Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Прикладной Информатики

Отчет по практике

Вариант номер 9

Научный руководитель: Карчева Наталия, преподаватель

Автор: Шелестян Анастасия, гр IA2304

Кишинэу,2024

**Вариант 9**

Описать структуру с именем ORDER, содержащую следующие поля:

* Расчётный счёт плательщика;
* Расчётный счёт получателя;
* Перечисляемая сумма в руб.

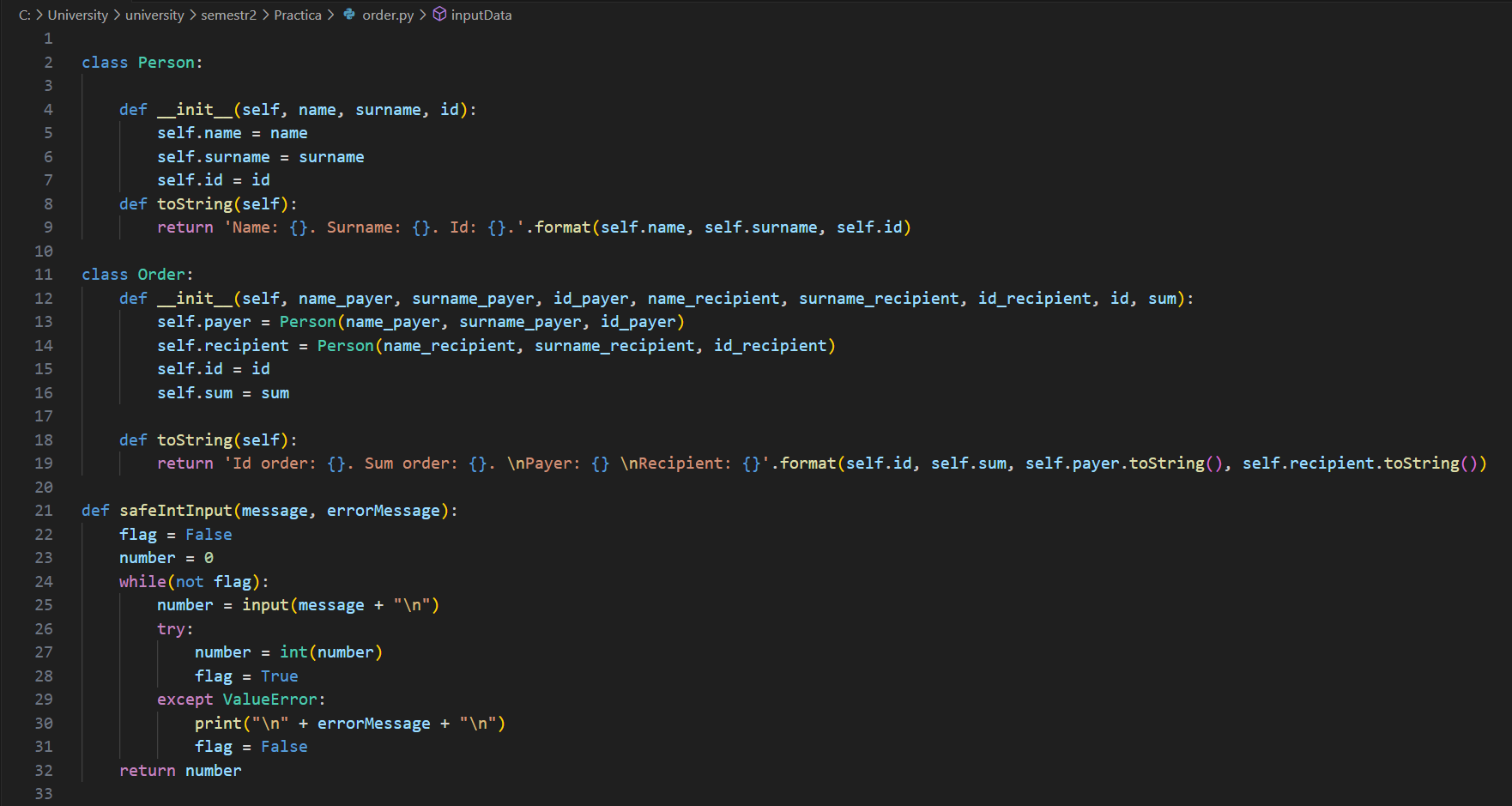
Написать программу, выполняющую следующие действия:

* Ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ORDER; записи должны быть размещены в алфавитном порядке по  расчётным счётом плательщиков;
* Вывод на экран информации о  сумме, снятой с расчётного счёта плательщика, введённого с клавиатуры;
* Если  такого расчётного счёта нет, выдать на дисплей соответствующее сообщение.

