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»
Вариант 20

Выполнила:
Михеева Елена Александровна
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 Р.А.-доцент департамента цифровых, роботехнических систем и электроники института перспективной инженерии

_____
(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:

Ссылка на Github: https://github.com/helendddd/Oop_3.git

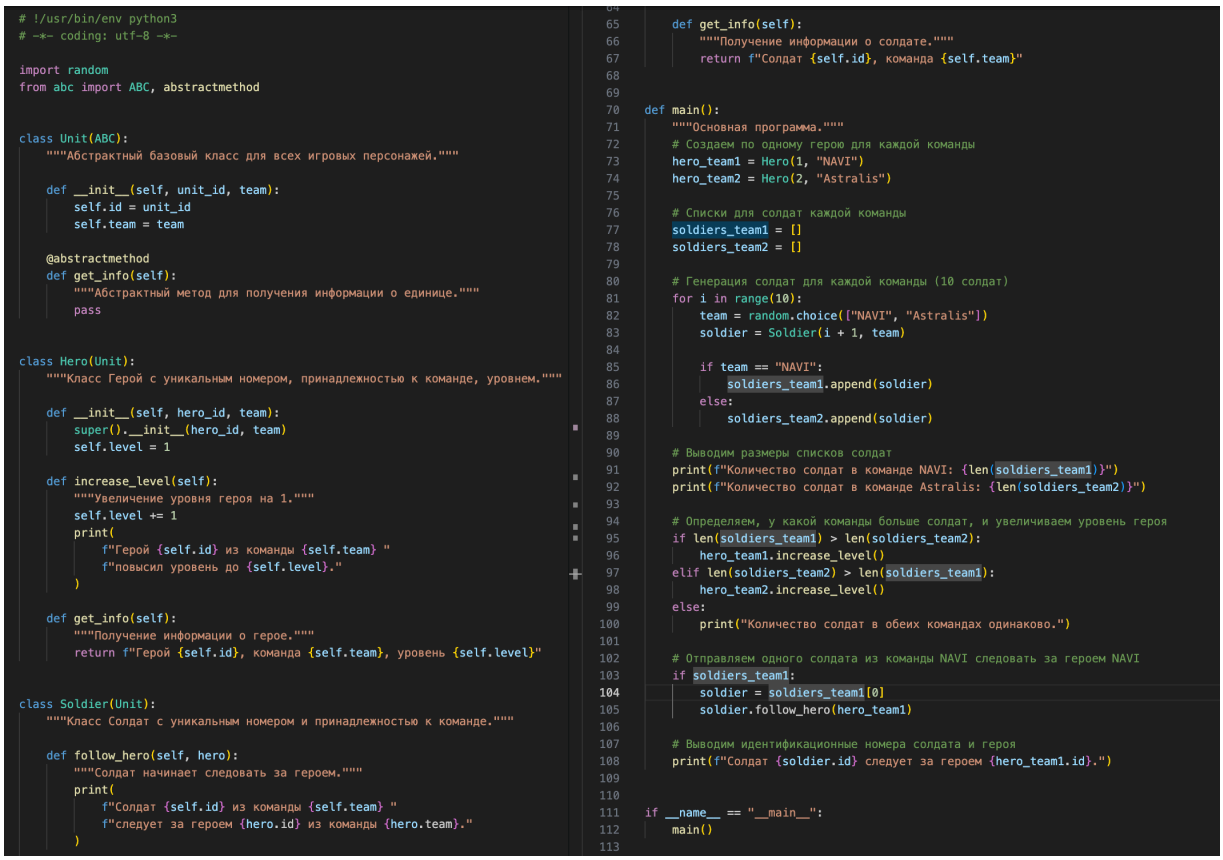
1. Была разработана программа по следующему описанию:

«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.

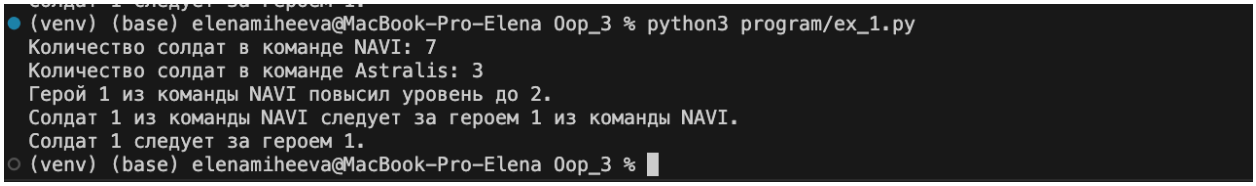
Измеряется длина списков солдат противоборствующих команд и выводится на экран. У героя, принадлежащего команде с более длинным списком, увеличивается уровень.

