

1. По текстовому описанию задачи построить диаграмму прецедентов.

На рисунке 1 указана диаграмма прецедентов. По ней можно понять, какой функционал будет иметь для работников отеля, или для руководителей групп.

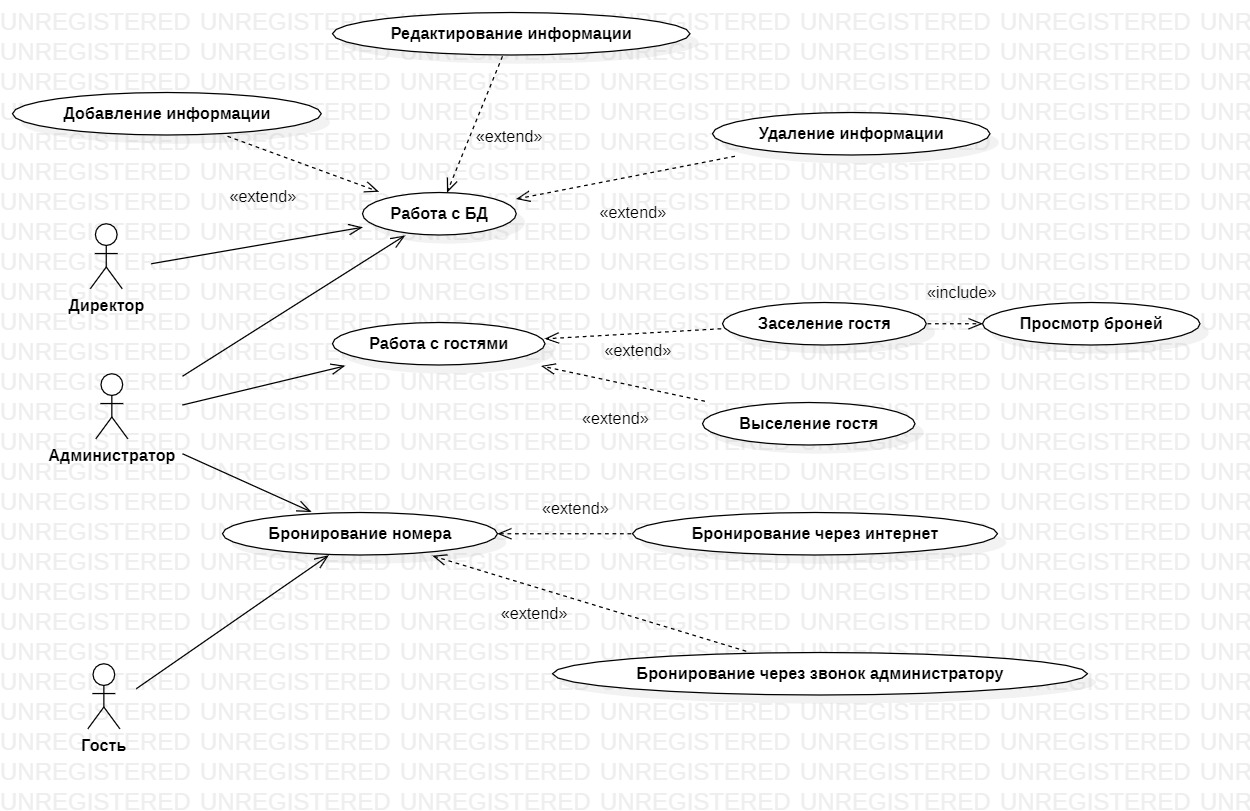


Рисунок 1

1. Классы по диаграмме прецедентов соответственно

На рисунке 2 показана диаграмма классов, по которой, предположительно будет создаваться программа, если не возникнет иных способов реализации задачи.