**Цель работы программы:**

Целью работы программы является автоматизация процесса управления заказами с возможностью ввода данных, их обработки и анализа. Основные задачи программы включают: ввод данных, сортировка данных, запись данных, анализ данных.

**Код программы:**

****

****

**Класс Person:**

**class Person:** начинает определение класса Person.

**\_\_init\_\_(self, name, surname, id):** конструктор класса, который инициализирует атрибуты объекта (self) с заданными name (имя), surname (фамилия) и id (идентификатор).

**self.name = name, self.surname = surname, self.id = id:** присваивание входных параметров (аргументов конструктора) атрибутам объекта self.

**def toString(self):** метод toString, который возвращает строку с данными объекта Person, форматируя имя, фамилию и идентификатор в заданном формате.

**Класс Order:**

**class Order**: начинает определение класса Order.

\_\_init\_\_(self, name\_payer, surname\_payer, id\_payer, name\_recipient, surname\_recipient, id\_recipient, id, sum): конструктор класса, который инициализирует атрибуты объекта (self).

**self.payer** = Person(name\_payer, surname\_payer, id\_payer): создает объект класса Person для плательщика с заданными именем, фамилией и идентификатором.

**self.recipient** = Person(name\_recipient, surname\_recipient, id\_recipient): создает объект класса Person для получателя с заданными именем, фамилией и идентификатором.

**self.id = id**: присваивает идентификатор заказа.

**self.sum** = sum: присваивает сумму заказа.

**def toString(self):** метод toString, который возвращает строку с данными объекта Order, включая идентификатор заказа, сумму, а также информацию о плательщике и получателе, используя метод toString из объектов Person.

**Функция safeIntInput():**

Используется для безопасного ввода целочисленных значений с обработкой исключений (ввод должен быть числом).

**Функция inputData():**

Определение функции inputData:

**def inputData():** определяет функцию inputData, которая не принимает аргументов.

**payer\_name = input**("Enter name payer \n"): запрос ввода имени плательщика.

**payer\_surname = input**("Enter surname payer \n"): запрос ввода фамилии плательщика.

**payer\_id = safeIntInput**("Enter ID payer", "Payer ID must be only number! Try again"): запрос ввода идентификатора плательщика с использованием функции safeIntInput для безопасного ввода целого числа.

Аналогично запрашивается информация о получателе и заказе (имя, фамилия, идентификатор, сумма и ID заказа).

**return Order**(...): создание и возврат нового объекта Order с введенными данными.

**Функция newArray(size):**

**def newArray(size):** определяет функцию newArray, которая принимает аргумент size (размер массива).

**newArray** = []: создает пустой список newArray для хранения объектов Order.

**for i in range(size)::** цикл for, который повторяется size раз.

print("\nEnter data for order", format(i + 1)): выводит сообщение о необходимости ввода данных для каждого заказа.

newArray.append(inputData()): добавляет введенные данные заказа (объект Order) в список newArray.

**sortedArr** = sorted(newArray, key=lambda order: order.payer.surname): сортирует список newArray по фамилии плательщика (payer.surname) и сохраняет отсортированный список в sortedArr.

**with open**("Order.txt", "a") as file:: открывает файл "Order.txt" для дозаписи (добавления новых данных).

**for element in sortedArr::** для каждого элемента в отсортированном списке sortedArr выполняется:

**file.write(element.toString()):** записывает строковое представление объекта element в файл.

return sorted.

