DVGB07 - Labbrapport Media Store

Grundversion

Melker Lurén Niklasson

29 April 2025

1 Ändringar (laboration 5)

I laboration 5 uppdaterade jag klassen `InventoryControl` med en ny knapp `syncButton` och tillhörande funktion `syncButton_Click()`. Jag skapade ytterligare en funktion, `GetProductFromXmlNode(XmlNode node)` som extraherar produkt-id, namn, pris och antal från en nod (produkt) i centrallagret. Funktionen returnerar därefter en ny instans av klassen `Product` med de extraherade värdena.

Funktionen `syncButton_Click()` körs när användaren trycker på knappen med texten "Sync Inventory" och hämtar produktdata från centrallagrets XML-fil. För varje produkt i lagret skapas ett nytt `Product`-objekt genom funktionen `GetProductFromXmlNode(XmlNode node)`. De sparas sedan i det lokala lagret.

1.1 Problem

Inga större problem uppstod under arbetet med laboration 5. Jag uppskattar att det tog ungefär 1h att läsa mig in i uppgiften och APIet och 1h att implementera den nya lösningen och funktionaliteten.

1.2 Slutsats

Funktionerna gör det möjligt för en anställd användare att manuellt synkronisera det lokala lagret med centrallagret. En alternativ lösning hade varit att automatiskt uppdatera det lokala lagret varje gång centrallagret uppdaterades. De nya funktionerna kan enkelt anpassas för en sådan alternativ lösning.

Uppgiften var ganska enkel, men intressant och lärorik. Jag lärde mig hur man enkelt hanterar läsning från XML-filer.

2 Antaganden

Under arbetets gång har jag gjort flera antaganden. Det första antagandet jag gjorde var att vid skapandet av en ny produkt, exempelvis en ny bok, så skulle ett unikt produkt-id (PID) automatiskt tilldelas produkten. Detta för att

underlätta arbetet för personalen när de ska lägga till nya produkter i lagret. Om lagret skulle vara jättestort och flera hundra olika produkter skulle finnas så skulle det bli väldigt jobbigt för personalen att hålla reda på vilka PID som är upptagna och inte, även fast en kontroll för duplicerade PID skulle finnas så blir det mycket gissande fram och tillbaka.

Ytterligare ett antagande gäller produktklasserna. Eftersom alla specificerade produkter (böcker, spel och filmer) måste ha attributen PID, namn och pris valde jag att göra en klass "Product" som varje enskild produkttyp kan ärva av. Detta gör det även enklare om man vill lägga till fler produkttyper, eller utöka arvsträdet. Exempelvis så skulle man kunna dela upp klassen "Games" i två sub-klasser, "VideoGames" och "BoardGames".

3 Översikt

Programmet består av två vyer, en vy för kunder och en vy för personal. Kundvyn består av en kundvagn och tre listor, en respektive lista för varje kategori av produkter. Användaren kan lägga till och ta bort produkter i kundvagnen. Kundvagnen visar upp namn, pris och antal för varje tillagd produkt och det totala priset visas. Efter att en kund har köpt en eller flera produkter uppdateras lagerstatusen för varje vara.

Personalvyn består av samma tre listor som kundvyn, men ingen kundvagn. Istället finns fyra olika typer av knappar för olika funktioner. Det finns knappar för att beställa mer av en vald produkt (öka antalet i lagret), man kan även minska antalet av en produkt eller ta bort den helt. Det finns även knappar för att lägga till nya produkter, en för respektive typ, bok, spel och film.

Programmet kontrollerar användarinmatning för att inte tillåta exempelvis negativa priser eller köp av fler antal produkter än vad som finns i lager.

4 Detaljerad beskrivning

Kapitlet beskriver programmets olika klasser i större detalj. Variabler och metoder beskrivs kortfattat för varje klass. För ytterligare förtydligande se Klassdiagram.

4.1 MainForm

MainForm representerar huvudfönstret och innehåller de två olika vyerna, InventoryControl och StoreControl.

Variabler:

- BindingList¡Book¿ books - Innehåller en lista med alla bokobjekt som är tillgängliga i lagret
- BindingSource booksSource - Används för att binda listan till gränssnittet
- BindingList¡Game¿ games

