Laboration 4 - Affärssystem (Grundversion)

C#.NET
DVGB07

Yazan Al Khalili

DVGB07 7,5 hp VT2025
2025-04-09

1. Antaganden

Under utvecklingen av programmet har ett antal antaganden gjorts för att kunna tolka och genomföra kraven i laborationen på ett tydligt sätt. Dessa antaganden är följande:

- Produkter spara i en CSV-fil med rubrikraden:
  Typ,ID,Namn,Pris,Extra1,Extra2,Extra3,Extra4,LagerAntal , där **Typ** anger om det är en bok, ett spel eller en film.
- Produktens lagerantal lagras som ett heltal och uppdateras manuellt via funktionen “Lägg till leverans”
- Programmet antar att all användarinmatning sker via **InputBox**, men det finns viss grundläggande validering (t.ex. Ogiltigt pris, tomt namn, eller ID-konflikt).
- Systemet stöder endast försäljning av enstaka produkter åt gången (inga kvantiteter i kundkorgen).
- Vid avslut av programmet sparas all aktuell information tillbaka till samma CSV-fil.
- Vid programstart läses filen in automatiskt. Om filen inte finns laddas inga produkter in.

Programmet initieras med slumpmässiga lagerantal:
Vid programmets start är produkternas lagerantal förifyllda med slumpmässiga värden i CSV-filen. Detta är ett antagande för att simulera ett existerande lager från början, istället för att börja med tomt lager (0). Det gör det möjligt att testa funktioner som leverans och försäljning direkt vid uppstart utan att först behöva lägga till leverans manuellt.

2. Översikt

Programmet är ett lager- och kassasystem för en butik som säljer böcker, spel och filmer. Det består av ett grafiskt gränssnitt (WinForms) med två huvudsakliga flikar: **Lager** och **Kassa**.

I **Lager-fliken** kan användaren:

- Lägga till nya produkter (bok, spel eller film)
- Ta bort befintliga produkter
- Uppdatera lagersaldot för en produkt via "Lägg till leverans"

Alla produkter visas i tabeller beroende på typ (Bok, Spel, Film). Lagerantalet visas och uppdateras direkt i gränssnittet.

I **Kassa-fliken** kan användaren:

- Se en lista med samtliga produkter
- Lägga till produkter i en kundkorg
- Ta bort produkter från kundkorgen
- Slutföra ett köp, vilket rensar kundkorgen
  När en produkt läggs till i kundkorgen minskas lagersaldot automatiskt. Om en produkt har 0 i lager kan den inte läggas till i kundkorgen.

Programmet laddar alla produkter från en CSV-fil vid start och sparar tillbaka till filen vid avslut.

3. Detaljerad Beskrivning

Programmet är uppbyggt kring flera klasser, där huvuddelen av logiken finns i Form1.cs, som representerar själva användargränssnittet. Programmet hanterar tre typer av produkter: Bok, Spel och Film, som alla ärver från basklassen Produkt.

Klasstruktur (förenklat):

Produkt (basklass)
├── Bok
├── Spel
└── Film

Produkt:

En basklass med egenskaper som ID, Namn, Pris och LagerAntal. Klassen innehåller även en virtuell metod ToCSV() som används för att spara produkten till fil.

Bok / Spel / Film:

Underklasser till Produkt. Varje klass innehåller ytterligare information:

- Bok: Författare, Genre, Format, Språk
- Spel: Plattform
- Film: Format, Speltid

Form1.cs:

allaProdukter

En lista med alla produkter i systemet (oavsett typ). Denna används för att lagra och söka efter produkter.

kundkorg

En lista med produkter som lagts till i kundkorgen under ett köp.

LoadProductsFromCSV()

Läser in alla produkter från en CSV-fil och skapar rätt objekt beroende på typ (Bok, Spel, Film). Data visas i rätt DataGridView för respektive produktkategori.

SaveProductsToCSV()

Skriver all produktdata tillbaka till fil vid programavslut, inklusive aktuellt lagerantal.

LagerAddButton_Click()
Användaren får via InputBox-dialoger ange information för en ny produkt. Produkten läggs till i listan “allaProdukter” och visas i gränssnittet.

LagerRemoveButton_Click()
Användaren kan välja att ta bort en produkt från lagret. Produkten tas bort från listan och alla relaterade tabeller uppdateras.

