МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №3

Наследование и полиморфизм в языке Python

По дисциплине «Объектно-ориентированное программирование»

Выполнил студент группы ИВТ-б-о-20-1

Галяс Д. И. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2022

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

Ссылка на репозиторий: https://github.com/DIMITRY-GALYAS1/Laba-4.3.git

Ход работы:

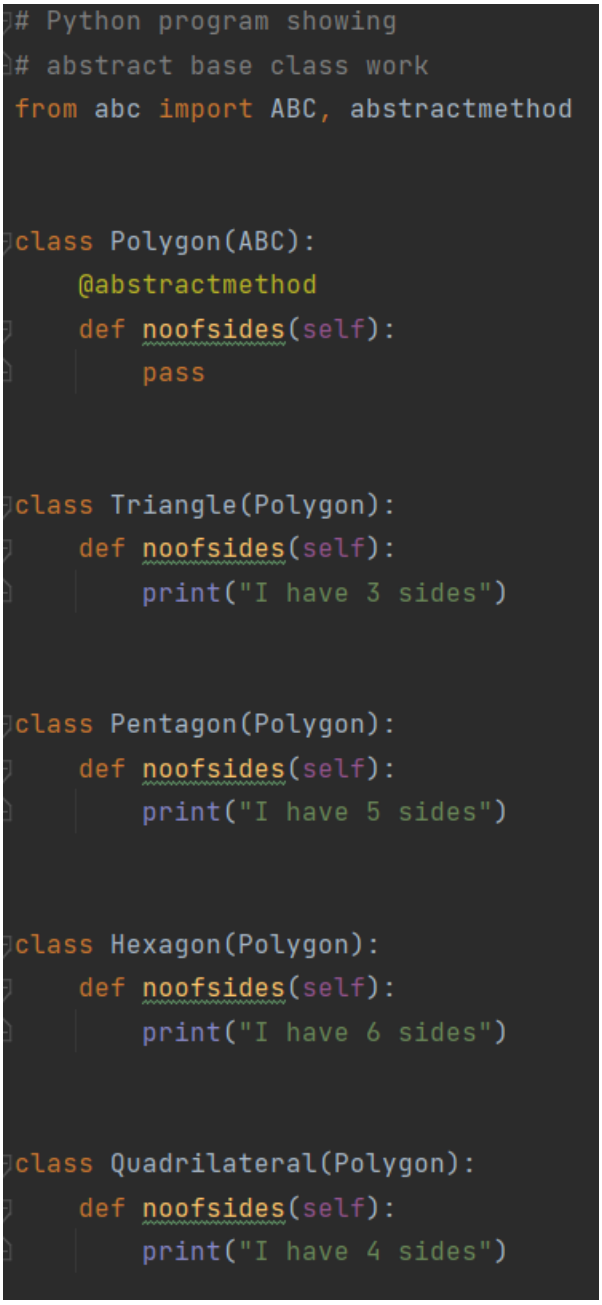
1. Создал новый репозиторий на github, после клонировал его и создал в папке репозитория новый проект PyCharm.

2. Выполнил первый пример.

```
#!/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()

    def __reduce(self):

        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

    @property
    def numerator(self):
```

