

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
дисциплины «Объектно-ориентированное программирование»

Выполнил:
Кожуховский Виктор Андреевич
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных систем
», очная форма обучения

(подпись)

Проверил:
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Наследование и полиморфизм в языке Python

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

Порядок выполнения работы:

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.
3. Выполнил клонирование созданного репозитория.
4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.
5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow.
6. Создал проект в папке репозитория.
7. Проработал примеры лабораторной работы.
9. 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.

```

2  # -*- coding: utf-8 -*-
3
4  import random
5
6
7  class Soldier:
8      def __init__(s, id, team):
9          s.id = id
10         s.team = team
11
12     def follow_hero(s, hero):
13         if hero.team == s.team:
14             hero.level_up(1)
15             return f"Soldier {s.id} is following hero {hero.id}"
16         else:
17             return (f"Soldier {s.id} cannot follow hero {hero.id} " +
18                     "because they are not in the same team.")
19
20
21 class Hero:
22     lvl = 0
23
24     def __init__(h, id, team):
25         h.id = id
26         h.team = team
27
28     def __str__(h) -> str:
29         return f"Hero {h.id} of {h.team}"
30
31     def level_up(h, lvl):
32         h.lvl += lvl
33         return f"Hero {h.id} leveled up by {h.lvl}"
34
35
36 if __name__ == '__main__':
37     sr = Hero(2, "red")
38     sb = Hero(1, "blue")
39
40     red_soldiers = []
41     blue_soldiers = []
42
43     for i in range(10):
44         team = random.choice(["red", "blue"])
45         soldier = Soldier(i + 1, team)
46         if team == "red":
47             red_soldiers.append(soldier)
48         else:
49             blue_soldiers.append(soldier)
50
51     print(f"Red soldiers count: {len(red_soldiers)}")
52     print(f"Blue soldiers count: {len(blue_soldiers)}")
53
54     if len(red_soldiers) > len(blue_soldiers):
55         soldier_following = red_soldiers[random.randint(
56             0, len(red_soldiers) - 1)]
57     else:
58         soldier_following = blue_soldiers[random.randint(
59             0, len(blue_soldiers) - 1)]
60
61     print(soldier_following.follow_hero(
62         sr if soldier_following.team == "red" else sb))
63
64     print(
65         f"Soldier ID: {soldier_following.id}, Hero ID: " +
66         f"{sr.id if soldier_following.team == 'red' else sb.id}, " +
67         f"Team: {soldier_following.team}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SEARCH ERROR

```

Red soldiers count: 7
Blue soldiers count: 3
Soldier 3 is following hero 2
Soldier ID: 3, Hero ID: 2, Team: red

```

Рисунок 1. Выполнение общего задания

8. Выполнил индивидуальное задание для варианта 14.

Задание 1

Реализовать класс-оболочку Number для числового типа float. Реализовать методы умножения и вычитания. Создать производный класс Real, в котором реализовать метод, вычисляющий корень произвольной степени, и метод для вычисления числа в данной степени.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  class Number:
6      def __init__(self, value: float):
7          self.value = value
8
9      def __mul__(self, other):
10         if isinstance(other, Number):
11             return Number(self.value * other.value)
12         raise ValueError()
13
14     def __sub__(self, other):
15         if isinstance(other, Number):
16             return Number(self.value - other.value)
17         raise ValueError()
18
19     def __repr__(self):
20         return f"Number({self.value})"
21
22
23     class Real(Number):
24         def root(self, n: float):
25             if n == 0:
26                 raise ValueError("Степень не может быть нулевой.")
27             return Number(self.value ** (1 / n))
28
29         def power(self, n: float):
30             return Number(self.value ** n)
31
32
33     if __name__ == "__main__":
34         num1 = Number(10.0)
35         num2 = Number(5.0)
36
37         print(f"Умножение: {num1 * num2}")
38         print(f"Вычитание: {num1 - num2}")
39
40         real_num = Real(27.0)
41
42         print(f"Кубический корень: {real_num.root(3)}")
43         print(f"27 в кубе: {real_num.power(3)}")
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS SEARCH ER

Умножение: Number(50.0)
Вычитание: Number(5.0)
Кубический корень: Number(3.0)
27 в кубе: Number(19683.0)

Рисунок 2. Код решения индивидуального задания 1 и его выполнение

Задание 2

Создать абстрактный класс Norm с виртуальной функцией вычисления нормы и модуля. Определить производные классы Complex, Vector3D с собственными функциями вычисления нормы и модуля. (Модуль для комплексного числа вычисляется как корень из суммы квадратов действительной и мнимой частей; норма для комплексных чисел вычисляется как модуль в квадрате. Модуль вектора вычисляется как корень квадратный из суммы квадратов координат; норма вектора вычисляется как максимальное из абсолютных значений координат.)

