

# Laboration 4

4/8/2022

SEN 3/5 Poäng

Försök 3

Feedback från granskning  
5/9/2022

Poäng för försök 3:

3/5



Lägg till kommentar

## Obegränsat antal försök tillåts

### Information

Denna laboration går ut på att både **designa** och **implementera** ett affärssystem. Affärssystemet ska användas till en fysisk affär som säljer media (t.ex. böcker, cd-skivor, spel). Systemet ska användas av anställda i affären och kunna hantera både kassahantering och lagerhållning.

Börja med att rita upp en klassstruktur för programmet, med ett förslag på klassdiagram och beskrivning av klasser. När programmet sen är färdigprogrammerat ska du diskutera skillnaden mellan din design och den faktiska implementationen i rapporten. **Rapporten ska innehålla både ett klassdiagram före och efter programmet implementerades.**

### Det inlämnade programmet ska:

- Vara implementerade i C#.NET med Windows Forms som grafiskt bibliotek.
- Vara implementerade i .NET Framework 4.6 eller nyare.
- Ska gå att köra på en svensk dator, med ett svenskt operativsystem.
- Ska inte under rimliga omständigheter krasha. Vi kommer att [monkey testa](https://en.wikipedia.org/wiki/Monkey_testing) ([https://en.wikipedia.org/wiki/Monkey\\_testing](https://en.wikipedia.org/wiki/Monkey_testing)) alla program, men vi kommer inte göra underliga ändringar i exekveringsmiljön eller liknande. Hantera alla fel med exceptions.
- Vara användarvänliga, med tydliga och beskrivande felmeddelanden, och inte bryta allvarligt mot [Microsofts rekommendationer för gränssnittsprogrammering](https://docs.microsoft.com/en-us/windows/win32/uxguide/guidelines) (<https://docs.microsoft.com/en-us/windows/win32/uxguide/guidelines>).

## Grundversion (för 3 poäng)

Följande funktionalitet ska finnas i systemet:

1. Lägga till nya produkter.
2. Ta bort produkter.
3. Lägga till en leverans av en eller flera produkter från grossit. *Här räcker det med att antalet uppdateras, leveransen behöver inte loggas.*

<https://canvas.kau.se/courses/13643/modules/items/373327>Försök  
<https://canvas.kau.se/courses/13643>

Systemet ska ha två separata vyer (t.ex. flikar eller fönster), en för **lagerarbete** och en för **kassabruk** (dock **max en instans** av varje vy). Vyerna ska båda ha en lättöverskådlig produktlista, och endast tillgång till relevant funktionalitet. Vyerna ska arbeta mot samma data som ni ska lagra i en **CSV-fil**. Vid försäljning ska varor kunna läggas till i en kundkorg innan köpet genomförs. Vid uppstart av programmet ska denna fil läsas in, och när programmet stängs av ska filen sparas. Databasen ska ligga på samma ställe som programmet (inga hårdkodade sökvägar). *Kundkorgen är tillfällig och behöver inte sparas till fil.*

Tänk på att göra systemet så generellt som möjligt, och se till att det är estetiskt tilltalande och användarvänligt för butikspersonal utan dataerfarenhet.

Kontrollfunktioner ska finnas så att:

- Om någon försöker lägga till en ny vara i sortimentet med ett redan upptaget varunummer ska detta ej godkännas.
- Om någon i personalen försöker ta bort en vara ur sortimentet, och dess lagerstatus inte är noll, ska en dialogruta dyka upp om man verkligen vill ta bort varan. I dialogrutan ska det gå att svara Ja eller Nej.
- Man ska inte kunna sälja varor som inte finns på lager.
- Produkter får inte kunna ha felaktig information (exempelvis avsaknad av obligatoriskt fält, negativa priser, bokstäver i varunumret, en speltid som är negativ eller inte en siffra)
- Varje varunummer ska ha ett unikt ID/varunummer.

**Produkterna i affären ska vara följande (\* = obligatoriskt fält, gråa bakgrund = ingen data från grossit):**

Böcker					
Namn*	Pris*	Författare	Genre	Format	Språk
Bello Gallico et Civili	449	Julius Caesar	Historia	Inbunden	Latin
How to Read a Book	149	Mortimer J. Adler	Kurslitteratur	Pocket	
Moby Dick	49	Herman Melville	Äventyr	Pocket	
Great Gatsby	79	F. Scott Fitzgerald	Noir	E-Bok	
House of Leaves	49	Mark Z. Danielewski	Skräck		
Dataspel					
Namn*	Pris*	Plattform			
Elden Ring	599	Playstation 5			
Demon's Souls	499	Playstation 5			



(<https://canvas.kau.se/courses/13643/modules/items/373327>)




(<https://canvas.kau.se/courses/13643>)

