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

Богдашов А.В. « »_____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.

Разработайте 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.

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__'. Создать класс Liquid (жидкость), имеющий поля названия и плотности. Определить методы переназначения и изменения плотности. Создать произ водный класс Alcohol (спирт), имеющий крепость. Определить методы пере назначения и изменения крепости.

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Создать класс Liquid (жидкость), имеющий поля названия и плотности. Определить
методы
переназначения и изменения плотности. Создать производный класс Alcohol (спирт),
имеющий крепость. Определить методы переназначения и изменения крепости.
"""


class Liquid:
    def __init__(self, name, density):
        self.name = name
        self.density = density

    def set_density(self, density):
        self.density = density

    def display(self):
        print(f"Жидкость: {self.name}, Плотность: {self.density} g/cm³")


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

    def set_strength(self, strength):
        self.strength = strength

    def display(self):
        super().display()
        print(f"Крепость алкоголя: {self.strength}%")


if __name__ == '__main__':
    water = Liquid("Вода", 1.0)
    water.display()

    vodka = Alcohol("Водка", 0.789, 40)
    vodka.display()

    # Изменение плотности и крепости
    water.set_density(0.998)
    vodka.set_strength(50)

    # Вывод измененных данных
    print("\nПосле модификации:")
    water.display()
    vodka.display()
```

Результат работы программы:

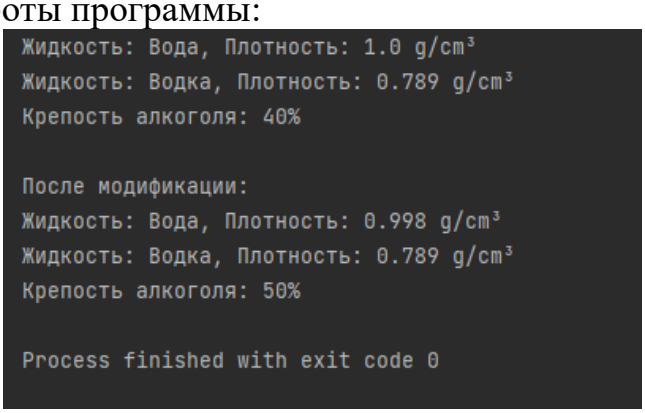

Рисунок 2. Результат работы программы

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 продемонстрироватьвсе варианты вызова переопределенных абстрактных методов. Написатьфункцию вывода, получающую параметры базового класса по ссылке идемонстрирующую виртуальный вызов.

Создать абстрактный базовый класс Body (тело) с абстрактными функциями вычисления площади поверхности и объема. Создать производные классы: Parallelepiped (параллелепипед) и Ball (шар) со своими функциями площади поверхности и объема.

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Создать абстрактный базовый класс Body (тело) с абстрактными функциями вычисления
площади поверхности и объема. Создать производные классы: Parallelepiped
(параллелепипед) и Ball (шар) со своими функциями площади поверхности и объема.
"""


from abc import ABC, abstractmethod
from math import pi


class Body(ABC):
    @abstractmethod
    def surface_area(self):
        pass

    @abstractmethod
    def volume(self):
        pass

    def __str__(self):
        return f"Площадь поверхности: {self.surface_area()}\nОбьём:
{self.volume()}"


class Parallelepiped(Body):
    def __init__(self, length, width, height):
        self.length = length
        self.width = width
        self.height = height

    def surface_area(self):
        return 2 * (self.length * self.width + self.length * self.height +
self.width * self.height)
```

```
    def volume(self):
        return self.length * self.width * self.height

    def __str__(self):
        return f"Параллелипипед:\n{super().__str__()}"


class Ball(Body):
    def __init__(self, radius):
        self.radius = radius

    def surface_area(self):
        return 4 * pi * self.radius**2

    def volume(self):
        return (4 / 3) * pi * self.radius**3

    def __str__(self):
        return f"Ball:\n{super().__str__()}"


if __name__ == "__main__":
    parallelepiped = Parallelepiped(3, 4, 5)
    print(parallelepiped)

    print("\n------------------\n")

    ball = Ball(2)
    print(ball
```

```
Параллелипипед:
Площадь поверхности: 94
Обьём: 60

------------------

Ball:
Площадь поверхности: 50.26548245743669
Обьём: 33.510321638291124

Process finished with exit code 0
```

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 выполнения данной работы были приобретены навыки по созданиюиерархии классов при написании программ с использованием языка программирования Python версии 3.10.