Отправьте одного из солдат первого героя следовать за ним. Выведите на экран идентификационные номера этих двух юнитов.»



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

import random
from abc import ABC, abstractmethod


class Unit(ABC):
    """Абстрактный базовый класс для всех игровых персонажей."""

    def __init__(self, unit_id, team):
        self.id = unit_id
        self.team = team

    @abstractmethod
    def get_info(self):
        """Абстрактный метод для получения информации о единице."""
        pass


class Hero(Unit):
    """Класс Герой с уникальным номером, принадлежностью к команде, уровнем."""

    def __init__(self, hero_id, team):
        super().__init__(hero_id, team)
        self.level = 1

    def increase_level(self):
        """Увеличение уровня героя на 1."""
        self.level += 1
        print(
            f"Герой {self.id} из команды {self.team} "
            f"повысил уровень до {self.level}."
        )

    def get_info(self):
        """Получение информации о герое."""
        return f"Герой {self.id}, команда {self.team}, уровень {self.level}"


class Soldier(Unit):
    """Класс Солдат с уникальным номером и принадлежностью к команде."""

    def follow_hero(self, hero):
        """Солдат начинает следовать за героем."""
        print(
            f"Солдат {self.id} из команды {self.team} "
            f"следует за героем {hero.id} из команды {hero.team}."
        )
    def get_info(self):
        """Получение информации о солдате."""
        return f"Солдат {self.id}, команда {self.team}"


def main():
    """Основная программа."""
    # Создаем по одному герою для каждой команды
    hero_team1 = Hero(1, "NAVI")
    hero_team2 = Hero(2, "Astralis")

    # Списки для солдат каждой команды
    soldiers_team1 = []
    soldiers_team2 = []

    # Генерация солдат для каждой команды (10 солдат)
    for i in range(10):
        team = random.choice(["NAVI", "Astralis"])
        soldier = Soldier(i + 1, team)

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

    # Выводим размеры списков солдат
    print(f"Количество солдат в команде NAVI: {len(soldiers_team1)}")
    print(f"Количество солдат в команде Astralis: {len(soldiers_team2)}")

    # Определяем, у какой команды больше солдат, и увеличиваем уровень героя
    if len(soldiers_team1) > len(soldiers_team2):
        hero_team1.increase_level()
    elif len(soldiers_team2) > len(soldiers_team1):
        hero_team2.increase_level()
    else:
        print("Количество солдат в обеих командах одинаково.")

    # Отправляем одного солдата из команды NAVI следовать за героем NAVI
    if soldiers_team1:
        soldier = soldiers_team1[0]
        soldier.follow_hero(hero_team1)

