OBS! Värdehandling!
Spara laborationsrapporten tills resultatet är registrerat i LADOK.

Laborationsrapport

Karlstads universitet

Kurs: DVGB07.....**C#.Net**.....
Kurskod Kursnamn Temin

Uppgift nr: 3B..... **Ansvarig lärare:**.....

Anm

Lab grupp
nr:.....

Medarbetare:

1) **Simon Köhlström**.....
Namn Personnummer

2).....
Namn Personnummer

3).....
Namn Personnummer

4).....
Namn Personnummer

Laborationen inlämnad: 14/04-13.....
Datum

OBS! Värdehandling!
Spara laborationsrapporten tills resultatet är registrerat i LADOK.

Härmed intygas på heder och samvete att detta arbete är utfört av undertecknade.

1) Simon Köhlström.....
Namnteckning

2).....
Namnteckning

3).....
Namnteckning

4).....
Namnteckning

Retur 1: ....
Åter senast Sign

Retur 2: ....
Åter senast Sign

Godkänd:.....
..
Datum

.....
Namnteckning

Innehållsförteckning

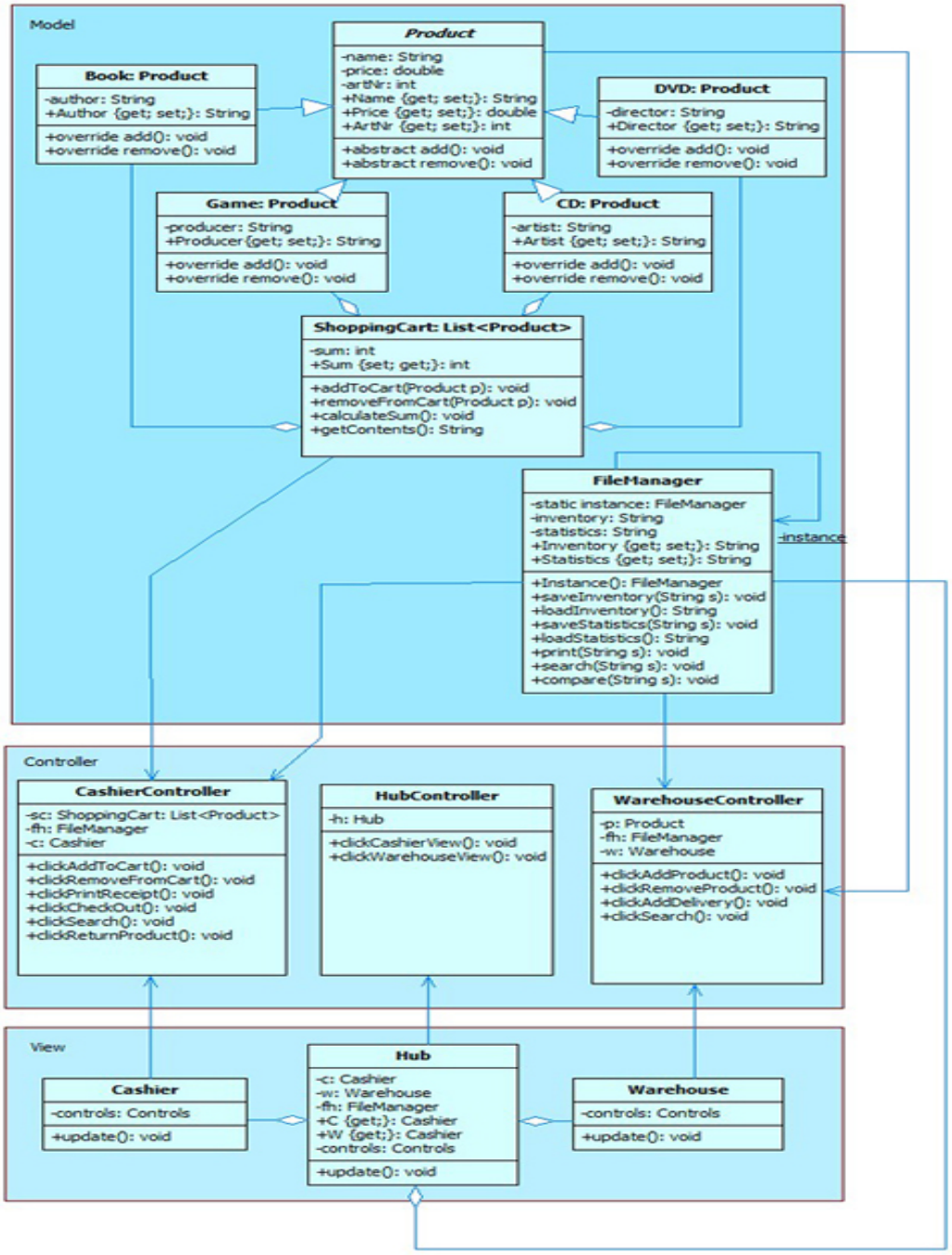
1. Antaganden

Det antaganden jag fått göra inkluderar bland annat strukturen på formulären vid lagerhantering. Både vid tillägg av ny artikel, borttagning och inleverans.
Jag antog att en flikstruktur för tillägg av ny produkt vore det bästa och sedan använda artikelnummer som ”handle” för borttagning och inleverans. Jag fick också göra ett antagande om huruvida listning av produkter skulle ske direkt i vyn, eller om man skulle trycka upp en ruta för det och hur ofta listan ska uppdateras. Jag valde att ha det direkt i vyn och uppdatera listan vid ändringar i fil samt vid start av program och byte av vy.
Jag gjorde även ett antagande om att produkterna skulle sparas till fill enligt strukturen (exempelvis) ”Typ: artnr . namn . pris . artist . mängd”. Jag antog också att det bästa sättet att spara ned produkterna till fil enligt en given form var att instansiera objekt av produkterna och spara ned till fil genom att hämta dess attribut, både på kassasidan och lagersidan.

Eftersom vart sökfunktionen inte var specificerad vart den skulle ligga så antog jag att en ruta att öppna för den vore det bästa. Även här fick jag göra antagande om strukturen på formuläret.
Formuläret för hantering av kundvagnen i kassavyn var inte heller specificerad, så där fick jag anta hur strukturen skulle vara likväl.

2. Översikt

Det här programmets syfte är att hantera, sälja och lagerhålla mediaprodukter.



Hub:

