**Subject 1 - OOP practical test**

Time: 1h 30min

29 May 2020

1. Define 2 classes: **Person** and **Student;** there must be an inheritance relationship between the **Student** class and the **Person** class.

Each person has:

* an identification number (cannot be changed)
* a first name and last name.

In addition to these fields that should be inherited from **Person**, each student has a student enrollment number (cannot be changed). The fields from the **Person** class should be visible only to derived class instances.

Overload the insertion operator (operator <<) for these classes.

1. Define getter and setter (if needed) methods for all the fields from the **Person** and **Student** classes.
2. Now design a new class **CoffeeShop** to simulate how clients are served in the “Students come first” coffee shop. The restaurant has the following rule: anyone can buy coffee, but students have priority and will be served before regular persons.

This class should have (at least) the following methods:

* void enqueue(Person \*p) - (!polymorphic method) to simulate that a person (instance of class Person or Student) set in the queue, waiting to be served. For simplicity, we assume that all the persons arrive at the same moment.
* void serveClients(); - display the way in which the clients from the queue will be served.

1. Display how many instances of the Person and Student classes were created.   
   Hint: You might need to use static class variables for this.

Example

**If the clients of the coffee shop are:**

(Ionel Mateescu, person), (Mihai Andrei, student), (Claudia Matei, person), (Carmen Ispas, student), (Sorina Sabou, person), (Melinda Ion, student)

**they should be saved in the following order:**

S: Mihai Andrei,

S: Carmen Ispas,

S: Melinda Ion,

P: Ionel Mateescu,

P: Claudia Matei,

P: Sorina Sabou

**Grading**

Ways to lose points (max 1 point), even if your program work perfectly:

* Your class ignores encapsulation, for example you define everything as public. **(- 0.2 p)**
* You don’t use meaningful names for your class members and methods: for example, your methods are called void m1(), void m2(char \* s) **(-0.2 p)**
* You don’t use indentation and you don’t separate the implementation into a *header* and a *source* file. For example you write the class MyString in a header file on a single line. **(-0.2 p)**
* You don`t mark the required methods as *const* (for example the getters) in their declaration, to ensure that you cannot change any member data through that method. **(-0.2 p)**
* You don’t use a consistent coding style. **(-0.2p)**

**If your project does not compile, it will not be graded!**

**As a rule of thumb, don’t write more than 10 lines of code without compiling your project.**

**If your project does not compile, fix the errors and only when your project is error free you should move on to the next point.**

Grade

|  |  |
| --- | --- |
| Class declaration, constructors, destructor | 2 |
| Inheritance, correct access modifiers for variables | 1 |
| Static members (number of totally created instances) | 1 |
| Insertion operator overloading operator<< | 1 |
| Coffee shop class + enqueue method | 2 |
| Display the order in which clients are served | 1 |
| Coding style (see Ways to lose points) | 1 |
| Default | 1 |
| **Total** | **10** |

Overload stream operators:

<https://www.tutorialspoint.com/cplusplus/input_output_operators_overloading.htm>