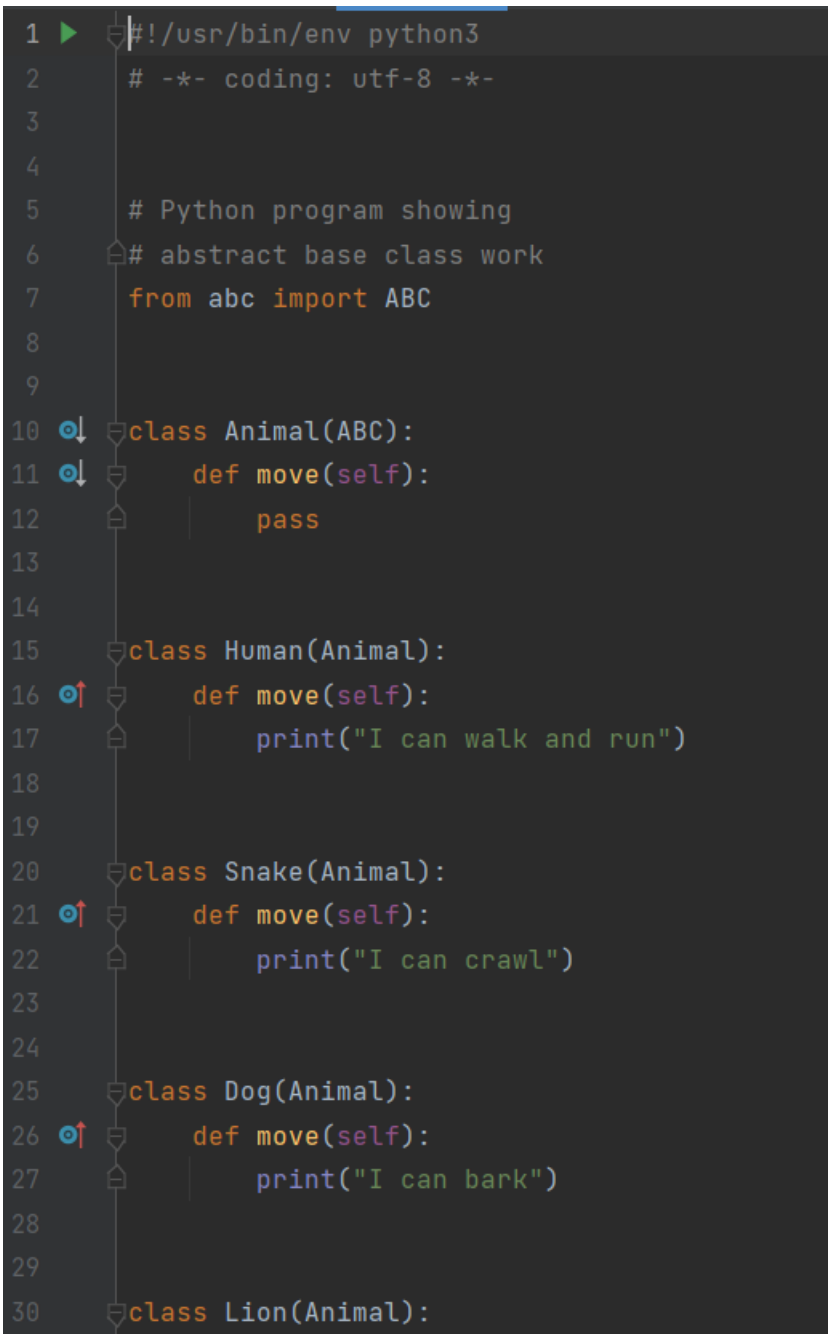

Рисунок 3 – Пример 3

```python
class Unit:
    def __init__(self, id, team):
        self.id = id
        self.team = team


class Soldier(Unit):
    def follow(self, hero):
        print(f"Солдат {self.id} идет за героем {hero.id}")


class Hero(Unit):
    def level_up(self):
        print(f"У героя {self.id} уровень повышен!")


if __name__ == '__main__':
    team1 = []  # список для хранения солдат первой команды
    team2 = []  # список для хранения солдат второй команды

    hero1 = Hero(1, 1)  # создаем героя первой команды с номером 1
    hero2 = Hero(2, 2)  # создаем героя второй команды с номером 2

    # генерируем 10 солдат
    for i in range(10):
        id = i + 1
        # определяем случайным образом принадлежность солдата команде
        team = random.choice([1, 2])
        if team == 1:
```

