

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ  
ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ТЕЛЕКОММУНИКАЦИЙ  
КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ**

**Лабораторная работа №1 по дисциплине «ООП»**

Выполнил:

Хасензода Муборакшох Латиф

Студент 2 курса группы ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

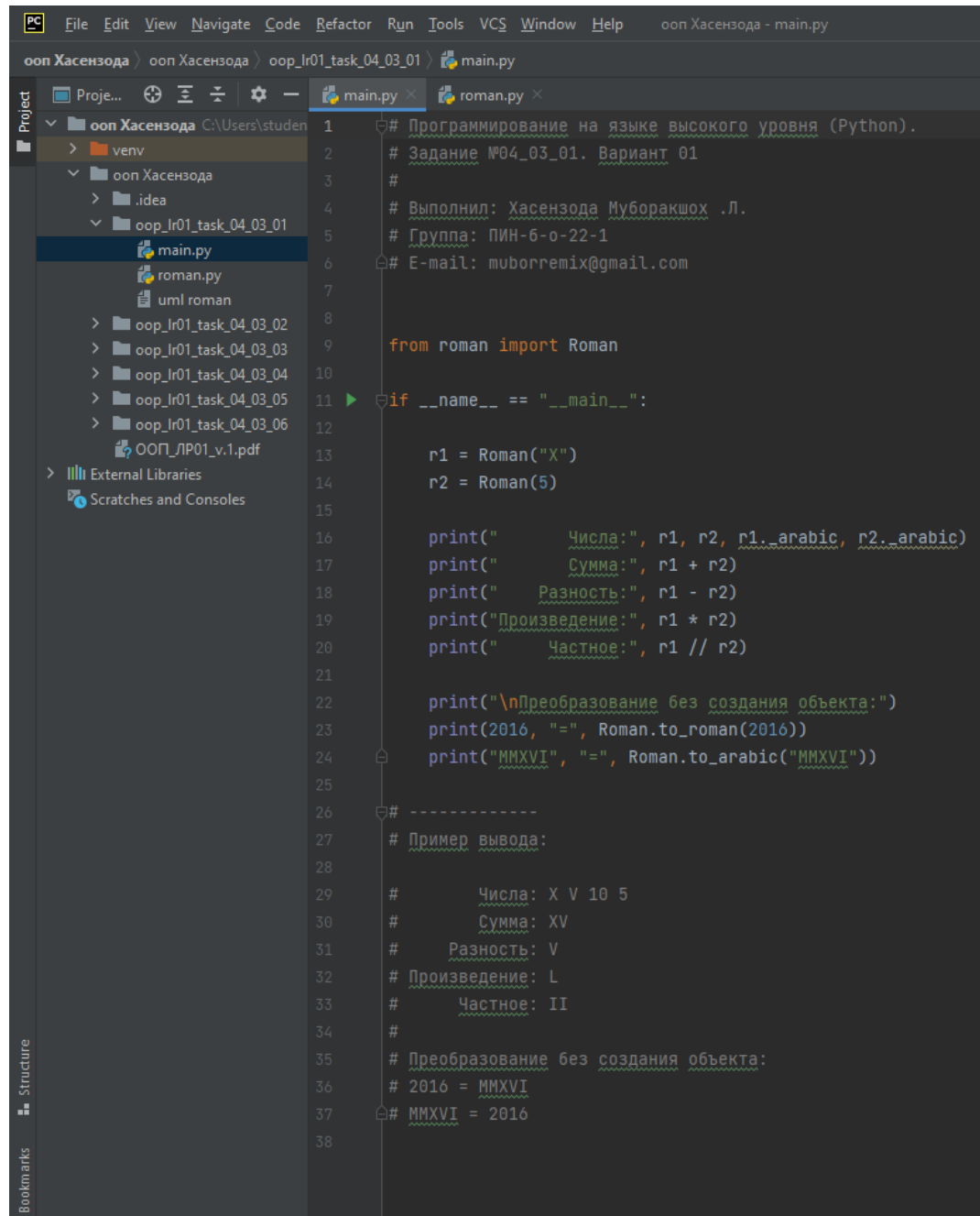
Ставрополь, 2024 г.

**Тема:** Основы объектно-ориентированного программирования на ЯП Python

**Цель работы:** изучить базовые понятия (классы, подклассы и методы)  
Реализовать фундаментальные принципы объектно-ориентированного программирования.

**Выполнение работы:**

#### 4.3.1. Римское число



The screenshot shows an IDE window with a project named 'oop Хасензода'. The project structure includes a 'venv' folder, a '.idea' folder, and a sub-project 'oop\_lr01\_task\_04\_03\_01' containing 'main.py', 'roman.py', and 'uml roman'. The 'main.py' file is open, showing the following code:

```
1  # Программирование на языке высокого уровня (Python).
2  # Задание №04_03_01. Вариант 01
3  #
4  # Выполнил: Хасензода Муборакшох .Л.
5  # Группа: ПИН-6-о-22-1
6  # E-mail: muborremix@gmail.com
7
8
9  from roman import Roman
10
11 if __name__ == "__main__":
12
13     r1 = Roman("X")
14     r2 = Roman(5)
15
16     print("    Числа:", r1, r2, r1._arabic, r2._arabic)
17     print("    Сумма:", r1 + r2)
18     print("    Разность:", r1 - r2)
19     print("    Произведение:", r1 * r2)
20     print("    Частное:", r1 // r2)
21
22     print("\nПреобразование без создания объекта:")
23     print(2016, "=", Roman.to_roman(2016))
24     print("MMXVI", "=", Roman.to_arabic("MMXVI"))
25
26 # -----
27 # Пример вывода:
28
29 #    Числа: X V 10 5
30 #    Сумма: XV
31 #    Разность: V
32 #    Произведение: L
33 #    Частное: II
34 #
35 # Преобразование без создания объекта:
36 # 2016 = MMXVI
37 # MMXVI = 2016
38
```

The screenshot shows an IDE window with the file explorer on the left and the code editor on the right. The file explorer shows a project structure with folders for tasks and files like `main.py`, `roman.py`, and `uml roman`. The code editor displays the `roman.py` file, which contains the implementation of the `Roman` class. The code includes comments in Russian, class attributes for min/max values, a list of Roman numeral symbols, and methods for initialization, checking, and arithmetic operations.