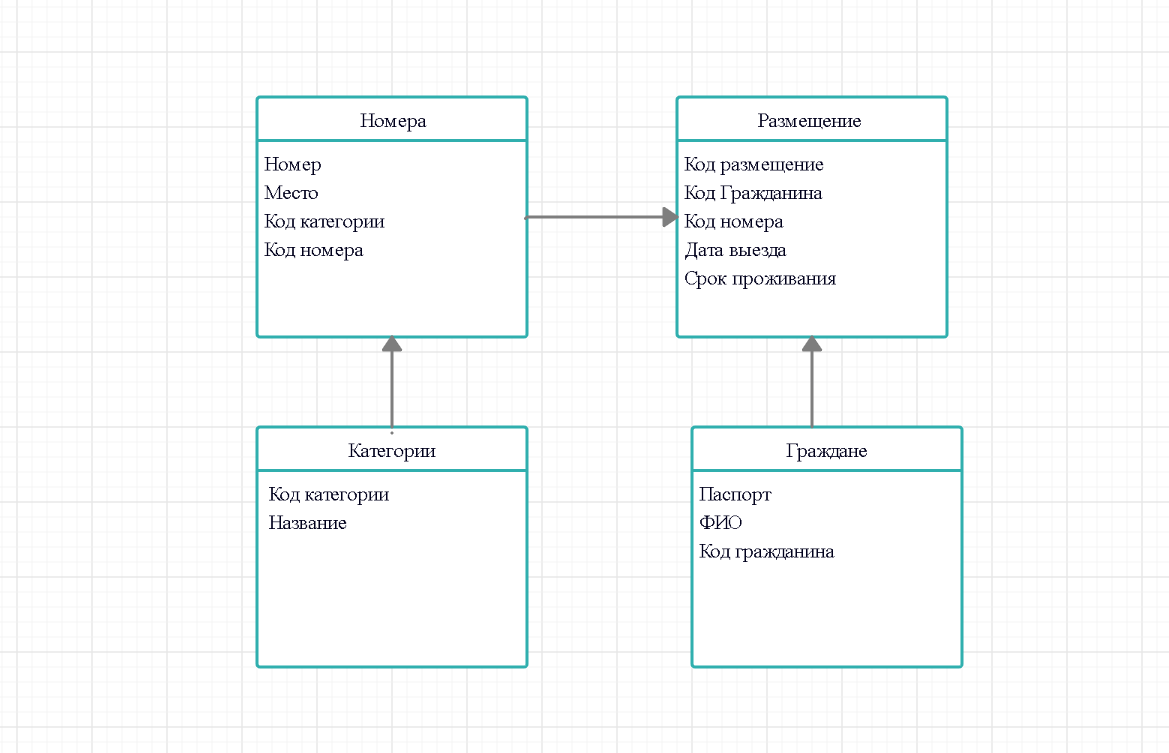


Рисунок 2

1. Разработать определение классов по диаграмме

После создания диаграммы классов по диаграмме прецедентов и описанию задачи было совершено заключение что описание является не полным и не четким .Классы будут реализованны в следующем порядке

А)Класс работы Граждане   
 Б)Класс работы Номера   
 4. Разработать проект с классами для выполнение задания

Код проекта представлен в файле “classes main.py”. Код так же будет приведён далее. Для правильной работы так-же необходимы файлы “Hostel” и “Number” с расширением .xlsx.

Код программы из файла “classes main.py”

import pandas as pd  
#класс для работы с таблицей “”  
class Table1:  
 def \_\_init\_\_(self, event\_type\_code, type\_name):  
 self.event\_type\_code = event\_type\_code  
 self.type\_name = type\_name  
  
 def PrintOut1():  
 df1 = pd.read\_excel('Hostel.xlsx')  
 print(df1[['Route Code', 'Route Name', 'Route Time', 'Route Category']])  
#класс для работы с таблицей “Граждане ”  
class Table2:  
 def \_\_init\_\_(self, event\_type\_code, type\_name):  
 self.event\_type\_code = event\_type\_code  
 self.type\_name = type\_name  
  
 def PrintOut2():  
 df2 = pd.read\_excel('Hostel.xlsx')  
 print(df2[["Leader's Code", 'NSL', 'Phone Number']])  
  
