



Технически университет – София
Факултет по компютърни системи и технологии

Курсов проект по Програмни езици

Изготвил: Магдалена Евгениева,
фак. номер: 501219012, група 36
Дата: 20.05.2022 г.

Проверил:
/доц. д-р инж. Иван Станков/

Задание:

Създайте приложение, което да поддържа информация за маршрутни таксите. данните за колите са марка, модел, години, колко местна е, с каква товароподемност и колко разход на гориво иска. Тези коли покриват някакви маршрути в града. За маршрута трябва да се знае възловите му точки и колко километра е дълъг и колко пъти на ден се обикаля.

Приложението да има възможност за въвеждане на произволен брой различни маршрутни таксите и маршрути(10 точки).

Да има възможност за избор на маршрутни таксите на което да се задава маршрут и да извежда информация колко гориво да се зареди за извършване не дневната обиколка (10 точки).

Класовете (най-малко 3 класа при реализацията) трябва да капсулира всичките детайли. Използват се private инстанции на променливите за съхраняване на различните детайли. Трябва да има най-малко два конструктора, public getters/setters за private инстанции на променливите (30 точки).

Необходимо е да извършвате проверка на входните данни (10 точки).

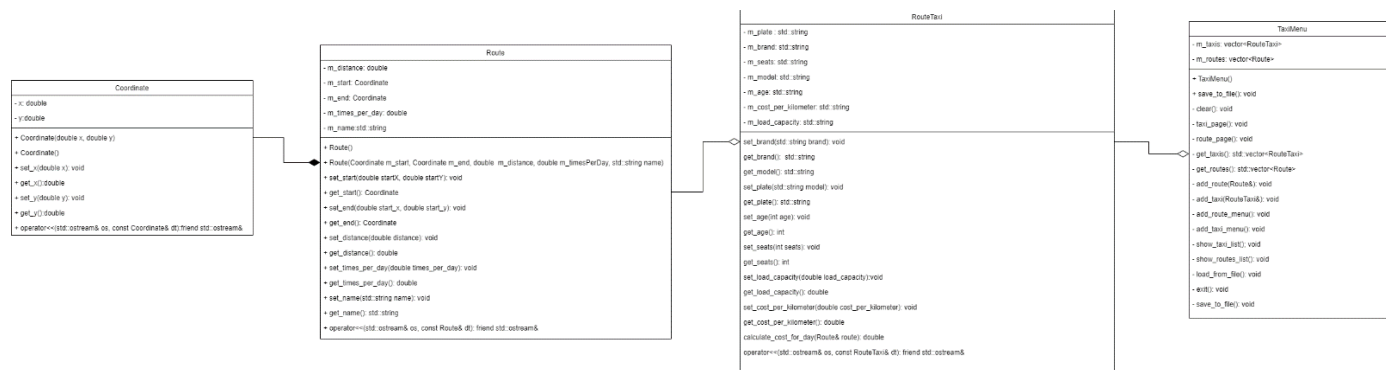
Да се предефинира операцията <<, която да се използва за извеждане на данните (10 точки). Данните да се четат и съхраняват във файл (20 точки).

Класовете да се опишат с UML клас диаграма (10 точки).