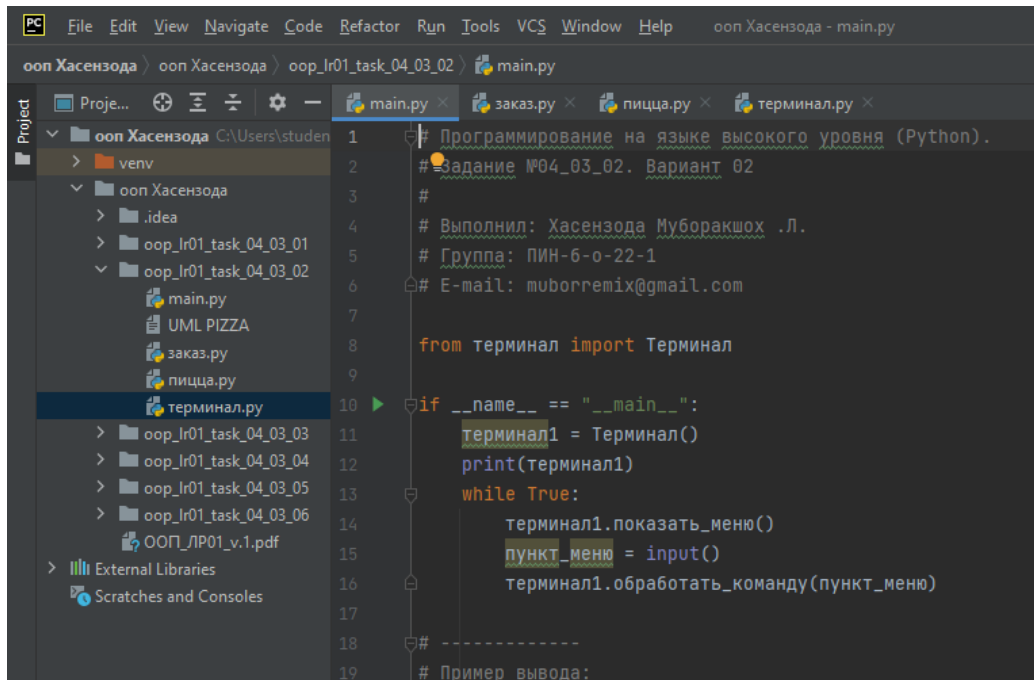
```
1  # Программирование на языке высокого уровня (Python).
2  # Задание №04_03_01. Вариант 01
3  #
4  # Выполнил: Хасензод Муборакшо .Л.
5  # Группа: ПИН-6-о-22-1
6  # E-mail: muborremix@gmail.com
7
8
9  class Roman:
10     ARABIC_MIN = 1
11     ARABIC_MAX = 3999
12     ROMAN_MIN = "I"
13     ROMAN_MAX = "MMMCMXCIX"
14
15     LETTERS = ["M", "CM", "D", "CD", "C", "XC", "L",
16               "XL", "X", "IX", "V", "IV", "I"]
17     NUMBERS = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]
18
19     def __init__(self, value):
20         if not isinstance(value, (int, str)):
21             raise TypeError("Cannot create Roman number from {}".format(type(value)))
22
23         if isinstance(value, int):
24             self.__check_arabic(value)
25             self._arabic = value
26         elif isinstance(value, str):
27             self.__check_roman(value)
28             self._arabic = self.to_arabic(value)
29
30     def __add__(self, other):
31         if isinstance(other, Roman):
32             result = self._arabic + other._arabic
33         elif isinstance(other, int):
34             result = self._arabic + other
35         else:
36             raise TypeError("Unsupported operand type for +: '{}' and '{}'".format(
37                 type(self).__name__, type(other).__name__))
38
39         return Roman(result)
```

The screenshot shows the same IDE window with the `uml roman` file open. It displays a UML class diagram for the `Roman` class. The diagram shows the class name, its attributes, and its methods. Below the diagram, there is a text description of the class and its attributes.

```
1  +-----+
2  |   Roman   |
3  +-----+
4  | - _arabic: int |
5  +-----+
6  | + __init__(value: Union[int, str]) |
7  | + __add__(other: Union[Roman, int]) |
8  | + __sub__(other: Union[Roman, int]) |
9  | + __mul__(other: Union[Roman, int]) |
10 | + __floordiv__(other: Union[Roman, int]) |
11 | + __truediv__(other: Union[Roman, int]) |
12 | + __str__() |
13 | + to_arabic(roman: str) |
14 | + to_roman(arabic: int) |
15 +-----+
16
17 Класс Roman представляет римское число и предоставляет методы для работы с ним.
18
19 Атрибуты класса:
20
21 _arabic - целочисленное значение римского числа.
```

