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
Колбасов В.С. « »_____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.

Ход работы

1. Изучив методические указания, приступил к разбору примеров.

```
C:\Users\vaaaa\anaconda3\envs\1\python.exe "C:/Users/vaaaa/Desktop/ООП/3/Пример 1.py"
3/4
Введите обыкновенную дробь: 6/9
2/3
17/12
1/12
1/2
8/9

Process finished with exit code 0
```

Рисунок 3.1 – Разбор работы кода первого примера

```
C:\Users\vaaaa\anaconda3\envs\1\python.exe "C:/Users/vaaaa/Desktop/ООП/3/Пример 2.py"
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides

Process finished with exit code 0
```

Рисунок 3.2 – Разбор работы кода второго примера

```
C:\Users\vaaaa\anaconda3\envs\1\python.exe "C:/Users/vaaaa/Desktop/ООП/3/Пример 3.py"
I can walk and run
I can crawl
I can bark
I can roar

Process finished with exit code 0
```

Рисунок 3.3 – Разбор работы кода третьего примера 2. Затем приступил к выполнению общего задания, используя изученные по теоретическим документам и примерам новые методы работы с классами и объектами.

```python
class Unit:
    id = 0

    def __init__(self, team):
        self.team = team
        self.id = Unit.id
        Unit.id += 1


class Hero(Unit):
    def __init__(self, team):
        super().__init__(team)
        self.level = 1

    def level_up(self):
        self.level += 1
        return f"Уровень героя {self.id} увеличен и равен: {self.level}"


class Soldier(Unit):

    def follow_to(self, hero):
        return f'Солдат {self.id}  идет за героем {hero.id}'


if __name__ == "__main__":
    hero1 = Hero("team1")
    hero2 = Hero("team2")
    team1, team2 = [], []

    for i in range(50):
        unit = Soldier(randint(0, 1))
        if unit.team == 0:
            team1.append(unit)
        else:
            team2.append(unit)
```

Рисунок 3.4 – Код общего задания

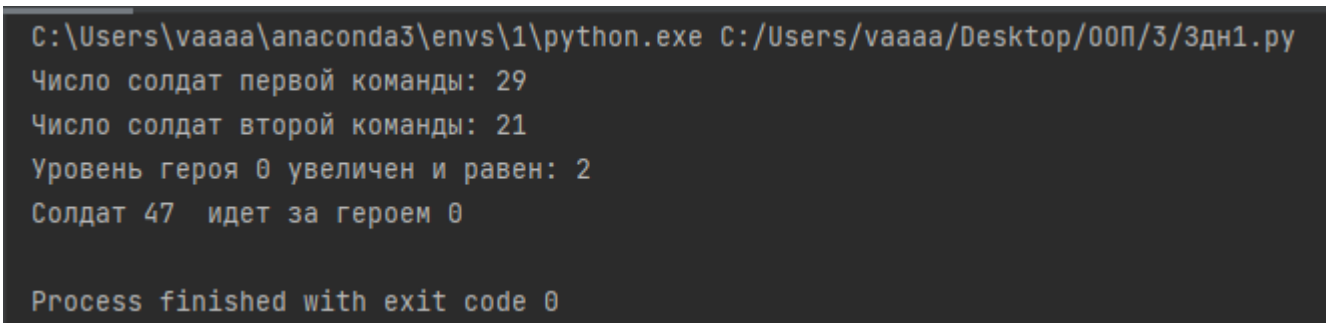

Рисунок 3.5 – Проверка кода общего задания

3. После чего приступил к выполнению индивидуальных заданий, закрепляя полученные знания.

```python
class Liquid:
    def __init__(self, name, density):
        self.__name = name
        self.__density = density

    @property
    def name(self):
        return self.__name

    @name.setter
    def name(self, inp):
        self.__name = inp

    @property
    def density(self):
        return self.__density

    @density.setter
    def density(self, inp):
        self.__density = inp


class Alcohol(Liquid):
    def __init__(self, name, density, strength):
        super().__init__(name, density)
        self.__strength = strength

    @property
    def strength(self):
        return self.__strength

    @strength.setter
    def strength(self, inp):
```

Рисунок 3.6 – Код первого задания

```
C:\Users\vaaaa\anaconda3\envs\1\python.exe C:/Users/vaaaa/Desktop/ООП/3/3дн2.py
Алкоголь 949 30
Вода 1000
Алкоголь 949 60

Process finished with exit code 0
```

Рисунок 3.7 – Проверка кода первого задания

```python
class Ellipse(Function):

    def __init__(self, a, b, x):
        self.__a = a
        self.__b = b
        self.__x = x
        self.__y = None

    def solve(self):
        if self.__a == 0:
            raise ValueError()

        self.__y = sqrt(1 + (self.__x ** 2 / self.__a ** 2) * self.__b ** 2)

    def display(self):
        return self.__y


class Hyperbola(Function):

    def __init__(self, a, b, x):
        self.__a = a
        self.__b = b
        self.__x = x
        self.__y = None
```

Рисунок 3.8 – Код второго задания

```
C:\Users\vaaaa\anaconda3\envs\1\python.exe C:/Users/vaaaa/Desktop/ООП/3/Здн3.py
14.035668847618199
0.7713892158398701

Process finished with exit code 0
```

Рисунок 3.9 – Результат выполнения кода

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 назназначение модуля abc языка программирования Python?

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

АБК 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 были приобретены простейшие навыки по работе с наследованием и абстрактными методами классов в языке программирования Python.