Denna klass sammanfogar de olika vyerna som programmet har. Huben har också en HubController kopplad till sig. Denna för att kunna hantera händelser när programmet laddas för första gången och när vyerna skiftas, eftersom saker måste uppdateras då
På kassasidan är det CashierView som håller bygger upp vyn. Den har vissa metoder för att uppdatera sig självt och lämna ut data från sig självt utifrån.
Detsamma gäller WarehouseView på lagersidan.

Lagersidan:
Genom lagervyn har man tillgång till att lägga till nya artiklar i sortimentet, lägga in inleveranser och ta bort artiklar ur sortimentet. Vyn har en WarehouseController kopplad till sig som hanterar alla knapptryck. Beroende på vilket formulär som det gäller utför controllern operationer mot en modellklass(FileManager) som agerar filhanterare och utför operationer anropade från controllern. Vid tillägg sparas artikel ned till fil, vid inleverans uppdateras existerande post i fil och vid borttagning raderas existerande post i fil. Produkter kommer även att instansieras i controllern för att erbjuda ett kontrollerat sätt att få in data till fil i rätt format. Baserat på vilken flik i vyn som är aktiv kommer en produkt av motsvarande typ instansieras vid tillägg och dess attribut sedan sparas ned till fil.

Kassasidan:
I den här vyn har man tillgång till att lägga till i kundvagn, ta bort ur kundvagn, genomföra köp(och skriva ut kvitto) och söka efter produkter. Vyn har en CashierController kopplad till sig som hanterar alla knapptryck. Beroende på vilket formulär det gäller utför controllern operationer mot en modellklass(FileManager) som agerar filhanterare och utför operationer anropade från controllern. Lagersidan och kassasidan arbetar för övrigt mot samma instans av FileManager. Vid försäljning av varor uppdateras poster i fil med de nya subtraherade mängderna. Vid returnering av vara ändras en post i fil genom att mängden för den räknas upp med ett.
CashierController har även en relation till modellklassen ShoppingCart (som i sin tur har en relation till produktklasserna i modellen) som är en lista som håller produkter av godtycklig typ. Denna används för att instansiera produkter och lägga till dessa i sig självt. Parametervärden för instansieringen hämtas från filinnehåll Genom listan är det lätt att lägga till och ta bort produkter, varefter vyn uppdateras. I den här klassen räknas även summa ut på samtliga artiklar.