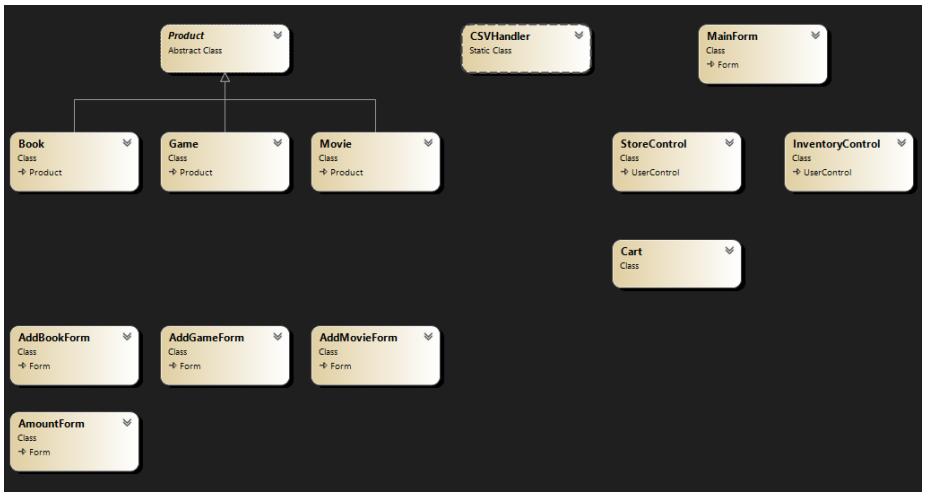

Figure 1: Klassdiagram

- BindingSource gamesSource
- BindingList¡Movie¿ movies
- BindingSource moviesSource
- BindingList¡Product¿ cartItems
- BindingSource cartSource

Metoder:

- **private void MainForm_Load(object sender, EventArgs e)**

  - Körs automatiskt när programmet startar (när formuläret laddas). Den ansvarar för att initiera och koppla samman alla datakällor med respektive kontroller i gränssnittet.

4.2 CSVHandler

CSVHandler hanterar kopplingen mellan programmet och .csv-filen.

Variabler:

- private static string filePath - filsökvägen för .csv-filen

Metoder:

- public static List¡Product¿ LoadProducts()

  - Läser in alla produkter från .csv-filen och returnerar en lista med produkter. Varje rad i filen parsas till en specifik produkt (Book, Game eller Movie) beroende på vilka attribut produkten har. Funktionen sparar också alla produkt ID:n (PID) som är använda.

- public static void SaveProducts(List¡Product¿ products)

  - Sparar en lista med produkter till .csv-filen. Funktionen itererar genom alla produkter och säkerställer att varje produkt har ett unikt PID. Om ett produkt-ID är ogiltigt eller redan använts tidigare, genereras ett nytt. Varje produkt skrivs sedan som en rad i filen baserat på sin typ. Till sist sparas den nya filen.

- public static int GeneratePID()

  - Returnerar ett nytt oanvänt PID.

4.3 InventoryControl

InventoryControl hanterar gränssnittet och funktionaliteten för personalvyn.

Variabler:

- `BindingSource booksSource` – Binder listan med böcker till respektive datagrid i gränssnittet.
- `BindingSource gamesSource` – Binder listan med spel till gränssnittet.
- `BindingSource moviesSource` – Binder listan med filmer till gränssnittet.
- `private List<Product> inventoryProducts` – En samlad lista med alla produkter som finns i lager.

Metoder:

- `public void LoadProducts()` – Läser in produkter och uppdaterar vyerna.
- `private void ClearGrids()` – Tömmer datagrids innan ny data laddas in.
- `private void HandleColumns()` – Anpassar kolumnerna beroende på produkttyp.
- `private bool AllValuesEmpty(DataGridView dataGrid, string columnName)` – Returnerar true om alla värden i en kolumn är tomma.
- `private bool AddProduct(Product product)` – Försöker lägga till en ny produkt i lagret, returnerar false om det misslyckas.
- `private DataGridView GetDataGridView()` – Returnerar det datagrid där en produkt är markerad av användaren.
- `private void RemoveSelectedProduct(DataGridView dataGrid, bool reduce)` – Tar bort vald produkt helt eller minskar antalet beroende på parameter om reduce är true/false.
- `private void RemoveAmountOfProduct(Product product, int amount)` – Minskar antalet för en specifik produkt.

- `private void dataGrid_SelectionChanged(object sender, EventArgs e)` – Hanterar markering av produkter i gränssnittet.
- `private void DeselectOtherGrids(object sender)` – Avmarkerar andra listor när en produkt markeras av användaren.
- `private void addBookBtn_Click(object sender, EventArgs e)` – Lägger till en ny bok i lagret.
- `private void addGameButton_Click(object sender, EventArgs e)` – Lägger till ett nytt spel i lagret.
- `private void addMovieButton_Click(object sender, EventArgs e)` – Lägger till en ny film i lagret.
- `private void orderProductButton_Click(object sender, EventArgs e)` – Ökar antalet i lager för vald produkt.
- `private void reduceStockButton_Click(object sender, EventArgs e)` – Minskar antalet i lager för vald produkt.
- `private void removeButton_Click(object sender, EventArgs e)` – Tar bort produkten helt från lagret.

4.4 StoreControl

StoreControl hanterar gränssnittet och funktionaliteten för kundvyn.

