DVGB07, vt20 C#.NET 34465

Laboration 3B

Version 3

Michael Sendow
2020-04-14

INNEHÅLL

ANTAGANDEN

Jag har antagit att ingen fullständig CSV implementation behövs då jag själv bestämt för att inte tillåta semikolon eller nyradstecken i textfälten. En användare kan aldrig bestämma produktnummer eller ändra ett gammalt vilket gör att vissa scenarion som beskrivs i labben inte kan uppstå. Jag har valt att aldrig ta bort en vara utan jag markerar den som inaktiv och döljer varan istället. Icke aktiva varor går inte att köpa. Jag har valt att använda en checkbox för att aktivera skrivare. Default är den utbockad. Den använder sig av en förhandsgranskardialog och skriver ut till standardskrivare. Produktfilen och kvittofilen antags ligga i samma katalog som programmet körs ifrån. Jag har valt att stödja tre olika produkttyper; böcker, film, och musik.

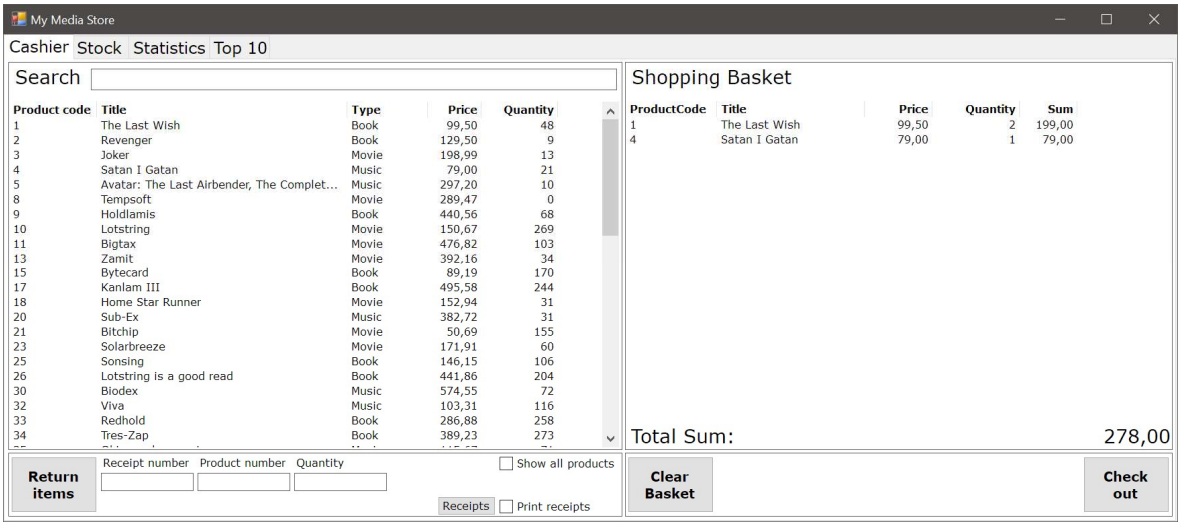
Tio-i-Topplistorna har jag antagit att man bara vill kunna se över tre olika tidshorisonter; All Time, detta år, och denna månad. Jag har valt att visa total försäljning/år/månad statistik i samma vy. Tyckte det blev renast så.

ÖVERSIKT

Programmet består av fyra olika flikar; *Cashier*, *Stock*, *Statistics* och *Top10*. *Cashier*-fliken är för kassapersonal. *Stock*-fliken är för lagerpersonal. *Statistics*- och *Top10*-flikarna är säljinformation och påverkar inte lagret. Tillsammans bör dessa flikar uppfylla kraven för version 3 av laborationen.

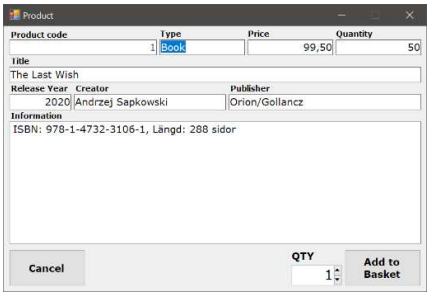
Programmet använder sig av två filer. En produktfil (products.csv) och kvittofil (sales.csv) som ligger i rot-katalogen för programmet. Jag har genererat upp 100 olika produkter och c:a 1050 olika kvitton via en sida som heter Mockaroo (www.mockaroo.com).

Cashier-fliken



Listan till vänster innehåller alla produkter i lagret. Listan går att sortera genom att klicka på de olika kolumnerna. I sökfältet går det att fritextsöka eller använda sig av nyckelordssökning. Se avsnitt Sökfunktion för mer information.

Genom att dubbelklicka på en vara i listan till vänster öppnas en ny dialog. Den nya dialogen visar all information om varan och har en väljare för hur många exemplar av varan kunden vill köpa. Om man väljer att klicka på *Add to Basket*-knappen läggs varan med valt antal upp i varukorgen. Lagret minskar med lika stort antal som lagts till i varukorgen. *Cancel*-knappen stänger dialogen utan att genomföra köp.



Under listan av varor finns möjlighet att returnera en vara. Fyll i kvittonummer, varunummer och antal varor som ska returneras och klicka på *Return items*-knappen. Om kvittot finns, varan finns på kvittot, och antalet inte överskrider antal sålda exemplar returneras varorna till lagret och produktlistan uppdateras.

Receipts-knappen kan användas för att lista alla kvitton i systemet.

Show all products-checkboxen gör så att produktlistan även visar icke aktiva produkter.

Print receipts-checkboxen aktiverar kvittoutskrift vid försäljning.

Clear Basket-knappen tömmer varukorgen och returnerar varorna till lagret.

Check out-knappen slutför försäljningen.

