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

Кафедра инфокоммуникаций

Объектно-ориентированное программирование

Отчет по лабораторной работе №4.3

Наследование и полиморфизм в языке Python

Выполнил студент группы

ИВТ-б-о-21-1

Урусов М.А « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил доцент
Кафедры инфокоммуникаций, старший преподаватель
Воронкин Р.А.

_____
(подпись)

Ставрополь 2023

Наследование и полиморфизм в языке Python.

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

Порядок выполнения работы:

Задание.

Код программы:

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

import random


class Soldier:
    def __init__(self, number, team):
        self.number = number
        self.team = team

    def go_to_hero(self, hero):
        print(f"Солдат {self.number} идет за героем {hero.number}")


class Hero:
    def __init__(self, number):
        self.number = number
        self.level = 1

    def increase_level(self):
        self.level += 1


if __name__ == "__main__":
    hero1 = Hero(1)
```

```
hero2 = Hero(2)

soldiers_team1 = []
soldiers_team2 = []

for _ in range(10):
    number = random.randint(1, 100)
    team = random.choice([1, 2])
    soldier = Soldier(number, team)

    if soldier.team == 1:
        soldiers_team1.append(soldier)
    else:
        soldiers_team2.append(soldier)

if len(soldiers_team1) > len(soldiers_team2):
    hero1.increase_level()
else:
    hero2.increase_level()

soldier_to_follow = random.choice(soldiers_team1)
soldier_to_follow.go_to_hero(hero1)

print(f"Идентификационный номер солдата: {soldier_to_follow.number}")
print(f"Идентификационный номер героя: {hero1.number}")
```

Результат работы программы:

```
Солдат 57 идет за героем 1
Идентификационный номер солдата: 57
Идентификационный номер героя: 1
```

Рисунок 1. Результат работы программы

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 Pair (пара целых чисел); определить методы изменения полей и операцию
сложения пар (a,b)+(c.b)=(a+b.c+b). Определить класс-наследник Long с полями: старшая часть числа и младшая часть числа. Перео пределить операцию сложения и
определить методы умножения и вычитания.

Код программы:

```
C:\Users\den-n\AppData\Local\Programs\Python\Python312\python.exe D:\git\Git\Lab_4.3\Task\Task_1.py
Работа с классом Pair:
Введите первое число: 12
Введите второе число: 13
Результат сложения пар:
Пара: (15, 17)

Работа с классом Long:
Введите первое число: 14
Введите второе число: 25
Результат сложения длинных чисел:
Пара: (18, 0)
Результат умножения длинного числа на 2:
Пара: (28, 50)
Результат вычитания длинных чисел:
Пара: (10, 50)

Process finished with exit code 0
```

```
        # Переопределение операции сложения
        new_first = self.first + other.first
        new_second = self.second + other.second
        # Обработка переполнения
        if new_second >= 100:
            new_first += new_second // 100
            new_second = new_second % 100
        return Long(new_first, new_second)

    1 usage  new *
    def multiply(self, number):
        # Умножение на число
        total_second = (self.first * 100 + self.second) * number
        return Long(total_second // 100, total_second % 100)

    1 usage  new *
    def subtract(self, other):
        # Вычитание чисел
        total_second_self = self.first * 100 + self.second
        total_second_other = other.first * 100 + other.second
        result = total_second_self - total_second_other
        return Long(result // 100, result % 100)

if __name__ == '__main__':
    print("Работа с классом Pair:")
    pair1 = Pair()
    pair1.read()
    pair2 = Pair( first: 3,  second: 4)
    result_pair = pair1 + pair2
    print("Результат сложения пар:")
    result_pair.display()

    print("\nРабота с классом Long:")
    long1 = Long()
    long1.read()
    long2 = Long( first: 3,  second: 75)
    result_long = long1 + long2
    print("Результат сложения длинных чисел:")
    result_long.display()
```

if __name__ == '__main__'

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

5 usages  MaksimUrusov *
class Pair:
    """
    Класс представляет пару целых чисел.
    """
    new *
    def __init__(self, first=0, second=0):
        self.first = first
        self.second = second

    2 usages  new *
    def read(self):
        self.first = int(input("Введите первое число: "))
        self.second = int(input("Введите второе число: "))

    5 usages (1 dynamic)  new *
    def display(self):
        print(f"Пара: ({self.first}, {self.second})")

    MaksimUrusov *
    def __add__(self, other):
        if isinstance(other, Pair):
            return Pair(self.first + other.first, self.second + other.second)
        else:
            raise ValueError("Операнд должен быть экземпляром класса Pair")

5 usages  new *
class Long(Pair):
    """
    Класс наследник Pair, реализующий длинные числа.
    """
    new *
    def __init__(self, first=0, second=0):
        super().__init__(first, second)

    new *
    def __add__(self, other):
```