Variabler:

- `BindingSource booksSource` – Binder listan med böcker till respektive datagrid i gränssnittet.
- `BindingSource gamesSource` – Binder listan med spel till gränssnittet.
- `BindingSource moviesSource` – Binder listan med filmer till gränssnittet.
- `private List<Product> inventoryProducts` – En samlad lista med alla produkter som finns i lager.
- `private Cart cart` - Skapar en ny instans av Cart.

Metoder:

- `public void LoadProducts()` – Läser in produkter och uppdaterar vyerna.
- `private void ClearGrids()` – Tömmer datagrids innan ny data laddas in.
- `private void HandleColumns()` – Anpassar kolumnerna beroende på produkttyp.

- `private bool AllValuesEmpty(DataGridView dataGrid, string columnName)` – Returnerar true om alla värden i en kolumn är tomma.
- `private void UpdatePrice()` - Uppdaterar det totala priset i gränssnittet.
- `private DataGridView GetDataGridView()` – Returnerar det datagrid där en produkt är markerad av användaren.
- `private void dataGrid_SelectionChanged(object sender, EventArgs e)` – Hanterar markering av produkter i gränssnittet.
- `private void DeselectOtherGrids(object sender)` - Avmarkerar andra listor när en produkt markeras av användaren.
- `private void addToCartButton_Click(object sender, EventArgs e)` - Lägger till vald produkt i kundvagnen.
- `private void removeFromCartButton_Click(object sender, EventArgs e)` - Tar bort en vald produkt.
- `private void buyButton_Click(object sender, EventArgs e)` - Köper produkter i kundvagnen och uppdaterar lagerstatus för köpta produkter.

4.5 Product

Basklass för produkter.
Variabler:

- public int PID - Unikt produkt ID.
- public int Price - Produktens pris.
- public int Stock - Antal av produkt.
- public string Name - Produktens namn.

Metoder:

- **public Product(int pid, string name, int price, int stock)**
  - Konstruktor, tilldelar produkten PID, namn, pris och antal.

4.6 Book

Book ärver av Product.
Variabler:

- public string Author - Bokens författare.
- public string Genre - Bokens genre.
- public string Format - Bokens format (inbunden, e-bok etc.).

- public string Language - Språket boken är skriven i.

Metoder:

- **public Book(int pid, string name, int price, int stock, string author, string genre, string format, string language)**

  - Konstruktor, tilldelar boken PID, namn, pris, antal, författare, genre, format och språk.

4.7 Game

Game ärver av Product.
Variabler:

- public string Platform - Spelets platform (PC, Xbox etc.).
- public string Genre - Spelets genre.

Metoder:

- **public Game(int pid, string name, int price, int stock, string genre, string platform)**

  - Konstruktor, tilldelar spelet PID, namn, pris, antal, platform och genre.

4.8 Movie

Movie ärver av Product.
Variabler:

- public string Genre - Filmens genre.
- public string Format - Filmens format (DVD, CD etc.).
- public string Length - Filmens längd i minuter.

Metoder:

- **public Movie(int pid, string name, int price, int stock, string genre, string format, int length)**

  - Konstruktor, tilldelar filmen PID, namn, pris, antal, genre, format och längd.

4.9 AmountForm

Pop-up fönster, används när användaren ska fylla i antal, exempelvis hur många exemplar av en bok som ska beställas till lagret.
Variabler:

- public int Amount - antal.

Metoder:

- **private void cancelButton_Click(object sender, EventArgs e)** - Avbryter.
- **private void continueButton_Click(object sender, EventArgs e)** - Fortsätter med inmatat antal. Kontrollerar också att input är större än 0.

4.10 AddBookForm

Pop-up fönster, används när personal ska lägga till en ny produkt i lager. Fönstret består av relevanta fält för respektive produkt (bok, spel eller film) som användaren får fylla i. Fälten namn, pris och stock måste fyllas i, de andra är valfria.

Variabler:

- public Book book - boken som ska läggas till.

Metoder:

- **private void cancelButton_Click(object sender, EventArgs e)** - Avbryter.
- **private void addButton_Click(object sender, EventArgs e)** - Fortsätter med inmatad information från användaren och skapar den nya produkten.
- **private bool CheckPositive(int value)** - Kontrollerar att ett värde är positivt. Returnerar false om värde är negativt och true om värdet är positivt.

4.11 AddGameForm & AddMovieForm

Består av samma funktionalitet som AddBookForm. Enda skillnaderna är de relevanta attributen för varje enskild produkt som kan anges när produkten ska skapas.