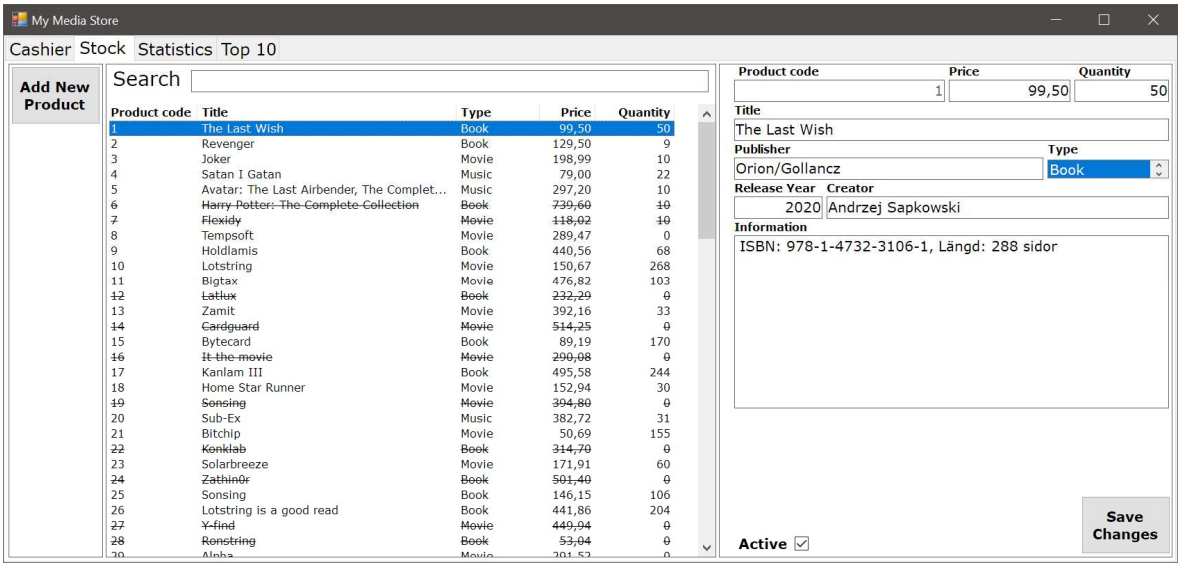
Klickar man på en vara i Shopping Basket får man upp en ny dialog. Här kan man välja en ny kvantitet. Väljer man 0 så tas produkten bort från varukorgen. Lagret samt varukorgen uppdateras med den nya kvantiteten när man klickar på *Update Quantity*-knappen.

Om varukorgen inte är tom när man försöker avsluta programmet blir man tillfrågad om man vill genomföra köpet eller inte.

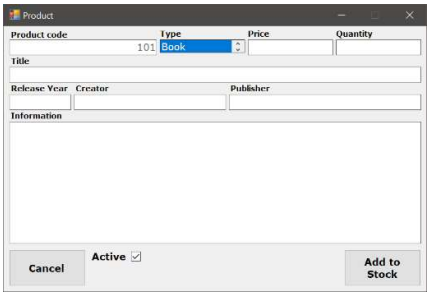
Exempel på Kvitto

```
Receipt 2020-04-14 22:26
Receipt number: 1008
#          Title                                Price        Qty                Total
-------------------------------------------------------------------------------------
3          Joker                                198,99       3                  596,97
13         Zamit                                392,16       1                  392,16
37         Opela                                270,63       1                  270,63
10         Lotstring                            150,67       1                  150,67
18         Home Star Runner                     152,94       1                  152,94
-------------------------------------------------------------------------------------
Total sum:                                                                     1563,37
```

Stock-fliken



För att lägga till en ny produkt klickar man på *Add New Product*-knappen. En ny dialog öppnas där man fyller i de värden för den nya produkten. *Product code*-fältet är det enda fält man inte kan ändra på då det tas fram från systemet. *Price, Quantity, Title* och *Release Year*-fälten är obligatoriska.

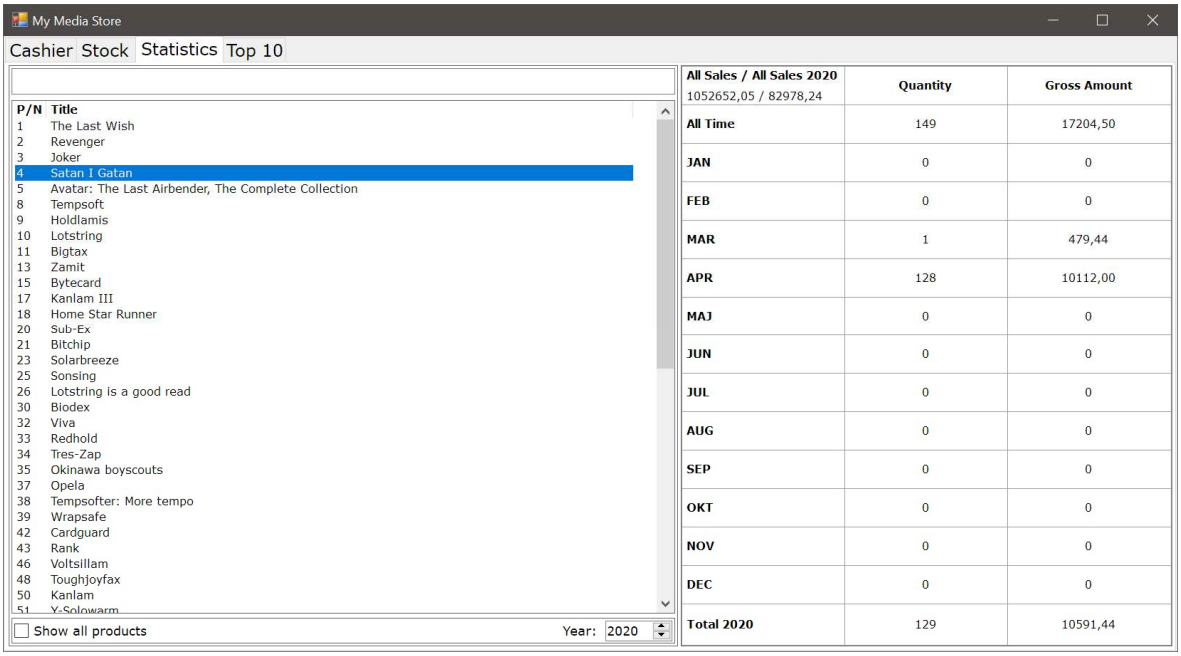


I listan väljer man den produkt man vill modifiera. Till höger i fliken visas produkten i sin helhet. Det enda fält man inte kan ändra på är P*roduct code. Active*-checkboxen visar om produkten går att handla eller inte.

