Code documentation for "Mini project 1"

This time the task was to create an inventory register for a company with at least 3 predefined offices in different countries. These countries were optional to enter, and that we would use the country's currency when entering the purchase price. There were also some requirements regarding how the inventory should be presented to the user, products that have an estimated economic life approaching 3 months should be marked in red, equipment that falls within a period of 6 months should be marked in yellow. We would also present the purchase value in SEK if it was not in SEK from the beginning.

I made a little mistake when I wrote my program code, read through the task and realized that this should not be a major problem. But when I read about it a little later, I realized that I had written some unnecessary code. There were no large amounts of code, but it felt a bit silly to throw away so I chose to expand the task a bit. I chose to enter an office / country / currency when entering equipment. However, this has been remedied so that there are a number of permanent offices to choose from. Has also added the ability to edit properties around these offices by just the speed. The code for entering offices that were a bit incorrect was easy to reuse to edit properties around the different offices, so the extra effort on my part was not so great.

Has also chosen to try to make it as easy as possible for the user when it comes to entering data, it saves data about the last entered object as you usually register several computers at once such as, it also counts the next available inventory number automatically if one is present in the system.

The program is based on the inheritance of classes, as well as the use of the date function and polymorphism. Wanted to include as much as possible of this week's lesson topics in the project, and that I also reused some of the old code from previous assignments. True to my habit, I have run with the MVC structure for handling the files. Have tried to divide the code into smaller functions to make it more readable. And that tried to write the code so that it can be easily expanded with new classes in the future or class variables. Of course this can be seen as a premium / overtime, but I have the opinion that a smaller effort from the beginning can lead to savings in the future. Admits that the imagination has been a bit bad about properties for different categories of equipment. They are just set to show my knowledge in terms of programming.

Had it been a real application, I would have made it directly as a website with database connection and with the possibility to add more categories than the predefined classes in the program. Likewise, I had added opportunities to enter an address for the offices etc.

Below are the various folders and files and their functions included in them.

Classes

> Folder that contains all definitions for all classes used by the program. What is common to all classes is in the base class Inventory.

Computer

Class used to store properties for computers within the company. It has a function that returns if the computer is portable.

setValue ()

A parameter: None a parameter

Out parameter: No out parameter

Function for updating private variables in the class.

showExtra ()

A parameter: None a parameter

Out parameter: Text string

Function that returns a text string that the computer is portable if the private variable indicating that it is just true.

Inventory

Our basic class that contains common information for all extended classes.

setValue ()

A parameter: None a parameter

Out parameter: No out parameter

Function for updating private variables in the more specific classes.

showExtra ()

A parameter: None a parameter

Out parameter: Empty text string

Function to present some extra features for a class.

MasterSettings

MasterSettings

Class that indicates some general settings in the service.

MastersettingsInventory

Class to make some default settings about how inventory ID should be designed.

MobilePhone

Class used to store properties for mobile phones within the company. It has a feature that returns if the phone supports dual SIM cards.

setValue ()

A parameter: None a parameter

Out parameter: No out parameter

Function for updating the private variable on dual SIM cards in the class.

showExtra ()

A parameter: None a parameter

Out parameter: No out parameter

Function to present if the mobile phone supports dual SIM cards.

Printer

Class used to manage printer properties within the company. It has a function that returns support for whether the printer supports PIN for printing, which is then used to retrieve the print queue from the print server. Can not think of the real name for this feature that is used among slightly larger companies.

setValue ()

A parameter: None a parameter

Out parameter: No out parameter

Function for updating the private variable if the printer supports class PIN code.

showExtra ()

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting whether the printer supports dual SIM cards.

ValidCurrency

Class to handle valid currencies in the program, of course we could use the information from freecurrency therapy instead of hard-coding these currencies that are included in the service. Currency conversion came in a little later in the coding and I feel that there is no point in adjusting this as it will not be a commercial product.

ValidOffice

Class to handle our various offices in the service

Control

CheckItemsOffice

CheckItemsOffice ()

Out parameter: True or False

Function for checking if an office has some connected equipment when closing an office. False returns if there is connected equipment connected to the office.