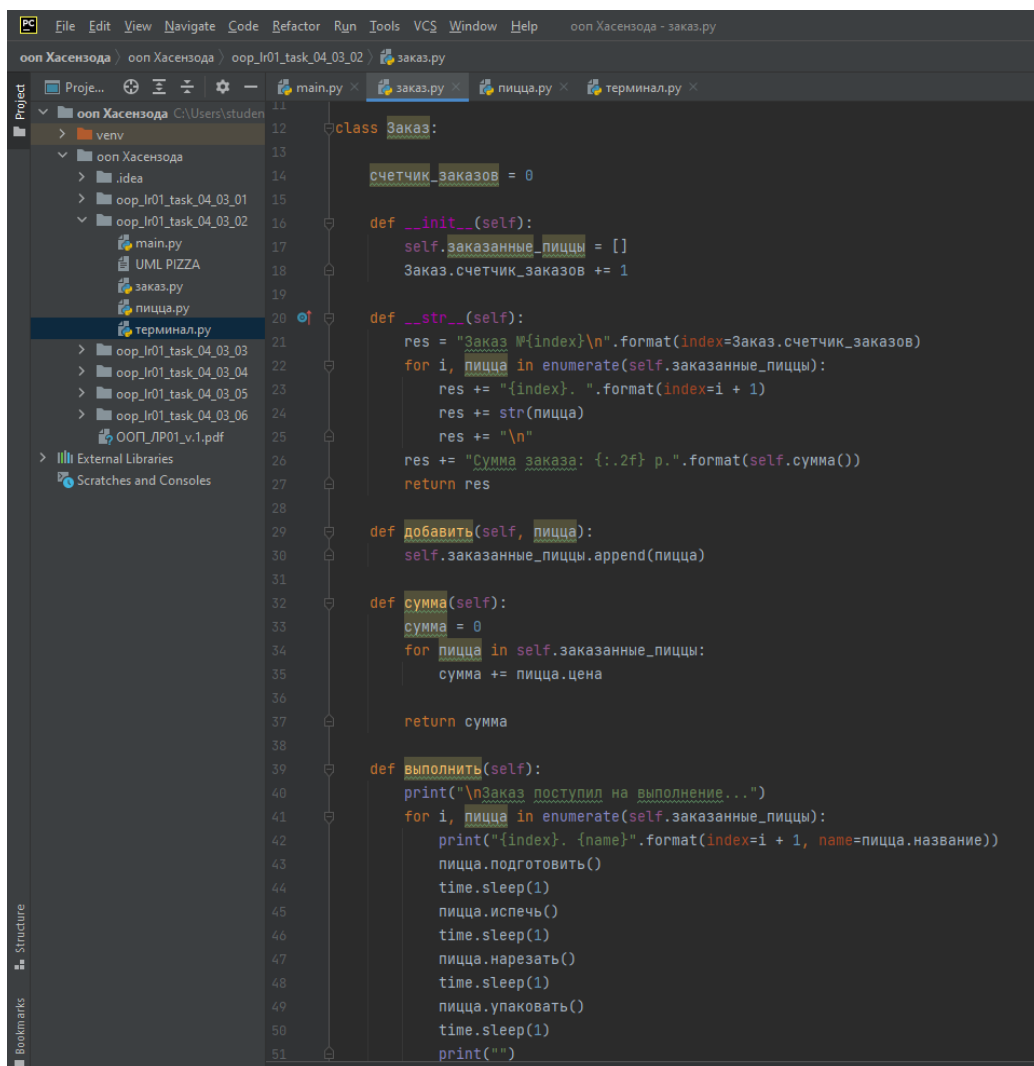
## UML-диаграмма

### 4.3.2. Пиццерия



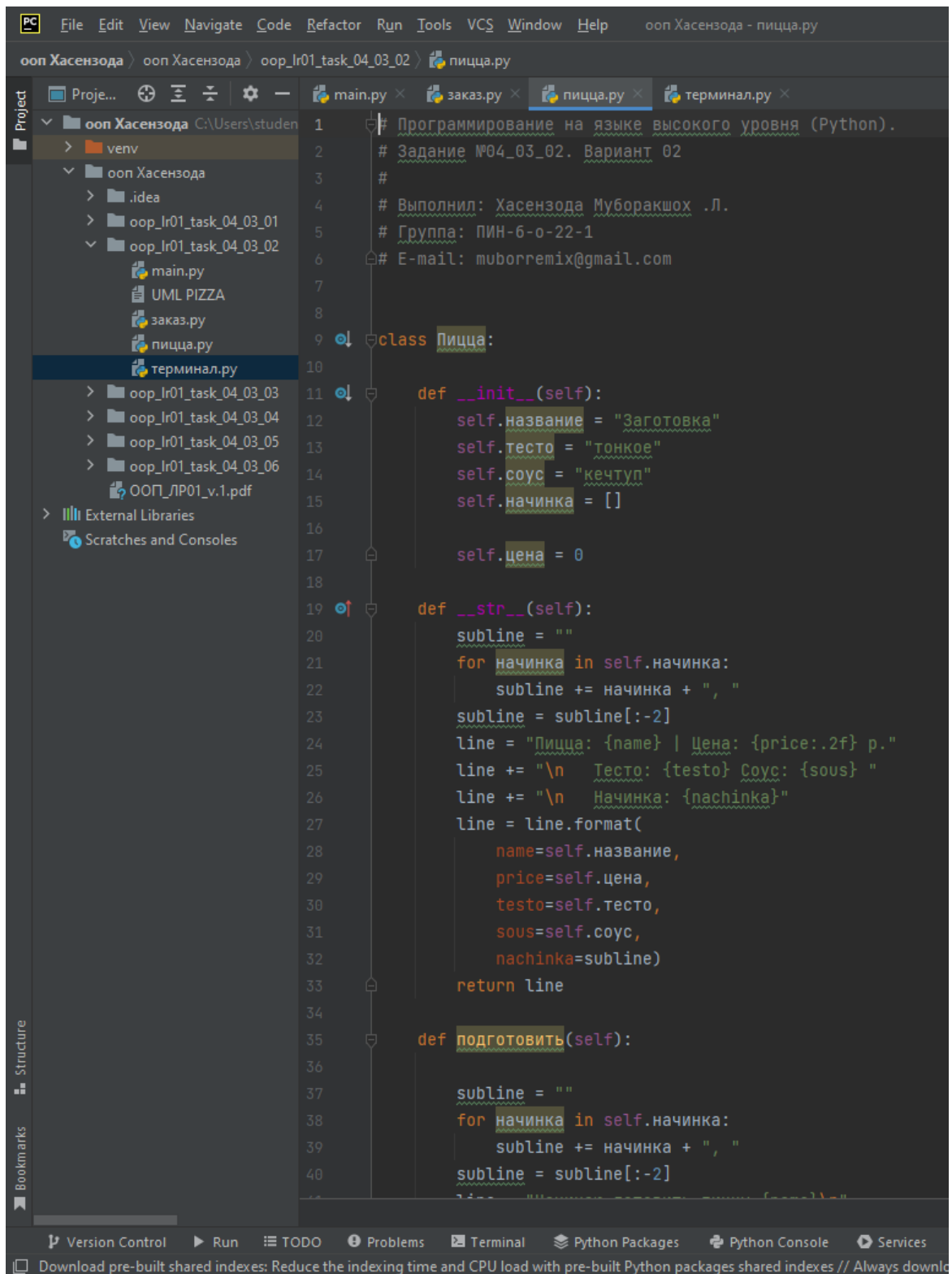
The screenshot shows an IDE with the 'main.py' file open. The project structure on the left includes a 'venv' directory and several task folders. The code in 'main.py' is a Python script that interacts with a 'Терминал' (Terminal) object. It includes comments in Russian and a loop that repeatedly shows a menu and processes commands.

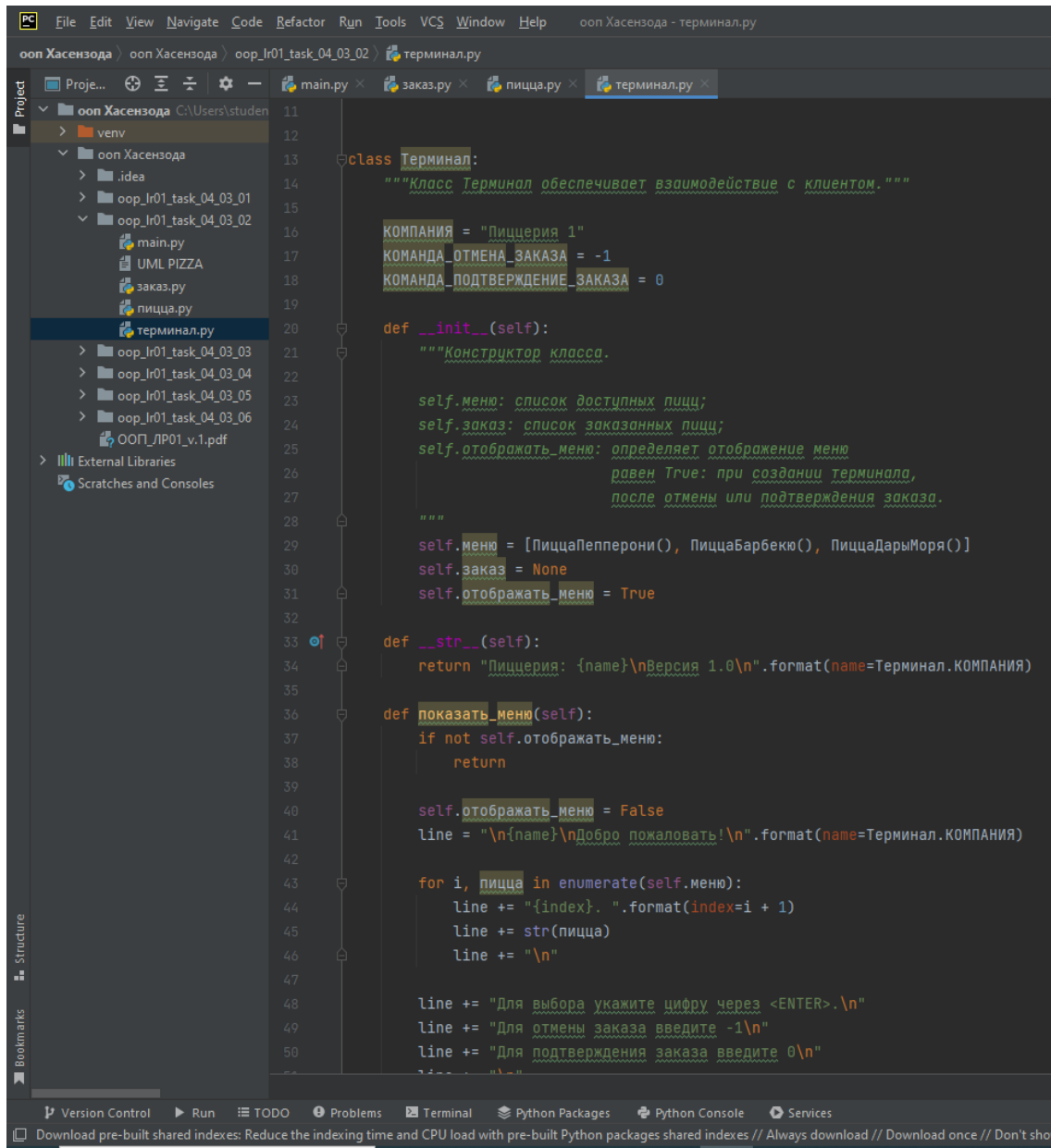
```
1  Программирование на языке высокого уровня (Python).
2  # задание №04_03_02. Вариант 02
3  #
4  # Выполнил: Хасензода Муборакшох .Л.
5  # Группа: ПИН-6-о-22-1
6  # E-mail: muborremix@gmail.com
7
8  from терминал import Терминал
9
10 if __name__ == "__main__":
11     терминал1 = Терминал()
12     print(терминал1)
13     while True:
14         терминал1.показать_меню()
15         пункт_меню = input()
16         терминал1.обработать_команду(пункт_меню)
17
18 # -----
19 # Пример вывода:
```