Om man ändrat något fält och försöker lämna fliken, välja ny produkt eller stänga programmet blir man tillfrågad om man vill spara ändringarna.

I sökfältet går det att fritextsöka eller använda sig av nyckelordssökning. Se avsnitt Sökfunktion för mer information.

Statistics-fliken



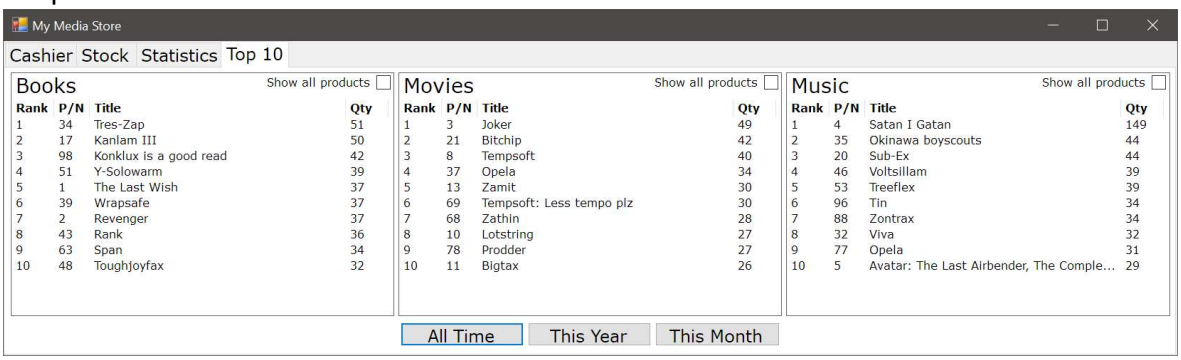
Högst upp finns en sökruta. Se avsnitt [Sökfunktion] för mer information.

I listan väljer man produkt man vill se säljstatistik över. Till höger visas all statistik för vald produkt. Längst ned till vänster finns en checkbox för att även visa icke aktiva produkter. Och till höger om den finns möjlighet att välja vilket år man är intresserad att se säljdata över.

Högst upp till vänster i säljstatistikgridden finns en summering av alla sålda varor i systemet samt den totala försäljningen av alla produkter för valt år.

All Time-raden visar total försäljning av en produkt i systemet. Raderna under är försäljning per månad för vald produkt och år. Raden längst ned visar en summering av månaderna.

Top 10-fliken



Här är 10-i-toplistor för de olika produkttyperna jag valt att ha i affären. Det går att per produkttyp välja att visa icke aktiva produkter eller inte. Efter att man bockat i eller ur en checkbox behöver man klicka på en knapp för att uppdatera siffrorna.

All Time-knappen visar lista baserad på all försäljning i systemet.

This Year-knappen visar lista baserad på all försäljning i systemet för i år.

This Month-knappen visar lista baserad på all försäljning i systemet för aktuell månad.

Alla knappar uppdaterar samtliga listor.

SÖKFUNKTION

Alla sökfält stödjer fritextsökning eller nyckelordssökning. Fritextsökningen matchar mot alla fält i en produkt. Nyckelordssökning fungerar genom att ange `nyckelord;sökord` och flera nyckelord kan användas genom att ha semikolon mellan alla söktermer.

Nyckelord

ProductCode; - Matchar mot produktnummer
Title; – Matchar mot titel.
Type; – Matchar mot produkttyp.
Price; – Matchar mot pris. Stödjer operatorer < och >.
Quantity; – Matchar mot kvantitet. Stödjer operatorer < och >.
ReleaseYear; – Matchar mot utgivningsår. Stödjer operatorer < och >.
Creator; – Matchar mot artist, författar och dylikt.
Publisher; – Matchar mot utgivare
FreeText; – Matchar mot Informations-fältet
Status; – Matchar mot produktens status.

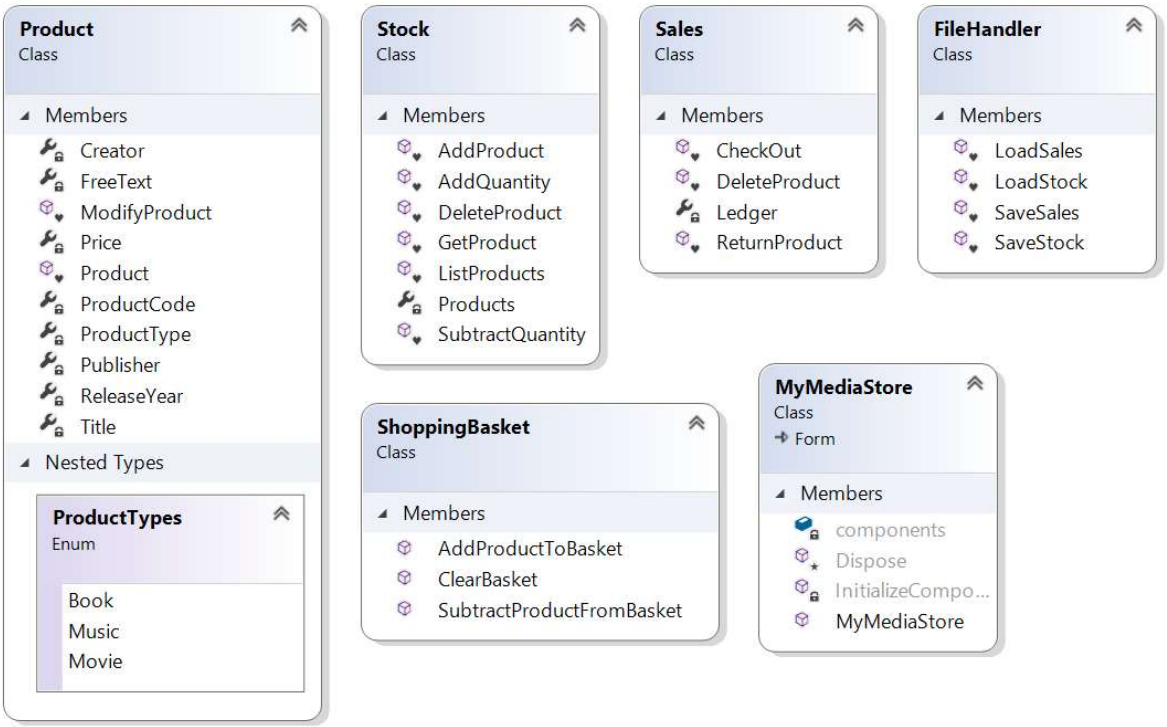
Exempel

`Quantity;5`
Returnerar alla produkter som har kvantitet 5 i lager.