CheckSyntax

A class with a number of common private variables and private functions to verify the syntax for different SKUs or category ids that exist.

ClassExist (string className)

A parameter

string className, the class name we are going to verify exists

Out parameter:

Bool, true if the class exists, otherwise false.

convertFieldsToDictonary ()

A parameter: None a parameter

Out parameter: No out parameter

Converts a fieldInfo variable type to a dictonary to facilitate further coding.

checkForceHyphen ()

A parameter: None a parameter

From parameter: bool

Function to verify if received string to verify contains the correct number of hyphens, etc. Returns true if everything is okay, otherwise false.

checkMinLengthLeftSide ()

A parameter: None a parameter

Ut bool

Function to verify the minimum length of category id or for the left part if we have an article number that consists of two different parts that are tied together with a hyphen.

check (string className = null, string InputString = null)

In parameters:

className classname that we work by and its associated parameters

InputString close as we check the syntax

From parameter: bool

Main function for verifying the syntax for a category ID or inventory ID that we want to verify. Returns true if all conditions are correct otherwise false.

FileManagment

readRegister ()

A parameter: None a parameter

Out parameter: No out parameter

Function for loading our equipment and offices into the service.

saveRegister ()

A parameter: None a parameter

Out parameter: No out parameter

Function to save our equipment and offices from the service.

Model

CommonFunction

Class to handle various common functions in the service.

FirstLetterToUpper (string str)

In parameter: String that should be capitalized on the first character in the string

Out parameter: Close with the first character as a capital letter

Function for capitalizing the first character in the string that enters the function.

GetMonthsBetween (DateTime from, DateTime to)

A parameter:

DateTime from. Start date that we will compare from

DateTime to: End date to compare with

Out parameter: The difference between two incoming dates in months.

GetDifferenceInYears (DateTime startDate, DateTime endDate)

A parameter:

DateTime startDate, start date being compared

DateTime endDate end date that we will compare against

Out parameter: Number of years between the specified dates

Function that calculates the number of years between two dates and takes into account the month.

ExchangeRates

Class for retrieving the current exchange rates online, ie the rates apply internationally and not at the companies that exchange currency.

getExchangeRates ()

A parameter: None a parameter

Out parameter: No out parameter

Trying to retrieve the current exchange rates and put these in a public variable that can be used in different places in the service.

MoveInventorysToOffice

Class for moving equipment connected from one office to another.

moveInventorysToOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function for moving all equipment in one office to another, uses a global memory variable to keep track of from which office the equipment is to be moved.

SaveEditInventorys

Class to save the edited inventory to our inventory register.

saveEditInventory ()

A parameter: None a parameter

Out parameter: No out parameter

Function that updates the information in our register based on information in global memory variables.

SaveEditOffice

Class to save office editing.

saveEditOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function to save editing properties around an office.

SetupDefaultOffice

Class to set up a number of standard offices if there are none already in the service.

setupDefaultOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function that sets up a number of standard offices in the service, saving these to a global variable.

View

AddInventory

Class to add inventory to our register.

showInputInventoryID ()

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting which inventory ID to enter for the product. It retrieves the highest ID found in the register and adds 1 if possible.

showPurchaseDate ()

A parameter: None a parameter

Out parameter: No out parameter

Function to present to the user that he must enter the date of purchase of the equipment. Uses old value if possible, otherwise today's date.

showBrand ()

A parameter: None a parameter

Out parameter: No out parameter

Function for specifying the manufacturer of the equipment, if possible using previously used manufacturers from the global memory variable.

showModel ()

A parameter: None a parameter

Out parameter: No out parameter

Function to specify model of inventory, if possible use previously used model if the global variable is set!

showOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function for indicating which office the inventory is located in. Uses value from global memory if the variable is set!

showPurchasePrice ()

A parameter: None a parameter

Out parameter: No out parameter

Function for entering the purchase price in the currency that applies to the particular office. Uses value from global memory if the variable is set!

