BLG252E - Object Oriented Programming
Homework – 1

```
Assignment Date: 12.03.2018
Due Date       : 26.03.2018 at 23:55
```

In this assignment, you will design and implement a program according to object-oriented programming approach, which simulates a workday of civil registry for the delivery of new identity cards.

Scenario:

The government decided to introduce new identity cards in 2017 and asked that all citizens must renew until 2020. Therefore, citizens started to take an appointment with a specific time (i.e. consisting of date & hour information) from civil registries to apply for their new cards.

The reservation system allows appointments to occur in the same time slot up to 3 citizens. The time slots are between 8.00 and 16.00. Each slot is 30 minutes. The civil registry has two officers: one officer for citizens with an appointment, one officer for citizens without an appointment. Each citizen is served between her/his appointment time slot.

Citizens with an appointment must be at the civil registry office before the appointment slot starts. Citizens are served by an officer according to FIFO approach. That is, citizens with an appointment in the same time slot, are served considering when they are entered the office. Let’s say that A, B and C are three different citizens with an appointment in the same time slot, 8.00-8.30. When they enter the office, the current time is 7.59, 7.19 and 7.45 for A, B and C respectively. Therefore, an officer serves to B, C and A, in order. By this way, when each citizen’s turn comes, the officer gets his/her application and deliver the new identity card as well.

However, there are still some citizens who go to the civil registry without an appointment. Therefore, the civil registry has two lists: for citizens with an appointment and for citizens without an appointment. When a citizen without an appointment arrives at the registry, the program only considers the clock when s/he enteres the civil registry office in contrast with the citizens with an appointment. That is, citizens without an appointment, are served considering when they are entered the office. Let’s say that X, Y and Z are three different citizens without an appointment. When they enter the office, the current time is 8.50, 8.20 and 8.35 for X, Y and Z respectively. Therefore, an officer serves to Y, Z and X, in order. So, when her/his turn comes, the officer gets his/her application and deliver the new e-identity card as well.

To sum up, you will design and implement a program which can be used in the simulation of this scenario and print the order of citizens served are printed to console. For detail, please check the “Sample Scenario” section.

Implementation:

The program firstly reads all citizens from an input file(as *input.txt*), then insert them to the related lists. After all citizens are served, the program terminates.

You will firstly design a Time class with attributes as day, month, year, hour, minute in order to be used for the description of date and clock information.

CivilRegistry class has two lists (for citizens with an appointment or no without an appointment). Citizens are inserted to and removed from lists according to the scenario explained above.

Each citizen is represented by an instance of Citizen class. Each citizen has an identity number, a name, a surname, the status of having an appointment, the date of appointment, appointment slot, and the time when a citizen comes to civil registry values. The appointment date and entry time information are held in an instance of Time class. This attribute is set to null if the citizen has no appointment.

Note: An application takes 10 minutes for each citizen.

You are expected to implement following classes and functionalities:

Time:

Attributes:

- int day
- int month
- int year
- int hour
- int minute

Methods:

- **Constructor**: The constructor should take values according to attribute values.
- **getTime()**: returns the time information in the form of "hh.mm dd/mm/yyyy" (one space between minute and day).
- **< operator**: compares two Time objects (this and argument) and returns true if time value of this is earlier than the time value of the argument.
- **> operator**: compares two Time objects (this and argument) and returns true if time value of this is later than the time value of the argument.
- **== operator**: compares two Time objects (this and argument) and returns true if time value of this is equal to the time value of the argument.

CivilRegistry:

Attributes:

- List <Citizen> wApp
- List <Citizen> wOutApp

Methods:

- **Constructor**: The constructor may take values according to attribute values.
- **insertCitizen()**: Insert a new citizen to the related list. This function must take a Citizen object as an argument.
- **removeCitizen()**: Remove the citizen who is served from the related list. This function must take an int value as an argument to specify the list from which a citizen is removed.

Citizen:

Attributes:

- char *idNo
- char *name
- char *surname
- bool hasApp
- Time *appTime
- char *appSlot

Methods:

- **Constructor**: The constructor should take values according to attribute values.
- **getAppTime()**: return the appointment time (date&entryClock) of the citizen.

Implementation Notes

- Make sure that there is no memory leak in your code.
- You may want to implement additional private methods.
- Be careful with the methods/attributes that are supposed to be constant, static, private/public.
- You can add getters/setters when they are necessary; however, it is NOT allowed to add new classes or new attributes/methods to the classes.
- Use comments wherever necessary in your code.
- You are allowed using STL containers.
- Your program should compile and run on Linux environment using g++ (version 4.8.5 or later). You can test your program on ITU's Linux Server using SSH protocol. Include all necessary header files to your code. Do not use precompiled header files and Windows specific header files and functions.