Genom kassavyn har man även tillgång till att trycka upp sökfunktionen. Det är ett nytt formulär som kommer upp och erbjuder att man fyller i att söka på antingen artnr, namn eller författare/artist/regissör/producent. Man kan söka på alla samtidigt, men det ger dubletter. Sökdialogen kan ligga kvar på skärmen genom hela exekveringen av programmet om så önskas.

3. Detaljerad beskrivning

Model

- **Product**
  - En abstrakt klass som innehåller variablar och attribut för artikelnummer, namn, pris och mängd. Attributen är limiterad i hur de kan sättas. Ett namn måste ha en längd och får inte innehålla siffror. Ett artikelnummer måste bestå av 6 siffor. Ett pris måste ha ett värde.
  - **Subklasser**
    - **Book: Product**

- Ärver från product. Har egen variabel och attribut för författaren till en specific bok. Instansiering kräver att alla ärvda attribut och det egna attributet sätts.

- **Cd: Product**
  - Ärver från product. Har egen variabel och attribut för artisten på en specific skiva. Instansiering kräver att alla ärvda attribut och det egna attributet sätts.
- **Dvd: Product**
  - Ärver från product. Har egen variabel och attribut för regissören till en specific film. Instansiering kräver att alla ärvda attribut och det egna attributet sätts.
- **Game: Product**
  - Ärver från product. Har egen variabel och attribut för producenten till ett specifict spel. Instansiering kräver att alla ärvda attribut och det egna attributet sätts.

- **ShoppingCart: List**
  - Ärver från ”List” och är således en lista, typad till att bestå av en godtycklig samling produkter.
  - **Metoder:**
    - **void addToCart(Product p)**
      -Tar en produkt som invärde och lägger till den i listan
    - **void addToSum(double i)**
      -Tar ett decimaltal som invärde och adderar det till attributet för summa.
    - **String getContents()**
      -För varje produkt som för tillfället finns i listan tilldelas en sträng attributen för produkten. Typen av produkt avgörs så att de unika attributen även kan läggas till.
    - **bool hasProduct(String s)**
      -Itererar genom listan och söker efter en produkt med ett artnr som matchar det i det inkommande invärdet. True returneras om produkten hittas, annars false.
    - **Product getProduct(String s)**
      -Itererar genom listan och söker efter en produkt med ett artnr som matchar det i det inkommande invärdet. Produkten i listan returneras om produkten hittas, annars returneras null.

- **FileManager**
  -Skapad utefter singletonmönstret. Har en variabel som håller en statisk instans av sig självt, en privat konstruktor och en statisk metod som instansierar klassen, fast enbart om en instans inte redan finns. Detta garanterar enbart en instans av klassen. Andra klassvariablar innefattar en StreamWriter (StreamReader ligger lokalt i metod) och strängar för lagring av filinnehåll och produktsträngar, inklusive en lista för förenklad utsökning och hantering. Klassen har ett attribut för hämtning av filinnehåll.
  - **Metoder:**
    - **void convertAndSave(String type, int artnr, String name, double price, String unique, int amount)**
      -Anropas i controller och tar invärden från en produkt. Dessa konverteras till en sammansatt sträng. Om artikeln inte redan finns så anropas funktionen save(String s), men den sammansatta strängen som invärde. Därefter rensas litstan i klassen, filen återöppnas och hämtar således in den nya sparade raden. Sedan sorteras listan.