AddDeliveryButton_Click()
Användaren anger en produkts ID och hur många enheter som ska läggas till i lager. Lagersaldot uppdateras direkt i både lagertabell och kassalista.

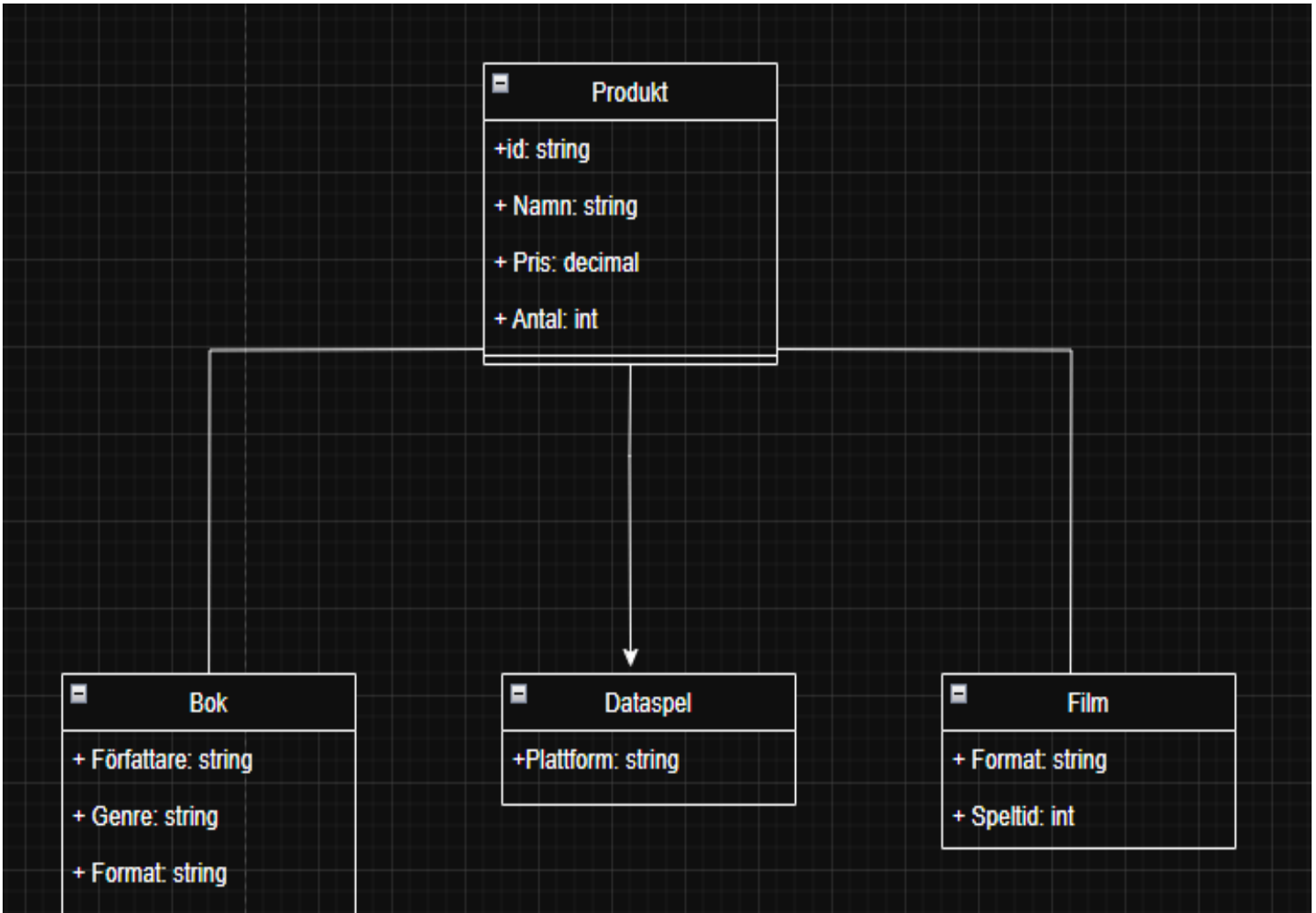
AddKundkorgenButton_Click() / RemoveKundKorgenButton_Click()
Lägger till eller tar bort produkter från kundkorgen. Lager antalet minskas eller ökas beroende på åtgärd. Det går inte att lägga till en produkt i kundkorgen om lagret är 0.