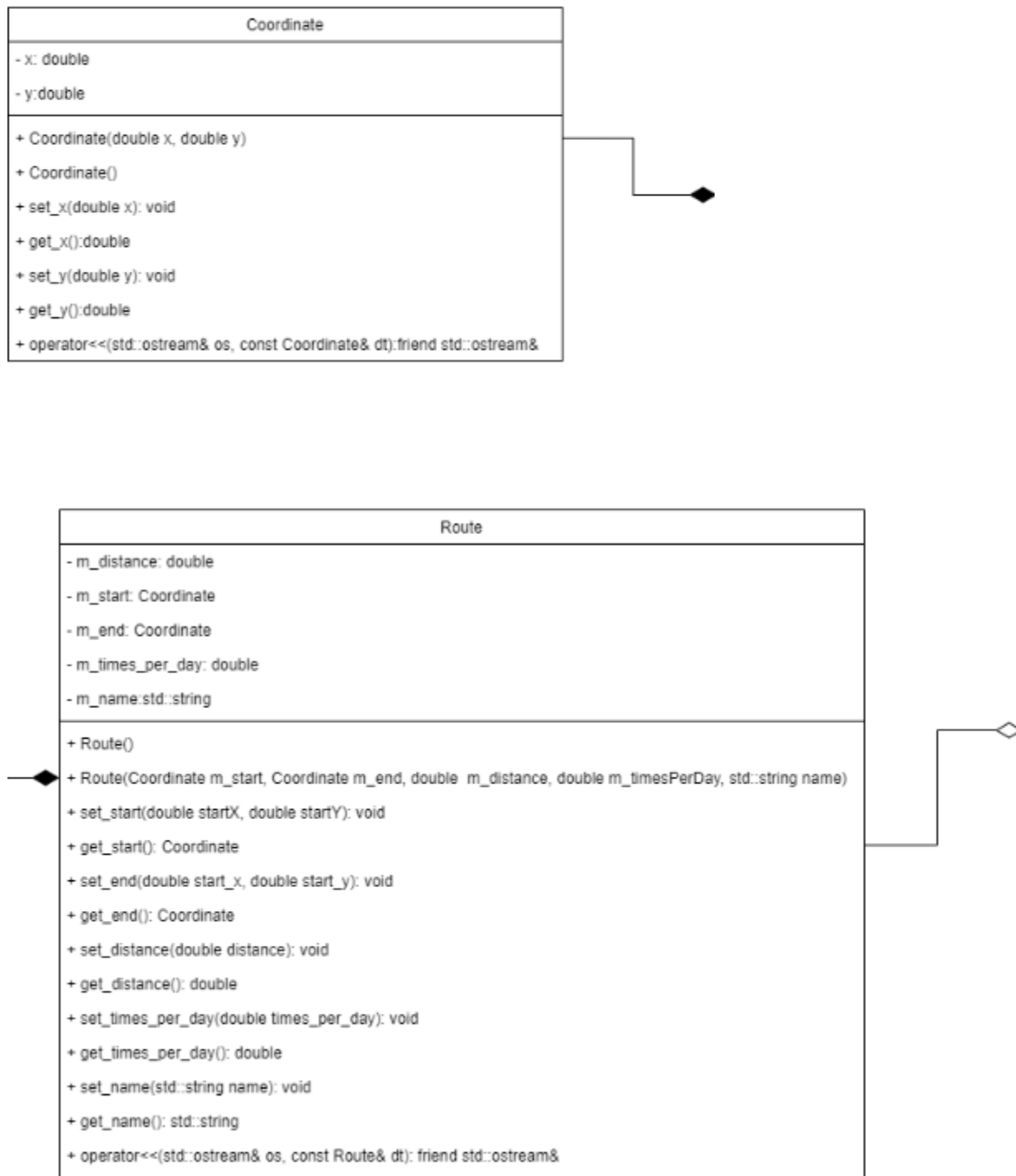
Задължително данните да се въвеждат динамично, чрез меню.

