**Projektrapport:**

**Projekt C: Shopping app för Cheap Cheating Shoppers**

Författare: Samir Ahmad, [h24samah@du.se](mailto:h24samah@du.se) och Ludwig Lindfors [h24ludli@du.se](mailto:h24ludli@du.se)

Datum: 2024-12-30

Kursnamn: GIK299 – Objektorienterad Programmering

Examinator: Pär Eriksson, [pei@du.se](mailto:pei@du.se)

**Innehåll**

[**Titel** 1](#_Toc186465664)

[**Syfte** 1](#_Toc186465665)

[**Egna user stories** 1](#_Toc186465666)

[**Egna tolkningar, övervägande och val av ställda userstories** 3](#_Toc186465667)

[**ChatGPT eller liknande AI-verktyg** 3](#_Toc186465668)

[**Metod** 5](#_Toc186465669)

[**Design** 5](#_Toc186465670)

[**Samarbete** 6](#_Toc186465671)

[**Testning** 6](#_Toc186465672)

[**Arkitektur/Design** 7](#_Toc186465673)

[**Resultat** 9](#_Toc186465674)

[**Slutsats** 12](#_Toc186465675)

[**Referenser** 13](#_Toc186465676)

[**Appendix** 13](#_Toc186465677)

[**Parprogrammeringslogg** 13](#_Toc186465678)

# **Titel**

Projekt C: Shopping app för Cheap Cheating Shoppers

# **Syfte**

Vi har skapat ett program som simulerar en online bokhandel. Det är utvecklat som Console applikation i C# och är baserad på Objektsorientering principer.

Programmet har en huvudmeny där användaren får välja mellan tre olika alternativ:

1. Starta programmet som administratörer
2. Starta programmet som köpare
3. Avsluta programmet.

Programmet enligt uppgiftens beskrivning ger administratören behörighet att ”hantera produktlistan och lägga till eller ta bort produkter till försäljning från webbplatsen” samt visar alla tillgänglige böcker i systemet (lagret), visar försäljningsdata och visar de mest populära böcker baserat på försäljning. Data sedan kan används för bättre marknadsföring och produktkatalog.

Vidare ger programmet enlgit uppgiftens beskrivning köparen behörighet att ”kunna se olika produkter till försäljning” och när köparen hittar den boken eller dem bokarna som hen blir intresserade av ”köpa den genom att lägga den till kundvagnen.”. Vidare har köparen behörighet att ”kunna hantera produkter i kundvagn tex lägg till eller ta bort och uppdatera.” och kan ”se vad jag ska betala totalt för produkterna i kundvagnen.”.

Programmet har 4 klasser och 17 metoder, se UML diagrammet för vidare information. Programmet är anpassat till null hantering och validerar användarens input med hjälp av if, while, try-catch och .Nets egna metoder.

# **Egna user stories**

Utöver uppgiftens egna userstories har vi lagt nedan fyra user story:

1. **User story 1:** Som köpare vill jag få 20 kronor i rabatt om det totala inköpspriset överstiger 200 kr. Jag vill kunna se rabatten både när jag kollar på mitt totalbelopp och när jag är i kassan och vill genomföra köpet.
   * Vi har implementerat en funktionalitet som automatiskt ger 20 kronor i rabatt om användarens totala inköp överstiger 200 kr. Rabatten visas på två ställen i programmet:
     1. I menyn för köpare BuyerMenu(), alternativ 5: **"Se totalbelopp"** – Här kan användaren se sitt totala belopp före och efter rabatten, om rabatten är tillämplig.
     2. I alternativ 6: **"Gå till kassan"** – När användaren går till kassan för att slutföra sitt köp visas den totala summan inklusive eventuell.
2. **User story 2:** Som köpare vill jag kunna välja mitt betalningssätt (Kreditkort, Klarna eller kontant vid leverens” i kassan.
   * Vi har implementerat en funktionalitet i kassan under ”Gå till kassan” som ger köparen möjlighet att välja mellan tre olika betalningsmetoder i kassan: **Kreditkort**, **Klarna**, eller **Kontant vid leverans**. Denna funktionalitet aktiveras när användaren når kassan och är redo att genomföra sitt köp. Efter att betalningsmetoden har valts får köparen även ett kvitto som bekräftar köpet.
3. **User story 3:**  Som administratör vill jag kunna se försäljningsstatistik så att jag kan uppdatera lagret vid behov.
   * Vi har implementerat en funktionalitet som genererar och visar försäljningsstatistik baserat på de böcker som köparen tidigare har köpt. Denna funktionalitet är direkt kopplad till metoden Checkout() – när betalningen genomförs uppdateras antalet sålda enheter i SalesCount vilket är statstik behållare för bokar i Book-klassen. Med denna funktionalitet kan administratören se aktuella försäljningsdata och få en översikt över hur många enheter av varje bok som har sålts.
4. **User story 4:** Som administratör vill jag kunna se de mest populära böckerna baserat på försäljning så att jag kan spåra köparens preferenser och optimera produktkatalogen i framtiden.
   * Vi har implementerat en funktionalitet DisplayMostPopularBooks() som använder en metod för att sortera böckerna baserat på försäljningsstatistik (genom en "Bubble sort"-algoritm). Denna funktion visar de mest sålda böckerna högst upp i listan vilket gör att administratören kan se vilka böcker som är populärast under en viss period. Data från denna funktionalitet kan sedan ger administratören insikter om köparens preferenser som kan även avvänds vid optimering av produktkatalogen i framtiden.