**Функция getTotalSum(arr, id):**

**def getTotalSum(arr, id):** определяет функцию getTotalSum, которая принимает два аргумента: arr (массив заказов) и id (идентификатор плательщика, для которого нужно вычислить сумму).

**sum = 0:** переменная sum инициализируется нулевым значением для накопления общей суммы заказов.

**flag = False**: переменная flag инициализируется как False для отслеживания наличия плательщика с указанным id.

**for order in arr:** цикл for, который перебирает каждый элемент (заказ) в массиве arr.

**if id == order.payer.id:** условие проверяет, соответствует ли идентификатор плательщика (id) идентификатору плательщика в текущем заказе (order.payer.id).

**sum += order.sum:** если условие выполняется, сумма заказа (order.sum) добавляется к общей сумме sum.

**flag = True:** устанавливает flag в True, чтобы указать, что плательщик с указанным id был найден.

**if flag**: условие проверяет, был ли найден плательщик с указанным id.

print("Total sum for payer id", format(id), "is", format(sum)): если плательщик найден, выводит общую сумму заказов для указанного id.

**else:** в противном случае (если плательщик не найден):

print("No payer with id", format(id), "found"): выводит сообщение о том, что плательщик с указанным id не был найден.

**Использование функций newArray и getTotalSum:**

**arr = newArray(2):** создает массив arr, содержащий 2 заказа (объекта Order), полученных с помощью функции newArray.

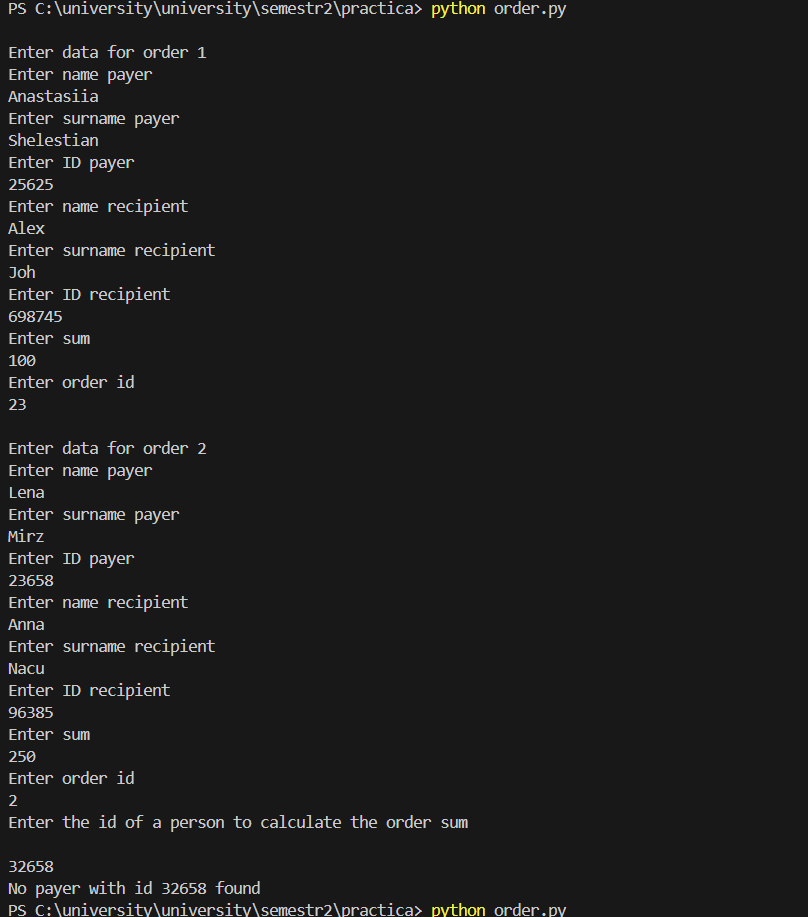
**Функция newArray** позволяет пользователю ввести данные о двух заказах, сортирует их по фамилии плательщика и записывает в файл "Order.txt".

**id = safeIntInput**("Enter the id of a person to calculate the order sum \n", "Error! Id must be a number!"): запрашивает у пользователя ввод идентификатора плательщика с помощью функции safeIntInput. Введенный идентификатор сохраняется в переменной id.

