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

roman.py

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*)

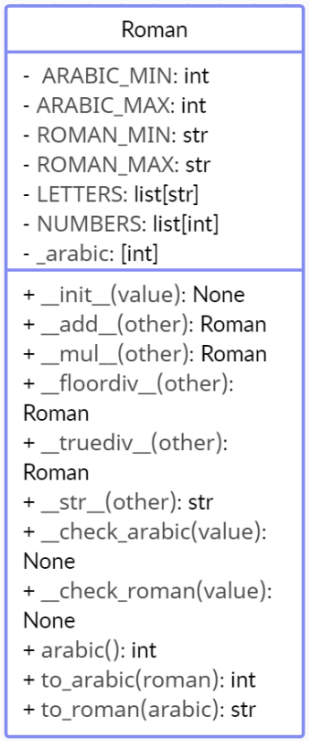


Рисунок 1 - Рисунок 1 – UML диаграмма класса *Roman*

№ 4.3.2 Листинг приведён в файле

main.py

заказ.py

пицца.py

терминал.py

test.py

Реализованы модули *заказ.py, пицца.py, терминал.py.* 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)

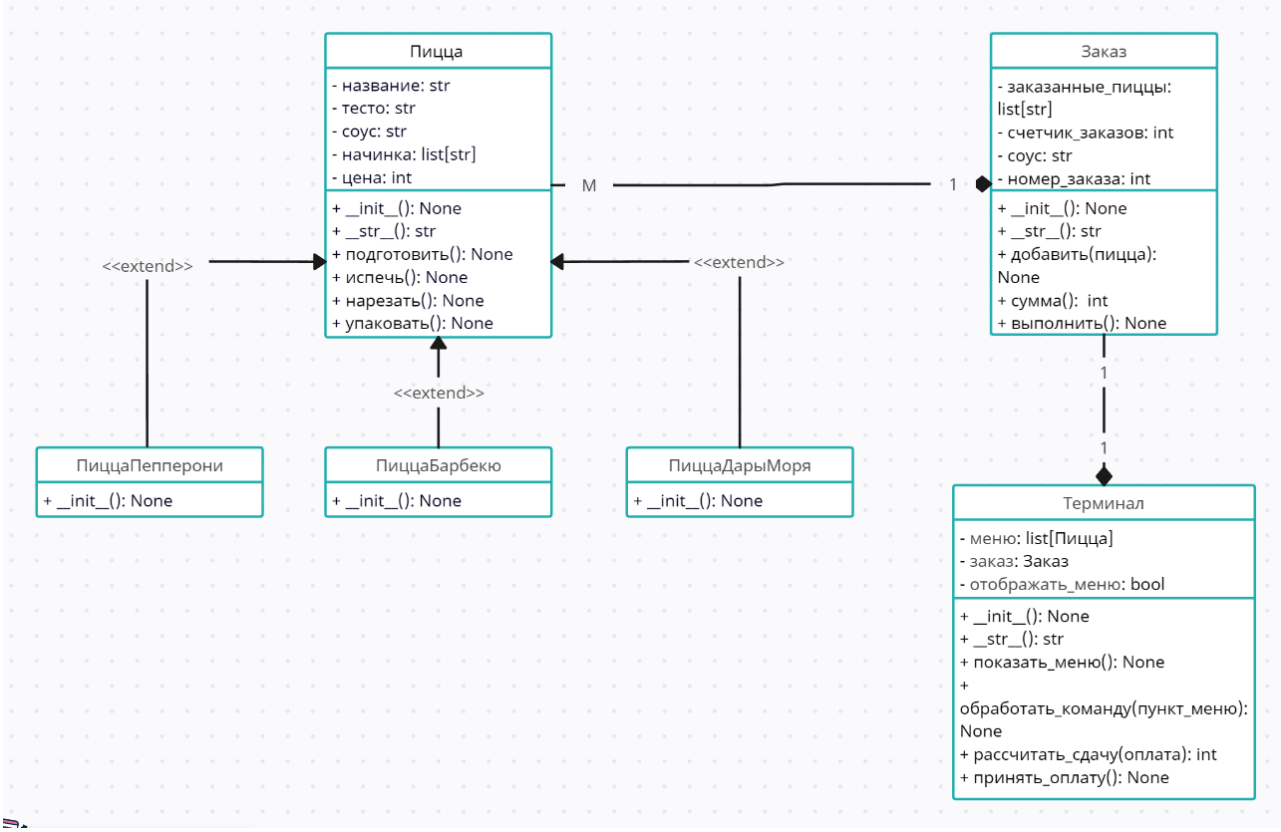


Рисунок 2 – UML диаграмма программы для заказа пиццы

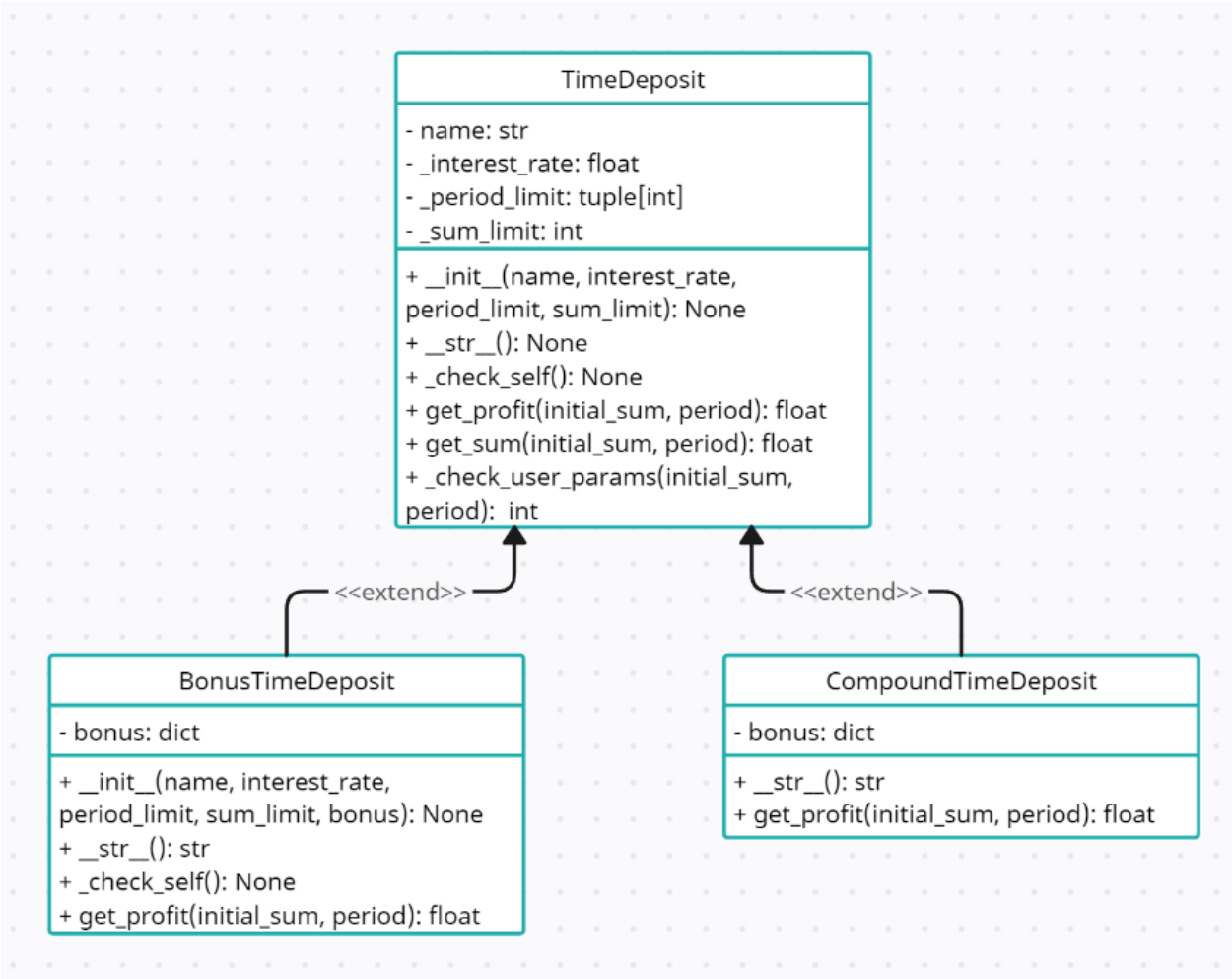
№ 4.3.3 Листинг приведён в файле

main.py

deposit.py

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

my_tyme.py

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*)