BetalaButton_Click()
Rensar kundkorgen och visar en bekräftelse.

Eventhantering
Programmet använder klassisk eventhantering i Windows Forms. Varje knapp i gränssnittet är kopplad till en metod, t.ex. AddKundkorgenButton_Click() som körs när knappen klickas.

4. Klassdiagram

Klassdiagram före programmets implementering:



Klassdiagram efter programmets implementering (genererat direkt från (Microsoft Visual Studios) genom att använda Class Diagram:

Skillnader mellan klassdiagram före och efter implementation

Det ursprungliga klassdiagrammet visade en förenklad struktur där klassen Produkt innehöll grundläggande egenskaper såsom id, namn, pris och antal. Dessutom fanns tre

underklasser (Bok, Datorspel och Film) som ärvde från Produkt och innehöll unika attribut som var specifika för respektive produkttyp, exempelvis Författare och Språk för böcker. Detta diagram fungerade som en teoretisk modell inför implementationen.

Efter implementationen har klassdiagrammet utvecklats till en mer komplett och praktiskt användbar struktur. Klassen Produkt innehåller nu inte bara egenskaper för ID, namn, pris och lagerantal (LagerAntal), utan också en metod ToCSV() som används för att spara produktinformation till fil. Alla underklasser (Bok, Spel, Film) har ärvt denna metod och fått sina specifika egenskaper definierade mer noggrant.

Den slutgiltiga modellen är mer verklighetstrogen och speglar programmets faktiska funktionalitet på ett bättre sätt.

5. Problem

1. **Design av användargränssnittet för lagerfliken**
   Ett problem som uppstod under designen av användargränssnittet var hur produkterna skulle visas i Lager-fliken. Ursprungligen var tanken att alla produkter skulle visas i en gemensam lista oavsett typ. Det visade sig dock bli rörigt och svårt att läsa. Därför valde jag istället att separera produkterna i tre olika tabbar: en för böcker, en för spel och en för filmer. Detta gjorde gränssnittet mer strukturerat och lättnavigerat för användaren.

2. **Lagring och läsning av data i CSV-format**
   En utmaning var att skapa en struktur som kunde spara och läsa olika produkttyper (bok, spel, film) från en gemensam CSV-fil. Eftersom varje typ har olika antal egenskaper krävdes extra logik för att korrekt hantera dessa skillnader vid både inläsning och sparning. Ett exempel är att vissa rader kan sakna kolumner (t.ex. en film har färre fält än en bok), vilket ledde till fel vid inläsningen om det inte hanterades rätt.

3. **Hantering av lagerantal vid försäljning och leverans**
   En annan utmaning var att se till att lagerantalet (antal produkter i lager) uppdateras korrekt vid varje försäljning eller leverans. I början saknades denna funktion helt, vilket innebar att produkterna kunde säljas även om de inte fanns i lager. Det krävdes logik som både minskade lagersaldot vid köp och ökade det vid leverans, samt hindrade köp när lagret var 0.

6. Sammanfattning

Projektet har resulterat i ett komplett lager- och kassahanteringssystem där användaren kan lägga till, ta bort och leverera produkter samt genomföra försäljningar. Programmet är uppdelat i två huvudsakliga delar: en lager flik där produkter hanteras och en kassa flik där kundkorgen och köp genomförs. Funktionaliteten är utformad enligt kravspecifikationen och förstärkt med extra valideringar och användarstöd, som färgmarkering av tomma celler och kontroll av lagerantal.

Under arbetets gång har jag stött på flera utmaningar, bland annat kring hur produkterna skulle organiseras i gränssnittet, validering av inmatning och uppdatering av visade data i flera vyer. Genom dessa problem har jag lärt mig mycket om både C#, WinForms och praktisk programstruktur. Jag har också insett vikten av att tänka igenom programlogiken tidigt för att minska risken för framtida buggar och omstruktureringar.

Projektet tog ungefär 2–3 dagar att färdigställa, inklusive kodning, testning och dokumentation. Jag är nöjd med resultatet och känner att jag fått ökad förståelse för både systemutveckling och strukturerad programmering.