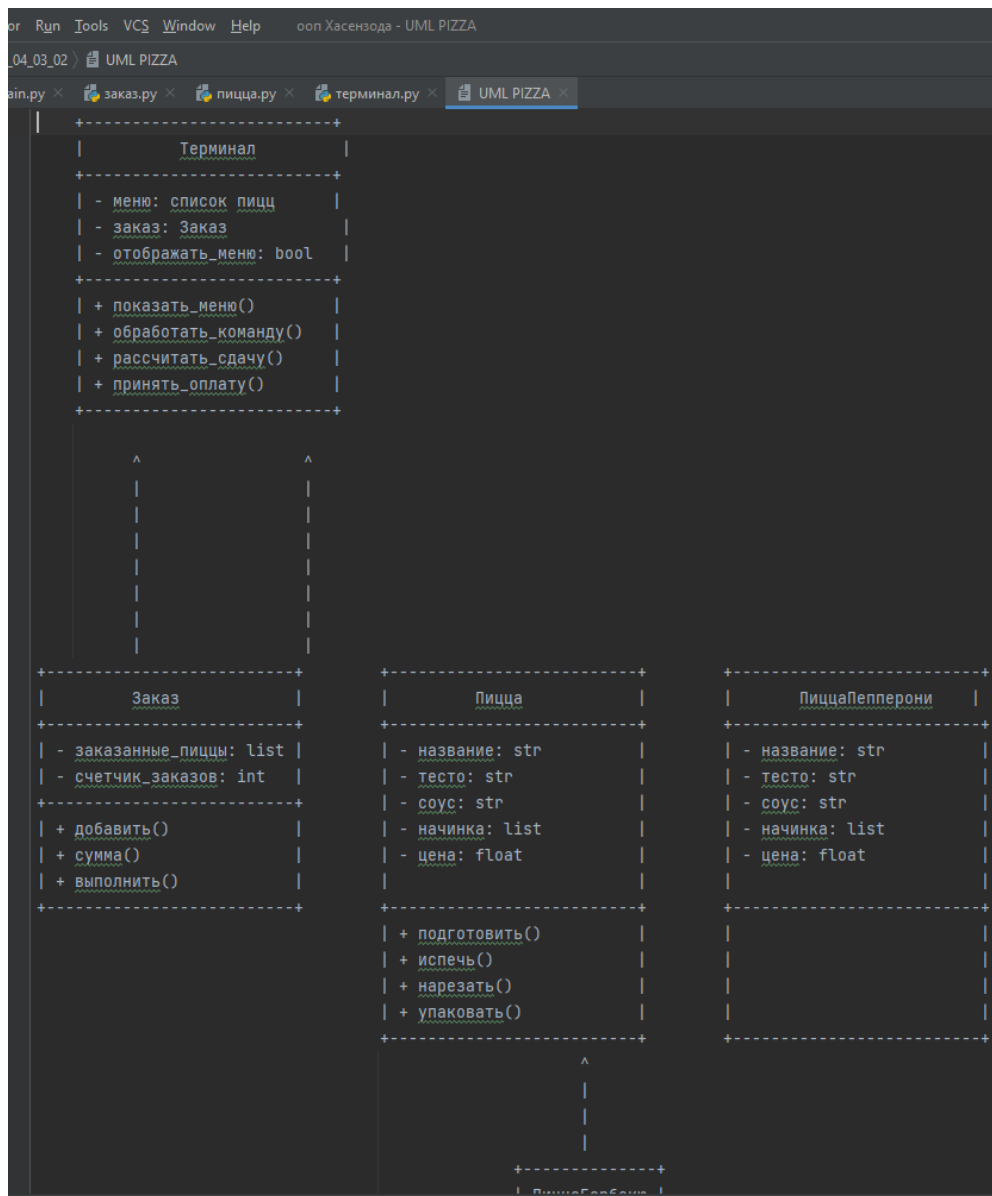


The screenshot shows an IDE with the 'заказ.py' file open. The project structure on the left is the same as the previous screenshot. The code in 'заказ.py' defines a 'Заказ' (Order) class with methods for initializing, displaying, adding items, calculating the total, and executing the order.

```
11 class Заказ:
12     счетчик_заказов = 0
13
14     def __init__(self):
15         self.заказанные_пиццы = []
16         Заказ.счетчик_заказов += 1
17
18     def __str__(self):
19         res = "Заказ №{index}\n".format(index=Заказ.счетчик_заказов)
20         for i, пицца in enumerate(self.заказанные_пиццы):
21             res += "{index}. ".format(index=i + 1)
22             res += str(пицца)
23             res += "\n"
24         res += "Сумма заказа: {:.2f} p.".format(self.сумма())
25         return res
26
27     def добавить(self, пицца):
28         self.заказанные_пиццы.append(пицца)
29
30     def сумма(self):
31         сумма = 0
32         for пицца in self.заказанные_пиццы:
33             сумма += пицца.цена
34         return сумма
35
36     def выполнить(self):
37         print("\nЗаказ поступил на выполнение...")
38         for i, пицца in enumerate(self.заказанные_пиццы):
39             print("{index}. {name}".format(index=i + 1, name=пицца.название))
40             пицца.подготовить()
41             time.sleep(1)
42             пицца.испечь()
43             time.sleep(1)
44             пицца.нарезать()
45             time.sleep(1)
46             пицца.упаковать()
47             time.sleep(1)
48         print("")
```







**UML-диаграмма pizza**

### 4.3.3. Банковские вклады

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help oop Хасензода - main.py
oop Хасензода oop Хасензода oop_lr01_task_04_03_03 main.py
Project
  oop Хасензода
    venv
      oop Хасензода
        .idea
        oop_lr01_task_04_03_01
        oop_lr01_task_04_03_02
        oop_lr01_task_04_03_03
          deposit.py
          Deposit uml
          main.py
        oop_lr01_task_04_03_04
        oop_lr01_task_04_03_05
        oop_lr01_task_04_03_06
        ООП_ЛР01_v.1.pdf
    External Libraries
    Scratches and Consoles
Structure
Bookmarks

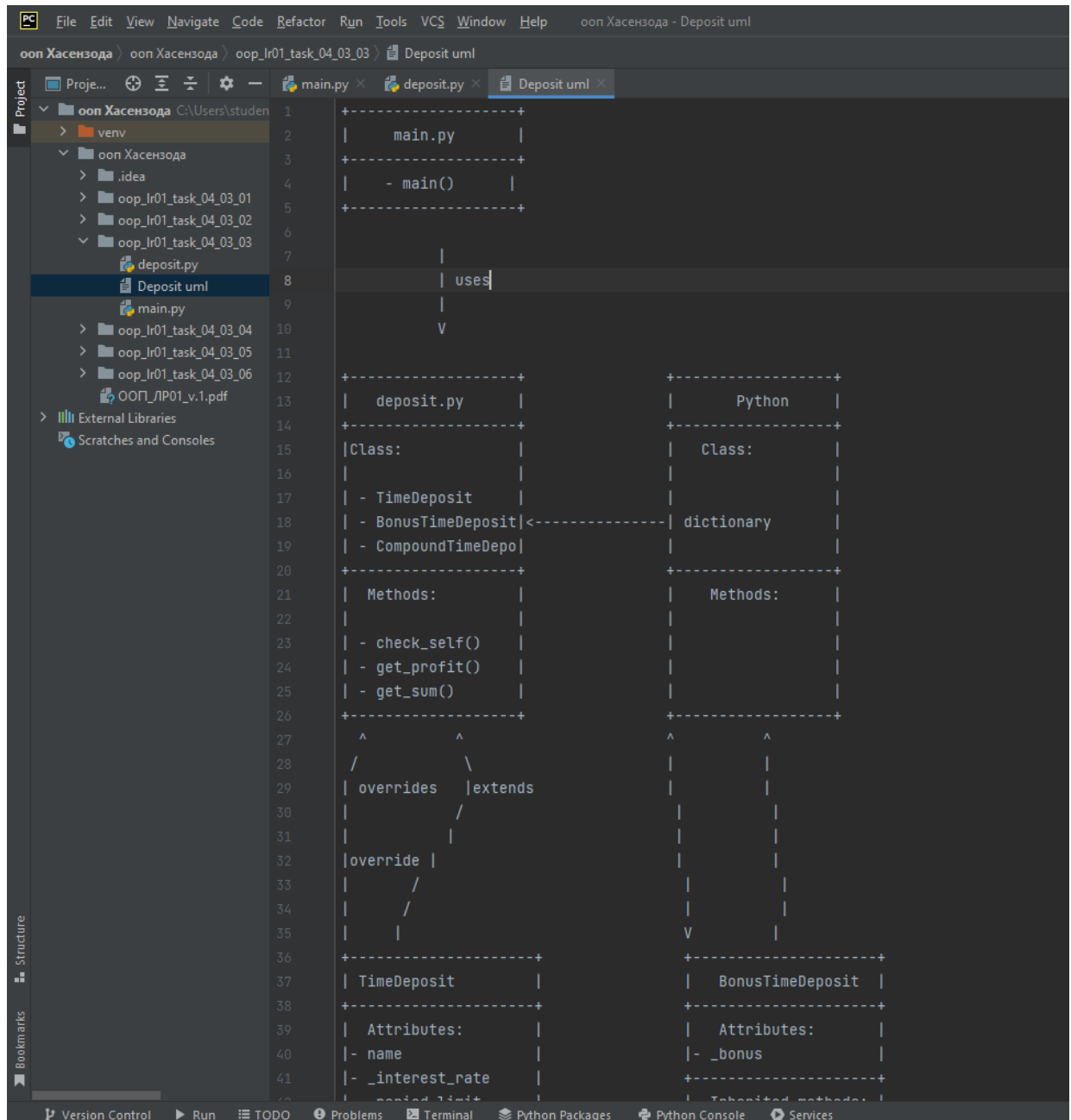
1 1# Программирование на языке высокого уровня (Python).
2 2# Задание № 04_03_03. Вариант 03
3 3#
4 4# Выполнил: Хасензода Муборакшох .Л.
5 5# Группа: ПИН-6-о-22-1
6 6# E-mail: muborremix@gmail.com1
7
8
9 from deposit import deposits
10
11 if __name__ == "__main__":
12     print("Добро пожаловать в систему подбора вкладов!")
13
14     while True:
15         print("\n----")
16         print("Нажмите 1, чтобы подобрать вклад, или что угодно для выхода.")
17
18         answer = input()
19         if answer == "1":
20
21             initial_sum = float(input("1/2: Введите начальную сумму вклада: "))
22             period = int(input("2/2: Введите срок вклада (мес.): "))
23
24             matched_deposits = []
25             for deposit in deposits:
26                 try:
27                     deposit._check_user_params(initial_sum, period)
28                     matched_deposits.append(deposit)
29                 except AssertionError as err:
30                     pass
31
32             if len(matched_deposits) > 0:
33                 print("{0:18} | {1:13} | {2:13}".format(
34                     "Вклад", "Прибыль", "Итоговая сумма"
35                 ))
36                 for deposit in matched_deposits:
37                     print("{0:18} | {1:8,.2f} | {3:4} | {2:8,.2f} | {3:4}".format(
38                         deposit.name,
39                         deposit.get_profit(initial_sum, period),
40                         deposit.get_sum(initial_sum, period),
41                         deposit.sum_limit
42                     ))
```

