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 №1
дисциплины
«Объектно-ориентированное программирование»
Вариант 13

Выполнил:
Рябинин Егор Алексеевич
3 курс, группа ИВТ-б-о-23-2,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии
Воронкин Роман Александрович
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г

Тема: Элементы объектно-ориентированного программирования в языке Python.

Цель: приобретение навыков по работе с классами и объектами при написании программ с помощью языка программирования Python.

Порядок выполнения работы:

Задание №1.

Парой называется класс с двумя полями, которые обычно имеют имена *first* и *second*. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать:

- метод инициализации `__init__`; метод должен контролировать значения аргументов на корректность;
- ввод с клавиатуры `read`;
- вывод на экран `display`.

Реализовать внешнюю функцию с именем `make_тип()`, где `тип` — тип реализуемой структуры. Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ = '__main__':` добавить код, демонстрирующий возможности разработанного класса.

Рисунок 1 – Задание №1

13. Поле first — дробное положительное число, катет a прямоугольного треугольника; поле second — дробное положительное число, катет b прямоугольного треугольника. Реализовать метод `hypotenuse()` — вычисление гипотенузы.

Рисунок 2 – Индивидуальное задание

```
class RightTrianglePair:
```

Рисунок 3 – Объявление класса, создание класса для работы с прямоугольным треугольником

```
def __init__(self, first: float = 1, second: float = 1):
    if first <= 0 or second <= 0:
        raise ValueError("Катеты должны быть положительными числами")

    self.__first = float(first)
    self.__second = float(second)
```

Рисунок 4 – Конструктор класса, инициализация объекта, проверка корректности данных, создание приватных атрибутов

```python
def read(self):
    self.__first = float(input("Введите первый катет: "))
    self.__second = float(input("Введите второй катет: "))

    if self.__first <= 0 or self.__second <= 0:
        raise ValueError("Катеты должны быть положительными числами")
```

Рисунок 5 – Метод ввода данных – чтение значений катетов с клавиатуры с проверкой корректности

```python
def display(self):
    print(f"Первый катет: {self.__first}")
    print(f"Второй катет: {self.__second}")
    print(f"Гипотенуза: {self.hypotenuse():.2f}")
```

Рисунок 6 – Метод вывода данных – отображение информации о треугольнике на экране

```python
@property
def first(self):
    return self.__first

@property
def second(self):
    return self.__second
```

Рисунок 7 – Свойства класса – предоставление доступа к приватным атрибутам

```python
def hypotenuse(self):
    return sqrt(self.__first**2 + self.__second**2)
```

Рисунок 8 – Метод вычисления гипотенузы – расчет гипотенузы по теореме Пифагора

```python
def make_right_triangle_pair(first: float, second: float) -> RightTrianglePair:
    if first <= 0 or second <= 0:
        raise ValueError("Катеты должны быть положительными числами")
    return RightTrianglePair(first, second)
```

Рисунок 9 – Внешняя функция – создание объекта треугольника с проверкой параметров

```python
if __name__ == "__main__":
    triangle = RightTrianglePair()
    triangle.read()
    triangle.display()
```

Рисунок 10 – Главный блок программы – создание объекта, ввод данных и вывод результата.

```
(OOP_lab1) PS C:\Users\4isto\OOP_lab1> python tasks/task1.py
Введите первый катет: 3
Введите второй катет: 4
Первый катет: 3.0
Второй катет: 4.0
Гипотенуза: 5.00
```

Рисунок 11 – Результат работы программы

Задание №2.

Составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

- метод инициализации `__init__`;
- ввод с клавиатуры `read`;
- вывод на экран `display`.

Номер варианта необходимо уточнить у преподавателя. В раздел программы, начинающийся после инструкции `if __name__ = '__main__':` добавить код, демонстрирующий возможности разработанного класса.

Рисунок 12 – Задание №2

13. Создать класс Goods (товар). В классе должны быть представлены поля: наименование товара, дата оформления, цена товара, количество единиц товара, номер накладной, по которой товар поступил на склад. Реализовать методы изменения цены товара, изменения количества товара (увеличения и уменьшения), вычисления стоимости товара.

Рисунок 13 – Индивидуальное задание

```python
class Goods:
    def __init__(self, name="", date="", price=0.0, quantity=0, number=0):
        if price < 0:
            raise ValueError("Цена не может быть отрицательной")
        if quantity < 0:
            raise ValueError("Количество не может быть отрицательным")
        if number < 0:
            raise ValueError("Номер накладной не может быть отрицательным")

        self.name = name
        self.date = date
        self.price = price
        self.quantity = quantity
        self.number = number
```

Рисунок 14 – Конструктор класса - инициализация объекта товара с проверкой корректности данных

```python
def read(self):
    self.name = input("Введите наименование товара: ")
    self.date = input("Введите дату оформления: ")

    self.price = float(input("Введите цену товара: "))
    if self.price < 0:
        raise ValueError("Цена не может быть отрицательной")

    self.quantity = int(input("Введите количество единиц товара: "))
    if self.quantity < 0:
        raise ValueError("Количество не может быть отрицательным")

    self.number = int(input("Введите номер накладной: "))
    if self.number < 0:
        raise ValueError("Номер накладной не может быть отрицательным")
```

Рисунок 15 - Метод ввода данных - интерактивный ввод информации о товаре с проверкой

```python
def display(self):
    print(f"Наименование: {self.name}")
    print(f"Дата оформления: {self.date}")
    print(f"Цена: {self.price}")
    print(f"Количество: {self.quantity}")
    print(f"Номер накладной: {self.number}")
    print(f"Общая стоимость: {self.total_cost()}")
```

