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

«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ И ПРОГРАММИРОВАНИЕ»

на тему

«Основы объектно ориентированного программирования на ЯП Python.»

**Выполнил:**

Дякин Владимир Дмитриевич

Студент 2 курса группы ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

**Руководитель работы:**

Щёголев А. А.

(ФИО, должность, кафедра)

Ставрополь, 2023 г.

**Цель работы:** изучить базовые понятия (классы, подклассы и методы)

Реализовать фундаментальные принципы объектно-ориентированного

программирования.

**Ход работы**

№ 4.3.1 Листинг приведён в файле

[main.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_01/main.py)

[roman.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_01/roman.py)

[test.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_01/test.py)

В модуле *roman.py* были доработаны несколько методов у класса *Roman*, включая обработку некорректных входных данных. В основном приложении реализован простой тест работоспособности класса *Roman.* Ниже представлена диаграмма класса *Roman* (Рисунок 1 – UML диаграмма класса *Roman*)

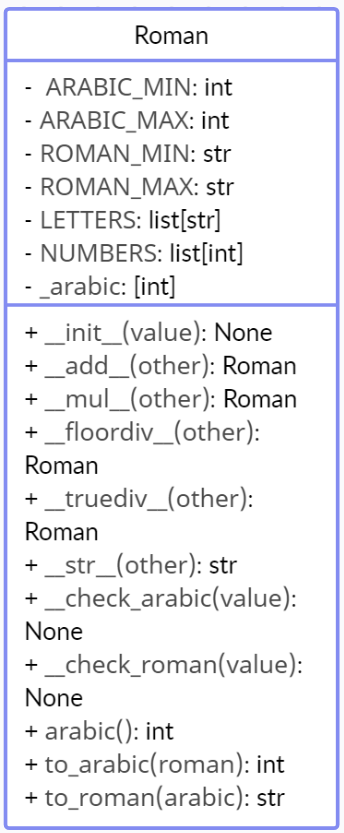
**

Рисунок 1 - Рисунок 1 – UML диаграмма класса *Roman*

№ 4.3.2 Листинг приведён в файле

[main.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_02/main.py)

[заказ.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_02/%D0%B7%D0%B0%D0%BA%D0%B0%D0%B7.py)

[пицца.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_02/%D0%BF%D0%B8%D1%86%D1%86%D0%B0.py)

[терминал.py   
test.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_02/test.py)

Реализованы модули *заказ.ру, пицца.ру, терминал.ру.* В их классах были реализованы методы и связи между друг другом. В основном приложении реализован алгоритм пиццерии, указанный в задании. Ниже представлена диаграмма классов (Рисунок 2 – UML диаграмма программы для заказа пиццы)