Рисунок 1. Пример 1

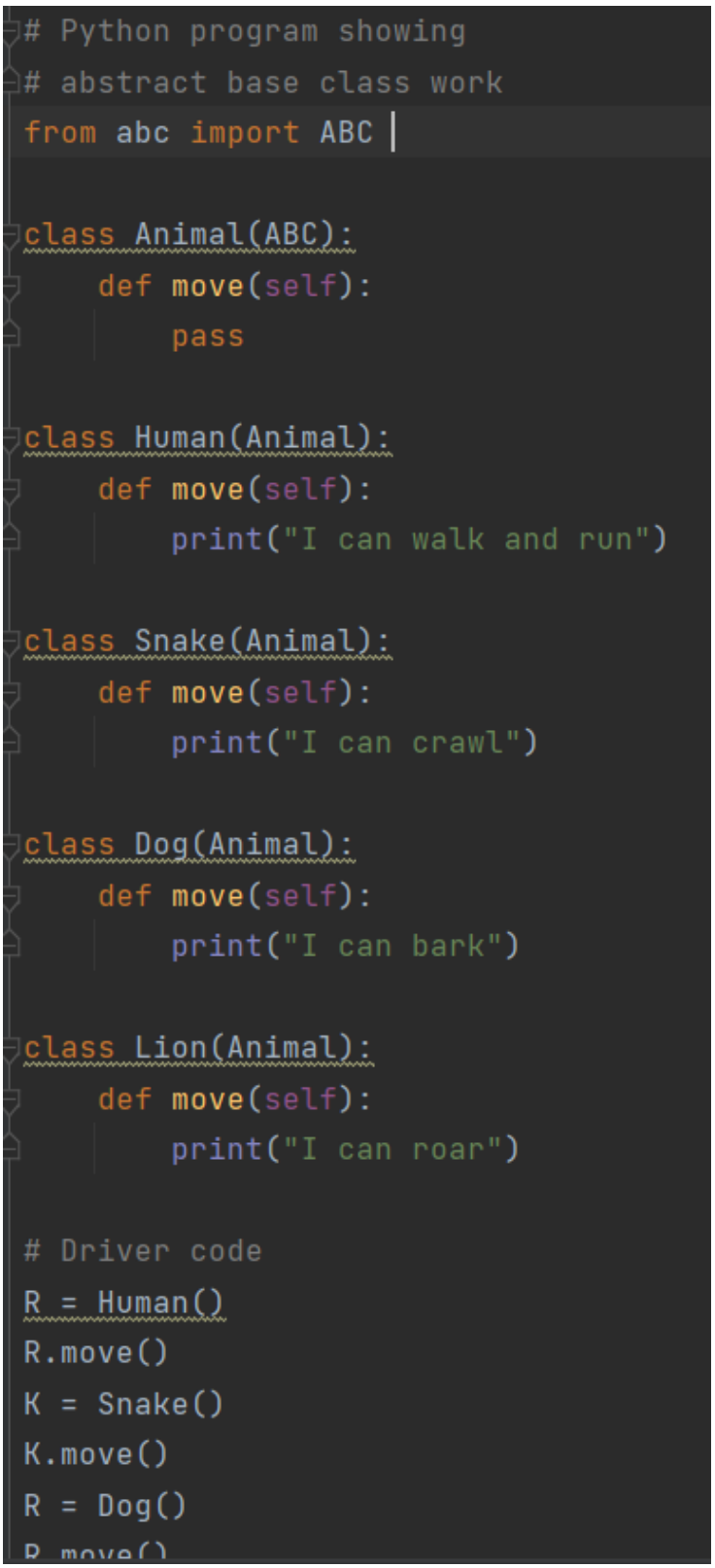
3. Выполнил второй пример.



```
# Python program showing
# abstract base class work
from abc import ABC, abstractmethod


class Polygon(ABC):
    @abstractmethod
    def noofsides(self):
        pass


class Triangle(Polygon):
    def noofsides(self):
        print("I have 3 sides")


class Pentagon(Polygon):
    def noofsides(self):
        print("I have 5 sides")


class Hexagon(Polygon):
    def noofsides(self):
        print("I have 6 sides")


class Quadrilateral(Polygon):
    def noofsides(self):
        print("I have 4 sides")
```

Рисунок 2. Пример 2

4. Сделал третий пример.



```python
# Python program showing
# abstract base class work
from abc import ABC

class Animal(ABC):
    def move(self):
        pass

class Human(Animal):
    def move(self):
        print("I can walk and run")

class Snake(Animal):
    def move(self):
        print("I can crawl")

class Dog(Animal):
    def move(self):
        print("I can bark")

class Lion(Animal):
    def move(self):
        print("I can roar")

# Driver code
R = Human()
R.move()
K = Snake()
K.move()
R = Dog()
R.move()
```

Рисунок 3. Пример 3

5. Затем выполнил общее задание.