```
Project
  oop Хасензода
    venv
      oop Хасензода
        .idea
        oop_lr01_task_04_03_01
        oop_lr01_task_04_03_02
        oop_lr01_task_04_03_03
          deposit.py
          Deposit uml
          main.py
        oop_lr01_task_04_03_04
        oop_lr01_task_04_03_05
        oop_lr01_task_04_03_06
        ООП_ЛР01_v.1.pdf
    External Libraries
    Scratches and Consoles
Structure
Bookmarks

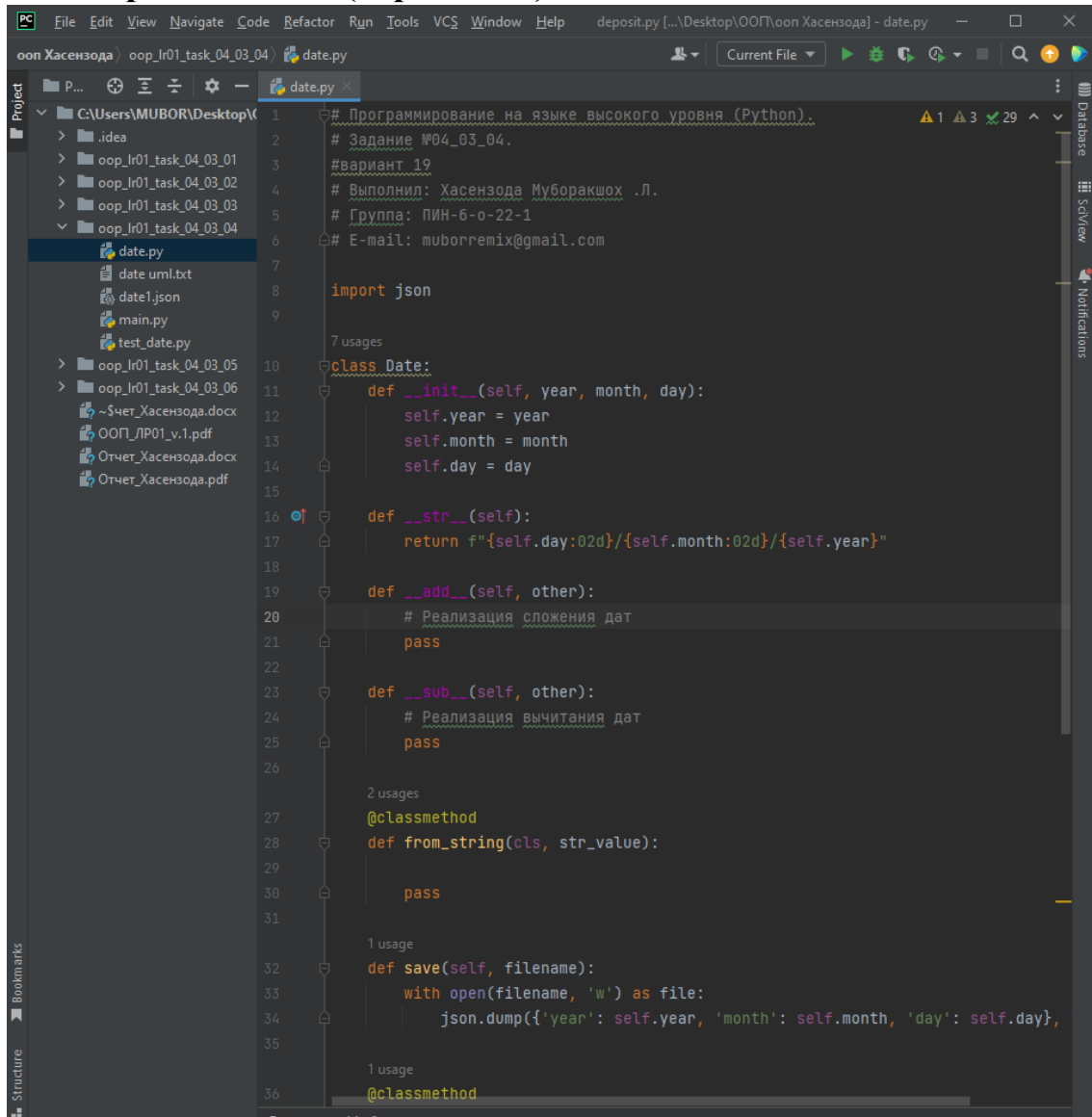
7
8
9 class TimeDeposit:
10     """Абстрактный класс - срочный вклад.
11
12     https://ru.wikipedia.org/wiki/Срочный_вклад.
13
14     Поля:
15     - self.name (str): наименование;
16     - self._interest_rate (float): процент по вкладу (0; 100];
17     - self._period_limit (tuple (int, int)):
18         допустимый срок вклада в месяцах (от; до);
19     - self._sum_limit (tuple (float, float)):
20         допустимая сумма вклада (от; до).
21
22     Свойства:
23     - self.currency (str): знак/наименование валюты.
24
25     Методы:
26     - self._check_self(initial_sum, period): проверяет соответствие данных
27         ограничениям вклада;
28     - self.get_profit(initial_sum, period): возвращает прибыль по вкладу;
29     - self.get_sum(initial_sum, period):
30         возвращает сумму по окончании вклада.
31
32     """
33
34     def __init__(self, name, interest_rate, period_limit, sum_limit):
35         self.name = name
36         self._interest_rate = interest_rate
37         self._period_limit = period_limit
38         self._sum_limit = sum_limit
39         self._check_self()
40
41     def __str__(self):
42
43         number_list_str = list(self._sum_limit)
44         for i, number_str in enumerate(number_list_str):
45             number_str = str(number_str)
46             new_line = ""
47             while len(number_str) > 3:
48                 new_line += " " + number_str[len(number_str)-3:]
49                 number_str = number_str[:-3]
```



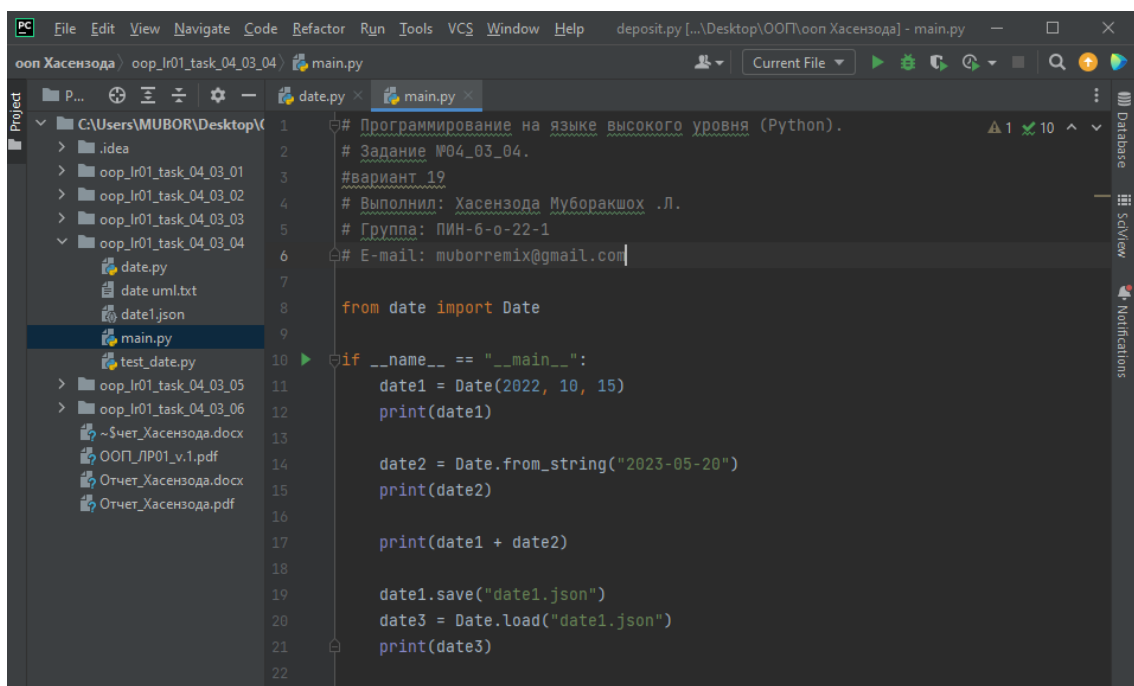
## UML-диаграмма депозит



### 4.3.4. Простой класс (вариант 19)