Рисунок 16 - Метод вывода данных - отображение полной информации о товаре

```
def change_price(self, amount):
    new_price = self.price + amount
    if new_price < 0:
        raise ValueError("Цена не может быть отрицательной")
    self.price = new_price
```

Рисунок 17 - Метод изменения цены - увеличение или уменьшение цены товара

```
def change_quantity(self, amount):
    new_quantity = self.quantity + amount
    if new_quantity < 0:
        raise ValueError("Количество не может быть отрицательным")
    self.quantity = new_quantity
```

Рисунок 18 - Метод изменения количества - увеличение или уменьшение количества товара

```
def total_cost(self):
    return self.price * self.quantity
```

Рисунок 19 - Метод расчета стоимости - вычисление общей стоимости товара

```
if __name__ == "__main__":
    goods = Goods()
    goods.read()
    goods.display()

    goods.change_price(100)
    goods.change_quantity(5)
    print("\nПосле увеличения цены и количества:")
    goods.display()
```

Рисунок 20 - Главный блок программы - демонстрация работы класса: создание объекта, ввод данных, изменение цены и количества, вывод результатов.

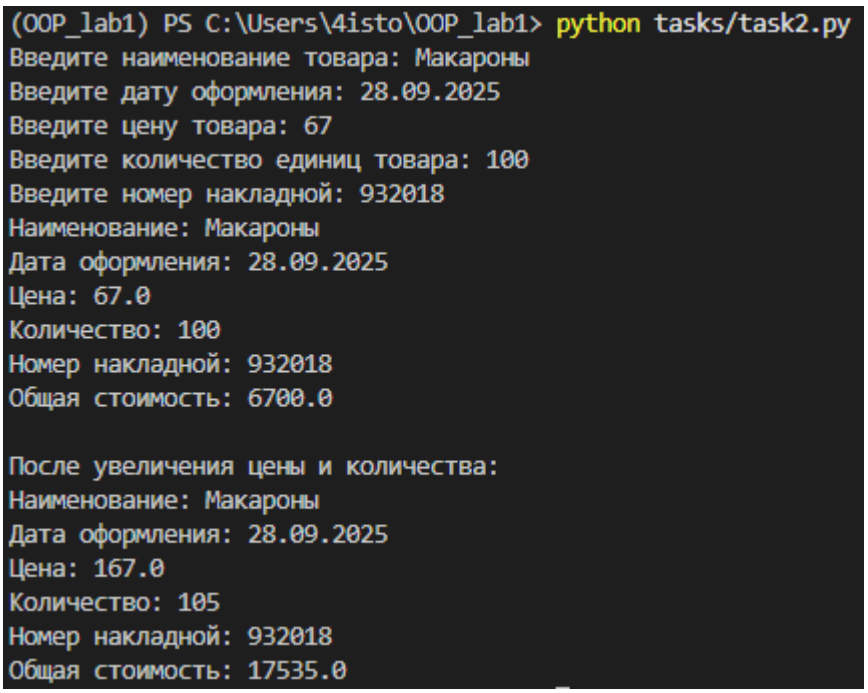

Рисунок 21 – Результат работы программы

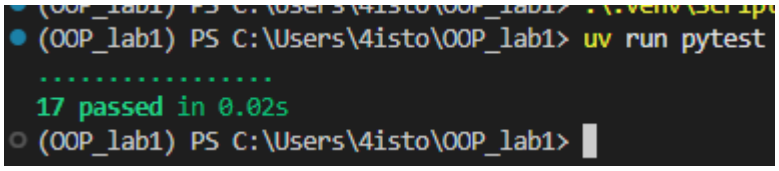

Рисунок 22 – Результат работы тестов для двух заданий

Контрольные вопросы:

1. Как осуществляется объявление класса в языке Python?

Класс объявляется с помощью ключевого слова class, за которым следует имя класса и двоеточие. Например: class MyClass:

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса принадлежат самому классу и разделяются всеми его экземплярами. Атрибуты экземпляра принадлежат конкретному объекту (экземпляру) класса и у каждого объекта могут быть свои значения.

3. Каково назначение методов класса?

Методы класса определяют поведение объектов этого класса. Они содержат код, который может работать с атрибутами экземпляра и выполнять операции, связанные с классом.

4. Для чего предназначен метод __init__() класса?

Метод init() является конструктором класса и вызывается при создании нового экземпляра. Он используется для инициализации атрибутов экземпляра.

5. Каково назначение self?

Self представляет ссылку на текущий экземпляр класса. Он используется для доступа к атрибутам и методам экземпляра внутри методов класса.

6. Как добавить атрибуты в класс?

Атрибуты добавляются в класс путем их объявления внутри методов, обычно в init(), или путем прямого присваивания значений атрибутам класса.

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

В Python управление доступом осуществляется через соглашения об именовании. Атрибуты и методы, начинающиеся с одного подчеркивания (_), считаются защищенными, а с двух подчеркиваний (__) - приватными. Однако это лишь соглашение, и прямой доступ все равно возможен.

8. Каково назначение функции isinstance?

Функция isinstance() проверяет, является ли объект экземпляром указанного класса или кортежа классов. Она возвращает True, если объект является экземпляром любого из указанных классов, и False в противном случае.

Вывод: в ходе лабораторной работы были приобретены навыки по работе с классами и объектами при написании программ с помощью языка программирования Python.