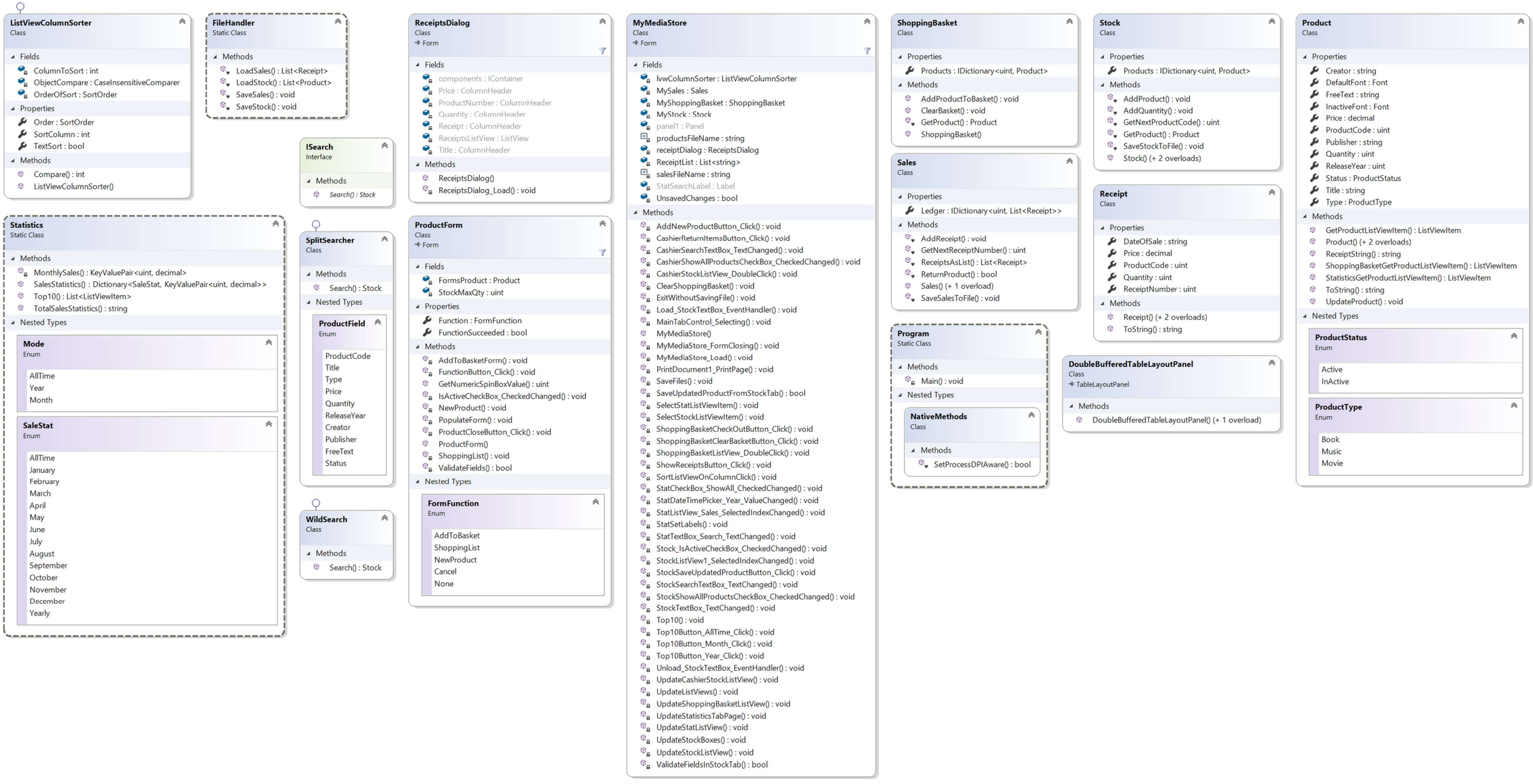
`Price;>100`
Returnerar alla produkter vars pris är större eller lika med 100

`Quantity;>10;Price;<500`
Returnerar alla produkter med en kvantitet större eller lika med 10 och vars pris är större eller lika med 500.

GAMMALT KLASSDIAGRAM



NYTT KLASSDIGRAM



IMPLEMENTATION

Applikationen består av huvudklassen MyMediaStore som ärver från Form och är huvudformulär. Ytterligare 13 klasser och ett interface är implementerade.

Klass: MyMediaStore

Huvudklass för mediabutiken. Ärver från Form. Består av ett formulär samt tillhandahåller alla metoder för att verka på formulärets kontroller.

Fält

Namn	Beskrivning
`private const string productsFileName = "products.csv";`	Sökväg till produktfil
`private const string salesFileName = "sales.csv";`	Sökväg till kvittofil
`private readonly ListViewColumnSorter lvwColumnSorter;`	Används för att kunna sortera alla ListViews
`private readonly ShoppingBasket MyShoppingBasket;`	Instans för en varukorg
`private readonly List<string> ReceiptList;`	Lista av strängar. Används vid kvittoutskrift.
`private Sales MySales;`	Instans för försäljningshistorik
`private Stock MyStock;`	Instans för varulager
`private ReceiptsDialog receiptDialog;`	Ett formulär som visar alla kvitton.
`private bool UnsavedChanges = false;`	Ett predikat för att hålla reda på om visa fält ändrats i formuläret.