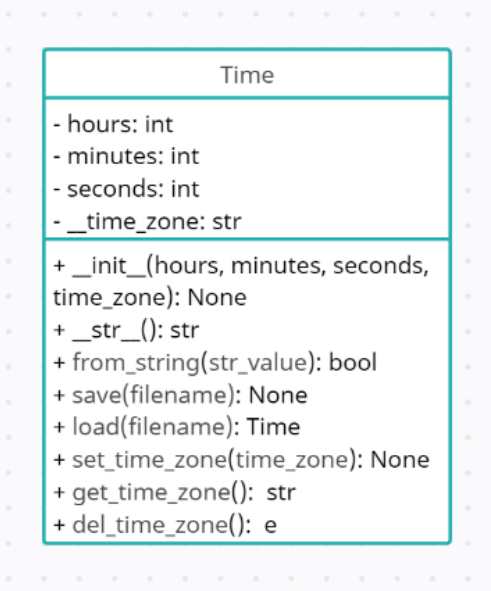


Рисунок 4 – UML диаграмма класса *Time*

№ 4.3.5 Листинг приведён в файле

main.py

timeContainer.py

my_time.py

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)*

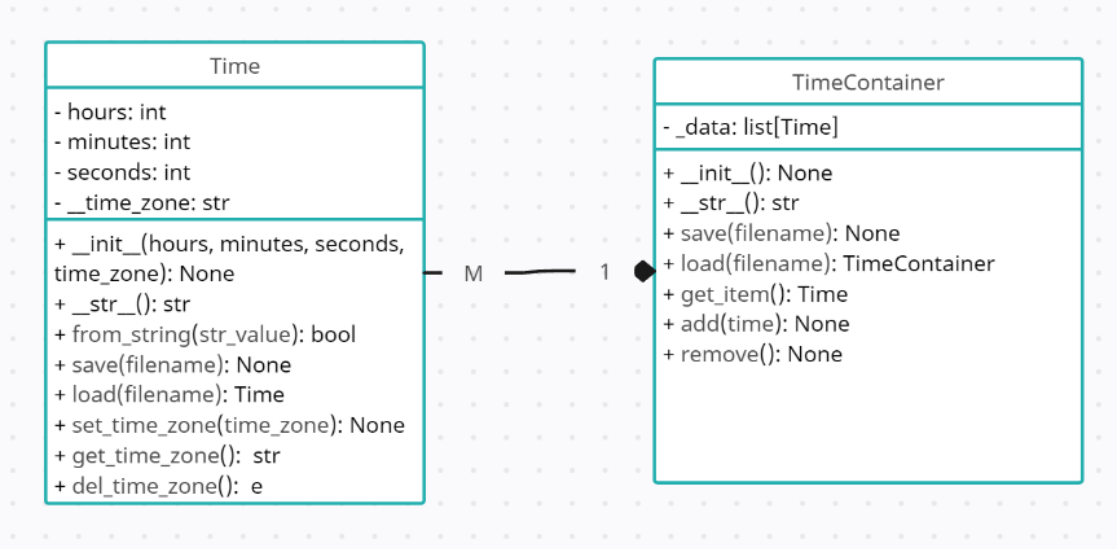


Рисунок 5 – UML диаграмма классов *Time* и *TimeContainer*

№ 4.3.6 Листинг приведён в файле

main.py

bilet.py

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*)*

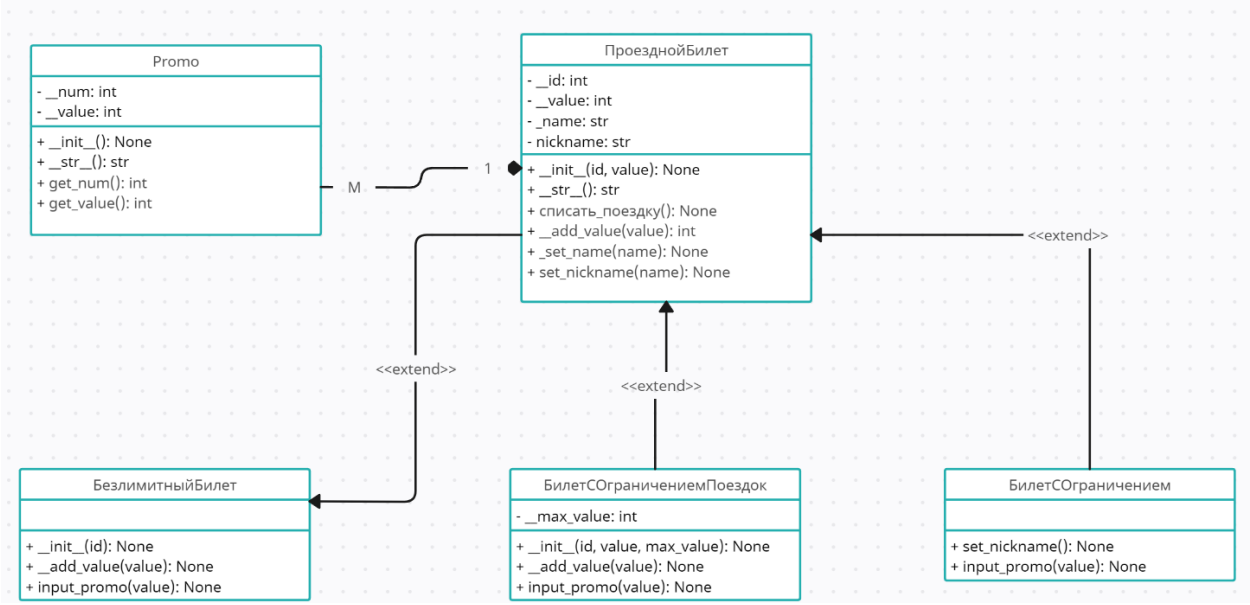


Рисунок 6 – UML диаграмма программы для работы с билетами

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.