UNIVERSIDADE DE AVEIRO
DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA
Desafio #3 de Programação por Objetos
17 de junho de 2022
Duração da apresentação: 15 minutos

Student 1 n.º _____ Name _____

Student 2 n.º _____ Name _____

- Download the exam base files from eLearning.
- Write the **student ID numbers** and **names** in comment on the first line of every file.
- The duration of the presentation is 15 minutes and consists of two parts. First the group has 10 minutes to present the developed software, starting by a usage demonstration followed by implementation details. The second part will be Q&A with the audience and the professor.
- Before you start writing your solution, read all the questions very carefully and check how the classes and their member functions are used in the main program provided.

The objective is to implement a set of classes to represent users and messages for a management system. Represent adequately all the entities requested below, creating constructors, requested getters, setters and other member functions, as well as operators that are fundamental to each class. Using the provided main program, test the developed classes incrementally, trying to reproduce the output listed at the end of this statement. (Remove the comments in the main program as you progress in your code.)

1. Adjust the implementation of the **User**, **Msg**, **EmailMsg,** and **TextMsg** classes to follow the additional requisites:
    - **Msg** class should be an abstract class;
    - **Msg** class should provide a pure virtual member function `GetType()` that should be implemented in all the derived classes - you should use an enumerate here;
    - implement a method to validate the email addresses and phone number passed to the constructors of **User** and **EmailMsg** classes (or to the corresponding setter methods). A valid e-mail should have the @ character and finish in .pt or .com, and a phone number should start with +351 and have 13 characters: in case of an invalid email, the corresponding methods should throw an exception;
    - Implement a method to validate that the body of a TextMsg has less than 40 characters passed to the constructors of **TextMsg** class or to the corresponding setter methods – The validation method should throw an exception.

Expected output for the given main file:

```
*** Part 1 ***

Type mobile: 0
Type email: 1

0
I love programming.
Mobile Msg 100 from +351234370555 to +351234370500 with content: I love programming.

1
I am ready :-)
Email 1 from a@ua.pt to m@ua.pt with content: I am ready :-)

ERROR: an email message is not a mobile message!

ERROR: User::setEmail - invalid email

ERROR: User::setMobile - invalid number

ERROR: TextMsg::TextMsg - invalid Text Message Body too long
ERROR: EmailMsg::EmailMsg - invalid src_ email address

I am ready :-)
```

2. Adjust the implementation of the **MsgManager** class in order to follow the additional requisites:

- Delete the attributes used to store Mobile Messages and Emails and consider only a single data structure to store messages that are created using dynamic memory allocation (vd. `main01.cpp - part2()`); the class must guarantee the release of this memory when the management system is destroyed; take into consideration the concept of polymorphism.
- Delete member functions **AddTextMsg** and **AddEmailMsg** and implement a new method **AddMsg(Msg *msg)** that allows using pointers to objects of type TextMsg or EmailMsg.
- Create a member function, **`saveOnFileFilter(const string &filename, int opt, const string &address)`**, to save filtered data on a text file (use the format that appears in the listing of part2, below). The function should interact with the user using a menu in order to:
    - opt = 1: print the list of emails sent to a given email address: each line contains the information about the sender and the text of the corresponding email;
    - opt = 2: print the list of emails sent by a given email address: each line contains the information about the receiver and the text of the corresponding email;
    - opt = 3: print the list of mobile messages sent to a given phone number: each line contains the name about the sender and the text of the corresponding message;
    - opt = 4: print the list of mobile messages received by a given phone number: each line contains the name of the sender and the text of the corresponding message.
- Implement a member function User GetUser(std::string mobile), to retrieve a user based on its mobile phone, and create a new member function changeMobile (User u, string new_mobile) that will update the mobile number of a user to the new_mobile number in the **MsgManager.**