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

"""
В некой игре-стратегии есть солдаты и герои. У всех есть свойство, содержащее уникальный
номер объекта, и свойство, в котором хранится принадлежность команде. У солдат есть
метод "иду за героем", который в качестве аргумента принимает объект типа "герой". У
героев есть метод увеличения собственного уровня.
В основной ветке программы создается по одному герою для каждой команды. В цикле
генерируются объекты-солдаты. Их принадлежность команде определяется случайно.
Солдаты разных команд добавляются в разные списки.
Измеряется длина списков солдат противоборствующих команд и выводится на экран. У
героя, принадлежащего команде с более длинным списком, увеличивается уровень.
Отправьте одного из солдат первого героя следовать за ним. Выведите на экран
идентификационные номера этих двух юнитов.
"""

import random


class Soldiers:
    id = 1

    def __init__(self):
        self.id = Soldiers.id
        Soldiers.id += 1

    def go_hero(self, hero):
        print(f'солдат с id {self.id} следует за героем {hero.id}')
```

Рисунок 4. Общее задание

6. Далее приступил к выполнению первого индивидуального задания.

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

"""
Создать класс Pair (пара чисел); определить методы изменения полей
и сравнения пар: пара p1 больше пары p2, если (first.p1 > first.p2)
или (first.p1 = first.p2) и (second.p1 > second.p2). Определить
класс-наследник Fraction с полями: целая часть числа и дробная часть
числа. Определить полный набор методов сравнения.
"""


class Pair:

    def __init__(self, first, second):
        self.first = first
        self.second = second

    def display(self):
        print(f'Большая пара чисел {self.first} {self.second}')

    def comparison(self, other):
        if isinstance(other, Pair):
            if self.first > other.first:
                return Pair(self.first, self.second)
            elif self.first == other.first and self.second > other.second:
                return Pair(self.first, self.second)
            elif self.first < other.first:
                return Pair(other.first, other.second)
            elif self.first == other.first and self.second < other.second:
```

Рисунок 5. Первое индивидуальное задание

7. Затем выполнил второе индивидуальное задание.

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

"""
Создать абстрактный базовый класс Number с абстрактными
методами – арифметическими операциями. Создать производные
классы Integer (целое) и Real (действительное).
"""

from abc import ABC, abstractmethod


class Number(ABC):

    @abstractmethod
    def add(self, a, b):
        pass

    @abstractmethod
    def sub(self, a, b):
        pass

    @abstractmethod
    def mul(self, a, b):
        pass

    @abstractmethod
    def div(self, a, b):
        pass
```

Рисунок 6. Второе индивидуальное задание

Контрольные вопросы:

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

Наследование — это возможность расширения (наследования) ранее написанного программного кода класса с целью дополнения, усовершенствования или привязки под новые требования.

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 способность выполнять действие над объектом независимо от его типа. Это обычно реализуется путем создания базового класса и наличия двух или более подклассов, которые все реализуют методы с одинаковой сигнатурой.

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

Эта концепция адаптирована из следующего абдуктивного умозаключения:

Если что-то выглядит как утка, плавает как утка и крякает как утка, это наверняка и есть утка.

Концепция утиной типизация в основном принята в языках программирования, поддерживающих динамическую типизацию, таких

как Python и JavaScript. Общей особенностью этих языков является возможность объявления переменных без указания их типа.

При использовании пользовательских типов для определённых целей, реализация связанных функций важнее, чем точные типы данных.

Утиная типизация подчёркивает реализацию связанных выполняемых функций, а конкретные типы данных менее важны

4. Каково назначение модуля abc языка программирования Python?

Начиная с версии языка 2.6 в стандартную библиотеку включается модуль abc, добавляющий в язык абстрактные базовые классы.

Абстрактные базовые классы позволяют определить класс, указав при этом, какие методы или свойства обязательно переопределить в классах-наследниках.

5. Как сделать некоторый метод класса абстрактным?

Перед методом класса необходимо добавить декоратор модуля abc: @abstractmethod.

6. Как сделать некоторое свойство класса абстрактным?

Абстрактные классы включают в себя атрибуты в дополнение к методам, вы можете потребовать атрибуты в конкретных классах, определив их с помощью @abstractproperty.

7. Каково назначение функции isinstance?

Функция isinstance () в Python используется для проверки, является ли объект экземпляром указанного класса или нет.

Вывод: в ходе выполнения лабораторной работы приобрел навыки по созданию иерархии классов при написании программ с помощью языка программирования Python версии 3.x.