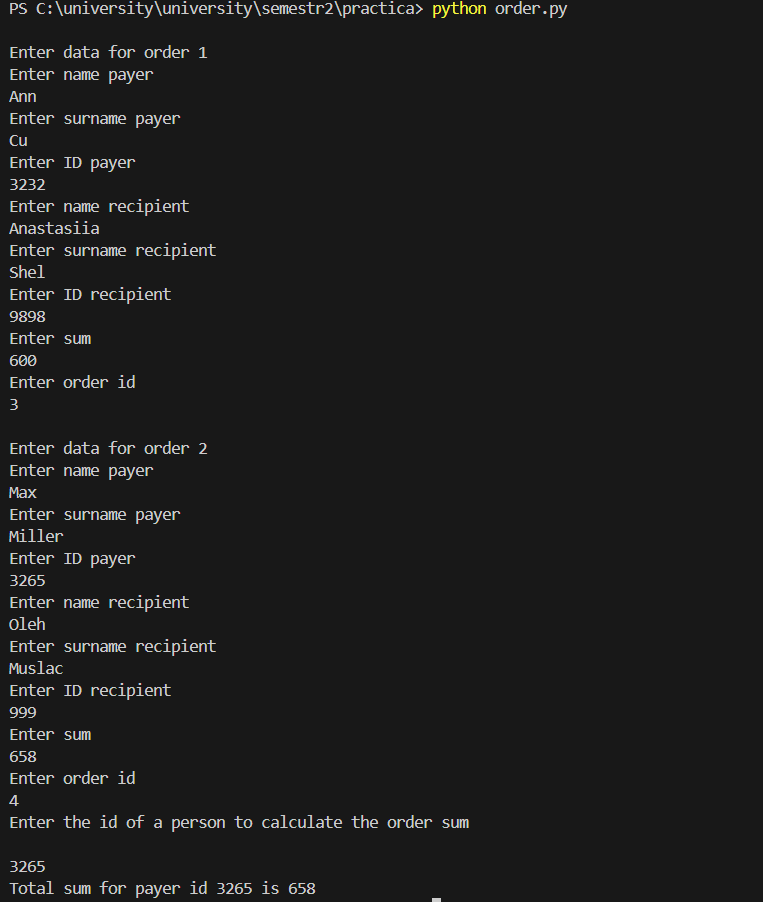
**getTotalSum(arr, id):** вызывает функцию getTotalSum для расчета общей суммы заказов для плательщика с указанным id.

**Выполненный код:**

Вводим данные плательщика и получателя также вводим номер заказа и сумму. Для проверки есть ли заказ вводим ID плательщика и программа нам выдает, что данного заказа нет, так как мы ввели несуществующий ID

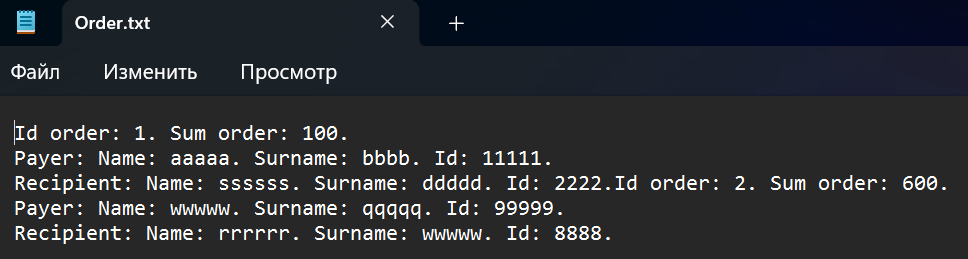
****

Вводим данные плательщика и получателя также вводим номер заказа и сумму. Для проверки есть ли заказ вводим ID плательщика и программа нам выводит сумму заказа.

****

**Вид в файле:**

До вписания в файл наши данные сортируются в алфавитном порядке и выводится на экран.

****

**Выводы:**

Таким образом, код представляет собой пример объектно-ориентированного программирования с использованием Python для управления данными о заказах и их анализа. Он демонстрирует эффективное использование классов, функций для ввода и обработки данных, а также методов для работы с файлами и анализа информации.