    # Выводим идентификационные номера солдата и героя
    print(f"Солдат {soldier.id} следует за героем {hero_team1.id}.")


if __name__ == "__main__":
    main()
```

Рисунок 1. Программа задания №1

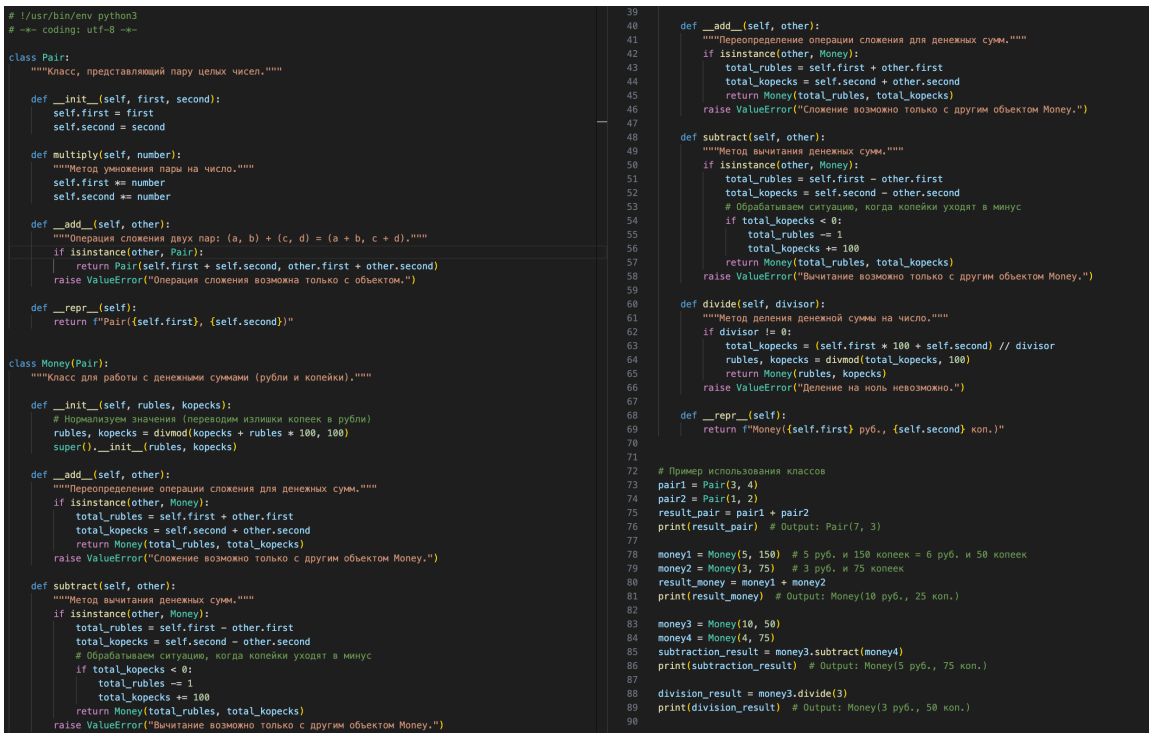


```
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_3 % python3 program/ex_1.py
Количество солдат в команде NAVI: 7
Количество солдат в команде Astralis: 3
Герой 1 из команды NAVI повысил уровень до 2.
Солдат 1 из команды NAVI следует за героем 1 из команды NAVI.
Солдат 1 следует за героем 1.
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_3 %
```

Рисунок 2. Результат работы программы

2. Приступили к выполнению индивидуального задания согласно варианту №20.

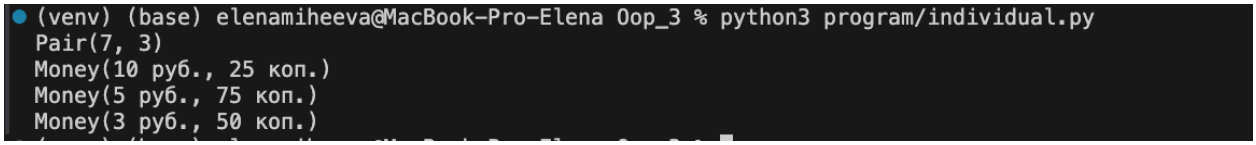
Задание: создать класс Pair (пара целых чисел); определить метод умножения на число и операцию сложения пар $(a,b) + (c,d) = (a + b,c + d)$. Определить класс-наследник Money с полями: рубли и копейки. Переопределить операцию сложения и определить методы вычитания и деления денежных сумм.



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

class Pair:
    """Класс, представляющий пару целых чисел."""

    def __init__(self, first, second):
        self.first = first
        self.second = second

    def multiply(self, number):
        """Метод умножения пары на число."""
        self.first *= number
        self.second *= number

    def __add__(self, other):
        """Операция сложения двух пар: (a, b) + (c, d) = (a + b, c + d)."""
        if isinstance(other, Pair):
            return Pair(self.first + self.second, other.first + other.second)
        raise ValueError("Операция сложения возможна только с объектом.")

    def __repr__(self):
        return f"Pair({self.first}, {self.second})"


class Money(Pair):
    """Класс для работы с денежными суммами (рубли и копейки)."""

    def __init__(self, rubles, kopecks):
        # Нормализуем значения (переводим излишки копеек в рубли)
        rubles, kopecks = divmod(kopecks + rubles * 100, 100)
        super().__init__(rubles, kopecks)

    def __add__(self, other):
        """Переопределение операции сложения для денежных сумм."""
        if isinstance(other, Money):
            total_rubles = self.first + other.first
            total_kopecks = self.second + other.second
            return Money(total_rubles, total_kopecks)
        raise ValueError("Сложение возможно только с другим объектом Money.")

    def subtract(self, other):
        """Метод вычитания денежных сумм."""
        if isinstance(other, Money):
            total_rubles = self.first - other.first
            total_kopecks = self.second - other.second
            # Обрабатываем ситуацию, когда копейки уходят в минус
            if total_kopecks < 0:
                total_rubles -= 1
                total_kopecks += 100
            return Money(total_rubles, total_kopecks)
        raise ValueError("Вычитание возможно только с другим объектом Money.")
```

```python
    def __add__(self, other):
        """Переопределение операции сложения для денежных сумм."""
        if isinstance(other, Money):
            total_rubles = self.first + other.first
            total_kopecks = self.second + other.second
            return Money(total_rubles, total_kopecks)
        raise ValueError("Сложение возможно только с другим объектом Money.")

    def subtract(self, other):
        """Метод вычитания денежных сумм."""
        if isinstance(other, Money):
            total_rubles = self.first - other.first
            total_kopecks = self.second - other.second
            # Обрабатываем ситуацию, когда копейки уходят в минус
            if total_kopecks < 0:
                total_rubles -= 1
                total_kopecks += 100
            return Money(total_rubles, total_kopecks)
        raise ValueError("Вычитание возможно только с другим объектом Money.")

    def divide(self, divisor):
        """Метод деления денежной суммы на число."""
        if divisor != 0:
            total_kopecks = (self.first * 100 + self.second) // divisor
            rubles, kopecks = divmod(total_kopecks, 100)
            return Money(rubles, kopecks)
        raise ValueError("Деление на ноль невозможно.")

    def __repr__(self):
        return f"Money({self.first} руб., {self.second} коп.)"


# Пример использования классов
pair1 = Pair(3, 4)
pair2 = Pair(1, 2)
result_pair = pair1 + pair2
print(result_pair)  # Output: Pair(7, 3)

money1 = Money(5, 150)  # 5 руб. и 150 копеек = 6 руб. и 50 копеек
money2 = Money(3, 75)   # 3 руб. и 75 копеек
result_money = money1 + money2
print(result_money)  # Output: Money(10 руб., 25 коп.)

money3 = Money(10, 50)
money4 = Money(4, 75)
subtraction_result = money3.subtract(money4)
print(subtraction_result)  # Output: Money(5 руб., 75 коп.)

division_result = money3.divide(3)
print(division_result)  # Output: Money(3 руб., 50 коп.)
```

