

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
**Федеральное государственное автономное образовательное
учреждение высшего образования**
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Кафедра инфокоммуникаций
Институт цифрового развития

ОТЧЁТ

по лабораторной работе №4.3

Дисциплина: «Объектно-ориентированное программирование»

Тема: «Наследование и полиморфизм в языке Python»

Вариант 5

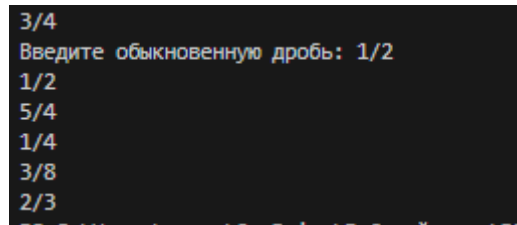
Выполнила: студентка 3 курса,
группы ИВТ-б-о-21-1
Диченко Дина Алексеевна

Ставрополь 2023

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

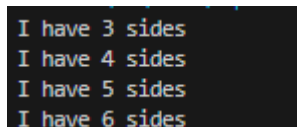
Практическая часть:

1. Создала репозиторий.
2. Клонировала репозиторий.
3. Дополнила файл .gitignore.
4. Организовала репозиторий в соответствии с моделью ветвления git-flow.
5. Проработала примеры.



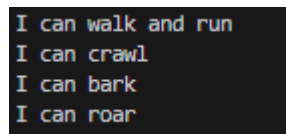
```
3/4
Введите обыкновенную дробь: 1/2
1/2
5/4
1/4
3/8
2/3
```

Рисунок 1. Результат выполнения примера 1



```
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides
```

Рисунок 2. Результат выполнения примера 2



```
I can walk and run
I can crawl
I can bark
I can roar
```

Рисунок 3. Результат выполнения примера 3

6. Разработала программу по следующему описанию.

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.

```
Количество солдат команды 1: 8
Количество солдат команды 2: 12
Количество солдат команды 3: 14
Уровень героя 'Команда 3' увеличен. Новый уровень: 2
Солдат 1 идет за героем 'Команда 1'
```

Рисунок 4. Результат выполнения общего задания

7. Выполнила индивидуальные задания.

Задание 1. Составить программу с использованием иерархии классов. Номер варианта необходимо получить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ == '__main__':` добавить код, демонстрирующий возможности разработанных классов.

Создать класс `Man` (человек), с полями: имя, возраст, пол и вес. Определить методы переназначения имени, изменения возраста и изменения веса. Создать производный класс `Student`, имеющий поле года обучения. Определить методы переназначения и увеличения года обучения.

```
Имя: Дмитрий, Пол: Мужчина, Возраст: 20, Вес: 70
Имя: Дима, Пол: Мужчина, Возраст: 21, Вес: 67

Имя: Алиса, Пол: Женщина, Возраст: 20, Вес: 55, Год обучения: 1
Имя: Алиса, Пол: Женщина, Возраст: 20, Вес: 55, Год обучения: 2
Имя: Алиса, Пол: Женщина, Возраст: 20, Вес: 55, Год обучения: 3
```

Рисунок 5. Результат выполнения индивидуального задания 1

Задание 2. В следующих заданиях требуется реализовать абстрактный базовый класс, определив в нем абстрактные методы и свойства. Эти методы определяются в производных классах. В базовых классах должны быть объявлены абстрактные методы ввода/вывода, которые реализуются в производных классах. Вызывающая программа должна продемонстрировать все варианты вызова переопределенных абстрактных методов. Написать функцию вывода, получающую параметры базового класса по ссылке и демонстрирующую виртуальный вызов.

Создать абстрактный базовый класс Triangle для представления треугольника с виртуальными функциями вычисления площади и периметра. Поля данных должны включать две стороны и угол между ними. Определить классы-наследники: прямоугольный треугольник, равнобедренный треугольник, равносторонний треугольник со своими функциями вычисления площади и периметра.

```
Прямоугольный треугольник:  
Введите длину первого катета: 3  
Введите длину второго катета: 4  
Площадь: 6.00  
Периметр: 12.00  
Равнобедренный треугольник:  
Введите длину основания: 8  
Площадь: 12.00  
Периметр: 18.00  
Равносторонний треугольник:  
Введите длину сторон: 5  
Площадь: 10.83  
Периметр: 15.00
```

Рисунок 6. Результат выполнения индивидуального задания 2

Контрольные вопросы:

1 Что такое наследование как оно реализовано в языке Python?

Наследование – способ создания нового класса на основе существующего.

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 – принцип ООП, который позволяет объектам различных типов обрабатываться с использованием обзего интерфейса. Полиморфизм,

как правило, используется с позиции переопределения методов базового класса в классе наследнике.

Проще всего это рассмотреть на примере. Добавим в наш базовый класс метод `info()`, который печатает сводную информацию по объекту класса `Figure` и переопределим этот метод в классе `Rectangle`, добавим в него дополнительные данные:

```
class Figure:
    def __init__(self, color):
        self.__color = color
    @property
    def color(self):
        return self.__color
    @color.setter
    def color(self, c):
        self.__color = c
    def info(self):
        print("Figure")
        print("Color: " + self.__color)

class Rectangle(Figure):
    def __init__(self, width, height, color):
        super().__init__(color)
        self.__width = width
        self.__height = height
    @property
    def width(self):
        return self.__width
    @width.setter
    def width(self, w):
        if w > 0:
            self.__width = w
        else:
            raise ValueError
    @property
    def height(self):
        return self.__height
    @height.setter
    def height(self, h):
        if h > 0:
            self.__height = h
        else:
            raise ValueError
    def area(self):
        return self.__width * self.__height

    def info(self):
        print("Rectangle")
        print("Color: " + self.color)
```

```
print("Width: " + str(self.width))
print("Height: " + str(self.height))
print("Area: " + str(self.area()))
```

Посмотрим, как это работает

```
>>> fig = Figure("orange")
>>> fig.info()
Figure
Color: orange
>>> rect = Rectangle(10, 20, "green")
>>> rect.info()
Rectangle
Color: green
Width: 10
Height: 20
Area: 200
```

Таким образом, класс наследник может расширять функционал класса родителя.

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

Тип или класс объекта не важен, пока объект поддерживает необходимые методы или свойства («Если что-то выглядит как утка, плавает как утка и крикает как утка, то это, вероятно, и есть утка»).

4 Каково назначение модуля abc языка программирования Python?

Для создания абстрактных классов.

5 Как сделать некоторый метод класса абстрактным?

С использованием декоратора @abstractmethod из модуля abc.

6 Как сделать некоторое свойство класса абстрактным?

С использованием модуля abc и декоратора @property в сочетании с абстрактным методом.

7 Каково назначение функции isinstance ?

Проверка принадлежности объекта к определенному типу или классу.

Вывод: в результате выполнения работы были приобретены навыки по созданию иерархии классов при написании программ с помощью языка программирования Python версии 3.x.