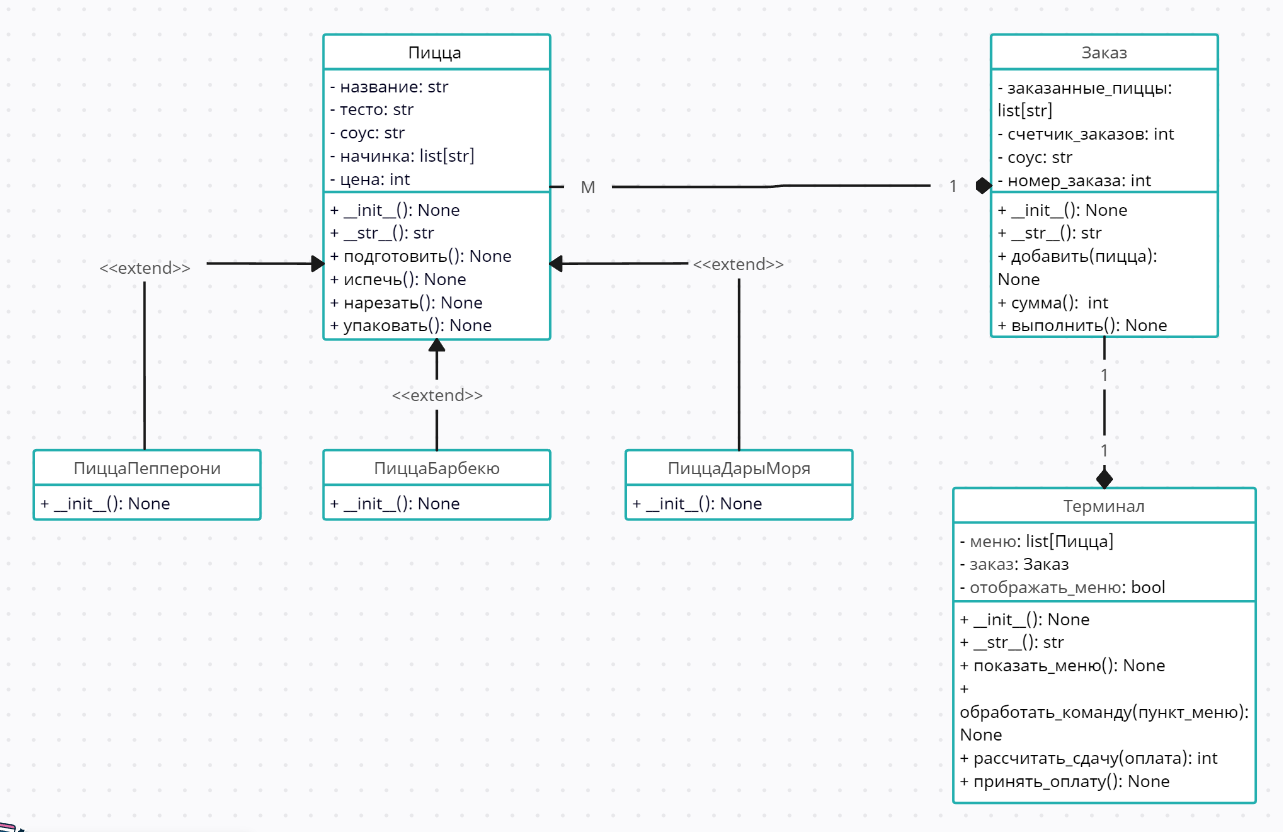


Рисунок 2 – UML диаграмма программы для заказа пиццы

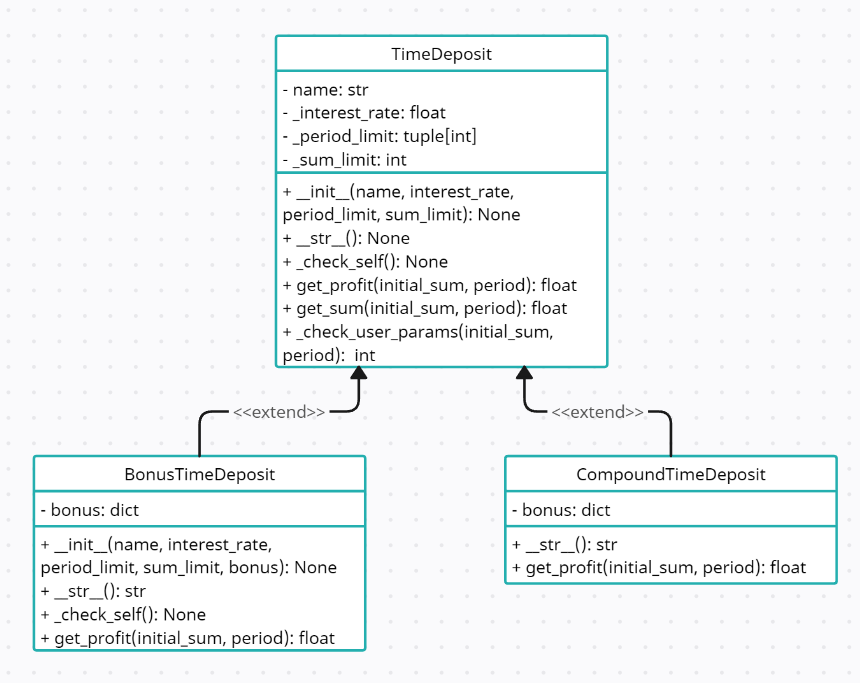
№ 4.3.3 Листинг приведён в файле

[main.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_03/main.py)