Результат работы:

```
C:\Users\den-n\AppData\Local\Programs\Python\Python312\python.exe D:\git\Git\Lab_4.3\Task\Task_1.py
Работа с классом Pair:
Введите первое число: 12
Введите второе число: 13
Результат сложения пар:
Пара: (15, 17)

Работа с классом Long:
Введите первое число: 14
Введите второе число: 25
Результат сложения длинных чисел:
Пара: (18, 0)
Результат умножения длинного числа на 2:
Пара: (28, 50)
Результат вычитания длинных чисел:
Пара: (10, 50)

Process finished with exit code 0
```

Код программы:

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 Figure с абстрактными методами вычисления площади
и периметра. Создать производные классы: Rectangle (прямоугольник), Circle (круг),
Trapezium (трапеция) со своими функциями площади и периметра. Самостоятельно
определить, какие поля необходимы, какие из них можно задать в базовом классе, а какие —
в производных. Площадь трапеции:

Код программы:

```python
from abc import ABC, abstractmethod
import math


3 usages  MaksimUrusov *
class Figure(ABC):
    """
    Абстрактный базовый класс для фигуры.
    """

    MaksimUrusov *
    @abstractmethod
    def area(self):
        """
        Метод для вычисления площади.
        """
        pass

    MaksimUrusov *
    @abstractmethod
    def perimeter(self):
        """
        Метод для вычисления периметра.
        """
        pass

    1 usage (1 dynamic)  MaksimUrusov *
    @abstractmethod
    def display(self):
        """
        Метод для вывода результатов на экран.
        """
        pass


1 usage  new *
class Rectangle(Figure):
    """
    Класс прямоугольника, наследуется от Figure.
    """

    new *
    def __init__(self, length, width):
        self.length = length
```

```
        self.length = length
        self.width = width

    1 usage  new *
    def area(self):
        return self.length * self.width

    1 usage  new *
    def perimeter(self):
        return 2 * (self.length + self.width)

    1 usage (1 dynamic)  new *
    def display(self):
        print(f"Прямоугольник со сторонами {self.length} и {self.width} имеет площадь {self.area()} и периметр {self.perimeter()}.")

1 usage  new *
class Circle(Figure):
    """
    Класс круга, наследуется от Figure.
    """
    new *
    def __init__(self, radius):
        self.radius = radius

    1 usage  new *
    def area(self):
        return math.pi * (self.radius ** 2)

    1 usage  new *
    def perimeter(self):
        return 2 * math.pi * self.radius

    1 usage (1 dynamic)  new *
    def display(self):
        print(f"Круг с радиусом {self.radius} имеет площадь {self.area():.2f} и периметр {self.perimeter():.2f}.")

1 usage  new *
class Trapezium(Figure):
    """
```

Результат работы программы:

```
print("Создаем число:")
number = Pair.make_pair( first: 5,  second: 99)
number.display()

print("\nВведите число для создания:")
number.read()
number.display()

multiplier = int(input("Введите множитель: "))
number.multiply(multiplier)
print(f"Результат умножения:")
number.display()
```

Рисунок 2. Результат работы программы

Ответы на вопросы:

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

Наследование — это когда один класс (подкласс) получает свойства и методы другого класса (суперкласса). Подкласс может наследовать все публичные атрибуты и методы своего суперкласса и добавлять свои собственные. В языке Python наследование реализуется с помощью ключевого слова class. Для создания подкласса нужно указать имя суперкласса в скобках после имени подкласса. Подкласс получает все атрибуты и методы суперкласса, их можно использовать напрямую или переопределить.

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

Полиморфизм — это возможность объектов разных классов иметь одно и то же имя метода, но каждый класс может предоставить свою собственную реализацию этого метода. Это позволяет использовать одинаковое имя метода для объектов различных классов, что упрощает программирование и повышает гибкость кода. В языке Python полиморфизм реализуется через наследование и переопределение методов. Если в подклассе метод с тем же именем переопределяется, то при вызове этого метода на объекте подкласса будет использоваться его реализация, а не реализация суперкласса. Это

позволяет использовать одинаковые методы с разным поведением для разных классов.

3. Что такое «утиная» типизация в языке Python?

«Утиная» типизация (англ. duck typing) — это концепция в языке программирования Python, основанная на философии «если она выглядит как утка, плавает как утка и крякает как утка, то это, вероятно, и есть утка». В контексте Python утиная типизация означает, что тип объекта определяется по его возможностям и методам, а не по его явно заданному типу. Иными словами, если объект обладает определенными методами, то мы можем использовать его как экземпляр нужного типа, не задумываясь о его фактическом классе или интерфейсе.

4. Каково назначение модуля abc языка Python?

Модуль abc (аббревиатура от "Abstract Base Classes") является частью стандартной библиотеки языка Python и предоставляет средства для определения абстрактных базовых классов.

5. Как сделать некоторый метод класса абстрактным?

Необходимо декорировать его методы как абстрактные, а реализацию выносить в классы-наследники.

6. Как сделать некоторое свойство класса абстрактным?

Можно потребовать атрибут в конкретных классах, определив их с помощью @abstractproperty.

7. Каково назначение функции isinstance?

Функция isinstance() проверяет, является ли объект экземпляром указанного класса или его подкласса.

Вывод: в ходе работы были приобретены навыки по созданию иерархии классов при написании программ с использованием языка программирования Python версии 3.10.