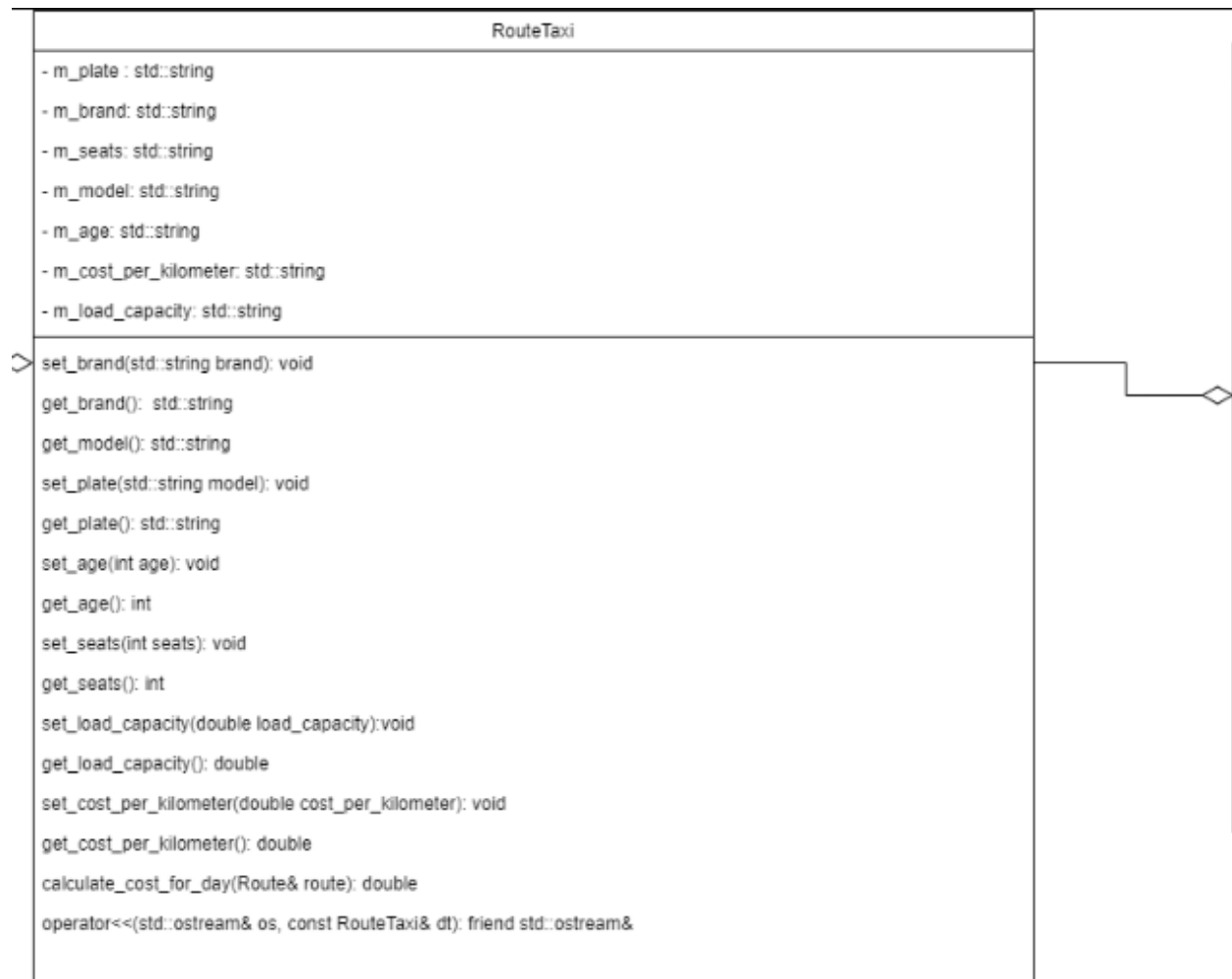
UML клас диаграма:

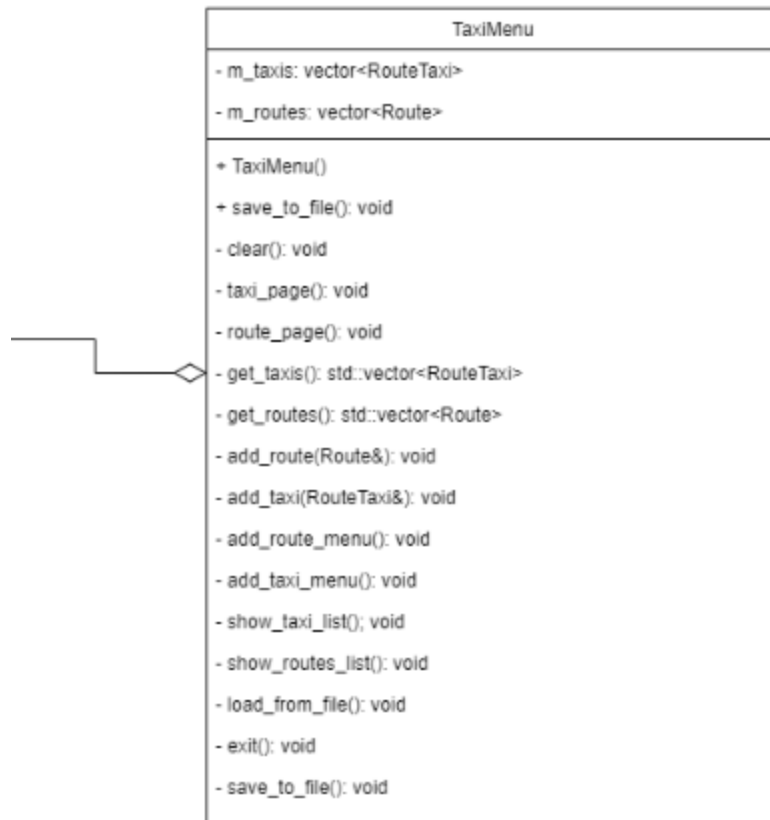
Цялостна UML диаграма:



UML диаграма на части:







Код на задачата:

Класа TaxiMenu съдържа следните функции:

```
+ TaxiMenu::TaxiMenu() { ... }  
  
+ int TaxiMenu::load_from_file() { ... }  
  
+ void TaxiMenu::exit() { ... }  
  
+ int TaxiMenu::save_to_file() { ... }  
  
+ std::vector<RouteTaxi> TaxiMenu::get_taxis() { ... }  
  
+ std::vector<Route> TaxiMenu::get_routes() { ... }  
  
+ void TaxiMenu::add_route(Route& route) { ... }  
  
+ void TaxiMenu::add_taxi(RouteTaxi& taxi) { ... }  
  
+ void TaxiMenu::add_taxi_menu() { ... }  
  
+ void TaxiMenu::add_route_menu() { ... }  
  
+ void TaxiMenu::start_page() { ... }  
  
+ void TaxiMenu::clear() { ... }
```

```
+ void TaxiMenu::taxi_page() { ... }  
  
+ void TaxiMenu::route_page() { ... }  
  
+ void TaxiMenu::show_taxi_list() { ... }  
  
+ void TaxiMenu::show_routes_list() { ... }
```

Функцията load_from_file() зарежда данните за масивите от таксите и маршрути съответно от файловете "taxis.txt", "routes.txt". Файлът taxis.txt е форматиран по следния начин „<номер >/<марка >/<модел възраст >/<модел>/<възраст>/<седалки>/<товароносимост разход>“

Функцията exit() извиква save_to_file().

Функцията get_taxis() връща списъка с въведени таксите.

Функцията get_routes() връща списъка с въведени маршрути.

Функцията save_to_file() запазва данните от таксите и маршрути съответно от файловете "taxis.txt", "routes.txt", и файловете се форматиран в следния формат „<номер >/<марка >/<модел възраст >/<модел>/<възраст>/<седалки>/<товароносимост разход>“

Функцията `start_page()` предоставя меню с опции:

```
1 TaxiMenu::start_page()

2
3
4 int option;
5 bool exit = false;
6 while (!exit)
7 {
8     clear();
9     std::cout << "Choose an option!\n";
10    std::cout << "1.See all taxis and choose or add route to taxi\n";
11    std::cout << "2.See all routes\n";
12    std::cout << "3.Add a taxi\n";
13    std::cout << "4. Add a route\n";
14    std::cout << "5.Save & exit\n";
15    std::cin >> option;
16
17    switch (option) {
18    case 1:
19        taxi_page();
20        //cout << "1.See all taxis and choose or add route to taxi";
21        break;
22    case 2:
23        route_page();
24        // cout << "2. See all routes";
25        break;
26    case 3:
27        add_taxi_menu();
28        // cout << "Add a taxi";
29        break;
30    case 4:
31        add_route_menu();
32        // cout << "4. Add a route";
33        break;
34    case 5:
35        // cout << "Exit";
36        TaxiMenu::exit();
37        exit = true;
38        break;
39    default:
40        std::cout << "Wrong option! Try again! Press any key to countinue.";
41        std::cin.get();
42        //exit()
43
44        break;
45    }
46 }
```

Функцията `taxi_page()` – това е менюто за селектиране на опции свързани с такситата, а именно

- взимане на такси. От тук следва избиране на път и смятане на разхода за деня
- добавяне на такси
- показване на менюто за маршрути

Функцията `route_page()` - менюто за маршрути. Може да се добавя нов маршрут.

Функцията `add_taxi_menu()` предоставя възможност за добавяне на таксите с възможност да се добавяне на табела, марка, модел, товароносимост и цената, която харчи таксито на километър.

Функцията `add_route_menu()` предоставя възможност за добавяне на маршрути. Като може да се добавят името на маршрута, началните и крайните координати, дължината на пътя и колко пъти се преминава през маршрута за ден.

Функцията `show_taxi_list()` показва списък с такситата въведени в системата.

Функцията `show_routes_list()` показва списък с маршрутите въведени в системата.