Metoder

Namn	Beskrivning
`public MyMediaStore()`	Konstruktor för formuläret
`private void MyMediaStore_Load(object sender, EventArgs e)`	När formuläret laddas laddar denna metod in produktfilen och kvittofilen. Uppdaterar alla ListViews och sätter lite startvärden för statistiksidan.
`private void AddNewProductButton_Click(object sender, EventArgs e)`	Öppnar en dialog för att lägga till ny produkt.
`private void CashierReturnItemsButton_Click(object sender, EventArgs e)`	Returnerar produkt till lagret.
`private void CashierSearchTextBox_TextChanged(object sender, EventArgs e)`	Sökfunktion i Cashier-fliken
`private void CashierShowAllProductsCheckBox_CheckedChanged(object sender, EventArgs e)`	Visa/dölj inaktiva produkter i Cashier-fliken
`private void CashierStockListView_DoubleClick(object sender, EventArgs e)`	Öppnar en dialog för att köpa en produkt
`private void MainTabControl_Selecting(object sender, TabControlCancelEventArgs e)`	Kallas på när man byter flik. Kontroll för att fråga om man vill spara ändringar på en produkt om sådan finns i Stock-fliken. Ändrar tillbaka bakgrundsfärgen vid YES/NO. Cancel gör inget.
`private void MyMediaStore_FormClosing(object sender, FormClosingEventArgs e)`	Kallas på när man stänger programmet. Kontrollerar att varukorgen är tom och att inga ändringar finns att spara. Om allt är som det ska så sparas produktfil och kvittofil.
`private void PrintDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)`	Utskriftsfunktion, modifierad för att läsa från en Lista av strängar istället för från en streamreader.
`private void ShoppingBasketCheckOutButton_Click(object sender, EventArgs e)`	Skapar upp kvitto för att printa samt ett kvitto för bokföringen.
`private void ShoppingBasketClearBasketButton_Click(object sender, EventArgs e)`	Tömmer varukorgen och återbördar exemplaren till lagret
`private void ShoppingBasketListView_DoubleClick(object sender, EventArgs e)`	Öppnar dialog som ger användaren möjlighet att ändra antalet varor kunden vill köpa

`private void ShowReceiptsButton_Click(object sender, EventArgs e)`	Öppnar ett enkelt formulär för att lista alla kvitton i bokföringen. Underlättar vid demonstrationssyfte.
`private void SortListViewOnColumnClick(object sender, ColumnClickEventArgs e)`	Sorterar listview via kolumnklick
`private void Top10Button_AllTime_Click(object sender, EventArgs e)`	Uppdaterar tio-i-topplistorna med all försäljning någonsin
`private void Top10Button_Month_Click(object sender, EventArgs e)`	Uppdaterar tio-i-topplistorna med all försäljning för innevarande månad
`private void Top10Button_Year_Click(object sender, EventArgs e)`	Uppdaterar tio-i-topplistorna med all försäljning för i år
`private void StatCheckBox_ShowAll_CheckedChanged(object sender, EventArgs e)`	Visa/dölj inaktiva produkter i Statistics-fliken
`private void StatDateTimePicker_Year_ValueChanged(object sender, EventArgs e)`	Uppdaterar formuläret för att återspegla valt år i datetime-väljaren
`private void StatListView_Sales_SelectedIndexChanged(object sender, EventArgs e)`	Kallas på vid val av produkt i listan över produkter i Statistics-fliken
`private void StatTextBox_Search_TextChanged(object sender, EventArgs e)`	Sökfunktion i Statistics-fliken
`private void Stock_IsActiveCheckBox_CheckedChanged(object sender, EventArgs e)`	Checkbox i Stock-tabben för att ändra en produkts status mellan aktiv och ej aktiv
`private void StockListView1_SelectedIndexChanged(object sender, EventArgs e)`	Kallas på vid val av produkt i listan över produkter i Stock-fliken
`private void StockSaveUpdatedProductButton_Click(object sender, EventArgs e)`	Uppdaterar vald produkt i Stock-fliken
`private void StockSearchTextBox_TextChanged(object sender, EventArgs e)`	Sökfunktion i Stock-fliken
`private void StockShowAllProductsCheckBox_CheckedChanged(object sender, EventArgs e)`	Visa/dölj inaktiva produkter i Stock-fliken
`private void StockTextBox_TextChanged(object sender, EventArgs e)`	Metod för att hålla reda på förändringar.
`private void ClearShoppingBasket()`	Hjälpmetod för att tömma varukorg
`private void ExitWithoutSavingFile()`	Hjälpmetod för att stänga utan att spara filer
`private void Load_StockTextBox_EventHandler()`	Hjälpmetod för att koppla på eventhanteraren för TextChanged i Stock-fliken
`private void SaveFiles()`	Hjälpmetod för att spara produktfil och kvittofil
`private bool SaveUpdatedProductFromStockTab()`	Hjälpmetod för att spara förändrad produkt i Stock-fliken. Returnerar true om det gick bra.
`private void SelectStatListViewItem()`	Hjälpmetod vid val av produkt i Statistics-fliken
`private void SelectStockListViewItem()`	Hjälpmetod vid val av produkt i Stock-fliken
`private void StatSetLabels()`	Hjälpmetod för att sätta rätt text i Statistics-fliken
`private void Top10(Product.ProductType productType, ListView listView, CheckBox checkBox, Statistics.Mode mode)`	Hjälpmetod för att bygga upp tio-i-topplistor
`private void Unload_StockTextBox_EventHandler()`	Hjälpmetod för att koppla loss eventhanteraren för TextChanged i Stock-fliken
`private void UpdateCashierStockListView()`	Hjälpmetod för att uppdatera listview i Cashier-fliken
`private void UpdateListViews()`	Hjälpmetod för att uppdatera alla listviews
`private void UpdateShoppingBasketListView()`	Hjälpmetod för att uppdatera varukorgens listview
`private void UpdateStatisticsTabPage(Dictionary<Statistics.SaleStat, KeyValuePair<uint, decimal>> keyValuePairs)`	Hjälpmetod för att rita upp Statistics-fliken
`private void UpdateStatListView()`	Hjälpmetod för att uppdatera listview i Statistics-fliken
`private void UpdateStockBoxes(uint productCode)`	Hjälpmetod för att rita upp en produkt i Stock-fliken
`private void UpdateStockListView()`	Hjälpmetod för att uppdatera listview i Stock-fliken
`private bool ValidateFieldsInStockTab()`	Kontrollerar att alla fält för en produkt i Stock-fliken är korrekt formaterade.

