

1. По текстовому описанию задачи построить диаграмму прецедентов.

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

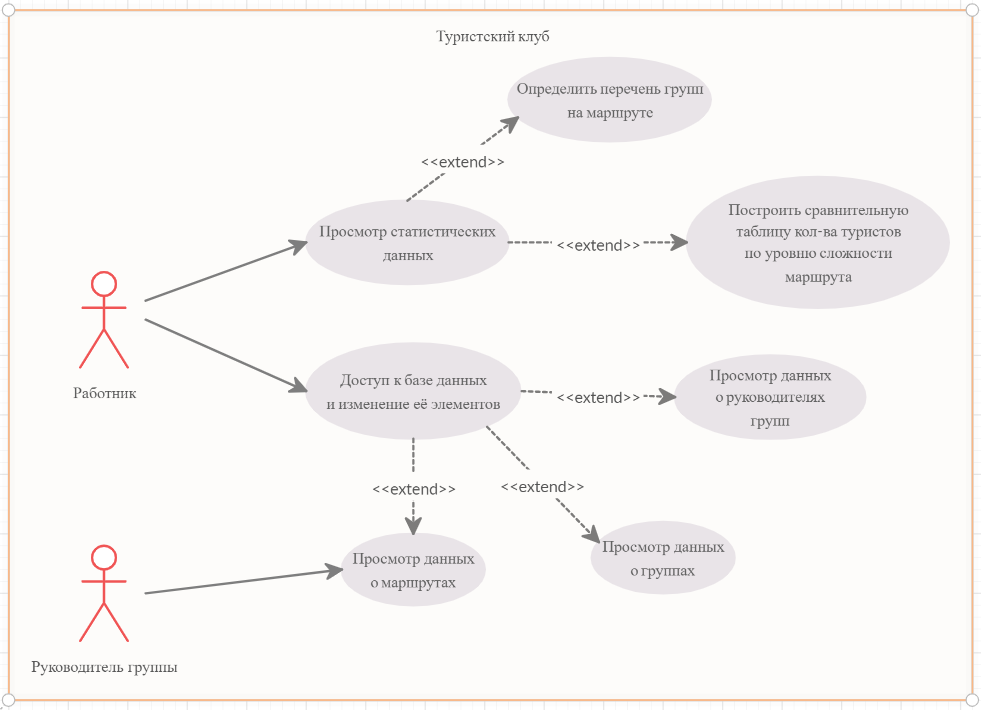


Рисунок 1. Диаграмма прецедентов

1. По описанию и диаграмме прецедентов построить диаграмму классов.

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

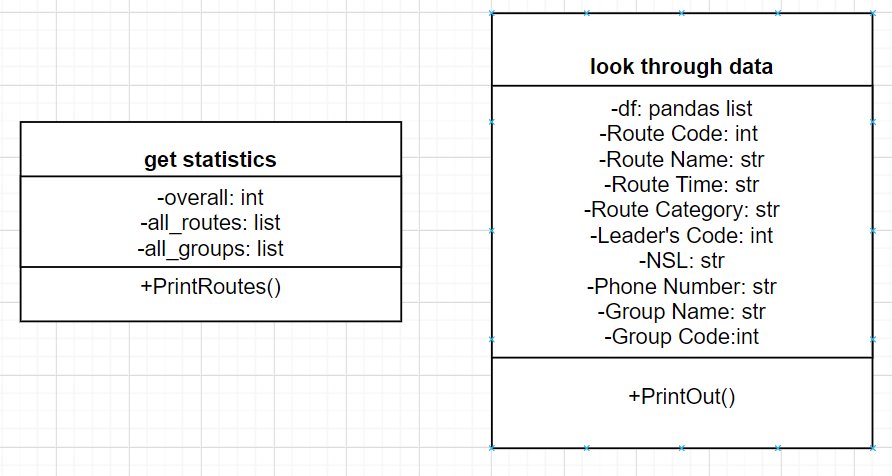


Рисунок 2. Диаграмма классов

1. Разработать определения классов по диаграмме.

После создания диаграммы классов по диаграмме прецедентов и описании задачи было совершено заключение, что диаграмма классов является нечетко определённой. Было решено создать программу с тремя классами для работы с каждой таблицей и более легкой работы. Классы будут реализованы в следующем порядке:

* 1. Класс для работы с таблицей “Маршруты”
  2. Класс для работы с таблицей “Руководители”
  3. Класс для работы с таблицей “Группы”

В каждом классе будет содержаться метод вывода таблицы.

1. Разработать проект с классами для выполнения задания.

Код проекта представлен в файле “classes main.py”. Код так же будет приведён далее. Для правильной работы так же необходимы файлы “Tourism” и “Group\_Data” с расширением .xlsx.

Код программы из файла “classes main.py”

import pandas as pd  
#класс для работы с таблицей “Маршруты”  
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('Tourism.xlsx')  
 print(df1[['Route Code', 'Route Name', 'Route Time', 'Route Category']])  
#класс для работы с таблицей “Руководители”  
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('Tourism.xlsx')  
 print(df2[["Leader's Code", 'NSL', 'Phone Number']])  
  
#класс для работы с таблицей “Группы”  
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('Group\_Data.xlsx')  
 print(df3)  
  
 def PrintRoutes():  
 df3 = pd.read\_excel('Group\_Data.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()

1. Показать отладку одного из модулей при разработке.