[deposit.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_03/deposit.py)

[test.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_02/test.py)

Реализован базовый класс *TimeDeposit,* на его основе были реализованы ещё два класса: *BonusTimeDeposit* и *CompoundTimeDeposit.* В основном приложении реализован алгоритм для вычисления прибыли от каждого из вкладов на заданный срок и заданную сумму. Ниже представлена диаграмма классов (Рисунок 3 – UML диаграмма программы для работы с депозитами)



(Рисунок 3 – UML диаграмма программы для работы с депозитами

№ 4.3.4 Листинг приведён в файле

[main.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_04/main.py)

[my\_tyme.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_04/my_time.py)

[test.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_04/test.py)

Реализован базовый класс *Time* для работы со временем в строковом формате*.* Он имеет возможность сохранять и загружать объект в json файл. Ниже представлена диаграмма класса *Time* (Рисунок 4 – UML диаграмма класса *Time*)

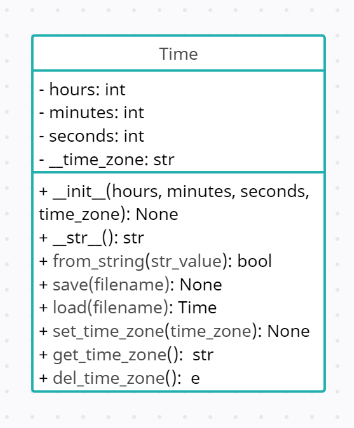


Рисунок 4 – UML диаграмма класса *Time*

№ 4.3.5 Листинг приведён в файле

[main.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_05/main.py)

[timeContainer.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_05/timeContainer.py)

[my\_time.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_05/my_time.py)

[test.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_05/test.py)

Реализован базовый класс *TimeContainer* для работы со объектами класса Time*.* Мы имеем возможность сохранять и загружать объект класса *TimeContainer* в json файл. Ниже представлена диаграмма классов *Time* и *TimeContainer (*Рисунок 5 – UML диаграмма классов *Time* и *TimeContainer)*

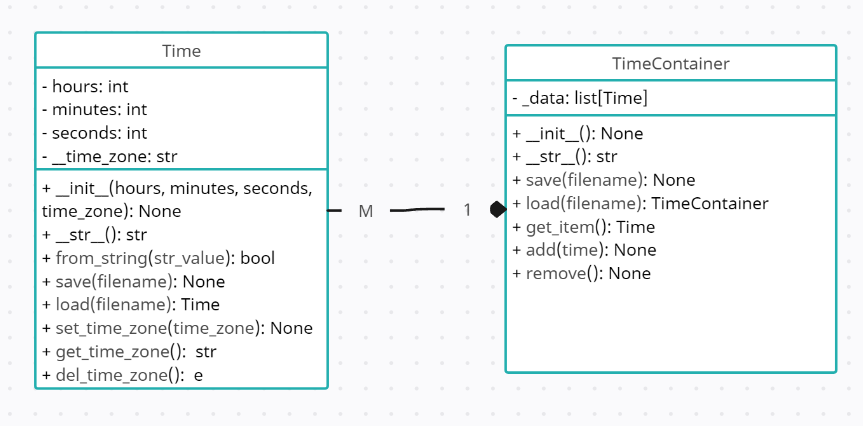


Рисунок 5 – UML диаграмма классов *Time* и *TimeContainer*

№ 4.3.6 Листинг приведён в файле

[main.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_06/main.py)

[bilet.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_06/bilet.py)