Klass: Product

Klass som representerar en produkt i lagret.

Properties

Namn	Beskrivning
`public static Font DefaultFont`	Defaultfont för ett listviewitem
`public static Font InactiveFont`	Defaultfont för en inaktiv produkt i ett listviewitem
`public string Creator`	Sträng som innehåller författare eller dylikt
`public string FreeText`	Sträng som innehåller fritext om produkten, t.ex. ISBN nummer.
`public decimal Price`	Priset på produkten
`public uint ProductCode`	Produktnummer
`public string Publisher`	Sträng som innehåller utgivare.
`public uint Quantity`	Antal exemplar av produkten i lager
`public uint ReleaseYear`	Produktens utgivningsår
`public ProductStatus Status`	Statusen på produkten. T.ex. om den är aktiv eller inte.
`public string Title`	Produktens titel
`public ProductType Type`	Vilken produktkategori, eller typ, produkten tillhör.

Enum

Namn	Beskrivning
`public enum ProductStatus`	Vilka olika statusar en produkt kan ha.
`public enum ProductType`	Vilka olika produkttyper en produkt kan tillhöra

Metoder

Namn	Beskrivning
`public Product(uint productCode, string title, ProductType type, decimal price, uint quantity = 0, string creator = "", string freeText = "", string publisher = "", uint releaseYear = 0, ProductStatus status = ProductStatus.Active)`	Klass konstruktor, tar in alla enskilda properites
`public Product(Product prod)`	Konstruktor som skapar en ny produkt utifrån en annan produkt.
`public Product()`	Konstruktor som skapar en tom produkt
`public ListViewItem GetProductListViewItem()`	Skapar en ListViewItem från produkten av ProductCode, Title, Type, Price, Quantity, Status. Sätter font baserat på Status.
`public string ReceiptString()`	Skapar en sträng för kvittoutskrift
`public ListViewItem ShoppingBasketGetProductListViewItem()`	Skapar en ListViewItem från produkten av ProductCode, Title, Price, Quantity, Status. Sätter font baserat på Status.
`public ListViewItem StatisticsGetProductListViewItem()`	Skapar en ListViewItem från produkten av ProductCode och Title. Sätter font baserat på Status.
`public override string ToString()`	Override av ToString(). Returnerar strängrepresentationen av produkten.
`public void UpdateProduct(string title, ProductType type, decimal price, uint quantity = 0, string creator = "", string freeText = "", string publisher = "", uint releaseYear = 0, ProductStatus status = ProductStatus.Active)`	Uppdaterar en produkt med nya värden

Klass: Stock

Klass som representerar ett varulager.

Properties

Namn	Beskrivning
`public IDictionary<uint, Product> Products`	Dictionary som innehåller alla varor i lagret. Key = ProductCode, Value = Product

Metoder

Namn	Beskrivning
`public Stock()`	Konstruktor som skapar ett tomt lager.
`public Stock(Stock stock)`	Konstruktor som skapar en kopia av ett existerande lager.
`public Stock(string filePathName)`	Konstruktor som skapar ett lager utifrån inläsning av fil
`internal void AddProduct(Product product)`	Lägger till en produkt till varulagret
`internal void AddQuantity(uint productCode, uint quantity)`	Ökar kvantitet på produkt i lager.
`internal uint GetNextProductCode()`	Metod för att få fram nästa lediga produktnummer.
`internal Product GetProduct(uint productCode)`	Hämtar en produkt från lagret
`internal void SaveStockToFile(string filePathName)`	Sparar varulagret till fil

Klass: DoubleBufferedTableLayoutPanel

Partiell klass för att aktivera dubbel buffring av TableLayoutPanel kontrollen.

Se referenser för mera information.

Metoder

Namn	Beskrivning
`public DoubleBufferedTableLayoutPanel() : base()`	Konstruktor
`public DoubleBufferedTableLayoutPanel(IContainer container) : base()`	Konstruktor

Klass: Sales

Klass som representerar försäljning.

Properties

Namn	Beskrivning
`public IDictionary<uint, List<Receipt>> Ledger { get; set; }`	Dictionary som innehåller alla kvitton. Key = Kvittonummer, Value = lista på kvitton tillhörande kvittonumret

Metoder

Namn	Beskrivning
`public Sales()`	Skapar en tom huvudbok utan försäljningshistorik
`public Sales(string filePathName)`	Läser in försäljningshistorik från fil
`internal void AddReceipt(Receipt receipt)`	Lägger till nytt kvitto till försäljningshistoriken
`internal uint GetNextReceiptNumber()`	Metod för att få fram ett ledigt kvittonummer.
`internal bool ReturnProduct(uint receiptNumber, uint productCode, uint QtyToReturn)`	Makulerar en försäljning. Kvittonummer, produkt och kvantitet måste finnas i historiken. Returnerar true om kvittot finns, produkten finns på kvittot, och antalet inte överskrider det som finns på kvittot.
`internal void SaveSalesToFile(string filePathName)`	Sparar försäljningshistorik till fil
`internal List<Receipt> ReceiptsAsList()`	Plattar ut dictionary listorna till en lista innehållandes alla enskilda kvitton.

Klass: ShoppingBasket

Klass som representerar en varukorg.

Properties

Namn	Beskrivning
`public IDictionary<uint, Product> Products { get; set; }`	Dictionary som innehåller alla varor i varukorgen. Key = ProductCode, Value = Product

Metoder

Namn	Beskrivning
`public ShoppingBasket()`	Konstruktor som skapar en tom varukorg
`public void AddProductToBasket(Product product, uint qty)`	Lägger till en produkt i varukorgen med Quantity = qty
`public void ClearBasket()`	Tömmer varukorgen på produkter
`internal Product GetProduct(uint productCode)`	Hämtar produkt från varukorgen.