showNotebook ()

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if you enter a computer in the inventory register. It asks if it is a laptop, the default is based on global memory variable.

showDualSIM ()

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if you enter a mobile phone in the inventory register. It asks if it supports dual SIM cards, the default is based on global memory variable

showPrintPin ()

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if you enter a printer in the inventory register. It asks if it supports printing PIN code to then retrieve the print queue from the print server, the default is based on global memory variable

GetProperties ()

A parameter: None a parameter

Out parameter: No out parameter

Function for retrieving all defined variables in a class, as well as the base class itself. Used to build the order in which different input options are presented to the user.

showAddArticle ()

A parameter: None a parameter

Out parameter: No out parameter

Function that presents the different options when an inventory is to be registered. The different options presented are based on the different variables that exist in the different classes.

resetMemParameters ()

A parameter: None a parameter

Out parameter: No out parameter

Function that resets some common memory parameters. Called when we switch between different types of equipment to be registered in the service.

addArticle ()

A parameter: None a parameter

Out parameter: No out parameter

Function that is called when we are to add new inventory to our register. The first thing it asks of the user is which category of inventory to register.

EditInventory

Class for editing inventory in our register.

editInputInventoryID ()

A parameter: None a parameter

Out parameter: No out parameter

Function for changing inventory ID on a product.

editInventory ()

A parameter: None a parameter

Out parameter: No out parameter

Function that calls up some sub-functions when we have to edit an inventory in the system. This is based on the various input variables that belong to the selected inventory ID.

ListInventorys

Class used to list our various inventory by selected sort from the user.

listInventorys ()

A parameter: None a parameter

Out parameter: No out parameter

Function that presents options for how we want to sort our equipment in the service.

Based on the choice, it asks various LAMDA questions whose results are saved in a variable. For the sake of readability of the code, the print itself is in a separate function.

ManageOffice

Class with functions for managing the various offices in the service.

addOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function for adding a new office, it is the very name of the office that is meant.

showCurrency ()

A parameter: None a parameter

Out parameter: No out parameter

Function for specifying which currency applies to that particular office.

editOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function for selecting an office to edit.

showValidateEmptyOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function that is called if an office to be deleted has equipment, you are asked to specify which office the equipment is to be transferred to.

deleteOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function called when an office is to be deleted. Request information from the user which office is deleted from the inventory register.

showCountry ()

A parameter: None a parameter

Out parameter: No out parameter

Function that asks the user which country the office should be connected to.

editOfficeMenu ()

A parameter: None a parameter

Out parameter: No out parameter

Function for calling the various sub-functions when we are editing an office.

showAddOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function for calling the various sub-functions when we are adding an office.

manageOffice ()

A parameter: None a parameter

Out parameter: No out parameter

Function that presents a menu system for managing everything around offices.

Menu

Class to present our menu system for the service.

displayMenu ()

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting a flexible menu system as it is based on a list is defined in the function.

RemoveInventory

Class to present information about deleting an inventory.

removeInventory ()

A parameter: None a parameter

Out parameter: No out parameter

Function to present which equipment is to be deleted. Perform some checks around the entered string.

SearchInventory

Class to search for certain text string in our article index.

searchInventory ()

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting which string to search for and also performs the actual search and saves the result in a variable to be presented to the user later.

ShowListInventorys

Class for presenting the contents of incoming variables with data.

showListInventorys (List <Inventory> sortedListCategory, bool showReadKey = true)

A parameter:

sortedListCategory the list to be presented to the user

showReadKey if we are to present a string that you have to press a key to move on.

Out parameter: No out parameter

Function that presents a list of all equipment that has entered the function. Marks equipment that is beginning to approach the position for exchange.

Program

Main file for the program itself. Initially, a number of global variables used by the program are defined in different places

Main ()

A parameter: None a parameter

Out parameter: No out parameter

The main function itself that is called when starting the program. Starts with downloading the exchange rates, then reads different classes from the hard drive. Finally, the menu for the service is presented.