- **void Save(String s)**
  -Sparar strängen från converAndSave olika typer av streamwriter instansieras beroende om filen för tillfället innehåller några produkter. Sedan skrivs strängen till fil och strömmen stängs.
- **void open()**
  -Instansierar en StreamReader. Läser sedan in samtliga rader från fil och lägger till i lista. Sedan sorteras listan. En sammansatt textsträng motsvarandes filinnehållet tilldelas också. Sedan stängs strömmen.
- **bool search(String s)**
  -Tar en sträng som invärde och söker igenom filinnehållet efter matchande innehåll. Returnerar true om något hittas, annars false.
- **updatePost(String artnr, String amount)**
  -Tar ett artikelnummer och mängd som invärde. Söker först igenom om artikelnummret finns i fil. Om så instanseras en Streamwriter och en strängarray deklareras. Sedan itereras filinnehållet igenom. På index där matchande artikelnummer hittas tas den strängen och splittas upp i sina olika attribut. Invärdet för mängd och den existerande mängden adderas och sedan byggs strängen upp igen på indexet. Listan sorteras och samtliga rader sparas ned till fil. En sammansatt textsträng motsvarandes filinnehållet tilldelas också. Sedan stängs strömmen.
- **List<String> getContents()**
  - returnerar listan med produkter.
- **void remove(String s)**
  - Tar ett artikelnummer som sträng, instansierar streamwriter. Söker igenom filinnehåll efter matchande artikelnummer. Om hittat så tilldelas en int värdet av mängden som ligger i strängen på aktuellt index.
  Sedan kontrolleras att värdet på mängden inte är noll, eftersom man skall få en varning om det finns varor av artikeln i lager. Om mängden inte är noll kommer alltså en bekräftelseruta upp. Om användaren svarar ja så raderas post där artikelnummer matchar. Om mängden var noll så görs detsamma fast utan bekräftelserutan. När produkten tagits bort sorteras listan, sedan skrivs alla rader ned till fil. Därefter stängs strömmen.
- **String getProduct(String s)**
  -Tar ett artikelnummer som en sträng. Söker sedan igenom filinnehåll för matchande artikelnummer. En sträng tilldelas värdet av index där matchning hittas. Den tilldelade strängen returneras sedan.
- **void Print(String s)**
  - Instansierar och initierar ett PrintDocument. Instansierar sedan en PrintDialog som tar dokumentet som underlag för utskrift. Den inkommande strängen ges till printdokumentet. Sedan visas dialogen och väntar på bekräftelse. Om ja så försöks utskrift göras.

View

- **Hub: Form**
  - Innehåller grunden för sidan. Huvudfönster innehållandes flikar för de två vyerna skapas här.
  De andra vy-klasserna är aggregerade här och filhanteraren instansieras. Filhanteraren kopplas till vyerna och en hubcontroller instansieras. Vyerna för kassa och lager samt

filhanteraren har en koppling till hubcontrollern. Eventhanterare för skifte mellan flikar och laddning av hub vid start sätts.

- **WarehouseView: Panel**
  - Innehåller komponenter som bygger upp vyn samt metoder för att hämta data och uppdatera vyn med information.
  - **Metoder:**
    - **void createControls()**
      -Instansierar alla kontroller
    - **void initializeControls()**
      -Sätter egenskaper för kontroller
    - **void setEventHandler()**
      -Knyter händelsehanterare till knappar.
    - **String getActiveTab()**
      -Hämtar den för tillfället aktiva fliken för formuläret där man lägger till produkter
    - **int getProductArtNr()**
      -Baserat på aktiv flik, hämtar motsvarande artikelnummer. Kontrollerar om artikelnummret har ett värde och består av enbart siffror.
    - **String getProductName()**
      -Baserat på aktiv flik, hämtar motsvarande artikelnamn. Kontrollerar att namn har ett värde och inte innehåller punkter.
    - **double getProductPrice()**
      -Baserat på aktiv flik, hämtar motsvarande artikelpris. Kontrollerar att pris har ett värde och består av siffror.
    - **String getProductUnique()**
      -Baserat på aktiv flik, hämtar motsvarande författare/artist/regissör/producent. Kontrollerar att namnet har ett värde och inte innehåller punkter.
    - **int getProductAmount()**
      -Baserat på aktiv flik, hämtar motsvarande artikelmängd. Kontrollerar att mängd har ett värde och enbart består av siffror.
    - **int getDeliveryArtNr()**
      -Hämtar artikelnummer från inleverans-formuläret. Kontrollerar att artikelnummer enbart består av siffror, att det inte är för långt och att det har ett värde.
    - **int getDeliveryAmount()**
      -Hämtar artikelmängd från inleverans-formuläret. Kontrollerar att artikelmängd enbart består av siffror, att det inte är för långt och att det har ett värde.
    - **int getRemoveArtNr()**
      -Hämtar artikelnummer från ta bort produkt-formuläret. Kontrollerar att artikelnummer enbart består av siffror, att det inte är för långt och att det har ett värde.
    - **Void getErrorMessage(String s)**
      -Hämtar ett meddellande med innehåll baserat på invärde.
    - **void updateProducts(String s)**
      -Uppdaterar textboxen med listan på produkter baserat på invärde.