Expected output for the given main file:

```
*** Part 2 ***

Users List (sorted by email):
       Ana +++ ana@ua.pt *** +351234370555
       Joao +++ joao@ua.pt *** +351234370111
       Maria +++ maria@ua.pt *** +351234370500
       Rita +++ rita@ua.pt *** +351234370000
       Ze +++ ze@ua.pt *** +351234234234
Email Messages List:
       Email 1 from maria@ua.pt to ana@ua.pt with content: The PpO challenge is today!
       Email 2 from ana@ua.pt to maria@ua.pt with content: I am ready!
       Email 3 from ana@ua.pt to ze@ua.pt with content: The content is what I studied.
       Email 4 from ze@ua.pt to maria@ua.pt with content: Classes and objects...
Mobile Messages List:
       Mobile Msg 51 from +351234370000 to +351234370111 with content: The PpO challenge is
indeed today!
       Mobile Msg 52 from +351234370555 to +351234370500 with content: I love programming.
       Mobile Msg 53 from +351234370000 to +351234234234 with content: How was the challenge
       Mobile Msg 54 from +351234370555 to +351234234234 with content: I love programming.

       Maria +++ maria@ua.pt *** +351234370500
       Maria +++ maria@ua.pt *** +351234370501
```

After the execution the following files should be created:

Contents of m1.txt:

```
Emails sent to Maria +++ maria@ua.pt *** +351234370500:
       Ana +++ ana@ua.pt *** +351234370555 -> I am ready!
       Ze +++ ze@ua.pt *** +351234234234 -> Classes and objects...
```

Contents of m2.txt:

```
Emails sent by Ana +++ ana@ua.pt *** +351234370555:
       Maria +++ maria@ua.pt *** +351234370500 -> I am ready!
       Ze +++ ze@ua.pt *** +351234234234 -> The content is what I studied.
```

Contents of m3.txt:

```
Text messages sent to Maria +++ maria@ua.pt *** +351234370500:
       Ana +++ ana@ua.pt *** +351234370555 -> I love programming.
```

Contents of m4.txt:

```
Text messages sent by Ana +++ ana@ua.pt *** +351234370555:
       Maria +++ maria@ua.pt *** +351234370500 -> I love programming.
       Ze +++ ze@ua.pt *** +351234234234 -> I love programming.
```

3. Implement a function in the main program, **LoadMessagesFromFile**, in order to retrieve, into an instance of the **MsgManager** class, the contents of a management system saved in a file. Use the text file "messages01.txt", provided in the exam base files.

Example of a file:

Messages01.txt

```
user Ze +351234234234 ze@ua.pt
user Rita +351234370000 rita@ua.pt
user Joao +351234370111 joao@ua.pt
user Maria +351234370500 maria@ua.pt
user Ana +351234370555 ana@ua.pt
email maria@ua.pt ana@ua.pt The PpO challenge is today!
email ana@ua.pt maria@ua.pt I am ready!
mobile +351234370000 +351234370111 The PpO challenge is indeed today!
email ana@ua.pt ze@ua.pt The content is what I studied.
mobile +351234370555 +351234370500 I love programming.
email ze@ua.pt maria@ua.pt Classes and objects...
mobile +351234370000 +351234234234 How was the challenge?
mobile +351234370555 +351234234234 I love programming.
```

Expected output for the given main file:

```
*** Part 3 ***
Users List (sorted by email):
       Ana +++ +351234370555 *** ana@ua.pt
       Joao +++ +351234370111 *** joao@ua.pt
       Maria +++ +351234370500 ***  maria@ua.pt
       Rita +++ +351234370000 *** rita@ua.pt
       Ze +++ +351234234234 *** ze@ua.pt
Email Messages List:
       Email 5 from maria@ua.pt to ana@ua.pt with content:  The PpO challenge is today!
       Email 6 from ana@ua.pt to maria@ua.pt with content:  I am ready!
       Email 7 from ana@ua.pt to ze@ua.pt with content:  The content is what I studied.
       Email 8 from ze@ua.pt to maria@ua.pt with content:  Classes and objects...
Mobile Messages List:
       Mobile Msg 55 from +351234370000 to +351234370111 with content:  The PpO challenge is
indeed today!
       Mobile Msg 56 from +351234370555 to +351234370500 with content:  I love programming.
       Mobile Msg 57 from +351234370000 to +351234234234 with content:  How was the
challenge?
       Mobile Msg 58 from +351234370555 to +351234234234 with content:  I love programming.
```