Класа `Coordinate` симулира точка на картата с координати `x` и `y`.

Има предефинира операцията `<<`.

Има два конструктора един с параметри и един без параметри.

```
1  #pragma once
2  #include <iostream>
3
4  class Coordinate
5  {
6  private:
7      double m_x;
8      double m_y;
9
10 public:
11     Coordinate(double x, double y);
12     Coordinate();
13     void set_x(double x);
14     double get_x();
15     void set_y(double y);
16     double get_y();
17     friend std::ostream& operator<<(std::ostream& os, const Coordinate& dt);
18
19 };
20
21
```


Класа RouteTaxi представя маршрутно такси.

Има private инстанции на променливите табела, марка, модел, товароносимост и цената, която харчи таксита на километър. И съдържа функция calculate_cost_for_day(), която при подаване на път пресмята разхода за деня и го връща. Има предефинирана операцията <<. Има два конструктора един с параметри и един без параметри.

```
#pragma once
#include <string>
#include "Route.h"
class RouteTaxi
{
private:
    std::string m_brand;
    std::string m_model;
    std::string m_plate;
    int m_age;
    int m_seats;
    double m_load_capacity;
    double m_cost_per_kilometer;
    //Route* m_route;
public:
    RouteTaxi(std::string plate, std::string brand, std::string model, int age, int seats, double load_capacity, double cost_per_kilometer);
    RouteTaxi();
    void set_brand(std::string brand);
    std::string get_brand();
    void set_model(std::string model);
    std::string get_model();
    void set_plate(std::string model);
    std::string get_plate();
    void set_age(int age);
    int get_age();
    void set_seats(int seats);
    int get_seats();
    void set_load_capacity(double load_capacity);
    double get_load_capacity();
    void set_cost_per_kilometer(double cost_per_kilometer);
    double get_cost_per_kilometer();
    //void set_route(Route* route);
    //Route* get_route();
    //bool has_route();
    double calculate_cost_for_day(Route& route);
    friend std::ostream& operator<<(std::ostream& os, const RouteTaxi& dt);
};
```

Класа Route представя маршрут.

Има private инстанции на променливите:

началните и крайните координати, дължината на пътя и колко пъти се преминава през маршрута за ден. Има предефинира операцията <<. Има два конструктора един с параметри и един без параметри.

```
#pragma once
#include <iostream>
#include "Coordinate.h"
#include <string>

class Route
{
private:
    Coordinate m_start;
    Coordinate m_end;
    double m_distance;
    double m_times_per_day;
    std::string m_name;
public:
    Route(Coordinate m_start, Coordinate m_end, double m_distance, double m_timesPerDay, std::string name);
    Route();
    void set_start(double startX, double startY);
    Coordinate get_start();
    void set_end(double start_x, double start_y);
    Coordinate get_end();
    void set_distance(double distance);
    double get_distance();
    void set_times_per_day(double times_per_day);
    double get_times_per_day();
    void set_name(std::string name);
    std::string get_name();
    friend std::ostream& operator<<(std::ostream& os, const Route& dt);
};
```

Примерни резултати при изпълнение на програма:

Пускаме следната поредица от команди:

```
3
AC345VC
Citroen
C5
```

12
4
10
3.5
4

Sofia-burgas
5
3
7
6
12
11

3
AC34986VC
Citroen
C3
12
2
13
3.2
4

Sofia-Plovdiv
5
5
7
7
15
12

Те създават 2 коли и 2 пътя. Виждаме следните резултати

```
1. AC345VC
2. AC34986VC
1.Take a taxi
2.Add a taxi
3.See taxi routes
4.Back
```

```
1. Sofia-burgas
2. Sofia-Plovdiv
1.Add a route
2.Back
```

В главното меню натискаме „5.Save & exit“


Проверяваме папката на проекта и виждаме 2 нови файла „taxi.txt“ и „routes.txt“. В тях са запаметени горните таксита и маршрути.

 routes.txt - Notepad

File Edit Format View Help

5/3/5/3/12/11/Sofia-burgas/

5/5/5/5/15/12/Sofia-Plovdiv/

 taxi.txt - Notepad

File Edit Format View Help

AC345VC/Citroen/C5/12/4/10/3.5/

AC34986VC/Citroen/C3/12/2/13/3.2/