- **CashierView: Panel**
  - Innehåller komponenter som bygger upp vyn samt metoder för att hämta data och uppdatera vyn med information.
  - **Metoder:**
    - **void createControls()**
      -Instansierar alla kontroller
    - **void initializeControls()**
      -Sätter egenskaper för kontroller
    - **void setEventHandler()**
      -Knyter händelsehanterare till knappar.
    - **int getArtNr()**
      -Hämtar artikelnummer från formuläret. Kontrollerar att artikelnummer enbart består av siffror, att det inte är för långt och att det har ett värde.
    - **int getAmount()**
      -Hämtar artikelmängd från formuläret. Kontrollerar att artikelmängd enbart består av siffror, att det inte är för långt och att det har ett värde.
    - **Label getPriceLabel()**
      -Hämtar etiketten som visar totalsumman av produkterna.
    - **String getCartContent()**
      -Hämtar sträng som motsvarar innehållet i kundvagnen.
    - **void getErrorMessage(String s)**
      -Hämtar ett meddellande med innehåll baserat på invärde.
    - **void getInputBox()**
      -skapar dialogrutan för sökfunktionen
    - **void hideSearch()**
      -Döljer dialogrutan för sökfunktionen
    - **void showSearch()**
      -Visar dialogrutan för sökfunktionen
    - **String getArtnrSearch()**
      -Hämtar artikelnummer från textboxen i dialogrutan för sökfunktionen
    - **String getNameSearch()**
      -Hämtar artikelnamn från textboxen i dialogrutan för sökfunktionen
    - **String getUniqueSearch()**
      -Hämtar författare/artist/regissör/producent från textboxen i dialogrutan för sökfunktionen
    - **void updateProducts(String s)**
      -Uppdaterar listan på produkter i vyn baserat på invärde.
    - **void updateCart(String s)**
      -Uppdaterar listan med innehåll i kundvagnen baserat på invärde
    - **void updateSearch(String s)**
      -Uppdaterar textrutan i dialogrutan för sökfunktionen med resultat av sökning baserat på invärde.

Controller

- **HubController**
  -Styr händelser vid start av program samt skifte mellan vyer.
  Asscoscierar med CashierView, WarehouseView och FileManager.
  - **Metoder:**

- **void onLoad(Object sender, Eventargs e)**
  -Öppnar innehåll från filhanterare, uppdaterar både kassa och lagervy med innehållet.
- **void onChange(Object sender, Eventargs e)**
  -Uppdaterar både kassa och lagervy med innehåll i fil vid byte av flik.

- **WarehouseController**
  -Styr händelser från knapptryckningar från WarehouseView. Assoscierar med WarehouseView och FileManager.
  - **Metoder:**
    - **void addClick(Object sender, Eventargs e)**
      -Baserat på aktiv flik i vyn försöks ett objekt av motsvarande produkttyp att instansieras. För att sedan kunna spara ned dess attribut till fil i FileManager och uppdatera vyn med den sparade produkten. Om instansieringen misslyckas visas ett varningsmeddelande.
    - **void deliveryClick(Object sender, Eventargs e)**
      -För inleverans. Kontrollerar först att angivet artikelnummer i vyn är sex siffror långt. Om det är det så kollas sedan om mängden angiven i vyn är skiljt i från noll. Om det även är så anropas metod för att uppdatera post i fil från FileManager, sedan uppdateras listan med produkter i vyn med filinnehållet.
    - **void removeClick(Object sender, Eventargs e)**
      -För borttagning av produkt. Kontrollerar att angivet artikelnummer i vyn är sex siffor långt. Om det är det så anropas metod för att ta bort post ur fil. Därefter uppdateras vyn.

- **CashierController**
  -Styr händelser från knapptryckningar från WarehouseView. Assoscierar med WarehouseView och FileManager.
  - **Metoder:**
    - **void addClick(Object sender, Eventargs e)**
      -Vid tillägg i kundvagnen. Kollar först om artikelnummer angivet från vyn är sex siffror långt och att mängden angivet i från vyn inte är noll. Om så söks filen igenom med metod från FileManager och matchar artikelnummer.Där matchning hittas splittas den motsvarande produktsträngen upp för att separera dess attribut. Därefter kontrolleras det att artikeln i fråga inte redan finns i kundvagnen. Finns den inte så instansieras en ny produkt av lämplig typ och läggs till i kundvagnen. Sedan uppdateras etiketten som håller totalsumman i vyn med ny summa och listan på produkter uppdateras för att innehålla den nytillagda produkten. Om artikeln i fråga istället redan fanns måste man först kontrollera om mängden av nästa tillägning överskrider tillgängligt saldo. Om det inte gör det så hämtas mängden av produkten som ligger i kundvagnen och adderas med den angivna mängden, detsamma gäller det sammanlagda priset av artiklarna av samma typ. Metod för att lägga till priset till den generella totalsumman anropas och etiketten som håller totalsumman i vyn uppdateras med den nya summan. Sedan uppdateras listan med innehållet i kundvagnen i vyn.
    - **void removeClick(Object sender, Eventargs e)**
      -Vid borttagning ur kundvagn. Itererar genom innehållet i kundvagnen. Där artikelnummer matchar angivet artikelnummer från vyn visas först en

bekräftelsruta som frågar om man verkligen vill ta bort artikeln permanent. Om ja så raderas produkten på aktuellt index i kundvagnen. Sedan uppdateras listan på produkter i kundvagnen i vyn med det nya innehållet.