Рисунок 4 – Общее задание

```python
import math


class Triangle:
    # конструктор класса, принимающий значения сторон a, b, c
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    # метод для изменения значений сторон
    def set_sides(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    # метод для вычисления углов
    def get_angles(self):
        alpha = math.degrees(math.acos((self.b ** 2 + self.c ** 2 - self.a ** 2)
                                       / (2 * self.b * self.c)))
        beta = math.degrees(math.acos((self.a ** 2 + self.c ** 2 - self.b ** 2)
                                      / (2 * self.a * self.c)))
        gamma = math.degrees(math.acos((self.a ** 2 + self.b ** 2 - self.c ** 2)
                                       / (2 * self.a * self.b)))
        return alpha, beta, gamma

    # метод для вычисления периметра
    def get_perimeter(self):
        return self.a + self.b + self.c
```

Рисунок 5 – Задание 1

```python
class Function(ABC):
    @abstractmethod
    def calculate(self, x: float) -> float:
        pass

    @abstractmethod
    def print_result(self):
        pass


# Определим класс Ellipse, который будет реализовывать вычисление эллипса
class Ellipse(Function):
    def __init__(self, a: float, b: float):
        self.a = a
        self.b = b
        self.result = 0.0

    def calculate(self, x: float) -> float:
        # Формула для вычисления эллипса
        self.result = sqrt(self.b * self.b * (1 - (x * x) / (self.a * self.a)))
        return self.result

    def print_result(self):
        # Вывод результата вычисления функции
        print(f"Результат вычисления функции (эллипс): {self.result}")


# Определим класс Hyperbola, который будет реализовывать вычисление гиперболы
class Hyperbola(Function):
    def __init__(self, a: float, b: float):
```

Рисунок 6 – Задание №2

Контрольные вопросы

1. Что такое наследование как оно реализовано в языке Python?

В организации наследования участвуют как минимум два класса: класс родитель и класс потомок. При этом возможно множественное наследование, в этом случае у класса потомка может быть несколько родителей. Не все языки программирования поддерживают множественное наследование, но в Python можно его использовать. По умолчанию все классы в Python являются наследниками от object, явно этот факт указывать не нужно. Синтаксически создание класса с указанием его родителя выглядит так:

class имя_класса(имя_родителя1, [имя_родителя2,…, имя_родителя_n])

2. Что такое полиморфизм и как он реализован в языке Python?

Полиморфизм - это возможность объектов с одинаковой сигнатурой методов вызывать разные реализации этого метода в зависимости от текущего типа объекта. В Python полиморфизм реализуется через вызов методов класса объекта без необходимости указывать явно тип объекта.

3. Что такое "утиная" типизация в языке программирования Python?

"Утиная" типизация - это стиль программирования, при котором проверка на соответствие типу объекта происходит во время выполнения, а не на этапе компиляции. В Python все объекты имеют общий тип object, и проверка соответствия типу может быть выполнена с помощью ключевого слова isinstance.

4. Каково назназначение модуля abc языка программирования Python?

По умолчанию Python не предоставляет абстрактных классов. Python поставляется с модулем, который обеспечивает основу для определения абстрактных базовых классов (ABC), и имя этого модуля - ABC. ABC работает, декорируя методы базового класса как абстрактные, а затем регистрируя конкретные классы как реализации абстрактной базы. Метод становится абстрактным, если он украшен ключевым словом @abstractmethod.

5. Как сделать некоторый метод класса абстрактным?

Для того чтобы сделать метод класса абстрактным, нужно создать абстрактный метод в базовом классе с помощью декоратора @abstractmethod. Этот метод не должен иметь реализации в базовом классе, и должен быть переопределен в каждом наследнике.

6. Как сделать некоторое свойство класса абстрактным?

Для того чтобы сделать свойство класса абстрактным, нужно создать абстрактное свойство в базовом классе с помощью декоратора @abstractmethod. Это свойство не должно иметь реализации в базовом классе, и должно быть переопределено в каждом наследнике.

7. Каково назначение функции isinstance?

Функция isinstance используется для проверки соответствия типа объекта указанному классу или его наследнику. Она принимает два аргумента: объект, тип которого нужно проверить, и класс или кортеж классов, с которым нужно сравнить тип объекта. Если объект является экземпляром указанного класса или его наследника, то функция возвращает True, в противном случае - False.