Klass: FileHandler

Statisk klass som sköter läsande och skrivande av produktfil och kvittofil

Metoder

Namn	Beskrivning
`internal static List<Receipt> LoadSales(string filePathName)`	Läser in försäljningshistorik från fil
`internal static List<Product> LoadStock(string filePathName)`	Läser in produktdata från fil
`internal static void SaveSales(Sales sales, string filePathName)`	Sparar försäljningshistorik till fil
`internal static void SaveStock(Stock stock, string filePathName)`	Sparar produktdata till fil

Klass: ListViewColumnSorter

Klass som implementerar IComparer-interfacet och används för att sortera alla ListViews. Denna klass är lånad från: https://support.microsoft.com/en-us/help/319401/how-to-sort-a-listview-control-by-a-column-in-visual-c [2020-03-21]. Jag har gjort några lätta modifieringar till den för att passa mina listviews.

Fields

Namn	Beskrivning
`private int ColumnToSort;`	Specifies the column to be sorted
`private readonly CaseInsensitiveComparer ObjectCompare;`	Case insensitive comparer object
`private SortOrder OrderOfSort;`	Specifies the order in which to sort (i.e. 'Ascending').

Properties

Namn	Beskrivning
`public SortOrder Order`	Gets or sets the order of sorting to apply (for example, 'Ascending' or 'Descending').
`public int SortColumn`	Gets or sets the number of the column to which to apply the sorting operation (Defaults to '0').
`public bool TextSort { get; set; }`	Specifies if sorting should be text or numerical.

Metoder

Namn	Beskrivning
`public ListViewColumnSorter()`	Class constructor. Initializes various elements
`public int Compare(object x, object y)`	This method is inherited from the IComparer interface. It compares the two objects passed using a case insensitive comparison.

Klass: ProductForm

Ärver från Form. Används för att visa ett generiskt formulär med produktdata som kan används till att lägga upp nya produkter, handla produkter eller ändra handlad volym.

Fields

Namn	Beskrivning
`private readonly Product FormsProduct;`	Den produkt som visas i formuläret.
`private readonly uint StockMaxQty;`	Maxvolym produkten kan ha i vissa lägen.

Properties

Namn	Beskrivning
`public FormFunction Function { get; set; }`	Vilken funktion formuläret ska fylla
`public bool FunctionSucceeded { get; set; }`	Predikat för att visa om funktionen lyckats eller ej

Enum

Namn	Beskrivning
`public enum FormFuntion`	De olika funktionerna formuläret kan anta.

Metoder

Namn	Beskrivning
`public ProductForm(Product product, FormFunction formFunction, uint maxQty = 1000000)`	Klasskonstruktor. Skapar formulär i enlighet med vald funktion
`private void FunctionButton_Click(object sender, EventArgs e)`	Om formulärfunktionen är *NewProduct* försöker vi spara produkten till FormsProduct och döljer formuläret. Vid annan funktion döljs formuläret.
`private void IsActiveCheckBox_CheckedChanged(object sender, EventArgs e)`	Om checkboxen i formuläret ändras sätter vi status på produkten till motsvarande
`private void ProductCloseButton_Click(object sender, EventArgs e)`	Om close-knappen klickas istället för Function-knappen sätts FunctionSucceeded = false
`public uint GetNumericSpinBoxValue()`	Hjälpmetod som läser av kvantitetsväljaren
`private void AddToBasketForm()`	Hjälpmetod som initierar formuläret om funktionen är AddToBasket
`private void NewProduct()`	Hjälpmetod som initierar formuläret om funktionen är NewProduct
`private void PopulateForm()`	Hjälpmetod som populerar alla kontroller i formuläret utifrån dess produkt
`private void ShoppingList()`	Hjälpmetod som initierar formuläret om funktionen är ShoppingList
`private bool ValidateFields()`	Hjälpfunktion som stämmer av att alla kontrollers värden i formuläret är giltigt formaterade. Returnerar true om så är fallet.

Klass: Receipt

Klass som representerar ett kvitto vid försäljning av en product.

Properties

Namn	Beskrivning
`public string DateOfSale { get; set; }`	Försäljningsdatum
`public uint ProductCode { get; set; }`	Produktnummer av såld vara
`public uint Quantity { get; set; }`	Antal sålda varor
`public uint ReceiptNumber { get; set; }`	Kvittonummer
`public decimal Price { get; set; }`	Priset på en vara vid försäljningstillfället.

Metoder

Namn	Beskrivning
`public Receipt(uint receiptNumber, uint productCode, string dateOfSale, uint quantity, decimal price)`	Klasskonstruktor. Skapar ett nytt kvitto bestående av inparametrarna.
`public Receipt(Receipt receipt)`	Klasskonstruktor. Skapar ett nytt kvitto utifrån ett annat kvitto.
`public Receipt()`	Klasskonstruktor. Skapar ett tomt kvitto.
`public override string ToString()`	Hjälpmetod för att representera ett kvitto som en textsträng.

Klass: ReceiptsDialog

Ärver från Form. En dialog för att visa alla kvitton i systemet. Inget krav för uppgiften men underlättar för att testa retur-funktionaliteten.

Metoder

Namn	Beskrivning
`public ReceiptsDialog(Stock stock, Sales sales)`	Klasskonstruktor. Skapar ett nytt kvitto bestående av inparametrarna.
`private void ReceiptsDialog_Load(object sender, EventArgs e)`	Centrerar dialogen över dess "parent" vid laddande av dialog.

Interface: ISearch

Kanske inte helt nödvändigt att använda sig av interface här men jag ville prova. Det är ett interface till att söka i ett object av klassen Stock.

Metoder

Namn	Beskrivning
`public ReceiptsDialog(Stock stock, Sales sales)`	Sökfunktion för att söka i lager Stock = Lagret att genomsöka searchString = Sträng att söka efter i lager Returnerar ett lager som matchar sökströngen

Klass: SplitSearcher

Klass som implementerar ISearch-interfacet. Delar upp söksträngen i nyckelord och söker igenom varje produkt med avseende på dem. Delar upp strängen med avseende på semikolon som skiljetecken.

Enum

Namn	Beskrivning
`public enum ProductField`	De olika properties som finns i ett objekt av klassen Product

Metoder

Namn	Beskrivning
`public Stock Search(Stock stock, string searchString)`	Sökfunktion för att söka i lager Stock = Lagret att genomsöka searchString = Sträng att söka efter i lager Returnerar ett lager som matchar sökströngen

Klass: WildSearch

Klass som implementerar ISearch-interfacet. Delar upp söksträngen i nyckelord och söker igenom varje produkt med avseende på dem. Delar upp strängen med avseende på semikolon som skiljetecken.