```
1  # Программирование на языке высокого уровня (Python).
2  # Задание №04_03_04.
3  # вариант 19
4  # Выполнил: Хасензода Муборакшоҳ .Л.
5  # Группа: ПИН-6-о-22-1
6  # E-mail: muborremix@gmail.com
7
8  import json
9
10 class Date:
11     def __init__(self, year, month, day):
12         self.year = year
13         self.month = month
14         self.day = day
15
16     def __str__(self):
17         return f"{self.day:02d}/{self.month:02d}/{self.year}"
18
19     def __add__(self, other):
20         # Реализация сложения дат
21         pass
22
23     def __sub__(self, other):
24         # Реализация вычитания дат
25         pass
26
27     @classmethod
28     def from_string(cls, str_value):
29
30         pass
31
32     def save(self, filename):
33         with open(filename, 'w') as file:
34             json.dump({'year': self.year, 'month': self.month, 'day': self.day},
35
36
37     @classmethod
```



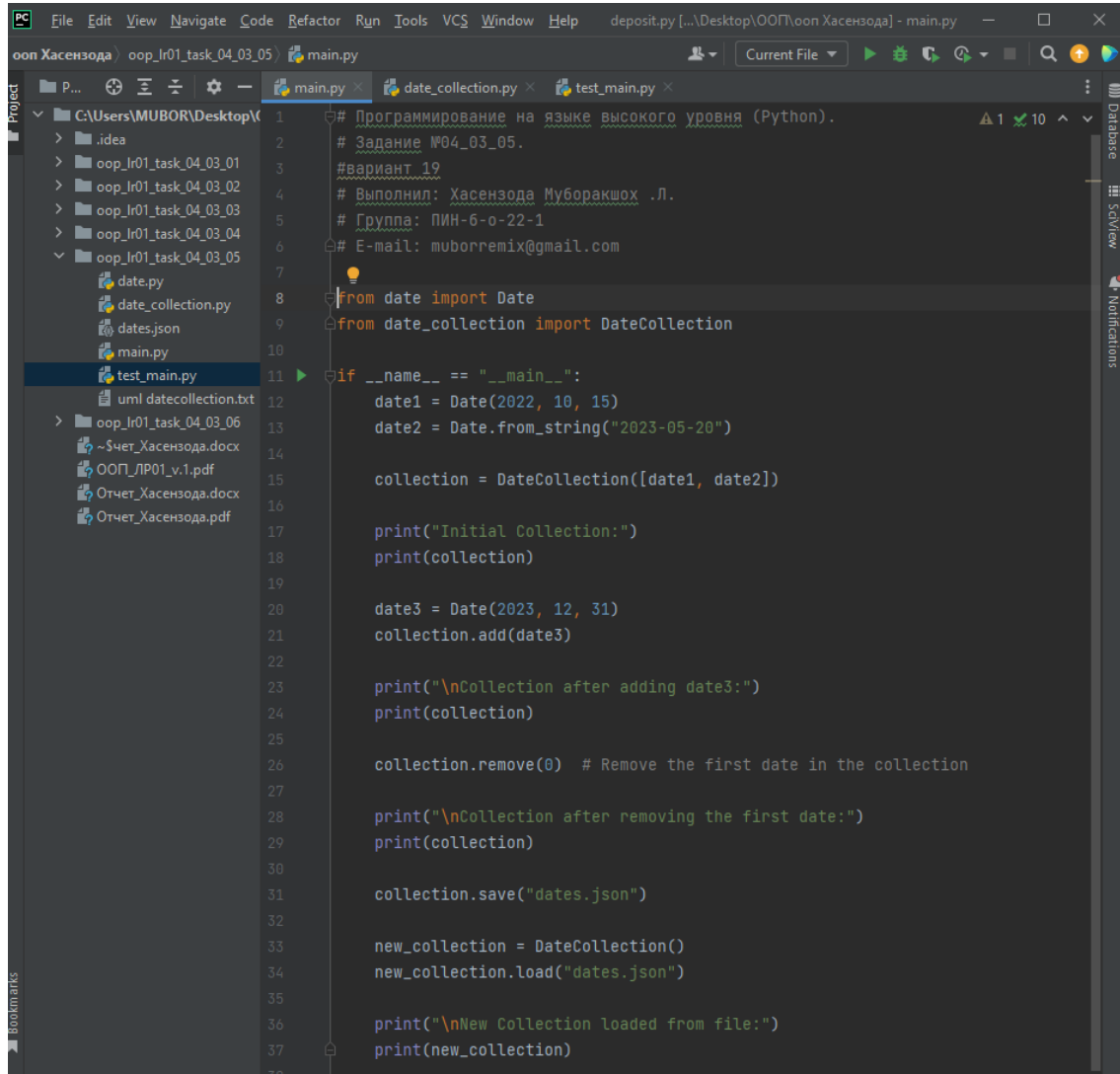
```
1  # Программирование на языке высокого уровня (Python).
2  # Задание №04_03_04.
3  # вариант 19
4  # Выполнил: Хасензода Муборакшоҳ .Л.
5  # Группа: ПИН-6-о-22-1
6  # E-mail: muborremix@gmail.com
7
8  from date import Date
9
10 if __name__ == "__main__":
11     date1 = Date(2022, 10, 15)
12     print(date1)
13
14     date2 = Date.from_string("2023-05-20")
15     print(date2)
16
17     print(date1 + date2)
18
19     date1.save("date1.json")
20     date3 = Date.load("date1.json")
21     print(date3)
22
```

