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

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 4.3 Наследование и полиморфизм в языке Python»

ОТЧЕТ
по лабораторной работе №31
дисциплины
«Основы программной инженерии»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Лабораторная работа 4.3 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.

Ход 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.

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

import random


class Unit:
    def __init__(self, number, name):
        self.__id = number
        self.__team = name

    @property
    def id(self):
        return self.__id

    @property
    def team(self):
        return self.__team


class Soldier(Unit):
    def follow_hero(self, hero):
        print(f"Солдат №{self.id} следует за героем №{hero.id}")
```

```python
class Hero(Unit):
    def __init__(self, number):
        super().__init__(number, "Hero")
        self.level = 1

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


if __name__ == '__main__':
    team1_hero = Hero(1)
    team2_hero = Hero(2)

    soldiers_1 = []
    soldiers_2 = []

    for i in range(10):
        team = random.choice([1, 2])
        soldier = Soldier(i + 1, team)

        if team == 1:
            soldiers_1.append(soldier)
        else:
            soldiers_2.append(soldier)

    print(f"Количество солдат в команде 1: {len(soldiers_1)}")
    print(f"Количество солдат в команде 2: {len(soldiers_2)}")

    if len(soldiers_1) > len(soldiers_2):
        team1_hero.increase_level()
    elif len(soldiers_2) > len(soldiers_1):
        team2_hero.increase_level()

    print(f"Уровень героя 1 команды: {team1_hero.level}")
    print(f"Уровень героя 2 команды: {team2_hero.level}")

    soldiers_1[1].follow_hero(team1_hero)
```

Листинг 1 – Код программы для общего задания

```
Количество солдат в команде 1: 7
Количество солдат в команде 2: 3
Уровень героя 1 команды: 2
Уровень героя 2 команды: 1
Солдат №3 следует за героем №1
```

Рисунок 1 – Результат выполнения программы

Индивидуальное задание

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 Triad (тройка чисел); определить методы увеличения полей на 1. Определить класс-наследник Time с полями: час, минута, секунда. Переопределить методы увеличения полей на 1 и определить методы увеличения на n секунд и минут.

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


class Triad:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def increase(self):
        self.a += 1
        self.b += 1
        self.c += 1


class Time(Triad):
    def __init__(self, hour, minute, second):
        super().__init__(hour, minute, second)

    def increase(self):
        self.c += 1
        if self.c >= 60:
            self.c -= 60
            self.b += 1
        if self.b >= 60:
            self.b -= 60
            self.a += 1

    def increase_seconds(self, n):
        self.c += n
        if self.c >= 60:
            self.c %= 60
            self.b += 1
        if self.b >= 60:
            self.b %= 60
            self.a += 1

    def increase_minutes(self, n):
        self.b += n
        if self.b >= 60:
            self.b %= 60
            self.a += 1


if __name__ == '__main__':
    time = Time(11, 20, 35)
    print(f"Время: {time.a}:{time.b}:{time.c}")

    time.increase()
    print(f"Вряме через секунду: {time.a}:{time.b}:{time.c}")
```

```
    time.increase_seconds(80)
    print(f"Время через 80 секунд: {time.a}:{time.b}:{time.c}")

    time.increase_minutes(30)
    print(f"Время через полчаса: {time.a}:{time.b}:{time.c}")
```

Листинг 2 – Листинг индивидуального задания 1

```
Время: 11:20:35
Вряме через секунду: 11:20:36
Время ещё через 80 секунд: 11:21:56
Время ещё через полчаса: 11:51:56
```

Рисунок 2 – Результат выполнения программы

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

from abc import ABC, abstractmethod
import math


class Container(ABC):
    @abstractmethod
    def sort(self):
        pass

    @abstractmethod
    def foreach(self):
        pass


class Bubble(Container):
    def __init__(self, data):
        self.data = data

    def sort(self):
        n = len(self.data)
        for i, item in enumerate(self.data):
            for j in range(0, n - i - 1):
                if self.data[j] > self.data[j + 1]:
                    self.data[j], self.data[j + 1] = self.data[j + 1],
self.data[j]
        return self.data

    def foreach(self):
        for i, item in enumerate(self.data):
            self.data[i] = math.sqrt(item)
        return self.data


class Choice(Container):
    def __init__(self, data):
        self.data = data
```

```
    def sort(self):
        for i, item in enumerate(self.data):
            mn = min(range(i, len(self.data)), key=self.data.__getitem__)
            self.data[i], self.data[mn] = self.data[mn], item
        return self.data

    def foreach(self):
        for i, item in enumerate(self.data):
            self.data[i] = math.log(item)
        return self.data


def print_container(container):
    print(f"Исходные данные: {container.data}")
    print(f"Сортированные данные: {container.sort()}")
    print(f"Обработанный список: {container.foreach()}")


if __name__ == '__main__':
    first_container = Bubble([9, 54, 7, 2, 1])
    print_container(first_container)

    second_container = Choice([22, 6, 7, 3, 8])
    print_container(second_container)
```

Листинг 3 – Код индивидуального задания 2

```
Исходные данные: [9, 54, 7, 2, 1]
Сортированные данные: [1, 2, 7, 9, 54]
Обработанный список: [1.0, 1.4142135623730951, 2.6457513110645907, 3.0, 7.3484692283495345]
Исходные данные: [22, 6, 7, 3, 8]
Сортированные данные: [3, 6, 7, 8, 22]
Обработанный список: [1.0986122886681098, 1.791759469228055, 1.9459101490553132, 2.0794415416798357, 3.091042453358316]
```

Рисунок 3 – Результат выполнения программы

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

Ответы на контрольные вопросы:

1. Что такое наследование как оно реализовано в языке Python? 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 - это возможность объектов с одинаковой сигнатурой методов вызывать разные реализации этого метода в зависимости от текущего типа объекта. В Python полиморфизм реализуется через вызов методов класса объекта без необходимости указывать явно тип объекта.

3. **Что такое "утиная" типизация в языке программирования Python?** "Утиная" типизация - это стиль программирования, при котором проверка на соответствие типу объекта происходит во время выполнения, а не на этапе компиляции. В Python все объекты имеют общий тип object, и проверка соответствия типу может быть выполнена с помощью ключевого слова isinstance.

4. **Каково назназначение модуля abc языка программирования Python?** По умолчанию Python не предоставляет абстрактных классов. Python поставляется с модулем, который обеспечивает основу для определения абстрактных базовых классов (ABC), и имя этого модуля - ABC. ABC работает, декорируя методы базового класса как абстрактные, а затем регистрируя конкретные классы как реализации абстрактной базы. Метод становится абстрактным, если он украшен ключевым словом @abstractmethod.

5. **Как сделать некоторый метод класса абстрактным?** Для того чтобы сделать метод класса абстрактным, нужно создать абстрактный метод в базовом классе с помощью декоратора @abstractmethod. Этот метод не должен иметь реализации в базовом классе, и должен быть переопределен в каждом наследнике.

6. **Как сделать некоторое свойство класса абстрактным?** Для того чтобы сделать свойство класса абстрактным, нужно создать абстрактное свойство в базовом классе с помощью декоратора @abstractmethod. Это свойство не должно иметь реализации в базовом классе, и должно быть переопределено в каждом наследнике.

7. Каково назначение функции isinstance? Функция isinstance используется для проверки соответствия типа объекта указанному классу или его наследнику. Она принимает два аргумента: объект, тип которого нужно проверить, и класс или кортеж классов, с которым нужно сравнить тип объекта. Если объект является экземпляром указанного класса или его наследника, то функция возвращает True, в противном случае - False.