Planescape Torment	99	PC	
Disco Elysium	399	PC	
Filmer			
Namn*	Pris*	Format	Speltid
Nyckeln till frihet	99	DVD	142 min
Gudfadern	99	DVD	152 min
Konungens återkomst	199	Blu-Ray	154 min
Pulp fiction	199	Blu-Ray	
Schindlers list	99	DVD	

## Version 2 (för 1 extrapoäng)

Lägg till:

- **Kvittofunktion:** Alla produkter en kund har köpt ska skrivas ut på ett kvitto (med en skrivare).
  - Har du ingen skrivare installerad på datorn går det att använda "Skriv ut till pdf". Finns inte alternativet kan du slå på det [så här](https://www.digitaltrends.com/computing/print-pdf-windows/)  (<https://www.digitaltrends.com/computing/print-pdf-windows/>).
- **Sökfunktioner:** Ska gå att söka på attribut som exempelvis produkt, lagerstatus och artist.
- **Återköp:** Kund ska kunna returnera en vara, varefter lagerstatusen ska ökas.

Programmet ska dessutom ha en tydlig separation mellan gränssnitt och logik. Detta skulle kunna implementeras genom ett backend-frontend, model-view-controller eller annat designmönster.

## Version 3 (för 1 extrapoäng)

Lägg till förutom tillägen i version 2:

- **Top-10-lista:** Det ska gå att få ut topplistor på mest sålda produkter för varje månad eller år.
- **Total försäljning:** Man ska även kunna hämta ut information om totalt antal produkter sålda varje månad och år.

För att implementera dessa två funktioner, behöver programmet lagra information om vilka produkter som är sålda, och när. Det är valfritt om denna information lagras i samma databas som produkterna, eller i en separat databas. Det senarenämnda är lite enklare.

## Inlämning



(<https://canvas.kau.se/courses/13643/modules/items/373327>)



(<https://canvas.kau.se/courses/13643/modules/items/373327>)

**dig på.** Du kan gärna [labmall.docx \(https://canvas.kau.se/courses/13643/files/1394894/download?wrap=1\)](https://canvas.kau.se/courses/13643/files/1394894/download?wrap=1) som mall för att skriva dokumentationen i. Om du väljer att använda en annan mall, se då till att ha en bra struktur och att inte missa viktig information.

**Uppgiften är individuell**, och när du lämnar in ditt svar intygar du samtidigt att det är du själv som gjort laborationen. Allt material som inte är ditt eget ska vara tydligt avgränsat och identifierat med källhänvisningar (se [Plagiering \(https://www.kau.se/bibliotek/skriva-referera/skriva/skriva-akademiskt/plagiering\)](https://www.kau.se/bibliotek/skriva-referera/skriva/skriva-akademiskt/plagiering)). **Alla inlämningar plagiatskontrolleras av flera automatiserade system.**

Programkoden ska lämnas in i en ZIP-fil (RAR, TAR.GZ och 7z går också bra), och dokumentationen ska lämnas in i pdf, separat från ZIP-filen. **Lägg alltså inte in dokumentationen i zip-filen, för då fungerar inte kommenteringsverktygen i Canvas. Skriv en kommentar vid inlämningen vilken version av uppgiften du försöker dig på.**

---

## Frequently Asked Questions

Här kommer några frågor som uppkommit under tidigare tillfällen.

### 1. Produkterna saknar vissa fält. Exempelvis har House of Leaves inget format.

Tabellen innehåller informationen ni fått från grossisten, och den saknar ibland viss information. En av utmaningarna i laborationen är hur ni lagrar detta i er CSV-fil. *(varför just House of Leaves inte har något format... mja det förstår ni om ni läser boken :))*

### 2. Produkterna i tabellen har inget varunummer eller ID.

Varunummer och ID är något ni får generera själva till produkterna.

### 3. Finns någon rekommenderad läsning om designmönster?

Eftersom nivån är högre än godkänd, är del av uppgiften att hitta bra information själv. Däremot kommer en video på Måndag om inte något allvarligt kommer ivägen.

### 4. Önskar också ett förtydligande om separation mellan gränssnitt och logik. Vad innebär det rent praktiskt?

Kolla på en enkel backend-frontend avdelning, alltså få programmets logik ut ur Form-klassen. Vill du ha ett lite mer avancerat designmönster kan du kika på Model-View-Controller. Finns även möjlighet här för erfarena studenter att göra någon ännu mer intressant lösning.

### 5. Metoderna som fångar upp användarens interaktion med gränssnittet, räknas de som logik eller gränssnitt?

Jag brukar räkna händelser i C#.NET som del av gränssnittet. De går att separera, men det ligger lite utanför den här kursens omfång.