```
1 # Программирование на языке высокого уровня (Python).
2 # Задание №04_03_04.
3 #вариант 19
4 # Выполнил: Хасензода Муборакшоҳ .Л.
5 # Группа: ПИН-6-о-22-1
6 # E-mail: muborremix@gmail.com
7
8 import ...
9
10
11 class TestDateMethods(unittest.TestCase):
12     def test_constructor(self):
13         date = Date(2022, 10, 15)
14         self.assertEqual(str(date), "15/10/2022")
15
16     def test_from_string(self):
17         date = Date.from_string("2023-05-20")
18         self.assertEqual(str(date), "20/05/2023")
19
20 # Другие методы тестирования
21
22 if __name__ == '__main__':
23     unittest.main()
```

## Uml диаграмма:

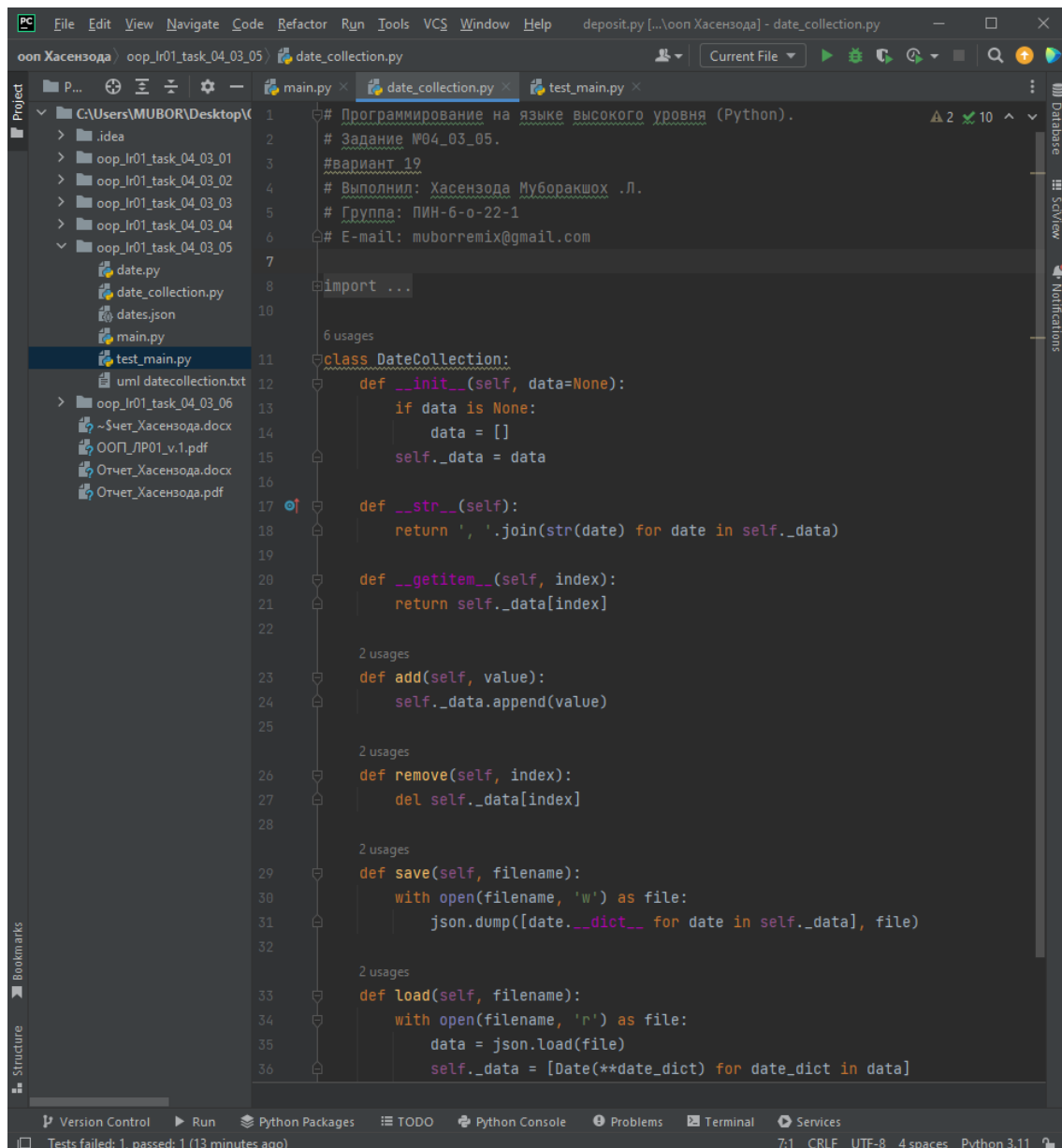
```
1 -----
2 |           Date           |
3 |-----|
4 | - year: int               |
5 | - month: int              |
6 | - day: int                |
7 |-----|
8 | + __init__(year, month, day) |
9 | + __str__(): string         |
10 | + __add__(other): Date      |
11 | + __sub__(other): Date      |
12 | + from_string(str_value): Date|
13 | + save(filename): void      |
14 | + load(filename): Date      |
15 | + is_valid(): bool          |
16 | + is_weekend(): bool        |
17 | + days_until(other_date): int|
18 |-----|
```

## 4.3.5. Класс-контейнер



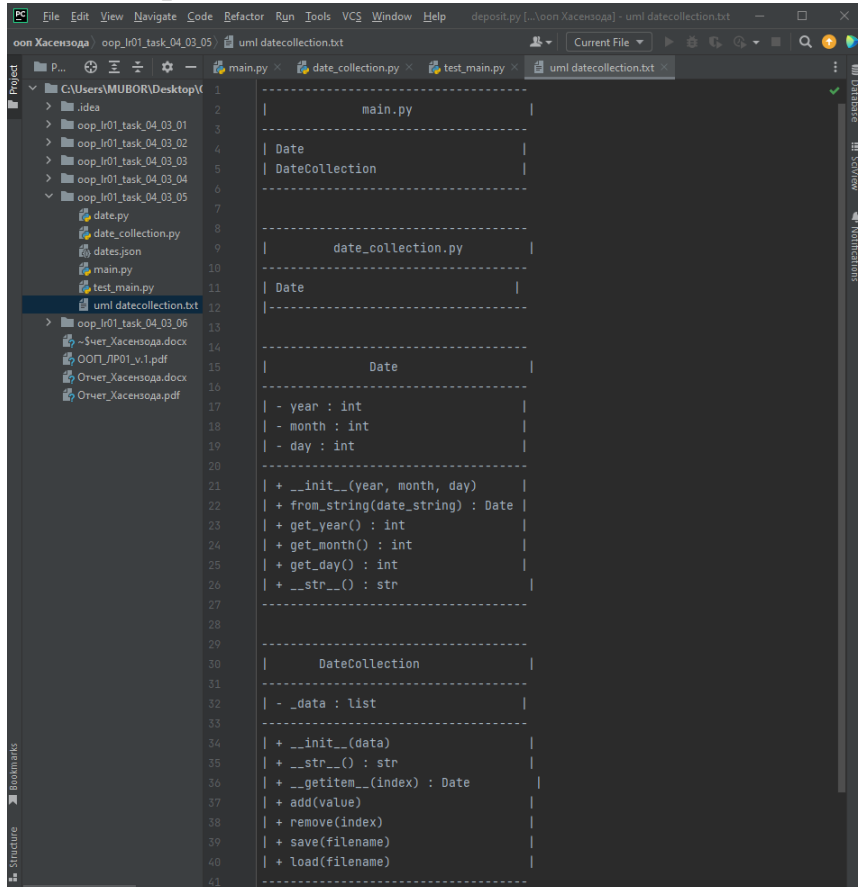
The screenshot shows an IDE window with a Python file named `main.py`. The code implements a `DateCollection` class and a `Date` class. The `DateCollection` class is a container for `Date` objects, supporting methods for adding, removing, and saving/loading data to a JSON file. The `Date` class represents a date and time.