Enum

Namn	Beskrivning
`public enum ProductField`	De olika properties som finns i ett objekt av klassen Product

Metoder

Namn	Beskrivning
`public Stock Search(Stock stock, string searchString)`	Sökfunktion för att söka i lager Stock = Lagret att genomsöka searchString = Sträng att söka efter i lager Returnerar ett lager som matchar sökströngen

Klass: Statistics

Statisk klass för att generera försäljningsstatistik från ett lager och försäljningshistorik

Enum

Namn	Beskrivning
`public enum Mode`	Vilken typ av statistik vill man räkna på: AllTime, Year, Month
`public enum SaleStat`	Vilken period gäller statistiken för. Består av 14 olika värden.

Metoder

Namn	Beskrivning
`public static Dictionary<SaleStat, KeyValuePair<uint, decimal>> SalesStatistics(uint productCode, DateTime dateTime, Stock stock, List<Receipt> receiptList)`	Beräknar försäljningsstatistik för en produkt i ett lager. KeyValuePair är av formen: Quantity, GrossAmount
`public static List<ListViewItem> Top10(Stock stock, List<Receipt> receiptList, Product.ProductType type, bool showOnlyActive, Mode mode)`	Metod för att bygga upp en lista av ListViewItem för att visa tio-i-topp information.
`public static string TotalSalesStatistics(Stock stock, List<Receipt> receiptList, DateTime dateTime)`	Metod som returnerar en textsträng som visar totalförsäljning och årligförsäljning av hela lagret.
`private static KeyValuePair<uint, decimal> MonthlySales(IEnumerable<Receipt> receipts, SaleStat key)`	Sammanställer månatlig försäljningsstatistik

PROBLEM OCH SAMMANFATTNING

Det tog ett tag att värka fram hur jag ville att programmet skulle se ut och fungera. Jag började med säljpersonalens flik och funktionalitet. Jag ville inte ha en separat vy eller funktion för att lista alla produkter utan jag ville visa det direkt i fliken. ListViews blev den kontroll jag landade i. Hittade en funktion för att kunna sortera dessa och gjorde en liten ändring i den för att kunna sortera vissa kolumner som numeriska och andra som text.

Jag tycker att GUI delen blev bra efter att jag fått bukt med TableLayoutPanels. Fick skapa en partiell klass som har stöd för dubbelbuffring. Jag har försökt göra det intuitivt för användaren genom att ändra färg på bakgrunden i Stock-fliken om man har osparade förändringar. Samma sak i Top-10 fliken så byter jag färg på den knapp man tryckt på så att man vet vad det är för listor man tittar på.

Jag la in en funktion för att kunna lista alla kvitton, mest för att kunna hitta valida kvitton utan att behöva öppna filen för kvitton och titta i den för att prova returfunktionaliteten. Utskriftsfunktionen är väldigt rudimentär men fyller sin funktion. Tog hjälp av en sida på nätet och modifierade den för att skriva ut från en lista av textsträngar istället för en fil.

Det blev många ListViews att hålla reda på och jag är till exempel inte helt nöjd med att ha lagt metoder i Products för att skapa olika ListViewItems. Den funktionaliteten skulle jag vilja flytta till en statisk klass som bara hanterar ListViews och ListViewItems.

I efterhand upptäckte jag DataGridView och BindingList som skulle vara ett bättre sätt att göra detta på. Och det är något jag skulle ändrat till om jag kommit på det tidigare i utvecklingen och haft tid att designa om.

Statistikfunktionaliteten gjorde att jag önskade att jag hade all data i SQL. Tills jag läste om LINQ och löste det via några LINQ uttryck. LINQ kändes väldigt kraftfullt och för en som är van vid SQL var det inte så svårt att ta till sig.

Trots alla klasser jag skapat och funktioner i de klasserna så är huvudformuläret över 1000 rader. Hade jag haft mera tid skulle jag vilja titta på vad som skulle kunna flyttas till de andra klasserna eller brytas ut till egna klasser. Jag skulle även vilja renodla klasserna så att de inte är så hårt knutna till huvudformuläret.

Jag tycker det funkade bra att använda klassen ProductForm som en generell dialog för olika ändamål genom att i konstruktorn för klassen ange önskad funktion. Även sättet att koppla på och av eventhanterare fungerade bra för att slippa hålla koll på om textboxar ändras programmatiskt eller av en användare. På det sättet behöver jag inte ha onödiga dialoger utan kan använda samma textboxar både för in- och utmatning.

Nyckelordssökningen tycker jag blev bra. Tacksamt att använda semikolon som ”trigger” för nyckelord då jag vet att produkterna inte kan innehålla det tecknet. Kanske inte helt nödvändigt att använda interface i det här fallet men jag ville testa på det.

Mitt klassdiagram för labb 3A tycker jag fortfarande håller för Version 1 av labben. Några metoder skulle tillkomma och några skulle inte behövas. Men i stora drag skulle det fungera.

Jag har använt mig mycket av Dictionary i denna labb och tycker det är ett smidigt sätt att gruppera data på särskilt när man kan ha KeyValuePairs som Value i ett dictionary.

REFERENSER

Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel
https://www.mockaroo.com/
[2020-04-04]

Printing Text File in C#
https://www.c-sharpcorner.com/article/printing-text-file-in-C-Sharp/
[2020-03-29]

How to sort a ListView control by a column in Visual C#
https://support.microsoft.com/en-us/help/319401/how-to-sort-a-listview-control-by-a-column-in-visual-c
[2020-03-21]

How to configure an app to run correctly on a machine with a high DPI setting (e.g. 150%)?
https://stackoverflow.com/questions/13228185/how-to-configure-an-app-to-run-correctly-on-a-machine-with-a-high-dpi-setting-e
[2020-02-20]

Double Buffered layout panel - removes flicker during resize operations.
https://www.richard-banks.org/2007/09/how-to-create-flicker-free.html
[2020-04-19]

Double Buffered Table Layout Panel
https://docs.telerik.com/devtools/winforms/knowledge-base/double-buffered-table-layout
[2020-04-19]