### 6. Vilka delar av programmet räknas som logik och vilka räknas som gränssnitt?

Finns ingen skarp avdelning, utan det beror lite på hur ditt program ser ut. Exempelvis hamnar ofta



<https://canvas.kau.se/courses/13643/modules/items/373327>



<https://canvas.kau.se/courses/13643/modules/items/373327>

varukorgen, och liknande. Tänk att du ska kunna skriva ett nytt gränssnitt och koppla på det på den befintliga logiken så smidigt som möjligt.

## 7. När man använder designmönstret MVC får man då inte ens använda sig av MessageBox(es) från kontroll klassen?

Jag tycker att detta är en avvägning man får ta som programmerare. Det kan ibland vara svårt att helt separera gränssnitt och logik och man måste ibland ta genvägar. Däremot håller jag inte med om att anropa en funktion i vyn för att visa en message box är mer "tightly coupled", däremot är det mer omständigt. Anledningen är att separera det grafiska från det logiska, inte separera klasserna från varandra. Betänk att om du vill byta ut ditt grafiska API mot något annat (exempelvis WPF), så ska du endast behöva ändra i vyn, men inte i kontrollerklassen. Har du grafisk kod i kontrollerklassen, måste båda dessa klasser modifieras (och testas) för att ditt gränssnittsbyte ska fungera.

## 8. Är det ett krav med flera olika klasser för att representera produkterna.

Nej. Ditt argument om att ha en generell klass för alla produkter kan jag både hålla med om, och inte hålla med om. I ett riktigt system tror jag att en dynamisk klass som innehåller en uppsättning attribut sparade i någon listform nog är den bästa vägen att gå, då affären kan utökas från gränssnittet, utan att ändra på programkoden. Däremot måste detta då göras snyggt, på ett smart, generellt och gärna testbart sätt. Att ha en produktklass med "x antal variabler" är inte det. Då är jag av åsikten att din lösning med en abstrakt klass och arv är bättre. Koden är testbar, och mer skyddad mot datatypsfel och annat.

## 9. Varför skapar man en backend-klass, och inte endast använder sig av en BindingList<T>?

Vi pratade om det i videon. Självklart kan du bara använda en BindingList<T> givet att du endast behöver de properties som finns i en BindingList, och ingen annan logik behövs i ditt frontend. Däremot behöver ditt program logik som inte är tillgängligt med en BindingList<T>, eller kan det komma att finnas ett framtida behov för sådan logik, är det hjälpsamt att ha en klass.

Låt oss säga att i en del av ditt program, vill du beräkna summan av värdet på alla produkter som är i lager. Om du då har en egen klass för din produktlista, kan detta implementeras i den klassen. Om du å andra sidan utför *logiken* i din form-klass, så ligger din programlogik delvis i backend och delvis i frontend.

Det finns flera fördelar med att ha logiken i ett backend. En fördel är om du vill byta ut ditt frontend mot ett annat gränssnitt. I sådant fall behöver du i bästa fall endast hantera gränssnittskod. Om du å andra sidan vill byta ut ditt backend (t.ex. mot ett webbapi), så är det lite lättare att limma ihop klasserna mellan ditt nya backend och ditt gamla frontend (detta blir ännu lättare med MVC). Dessutom om du vill använda dig av enhetstester, så är detta svårare att göra om du har din logik i en grafisk klass. Slutligen så hjälper separationen under själva programmeringen. Backendprogrammerare får hålla på med sitt, och frontendprogrammerare med sitt.

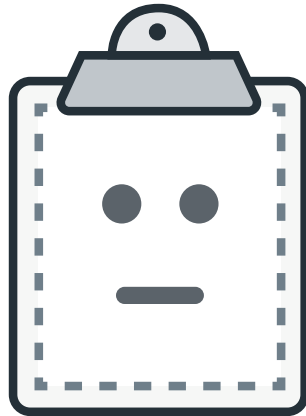


(<https://canvas.kau.se/courses/13643/modules/items/373327>)



(<https://canvas.kau.se/courses/13643/modules/items/373328>)

	Filnamn	Storlek	
	<a href="#">Shop Mana...mning.zip</a>	4,45 MB	✓
	<a href="#">Ghaith Ch...ort-2.pdf</a>	485 kB	✓



Förhandsvisning ej tillgänglig

Shop Management- 3 inlämning.zip



Ladda ner

([https://canvas.kau.se/files/1551598/download?  
download\\_frd=1&verifier=kQfDfCdFtYbCZUgoCG3tzWXt0mjVZ7KilyPTwOyp](https://canvas.kau.se/files/1551598/download?download_frd=1&verifier=kQfDfCdFtYbCZUgoCG3tzWXt0mjVZ7KilyPTwOyp))



(<https://canvas.kau.se/courses/13643/modules/items/373327>)



(<https://canvas.kau.se/courses/13643/modules/items/373328>)