При запуске программы, происходила ошибка компиляции. В окне запуска появлялся текст, отображённый на рисунке 3.



Рисунок 3. До применения средств отладки

В результате анализа была выявлена причина ошибки. Ошибку содержала 32 строка (класс “Table3”, метод “PrintOut3”). Из-за неверно указанного названия колонки, оператор библиотеки pandas не мог найти нужное название, и выдавал ошибку. После исправления, метод работал исправно.

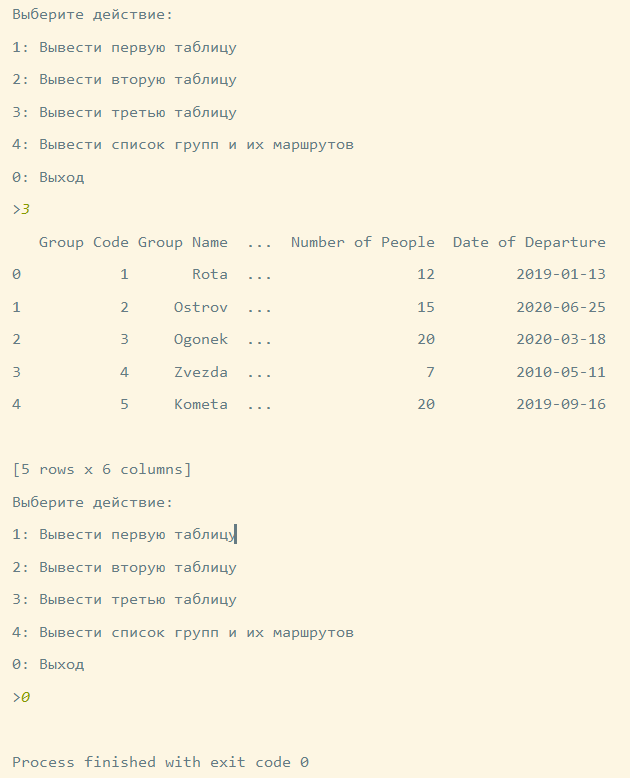


Рисунок 4. После применения средств отладки

1. 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.

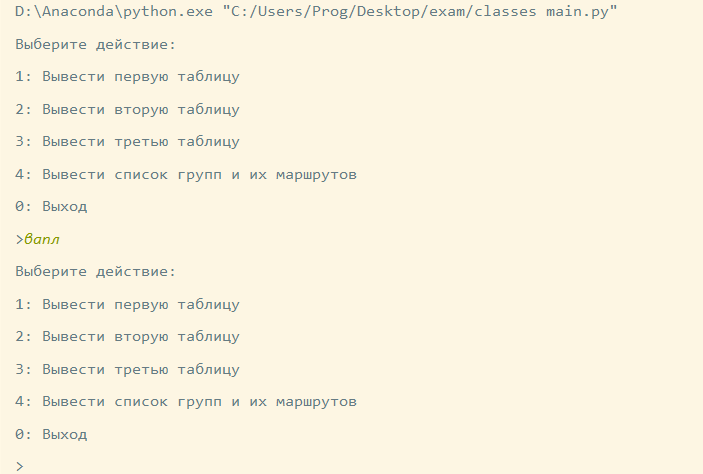


Рисунок 5. До применения предложенного решения

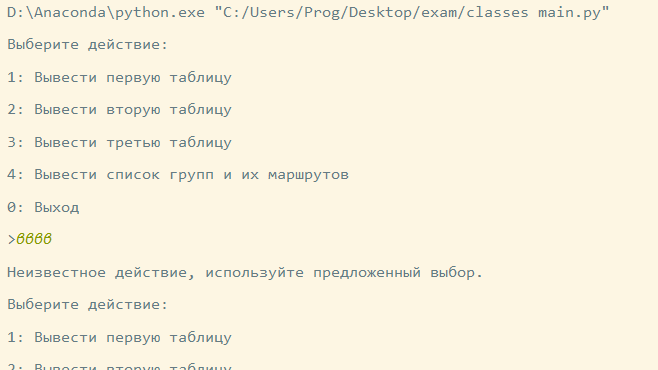


Рисунок 6. После применения предложенного решения

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

В коде файла “classes main.py”, методы всех классов не имеют циклов, от этого они являются линейными и выполняется за некоторое постоянное время, определяемое вычислительной мощностью компьютера, на котором программа запускается. Алгоритмы с таким константным временем всегда выражаются как:

**O(1)**

Рассмотрим метод “PrintOut1” класса “Routes”, из файла “classes main.py”.

Код метода PrintOut1.

def PrintOut1():  
 df1 = pd.read\_excel('Tourism.xlsx')  
 print(df1[['Route Code', 'Route Name', 'Route Time', 'Route Category']])

Алгоритм построчного выполнения при вызове этого метода будет иметь константное время. То есть Big O будет равным O(1).

Для всей программы значение O будет равным:

**O(7 + n),**

где 7 – все методы с линейным алгоритмом, а n – цикл while с потенциально бесконечным количеством итераций.