Programming for Engineers

Fall 2021-2022

Assignment 4

STUDENT INFORMATION					GRADE
Session No	Student No	Name and Surname	Dept.	Signature	
1	64180006	Gülsüm İkbal Avşar	CoE - EEE		

Problem Definition

A hospital keeps track of the **Doctors** and the **Patients** each doctor is responsible for. As private information: A doctor has name, age, social security number and office number as constants, and a list of his/her patients; A patient has a name, age, and social security number as constants. For patients you also need to store number of days stayed in the hospital and daily charge.

a) Define an abstract **Person** class that has virtual print() function for printing the information about a person.
b) Define **Doctor** and **Patient** classes as derived classes of **Person**. Add to their public part the necessary functions to get and set the values of the private variables. Use a static variable in **Patient** class to keep track of the total number of patients at the hospital.
c) Add other public/friend functions to do the following tasks:
   1) print() function in each class, to print the data in an organized (formatted) way. For instance, print the information of a given patient in a .txt file, each information on a separate line with title on the left side and use "Patient Information" as the header of the form.
   2) A friend function of the **Patient** class to calculate total expenses.
   3) Two constructors in the **Patient** class: **1.** default constructor; **2.** one constructor for setting private variables.
   4) The destructor of each class. When the patient leaves the hospital, the total hospital charge is printed on the screen and the corresponding patient object is destroyed.

Program Code

```
//Define an abstract Person class that has virtual print() function for printing the
//information about a person.
class Person {
private:
    const string name;
    const int age;
    const int ss_number;
public:
    // Constructor with no parameters
    Person() : name( s: ""), age(0), ss_number(0) {}
    // Constructor with parameters
    Person(string name, int age, int ss_number) : name(name), age(age), ss_number(ss_number) {}
    virtual void printInfo(){
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
        cout << "Social Security Number: " << ss_number << endl;
    }
    //getter and setter
    string getName() { return name; }
    int getAge() { return age; }
    int getSSNumber() { return ss_number; }
};
```

Figure-1/ Person Class

```cpp
//Define Patient class as derived classes of Person.
class Patient : public Person {
private:
    int days; //number of days stayed in hospital
    double daily_charge; //daily charge
public:
    void printInfo() { //Overriden function
    }
    //constructor with parameters
    Patient(string name, int age, int ss_number) : Person( name: name, age, ss_number) {
        days = 0; //at the beginning the number of days stayed is 0
        daily_charge = 0; //at the beginning the daily charge is 0
        //print the information of a given patient in a .txt file, each information on a separate line with
        //title on the left side and use "Patient Information" as the header of the form.
        ifstream myFile( s: "patient_information.txt");
        if (!myFile) {
            ofstream myFile( s: "patient_information.txt");
            //write "Patient Information" to the file
            myFile << "Patient Information" << endl;
        }
        //open a file named "patient.txt" in append mode
        ofstream patient_file( s: "patient_information.txt", ios::app);
        //write the patient information to the file
        patient_file << "Name: " << name << endl;
        patient_file << "Age: " << age << endl;
        patient_file << "Social Security Number: " << ss_number << endl;
        patient_file << "-----------------------------------------" << endl;
    }

    //setter functions
    int getDaysStayed() { return days; }
    double getDailyCharge() { return daily_charge; }

    //getter functions
    void setDaysStayed(int days_stayed) { this->days = days_stayed; }
    void setDailyCharge(double daily_charge) { this->daily_charge = this->daily_charge+daily_charge; }


    //The destructor of each class.
    ~Patient() {
        cout << "Total hospital charges: " << daily_charge << endl; //total hospital charges
    }

    // A friend function of the Patient class to calculate total expenses.
    friend double calculate_total_expenses(Patient patient);
};

double calculate_total_expenses(Patient patient) {
    double total_hospital_charge = patient.days * patient.daily_charge;
    return total_hospital_charge;
}
```

Figure-2/ Patient Class

```cpp
//Define Doctor class as derived classes of Person.
class Doctor : public Person {
private:
    const int officeNo;
    static int num_patients;
public:
    //constructor with parameters
    Doctor(string name, int age, int social_security_number, int office_number) : Person( name: name, age, social_security_number), officeNo(office_number) {}
    //getter and setter

    const int getOfficeNo() const {
        return officeNo;
    }

    static int getNumberOfPatients() {
        return num_patients;
    }

    static void setNumberOfPatients(int numberOfPatients) {
        num_patients = numberOfPatients;
    }

    void printInfo() {
    }
};
```

Figure-3/ Doctor Class

```cpp
int main() {
    //create a hospital with at least 3 doctors and 5 patients
    vector<Doctor*> doctors;
    vector<Patient*> patients;

    //create 3 doctors
    Doctor* doctor1 = new Doctor( name: "Ahmet Polat", age: 24, social_security_number: 64180006, office_number: 546);
    Doctor* doctor2 = new Doctor( name: "Emre", age: 23, social_security_number: 64180009, office_number: 683);
    Doctor* doctor3 = new Doctor( name: "Hasan", age: 22, social_security_number: 64180008, office_number: 420);
    //using push_back method, doctors added to the doctors vector that created above
    doctors.push_back(doctor1);
    doctors.push_back(doctor2);
    doctors.push_back(doctor3);

    //create 5 patients
    Patient* patient1 = new Patient( name: "Gulsum", age: 21, ss_number: 10004090);
    Patient* patient2 = new Patient( name: "Ikbal", age: 20, ss_number: 77080022);
    Patient* patient3 = new Patient( name: "Ahmet", age: 24, ss_number: 87654321);
    Patient* patient4 = new Patient( name: "Polat", age: 23, ss_number: 12345678);
    Patient* patient5 = new Patient( name: "Hilal", age: 18, ss_number: 75315985);

    patients.push_back(patient1);
    patients.push_back(patient2);
    patients.push_back(patient3);
    patients.push_back(patient4);
    patients.push_back(patient5);
    return 0;
}
```

Figure-4/ Main Function

```
Patient Information
Name: Gulsum
Age: 21
Social Security Number: 10004090
--------------------------------------------
Name: Ikbal
Age: 20
Social Security Number: 77080022
--------------------------------------------
Name: Ahmet
Age: 24
Social Security Number: 87654321
--------------------------------------------
Name: Polat
Age: 23
Social Security Number: 12345678
--------------------------------------------
Name: Hilal
Age: 18
Social Security Number: 75315985
--------------------------------------------
```

Figure-5/ Output TXT file