```
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3
4  from abc import ABC, abstractmethod
5  from math import sqrt
6
7
8  class Norm(ABC):
9
10     @abstractmethod
11     def CalculateNorm(self):
12         pass
13
14     @abstractmethod
15     def CalculateAbsolute(self):
16         pass
17
18
19  class Complex(Norm):
20     def __init__(self, real, img) -> None:
21         self.real = real
22         self.img = img
23
24     def CalculateAbsolute(self):
25         return sqrt(self.real ** 2 + self.img ** 2)
26
27     def CalculateNorm(self):
28         return self.CalculateAbsolute() ** 2
29
30
31  class Vector3D(Norm):
32     def __init__(self, x, y, z) -> None:
33         self.x = x
34         self.y = y
35         self.z = z
36
37     def CalculateAbsolute(self):
38         return sqrt(self.x ** 2 + self.y ** 2 + self.z ** 2)
39
40     def CalculateNorm(self):
41         return max(abs(self.x), abs(self.y), abs(self.z))
42
43
44  if __name__ == "__main__":
45     c = Complex(5, 2)
46     print("Комплексное число", c.real, "+", c.img, "i")
47     print("Модуль комплексного числа", c.CalculateAbsolute())
48     print("Норма комплексного числа", c.CalculateNorm())
49     v = Vector3D(1, 5, 7)
50     print("Вектор", v.x, v.y, v.z)
51     print("Модуль вектора", v.CalculateAbsolute())
52     print("Норма вектора", v.CalculateNorm())
53
54
55  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  COMMENTS
```

Комплексное число 5 + 2 i
Модуль комплексного числа 5.385164807134504
Норма комплексного числа 28.999999999999996
Вектор 1 5 7
Модуль вектора 8.660254037844387
Норма вектора 7

Рисунок 3. Код решения индивидуального задания 2 и его выполнение

9. Зафиксировал сделанные изменения в репозитории.
10. Выполнил слияние ветки для разработки с веткой master/main.
11. Отправил сделанные изменения на сервер GitHub.

Ссылка: <https://github.com/Viktorkozh/OOP-3>

Контрольные вопросы:

1. Как осуществляется объявление класса в языке Python?

class MyClass:

pass

Ключевое слово class используется для объявления класса, за которым следует имя класса и двоеточие.

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса: Общие для всех экземпляров класса. Объявляются внутри класса, но вне методов.

Атрибуты экземпляра: Уникальны для каждого экземпляра класса. Объявляются внутри метода `__init__`.

3. Каково назначение методов класса?

Методы класса определяют поведение объектов класса. Они могут изменять состояние объекта или выполнять действия, связанные с объектом.

4. Для чего предназначен метод `__init__()` класса?

Это инициализатор, который вызывается при создании нового экземпляра класса. Он используется для инициализации атрибутов экземпляра.

```
class MyClass:
```

```
    def __init__(self, value):  
        self.value = value
```

5. Каково назначение `self` ?

`self` представляет собой ссылку на текущий экземпляр класса и используется для доступа к атрибутам и методам класса изнутри.

6. Как добавить атрибуты в класс?

Атрибуты можно добавлять в методе `__init__` или в любом другом методе класса.

```
class MyClass:
```

```
    def __init__(self, value):  
        self.value = value
```

7. Как осуществляется управление доступом к методам и атрибутам в языке Python?

В Python используется соглашение об именовании для управления доступом:

Приватные атрибуты и методы начинаются с одного или двух подчеркиваний (например, `_private` или `__very_private`).

Публичные атрибуты и методы не имеют подчеркиваний.

8. Каково назначение функции `isinstance` ?

Функция `isinstance` используется для проверки, является ли объект экземпляром определенного класса или кортежа классов.

`isinstance(obj, MyClass)`

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