5 Problem

Det största problemet jag stötte på var klasshanteringen, jag visste inte hur jag skulle strukturera upp uppgiften eller hur jag skulle börja med projektet. Den största frågan var hur jag skulle hantera kopplingen mellan programmet och fil. Tillslut bestämde jag mig för att börja med det jag kände mig bekväm med och utgå där ifrån. Tillslut blev det totalt tre olika klasser som utgör hanteringen av input/output till fil. Om jag hade gjort om uppgiften hade jag istället skapat CSVHandler klassen och en till, exempelvis ControllerHandler. Detta eftersom InventoryControl och StoreControl har mycket duplicerad kod mellan sig och det hade kunnat hanterats mycket bättre.

Ett annat problem har varit att få gränssnittet att se bra ut. Jag har försökt att få gränssnittet att se bättre ut och att det ska anpassa sig bättre när man förminskar/förstorar fönstret, men det har inte gått så bra. Jag har försökt fram och tillbaka, men bestämde mig till slut att det var bättre att fokusera på funktionaliteten av koden istället för utseendet. Hade jag haft mer tid hade jag lagt den på gränssnittet.

6 Sammanfattning

Sammanfattningsvis har arbetet gått bra. Jag har lärt mig mycket om programmering i C# och kopplingar mellan kod oich gränssnitt. Som jag beskrev i kapitel 4 hade jag skapat en klass ControllerHandler för att minska duplicerad kod och få koden mer lättförståelig om jag hade gjort uppgiften igen. Jag hade även lagt mer tid på gränssnittet. Nackdelen med den nuvarande koden är att den kan bli ganska rörig och svårförståelig, programmet ser heller inte så tilltalande ut att kolla på och vissa funktioner/knappar kan bli dolda beroende på fönsterstorleken.

Det är väldigt svårt att uppskatta tidsåtgången för enskilda moment, men den mesta tiden har gått till att skapa klassen InventoryControl och allt som hänger ihop med den. Jag skapade den innan jag skapade StoreControl och det gick därför lättare när jag kom till StoreControl då jag redan hade färdiga funktioner och visste hur jag skulle hantera datan.

7 Klassdiagram

Det initiala klassdiagrammet jag gjorde var delvis korrekt, men saknade många klasser. Jag visste exempelvis inte att jag skulle behöva göra flera olika ”Forms” för exempelvis tilläggning av en ny bok. Jag visste heller inte hur jag skulle hantera de olika vyerna, men hade som tanke att göra det från MainForm.

Det slutgiltiga klassdiagrammet visar att istället för att hantera båda vyerna i MainForm så delade jag upp det för tydlighetens skull. Som jag beskrev ovan skulle det kunna bli ännu tydligare med ännu mer uppdelning.

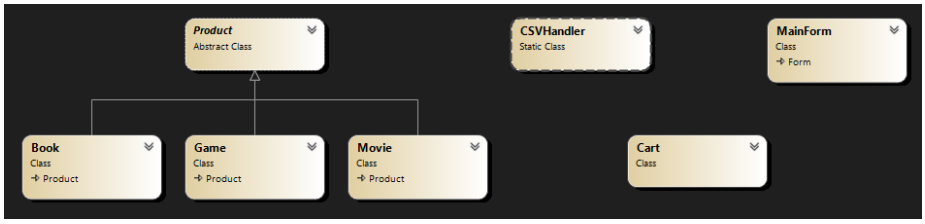

Figure 2: Initialt klassdiagram

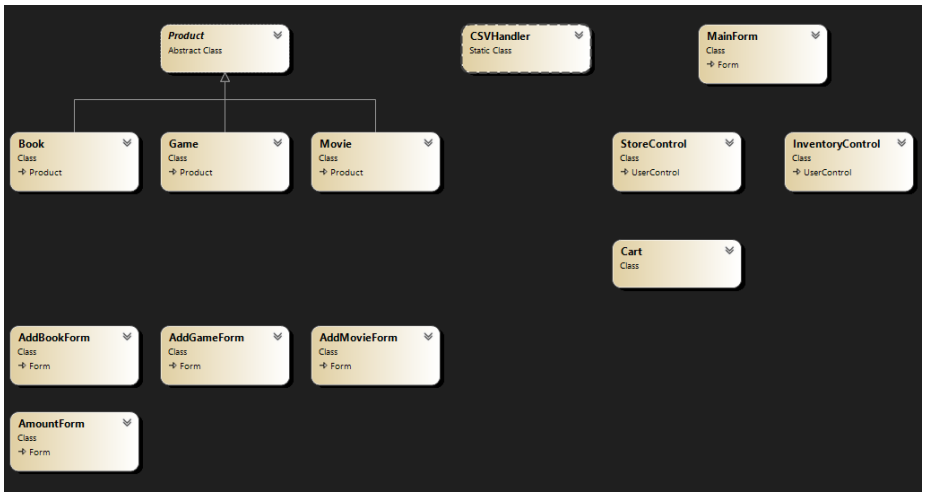

Figure 3: Slutgiltigt klassdiagram