[test.py](https://github.com/vremennyj/OOP/blob/main/1/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/task_04_03_06/test.py)

Реализован базовые классы *Promo* идля работы со объектами класса Time*.* Мы имеем возможность сохранять и загружать объект класса *TimeContainer* в json файл. Ниже представлена диаграмма программы для работы с билетами *(*Рисунок 6 – UML диаграмма программы для работы с билетами*)*

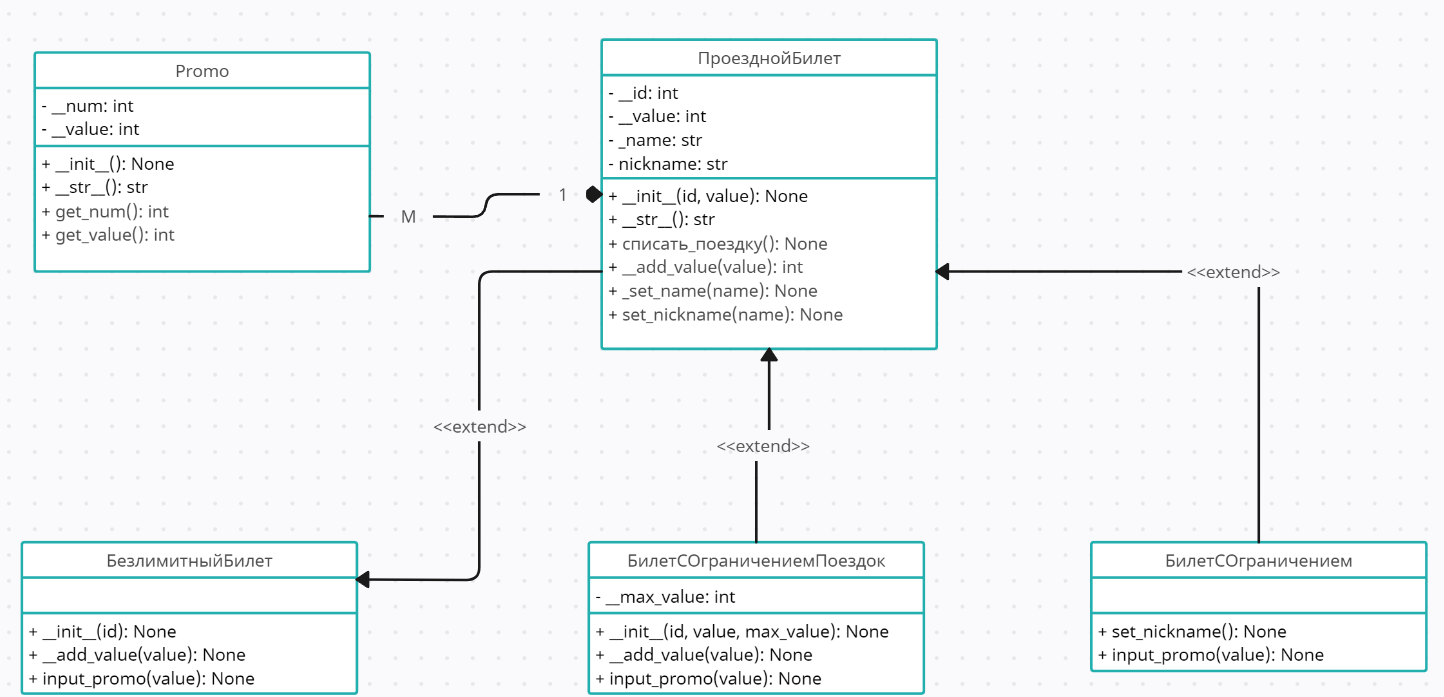


Рисунок 6 – UML диаграмма программы для работы с билетами

**Вывод:** изучены три основных парадигмы ООП, также на практике изучены методы и приёмы реализации собственных классов и последующей работы с ними. С помощью UML диаграмм мы визуализировали структуру созданных классов.