# EOOP - preliminary project

### Date:29.04.2022 Semester: Spring 2022

### Author and Group:!!!!!!!!!!!!!!

### Subject (Keyword): Fire Station

## **Description of the project**

### Overview of the project

### Fire Station have brigades consisting of firetrucks and personal and calls of fires. Station send brigades to calls.

### Class and data structures overview

### There are following classes: FireStation, FireBrigade, Person, FireTruck and Call.

### FireStation contain unlimited number of FireBrigades and Calls and method Menu in which manager call methods for add, remove, send, choose, manage and print FireBrigade and print, add, remove and choose call.

### FireBrigade contain pointer to call and unlimited number of FireTrucks and Personal and method Menu from which manager call methods to add, remove, choose, manage and print truck and personal, print call, return to station and send brigade.

### Person contain data about person and methods to print and update person data.

### FireTruck contain data about truck and methods to print and update truck data.

### Call contain data about call and unlimited number of brigades which send to call and methods to add, remove, choose and print and print call.

## **Case study (a memory map)**

**At the end of Preliminary Project.**

## **Declaration of the classes**

class FireStation

{

private:

    string name;                       // name of the fire station

    string address;                    // address of the fire station

    string phone;                      // phone number of the fire station

    vector<FireBrigade> fire\_brigades; // vector of fire brigades

    int free\_fire\_brigades;            // number of free fire brigades

    queue<Call> calls;                 // queue of calls

    void add\_fire\_brigade();

// add a fire brigade with parameters from the duty manager

    void remove\_fire\_brigade();

// Function which call FireBrigade\* chooseFireBrigade() and delete the fire brigade

    void manage\_fire\_brigade();

// Function which call FireBrigade\* chooseFireBrigade() and call FireBrigade::enter\_menu()

    void print\_fire\_brigades();

// Function which print all fire brigades in the vector0

    void print\_calls();                // Function which print all calls in the queue

    void add\_call();

// Function add a call to the queue with parameters from the duty manager

    void remove\_call();

// Function which call Call\* chooseCall() and delete the call

    void send\_brigades();

// Function which call Call\* chooseCall() and FireBrigade\* chooseFireBrigade() and call FireBrigade::send\_brigade(Call) and Call::add\_fire\_brigade(FireBrigade)

    FireBrigade \*choose\_brigade();

// Function which call print\_fire\_brigades() and ask the duty manager to choose a fire brigade

    Call \*choose\_call();

// Function which call print\_calls() and ask the duty manager to choose a call

public:

    FireStation();     // Constructor

    void enter\_menu();

// Function which ask the duty manager to choose an option and call the function which correspond to the option

};

class FireBrigade

{

private:

    vector<FireTruck> truck;       // vector of fire trucks

    vector<Person> personal;       // vector of persons

    Call \*call = nullptr;

// pointer to the call which the fire brigade is assigned to

    void add\_truck();

// add a fire truck with parameters from the duty manager

    void remove\_truck();

// Function which call FireTruck\* chooseTruck() and delete the fire truck

    void add\_person();

// add a person with parameters from the duty manager

    void remove\_person();

// Function which call Person\* choosePerson() and delete the person

    void manage\_trucks();

// Function which call FireTruck\* chooseTruck() and call FireTruck::update\_truck\_data()

    void manage\_personal();

// Function which call Person\* choosePerson() and call Person::update\_person\_data()

    FireTruck \*choose\_truck();

// Function which call print\_trucks() and ask the duty manager to choose a fire truck

    Person \*choose\_person();

// Function which call print\_personal() and ask the duty manager to choose a person

    void print\_trucks();

// Function which print all fire trucks in the vector

    void print\_personal();         // Function which print all persons in the vector

    void print\_call();

// Function which print the call which the fire brigade is assigned to

    void return\_to\_station();

// Function which return the fire brigade to the fire station

    void send\_brigade(Call \*call);

// Function which send the fire brigade to the call

public:

    FireBrigade();     // Constructor

    void enter\_menu();

// Function which ask the duty manager to choose an option and call the function which correspond to the option

};

class Person

{

private:

    string name;            // name of the person

    string surname;         // surname of the person

    string phone;           // phone number of the person

    string address;         // address of the person

    string email;           // email of the person

    string birth\_date;      // birth date of the person

    string passport\_number; // passport number of the person

public:

    Person();   // Constructor

    Person(string name, string surname, string phone, string address, string email,

string birth\_date, string passport\_number);

// Constructor with parameters

    void update\_person\_data();

// Function which ask the duty manager to update the person data

    void print\_person\_data();        // Function which print the person data

};

class FireTruck

{

private:

    string model; // model of the fire truck

    int number;   // number of the fire truck

    int year;     // year of the fire truck

    string VIN;   // VIN of the fire truck

    int mileage;  // mileage of the fire truck

public:

    FireTruck(); // Constructor

    FireTruck(string model, int number, int year, string VIN, int mileage);

// Constructor with parameters

    void update\_truck\_data();

    // Function which ask the duty manager to update the fire truck data

    void print\_truck\_data();  // Function which print the fire truck data

};

class Call

{

private:

    string address;                     // address of the call

    string type;                        // type of fire

    int priority;                       // priority of the call

    vector<FireBrigade \*> fireBrigades;

// vector of fire brigades which are assigned to the call

public:

    Call();                                          // Constructor

    ~Call();                                         // Destructor

    Call(string address, string type, int priority); // Constructor with parameters

    void add\_fire\_brigade(FireBrigade \*fireBrigade);

// Function which add a fire brigade which is assigned to the call

    void remove\_fire\_brigade();

// Function which delete the fire brigade which is assigned to the call and call FireBrigade::return\_to\_station()

    FireBrigade \*choose\_fire\_brigade();

// Function which call print\_fire\_brigades() and ask the duty manager to choose a fire brigade

    void print\_fire\_brigades();

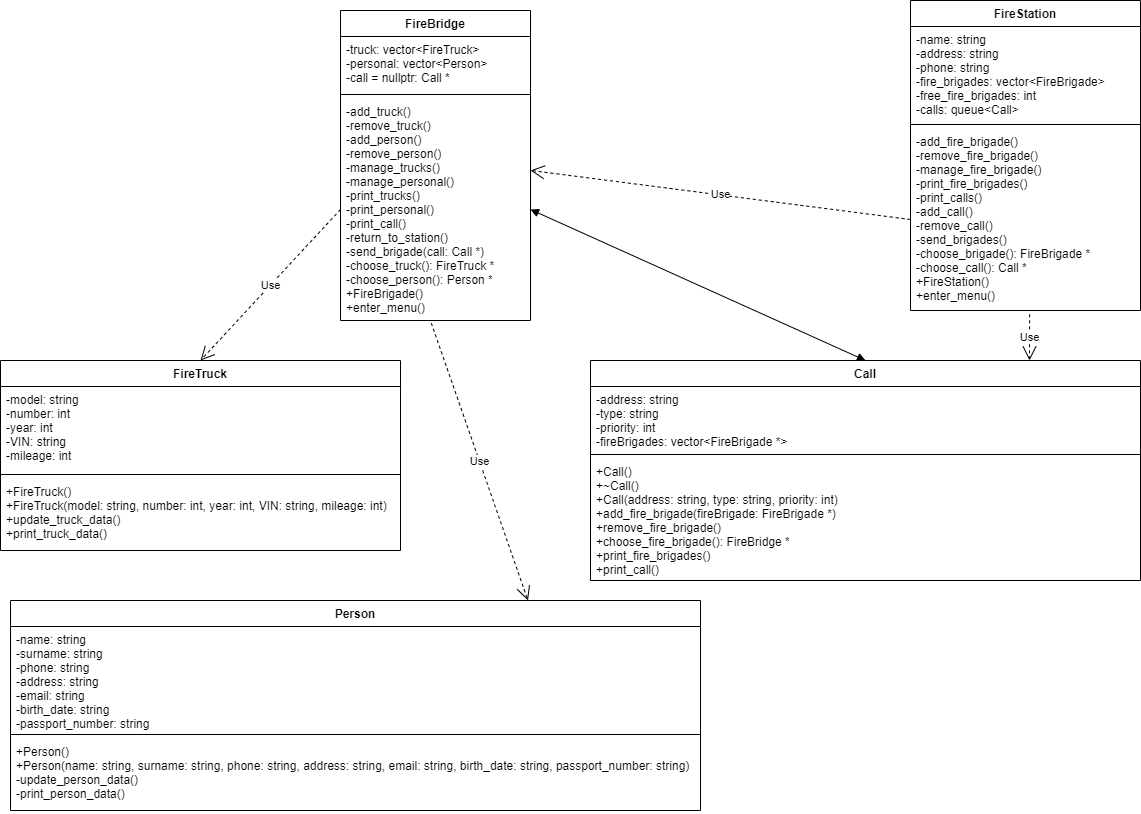
// Function which print all fire brigades which are assigned to the call

    void print\_call();       // Function which print the call

};

## **Functional test cases**

* + - 1. **Try to enter number negative or higher thin numbers in menu in FireBrigade \* FireStation::choose\_brigade(), Call \* FireStation::choose\_call(), void FireStation::enter\_menu(), FireBridge::enter\_menu(), FireTruck \* FireBridge::choose\_truck(), Person \* FireBridge::choose\_person(), Person::** **update\_person\_data(), FireTruck::** **update\_truck\_data(), FireBrigade \* Call::choose\_fire\_brigade().**
      2. **Try to enter mileage lower than before FireTruck::update\_truck\_data().**

****