MICHAŁ HADRYSIAK, 253136
MICHAŁ PIECHOWSKI, 253137

THE M&M'S SHOP – DOCUMENTATION

Remark: in the text below, notation "(*filename:line_number*)" is used to indicate the point in our code to which we are referring at the moment. For example: "(*main.cpp:17*)".

As you enter our shop, you encounter the login panel in which you can choose your status as a client (either standard user, premium user or admin). Two first options are visible and does not require any additional verification, but if you want to use the shop with admin's permissions, you have to type in the special combination, which is "23" (*Manager.cpp:110*), and then enter the special password, which is "adminadmin" (*Manager.cpp:92*). Simultaneously, initial database, stored in the external file, is being uploaded so that the store is not empty (*Manager.cpp:97*).

Once you log in as your preferred user, you encounter the various possibilities which our project has to offer. We will describe those possibilities as well as the code behind them in the following part of the documentation.

STANDARD/PREMIUM USER

Note: while you are in this menu, you are in fact in the method Workplace (*Manager.cpp:58*).

1. Show Products

After entering number "1" from the keyboard, method Printing (*Manager.cpp:9*) is initialized and you are asked to choose the type of products you want to print. Currently, there are only two different types of products (and therefore three options to choose, since one of them is displaying both), but we used enum type to declare types of products (*Product.h:7*) which means that the number of types of products is easy to enlarge. After entering the proper number, method showProduct is initialized (*Menu.cpp:35 or 45*) and appropriate list of products (depending on the type of user, standard user can see only non-premium products and premium user can see only premium products) is displayed. Method showProduct is using simple getters from class DB as well as supplementary method displayHeader (*Database.cpp:31*) used to align the header row and main method showOne (*Database.cpp:17*) which includes other additional methods (checkSize (*Database.cpp:27*) and getTheLongest (*Database.cpp:95*), they are used to align shown list of products using the setw() function included in <iomanip> library.

2. Add to Shopcart

Method ATC (*Manager.cpp:18*) is combining the functionality of method addToCart (*Shopcart.cpp:19*) and user's interface. Method addToCart adds product with the ID entered by user to the dynamic array declared as a field of Shopcart class.

3. Show Shopcart

Method showShopcart (*Shopcart.cpp:41*) is using the same aligning functionality as method showProduct and it is displaying the content of the dynamic array from the Shopcart class.

4. Save Shopcart

Method SaSh (*Manager.cpp:23*) is allowing user to enter the name of the file (*.csv and *.txt extensions are supported) in which the content of shopcart should be saved. Method saveShopcart (*Shopcart.cpp:29*) is using the method saveOne (*Database.cpp:43*) for all of the IDs stored in the dynamic array as well as it is adding the summary at the end of the file: number of products bought and total price.

5. Log Out From the Program

This method is simply breaking the loop in which method Workplace is executed and allows user to go back to the Login menu.

ADMIN

Note: while you are in this menu, you are in fact in the method AdminWorkplace (*Manager.cpp:71*).

1. Show Products

The procedure here is the same as in the similar option in users menu. The only difference is the fact that now both types (premium and non-premium) of products are displayed, as written in PrintMenu (*Menu.cpp:25*).

2. Add Product to Database

The method APTD (*Manager.cpp:29*) allows admin to add new product to the database. It asks admin for the parameters of the new product and passes them to the method addProduct (*Database.cpp:9*) which simply adds new product to the vector declared as a field of DB class.

3. Delete Product from Database

The method DPFD (*Manager.cpp:43*) allows admin to delete the product from the database. It asks admin for the ID of the product and passes it to the method deleteProduct (*Database.cpp:39*) which simply deletes said product from the vector declared as a field of DB class.

4. Upload Products from File

The method Upload (*Manager.cpp:48*) allows admin to add new products to the database from the external file (with *.txt or *.csv extension). It asks for the name of the file and then initializes eponymous method (*Database.cpp:61*) which reads the content of the file and stores it in the appropriate places in the vector declared as a field of DB class with the use of various functions from <string> library. This method also uses the method endeCode (*Database.cpp:103*), but it will be explained later.

5. Save Database to File

The method SDTF (*Manager.cpp:53*) allows admin to save whole database in the external file (with *.txt or *.csv extension). It asks for the name of the file and then initializes method saveAll (*Database.cpp:49*) which saves the content of the database vector to the said file in manner which allows the very same file to be later uploaded to the shop. This method also uses the method endeCode (*Database.cpp:103*), but it will be explained later.

Note that we used ";" character in the csv-related methods as a separator because our version of Excel supported it. We know that the character "," is more popular, though. If you want to change this separator for any other, just edit initial value of the char declared as a field of DB class (*Database.cpp:10*).

6. Log Out From the Program

This method is simply breaking the loop in which method AdminWorkplace is executed and allows user to go back to the Login menu.

STRUCTURE OF CLASSES

Menu

Class Menu is virtual class from which the classes Standard_Menu, Premium_Menu, Admin_Menu are inheriting. In main.cpp, we declare the pointer-object of the class Menu and three objects of subclasses. Depending on the chosen user/admin, pointer is pointing at appropriate menu and appropriate methods are initialized in the program.

In methods showProduct (*Menu.cpp: 35, 45 and 55*) the reference to the object of the DB class is declared which allows those methods to operate on DB's methods.

Product

In this class, the parameters of products (as well as all required getter and setters) are stored.

DB

In this class, the vector consisting of objects of class Product is created. All of the methods that involves operations on files are declared here, as well as supplementary methods which are used by other classes in order to display lists of products. The operator '[]' is overloaded in this class in order to allow other classes to get information about the elements of vector.

Shopcart

In this class, the dynamic array and the reference to the DB class object is declared. This class covers creating, displaying and saving shopcarts (to the external file) with the use of the DB methods.

Manager

Manager class consists of objects from every other class (but Product). It coordinates the GUI and combines the potential of every other class.

In main.cpp, objects from all of the classes (but Product) are declared and then the method LoginPanel (*Manager.cpp:93*) is initialized and in this method the whole program takes place.

ADDITIONAL FEATURES

Encrypting and decrypting files

In order to encrypt and decrypt the content of our database (so that only inside the program it is possible to read the database), we implemented method endeCode (*Database.cpp:103*) which is initialized in Upload and SaveAll methods (data is decrypted as it enters the program and encrypted when it leaves it).

We created our own algorithm for this process basing on the ROT13 cipher, but there are some differences:

1. In our version, we divided the algorithm in two parts: one for decryption and the second for encryption. That means that there is no limitation to latin latters and the whole ASCII code is used.

2. Character “.” is omitted in the algorithm because otherwise it would be changed to the “;” character (ASCII for “.” is 46 and for “;” is 59) and uploading encrypted database as well as opening it in Excel would be problematic.