#класс для работы с таблицей “Номера”  
class Table3:  
 def \_\_init\_\_(self, event\_code, event\_type\_code, event\_name, start\_time, full\_time, done\_tick):  
 self.event\_code = event\_code  
 self.event\_type\_code = event\_type\_code  
 self.event\_name = event\_name  
 self.start\_time = start\_time  
 self.full\_time = full\_time  
 self.done\_tick = done\_tick  
  
 def PrintOut3():  
 df3 = pd.read\_excel('Nomber.xlsx')  
 print(df3)  
  
 def PrintRoutes():  
 df3 = pd.read\_excel('Nomber.xlsx')  
 print(df3[["Group Name", "Route Code"]])  
  
# функция для цикла с главным меню  
def MainMenu():  
 print('Выберите действие:')  
 print('1: Вывести первую таблицу')  
 print('2: Вывести вторую таблицу')  
 print('3: Вывести третью таблицу')  
 print('4: Вывести список групп и их маршрутов')  
 print('0: Выход')  
 cycle = input('>')  
 return cycle  
  
  
cycle = MainMenu()  
# цикл выбора пунктов в главном меню  
while cycle != '0':  
 if cycle == '1':  
 Table1.PrintOut1()  
 elif cycle == '2':  
 Table2.PrintOut2()  
 elif cycle == '3':  
 Table3.PrintOut3()  
 elif cycle == '4':  
 Table3.PrintRoutes()  
 else:  
 print('Неизвестное действие, используйте предложенный выбор.')  
 cycle = MainMenu()

5. Показать отладку одного из модулей

В ходе написания задания при попытке запустить скрипт было получено данное сообщение:

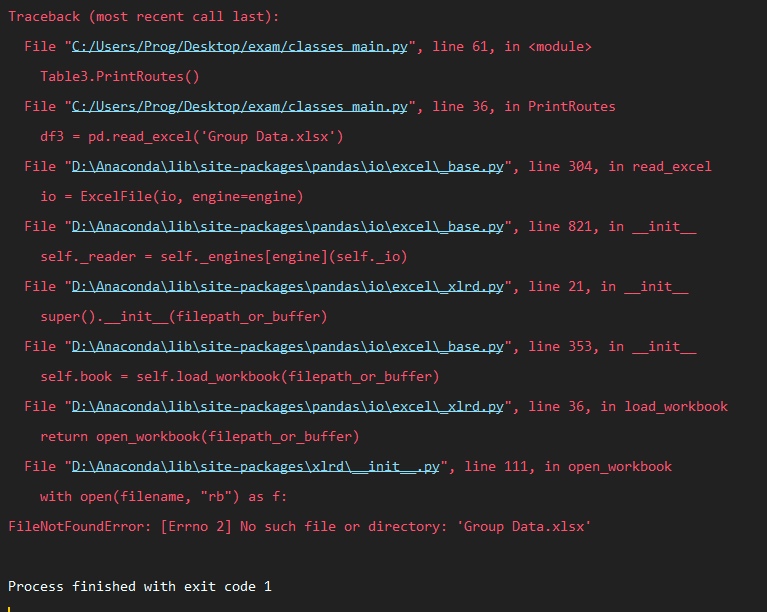


Рисунок3. До применения средств отладки

После получения данного сообщения были просмотрены 90 и 89 строки модуля Implementation.py и была обнаружена ошибка, которая впоследствии была устранена, а после попытки запуска скрипта получено данное сообщение:

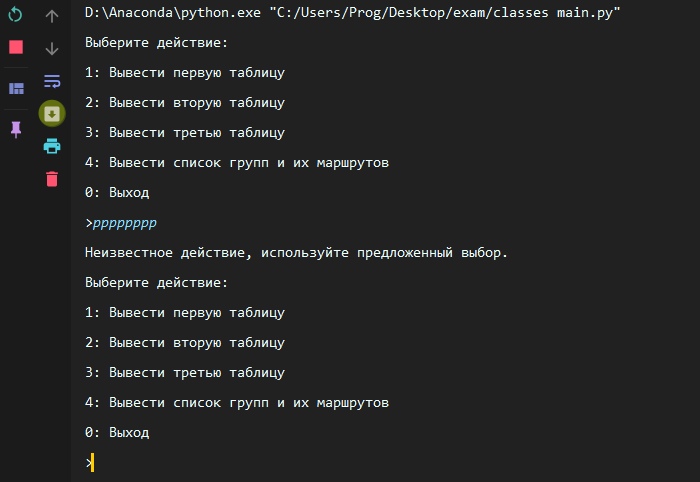


Рисунок4. После применения средств отладки

Это означает что ошибка была устранена и скрипт запустился.

6.Подготовить текстовые наборы и провести тестирование одного из модулей.

После передачи действия в главное меню, запускается бесконечный цикл, прерывающийся подачей символа “0”. Реагирует цикл только на символы “1”, “2”, “3”, “4” и “0”. При попытке ввода иных символов цикл переходит на новую итерацию, не производя никаких иных действий. Было принято решение создать подсказку, оповещающую, что на введённый символ реакции не будет.

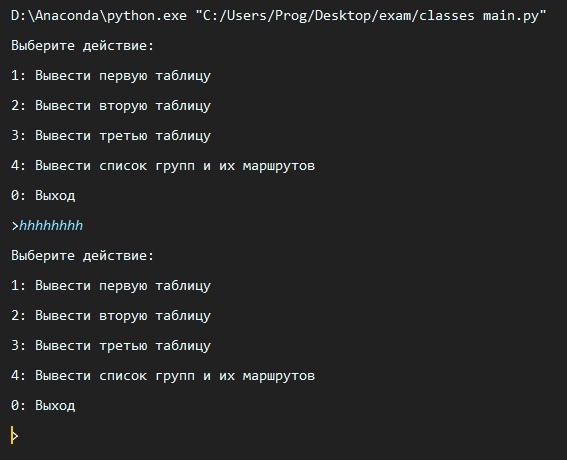


Рисунок 3 До применения предложенного решения

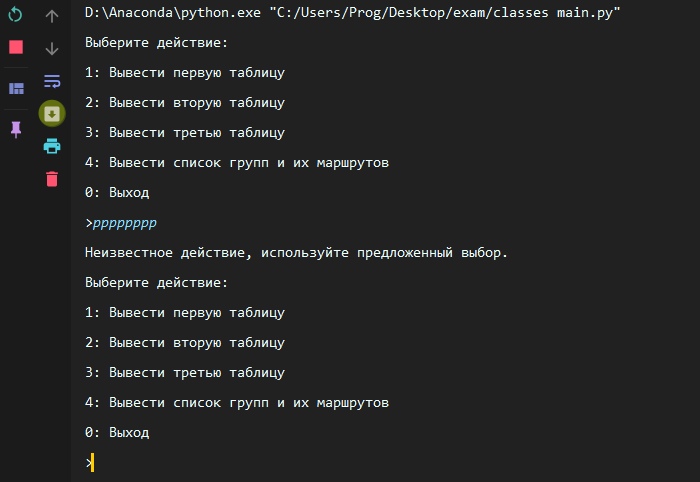


Рисунок 4 после

1. Определить значение временной сложности O для одного из методов. Указать значение O для оптимального кода.

Весь код держится на цикле “While”, начинающемся на строке 53. Из него происходит вызов методов из трёх классов. Цикл не прекратится, пока пользователь не введёт символ “0” в строку диалога. В противном случае цикл будет продолжаться бесконечное количество времени. Отсюда значение временной сложности Big O будет равняться O(N), так же оно является значением для оптимального кода.