```
1  # Программирование на языке высокого уровня (Python).
2  # Задание №04_03_05.
3  # вариант 19
4  # Выполнил: Хасензод Муборакшох .Л.
5  # Группа: ПИН-6-о-22-1
6  # E-mail: muborremix@gmail.com
7
8  from date import Date
9  from date_collection import DateCollection
10
11 if __name__ == "__main__":
12     date1 = Date(2022, 10, 15)
13     date2 = Date.from_string("2023-05-20")
14
15     collection = DateCollection([date1, date2])
16
17     print("Initial Collection:")
18     print(collection)
19
20     date3 = Date(2023, 12, 31)
21     collection.add(date3)
22
23     print("\nCollection after adding date3:")
24     print(collection)
25
26     collection.remove(0) # Remove the first date in the collection
27
28     print("\nCollection after removing the first date:")
29     print(collection)
30
31     collection.save("dates.json")
32
33     new_collection = DateCollection()
34     new_collection.load("dates.json")
35
36     print("\nNew Collection loaded from file:")
37     print(new_collection)
```

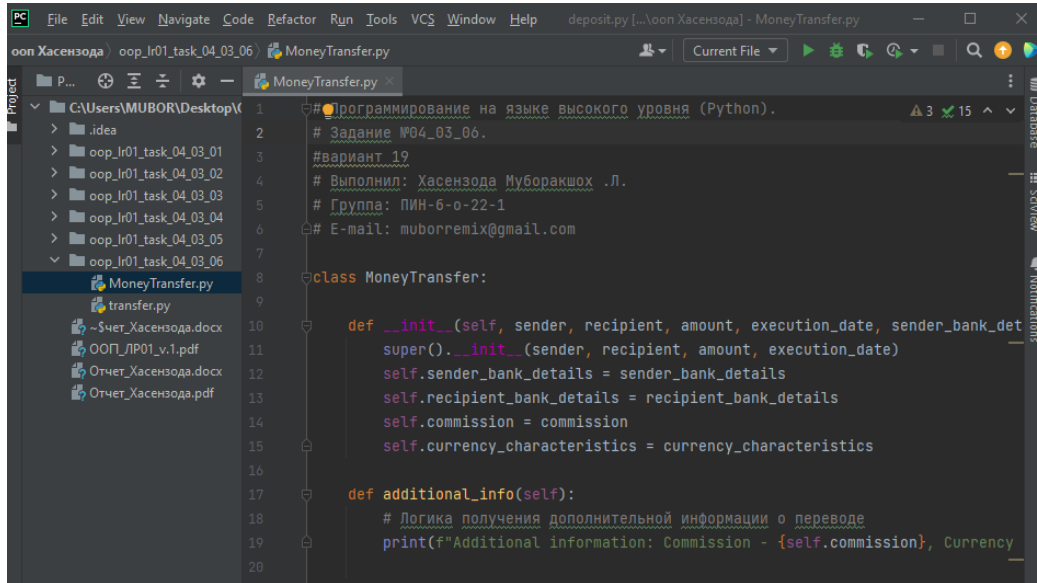


```
1  # Программирование на языке высокого уровня (Python).
2  # Задание №04_03_05.
3  # вариант 19
4  # Выполнил: Хасензода Муборакшоҳ .Л.
5  # группа: ПИН-6-о-22-1
6  # E-mail: muborremix@gmail.com
7
8  import ...
9
10
11 class TestDateCollection(unittest.TestCase):
12     def test_date_collection_operations(self):
13         date1 = Date(2022, 10, 15)
14         date2 = Date.from_string("2023-05-20")
15         date3 = Date(2023, 12, 31)
16
17         collection = DateCollection([date1, date2])
18
19         self.assertEqual(str(collection), "2022-10-15, 2023-05-20")
20
21         collection.add(date3)
22
23         self.assertEqual(str(collection), "2022-10-15, 2023-05-20, 2023-12-31")
24
25         collection.remove(0)
26
27         self.assertEqual(str(collection), "2023-05-20, 2023-12-31")
28
29         collection.save("test_dates.json")
30
31         new_collection = DateCollection()
32         new_collection.load("test_dates.json")
33
34         self.assertEqual(str(new_collection), "2023-05-20, 2023-12-31")
35
36 if __name__ == '__main__':
37     unittest.main()
```

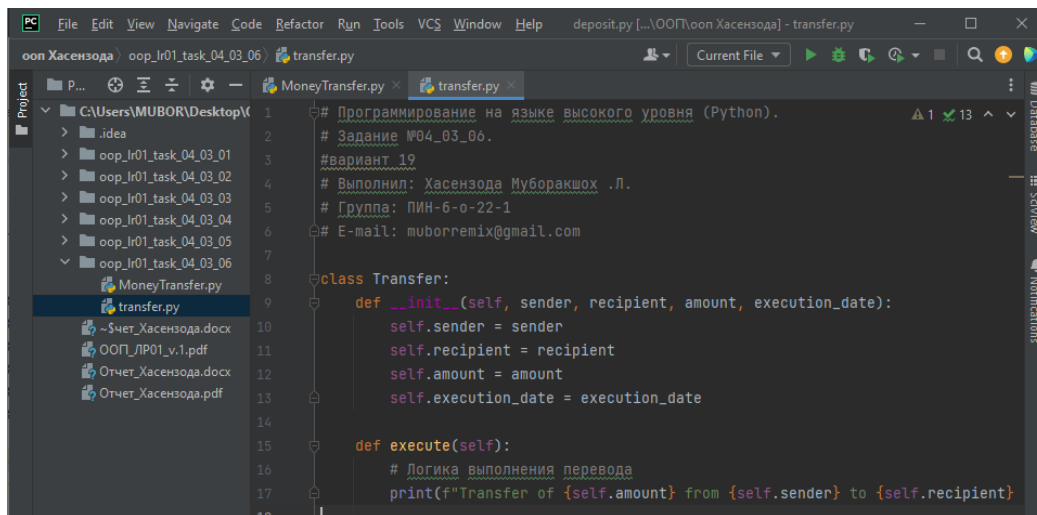
## Uml диаграмма:



## 4.3.6. Иерархия классов



```
1 # Программирование на языке высокого уровня (Python).
2 # Задание №04_03_06.
3 # вариант 19
4 # Выполнил: Хасензода Муборакшо .Л.
5 # Группа: ПИН-6-о-22-1
6 # E-mail: muborremix@gmail.com
7
8 class MoneyTransfer:
9
10     def __init__(self, sender, recipient, amount, execution_date, sender_bank_details, recipient_bank_details, commission, currency_characteristics):
11         super().__init__(sender, recipient, amount, execution_date)
12         self.sender_bank_details = sender_bank_details
13         self.recipient_bank_details = recipient_bank_details
14         self.commission = commission
15         self.currency_characteristics = currency_characteristics
16
17     def additional_info(self):
18         # Логика получения дополнительной информации о переводе
19         print(f"Additional information: Commission - {self.commission}, Currency
```



```
1 # Программирование на языке высокого уровня (Python).
2 # Задание №04_03_06.
3 # вариант 19
4 # Выполнил: Хасензода Муборакшо .Л.
5 # Группа: ПИН-6-о-22-1
6 # E-mail: muborremix@gmail.com
7
8 class Transfer:
9
10     def __init__(self, sender, recipient, amount, execution_date):
11         self.sender = sender
12         self.recipient = recipient
13         self.amount = amount
14         self.execution_date = execution_date
15
16     def execute(self):
17         # Логика выполнения перевода
18         print(f"Transfer of {self.amount} from {self.sender} to {self.recipient}
```