- **void purchaseClick(Object sender, Eventargs e)**
  -När ett köp ska genomföras. Kontrollerar först om det finns varor i kundvagnen. Om så visas sedan en bekräftelseruta som frågar om köpet skall genomföras. Om ja så itereras produkterna i kundvagnen igenom och uppdaterar posterna i filen där artikelnummer på produkt matchar artikelnummer i fil med metod från FileManager och subtraherar såld mängd från tidigare mängd.
  Sedan visas en bekräftelseruta på huruvida man vill skriva ut kvitto. Om ja så skrivs en sträng med en lista på de sålda produkterna, med priser och totalsumman med metod från FileManager. Sedan rensas kundvagnen och listan av produkter i vyn uppdateras med det nya innehållet i filen.
- **void closeSearch(Object sender, FormClosingEventArgs e)**
  -Överskrider stängningsoperationen på dialogrutan för sökfunktionen. Anropar metod från vyn för att dölja rutan istället.
- **void openSearchClick(Object sender, Eventargs e)**
  -Anropar metod från vyn för att göra dialogrutan för sökfunktionen synlig.
- **void searchClick(Object sender, Eventargs e)**
  -Vid sökning. Kontrollerar att textbox för artikelnummer, artikelnamn och författare/artist/regissör/producent inte är tomma. Itererar sedan igenom listan av produkter från FileManager och matchar mot söksträngar. Där matchning hittas appenderas raden till en sträng följt av en radbrytning. Sedan uppdateras textboxen för sökresultat i dialogrutan för sökfunktionen.
- **void returnClick(Object sender, Eventargs e)**
  -När vara ska returneras. Kollar om artikelnummer angetts och är i rätt format. Skickar sedan upp en bekräftelseruta som informerar att varan kommer att returneras. Om användaren klickar ok så uppdateras artikelpost med metod från FileManager och sedan uppdateras listan med produkter i vyn.

4. Problem

Jag har haft problem med att tolka uppgiften. Det kändes som att mina antaganden ibland skärde sig med vad som var förväntat. Någon av de efterfrågade kontrollfunktionerna blev inaktuell, då jag exempelvis använder artikelnummer som ”handle” för att lägga in leveranser. Det gav mig mycket huvudbry, men jag känner mig ändå nöjd med lösningen.
Jag har lärt mig att man vid sådana här tillfällen gör bäst i att samtala med den/de som har hand om uppgiften.

Jag har haft lite problem till och från med filhanteringen. Jag finner det svårt att få allt att hända i rätt sekvens och att allt sparas i rätt format. Det är också utmanande att lösa hur enskilda poster ska uppdateras. Den enda lösningen här var att läsa på internet och testa mig fram stegvis. Jag har således lärt mig en hel del om filströmmar och olika beteenden hos dem. Att en StreamWriter per default skriver över redan existerande filinnehåll om inte ytterligare parametrar anges exempelvis.

Annars har logiken i programmet varit ganska enkel att ha att göra med.

5. Sammanfattning

Sammanfattningsvis har det gått smidigt med utvecklingen av den här labben.
Att ha ett genomtänkt klassdiagram innan har hjälpt mycket.
Det var till viss del skönt att få ganska fria händer vad gäller strukturen på programmet och får tillfället att testa med abstrakta klasser, listor och uppdelning av programmet i olika sektioner. Där har jag lärt mig en hel del och jag har fått större vana för användning av attribut.

Jag är nöjd med klasstrukturen som jag valde, eftersom det har varit väldigt lätt och överskådligt att jobba med. Till en början funderade jag på hur vyerna skulle presenteras. Jag stod i valet mellan flikar och popup-formulär. Slutligen bestämde jag mig för flikar, då det känns som en mer effektiv och användarvänlig navigeringsmetod.

Ungefärlig tidsåtgång åt det här projektet har varit 35-40 timmar.