Рисунок 1. Программа индивидуального задания



```
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_3 % python3 program/individual.py
Pair(7, 3)
Money(10 руб., 25 коп.)
Money(5 руб., 75 коп.)
Money(3 руб., 50 коп.)
```

Рисунок 2. Результат работы программы

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.

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 один из ключевых принципов объектно-ориентированного программирования (ООП), который позволяет объектам разных классов обрабатывать данные одним и тем же способом. Полиморфизм можно описать как способность функции, метода или операции работать с разными типами объектов, предоставляя единый интерфейс для них.

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

Утиная типизация — это концепция, характерная для языков программирования с динамической типизацией, согласно которой конкретный тип или класс объекта не важен, а важны лишь свойства и методы, которыми этот объект обладает. Другими словами, при работе с объектом его тип не проверяется, вместо этого проверяются свойства и методы этого объекта. Такой подход добавляет гибкости коду, позволяет полиморфно работать с объектами, которые никак не связаны друг с другом и могут быть объектами разных классов. Единственное условие, чтобы все эти объекты поддерживали необходимый набор свойств и методов.

4. Каково назначение 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.

5. Как сделать некоторый метод класса абстрактным?

Метод становится абстрактным, если он украшен ключевым словом @abstractmethod.

6. Как сделать некоторое свойство класса абстрактным?

Для того чтобы сделать свойство класса абстрактным в Python, необходимо использовать модуль abc, который позволяет создавать абстрактные классы и методы. Абстрактное свойство или метод объявляется, но не реализуется в базовом классе, и подклассы обязаны его реализовать.

7. Каково назначение функции isinstance?

Функция «isinstance» используется для проверки, принадлежит ли объект к определенному классу или группе классов (кортежу классов). Возвращает «True», если объект является экземпляром указанного класса или его подклассов, и «False» в противном случае.

Вывод: были приобретены навыки по созданию иерархии классов при написании программ с помощью языка программирования Python.