Submission Notes

- You should put your class declarations and definitions in the “Time.h”, “Time.cpp”, “Citizen.h”, “Citizen.cpp”, “CivilRegistry.h” and “CivilRegistry.cpp” files respectively.
- You should also write a report to explain reasons why you need to use private/public/static/constant variables/methods in your program.

  After that, you should compress all files into an archive file named “<your_student_number>.zip”. Do NOT include any executable or project files in the archive file. You should only submit necessary files.
- Submissions are made through the Ninova system and have a strict deadline. Assignments submitted after the deadline will **NOT** be accepted. If you send your homework via e-mail, you will NOT get any points. Don’t wait until the last minute. Upload whatever you have, you can always overwrite it afterwards.
- This is not a group assignment and getting involved in any kind of cheating is subject to disciplinary actions. Your homework SHOULD NOT include any copy-paste material (from the Internet or from someone else’s paper/thesis/project). Check the “Academic honesty” section in the syllabus.
- For any questions about the assignment, contact **Tuğba PAMAY** via e-mail (**pamay@itu.edu.tr**).

Sample Scenario

- The program reads citizens’ information from an input file containing data in the form of Table 1.
- Firstly, citizens with or without an appointment are separated because they are inserted into different lists.
  - Therefore, while Tuğba, Ahmet, Talha and Berke are put into wApp list, Ali, Merve, Deniz and Lara are inserted to wOutApp list.
- Secondly, citizens are inserted to the related list according to their appointment times (date & time slot & entry clock).

  For citizens with an appointment:
  - Let’s firstly focus on citizens with an appointment: Tuğba and Ahmet (line 1 and 2) have an appointment in 8.00-8.30 time slot at 3.3.2018 and Talha and Berke (line 3 and 4) have an appointment in 8.30-9.00 slot at 03.03.2018 and 04.03.2018 respectively.
    While citizens are added to the related list, appDate information should be considered. Therefore, we are sure that Tuğba, Ahmet and Talha are definitely served before Berke.
  - Then, citizens with an appointment at the same date, are sorted according to slot values. Therefore, Tuğba and Ahmet are served before Talha.
    Tuğba also gets the service before Ahmet because she entered the office before him. Therefore, the order is Tuğba, Ahmet and Talha at 3.3.2018.

- Be careful that, even if the entryClock value of Talha is earlier than Ahmet's value; Ahmet is served before Talha due to their appointment time slots. Talha has an appointment in the time slot of 8.30-9.00, while Ahmet's appointment time slot is 8.00-8.30. Therefore, Talha cannot take the service before his time slot even if he enters the office previously.
- At the end, Berke is added to the list.
- After all citizens are added to wApp list, it consists of citizens with an appointment in this order: Tuğba, Ahmet, Talha and Berke.

For citizens without an appointment:

- For the citizens without an appointment, the sorting process is easier because they do not have a time slot constraint.
- Firstly, while citizens are added to the related list, appDate information should be considered. Therefore, we are sure that Ali, Merve and Deniz are definitely serviced before Lara.
- Then, citizens without an appointment at the same date, are sorted according to entryClock values. Therefore, the sequence of getting service is as Deniz, Merve and Ali at 03.03.2018.
- After this step, Lara is added to the related list.
- After all citizens are added to wOutApp list, it consists of citizens without an appointment in this order: Deniz, Merve, Ali and Lara.

- Program must print the order of citizens which are served. For this sample scenario, the output will be like:

```
Citizens with an appointment:
Tuğba
Ahmet
Talha
Berke

Citizens without an appointment:
Deniz
Merve
Ali
Lara
```

- After all citizens are inserted rankly and are served by considering the sequence of the list, all instances should be deleted and the program terminates.

	idNo	Name	Surname	hasApp	appDate	appSlot	entryClock
1	111	Tuğba	Esgin	TRUE	03-03-18	8.00-8.30	7.22
2	222	Ahmet	Tezmen	TRUE	03-03-18	8.00-8.30	7.59
3	555	Talha	Yardımcı	TRUE	03-03-18	8.30-9.00	7.45
4	103	Berke	Konak	TRUE	04-03-18	8.30-9.00	8.17
5	777	Ali	Fındık	FALSE	03-03-18	-	8.22
6	999	Merve	Egin	FALSE	03-03-18	-	8.13
7	101	Deniz	Yolcu	FALSE	03-03-18	-	8.09
8	100	Lara	Dinçtürk	FALSE	04-03-18	-	8.16

Table 1: The sample of input data