## **Egna tolkningar, övervägande och val av ställda userstories**

Nedan user storeis är tagen direkt från uppgiftsbeskrivning.

*1. Som shoppare vill jag kunna se olika produkter till försäljning för att när jag hittar det jag är intresserade av, köpa den genom att lägga den till kundvagnen.  
2. Som shoppare vill jag kunna hantera produkter i min kundvagn tex lägg till ta bort och  
uppdatera. Jag vill se vad jag ska betala totalt för produkterna i kundvagnen.  
3. Som administratör vill jag kunna hantera produktlistan och lägga till, redigera eller ta bort produkter till försäljning från webbplatsen*.

Att tolka dessa user stories var inte så svårt eftersom vi har erfarenhet av att arbeta med liknande applikationer i våra yrkesliv. Det tog inte mycket tid att förstå vad som behövde göras. Däremot var det mer tidskrävande att skriva koden.

Ibland stötte vi på problem när vi hoppade mellan olika lösningar. Då uppstod risken att vi översatte en user story till en helt annan lösning än vad som egentligen krävdes. Vi lärde oss att lösa detta var att bryta ner problemet i mindre delar och sedan matcha det mot user storyn igen. Vi valde också att skriva ner varje user story på papper med penna. Det gjorde att de blev en naturlig del av vår process och hjälpte oss att hålla rätt fokus. På så sätt kunde vi hela tiden försäkra oss om att vi var på rätt spår när vi utvecklade nya funktioner och skrev koden.

Vi valde ovan fyra egna user stories eftersom vi ansåg att de låg nära uppgiftsbeskrivningen och kunde komplettera programmet ytterligare. Vi delade upp dem jämnt mellan två för köparen och två för administratören.

För köparen ansåg vi att det är viktigt att kunna få rabatt vid större köp (över 200 kr) eftersom detta är en stark trend där de flesta företag erbjuder liknande fördelar. Dessutom är olika betalningsalternativ en viktig del av moderna e-handelsplattformar och därför inkluderade vi det i projektet. Naturligtvis hade vi fler idéer men eftersom uppgiften endast krävde fyra user stories höll vi oss till det. Vi kommer att utöka och förbättra programmet i framtiden.

För administratören såg vi det som avgörande att kunna uppdatera lagret vid behov. Därför gav vi funktionen att visa antalet sålda böcker och möjligheten att basera lagertillgång på försäljningsstatistik. Vidare ansåg vi att försäljningsstatistik är viktigt för marknadsföring och beslutsfattande och det ledde till implementeringen av en funktion som med hjälp av Bubble Sort visar de mest populära böckerna. Detta gör det enklare för administratören att optimera bokkatalogen och planera framtida inköp.

# **ChatGPT eller liknande AI-verktyg**

I denna uppgift har vi använt ChatGPT och Google och ställde nedan frågor.

Verktyg: ChatGPT och google

Fråga/prompt: Skillnad mellan enum, struct och klass?

På vilket sätt svaret användes: svaret hjälpte oss att förstå att enum är ett utmärkt val för att representera konstanta värden eftersom det ger bättre struktur och läsbarhet jämfört med att använda en klass med attribut.

Verktyg: ChatGPT

Fråga/prompt: Vad ska vi tänka vid testning av kod?

På vilket sätt svaret användes: Vi fick tillgång till mycket information och valde att filtrera bort det som inte var relevant eller som var överkurs, men ändå vi läste mycket om testning. Svaret hjälpte oss att koda effektiv och hantera användarens felinmatning i god tid.

Verktyg: ChatGPT

Fråga/prompt: Vad är symbolen? i C# ?

På vilket sätt svaret användes: Svaret hjälpte oss att förstå begreppet nullable types i C# och hur vi kan hantera dem. Vi använde ”?” i vår switch-meny i koden string? choice = Console.ReadLine();. Därefter försvann varningen som tidigare uppstod trots programmet kunde hantera null ett på annat sätt. Vi lärde oss att ”?” infördes i C# 8.0 för att explicit markera att en referenstyp (som en string) kan vara ”null”. För att fördjupa oss ställde vi samma fråga till Google och fann mycket användbar information på Stack Overflow och Raddit som vi har inkluderat i referenslistan.

Och många relaterade frågor till Google.

Vi tyckte att det var svårt att välja mellan klass och enum för "CartItem" men ChatGPT och Google hjälpte oss att bättre förstå skillnaderna med exempler särskilt med tanke på att enum är begränsat till konstanta värden och kan skapa problem om vi i framtiden behöver lägga till fler egenskaper i programmet.

ChatGPT gav oss också stöd när det gällde validering av null-hantering och vägledde oss i testning. Vi fick inspiration från ChatGPT och Google kod. Vi märkte att när programmet blev större och mer komplext blev ChatGPT och Google mer begränsat och tidkrävande. I detta läge valde vi ställa små och korta frågor.

I labb fyra kom vi fram till att det är bättre att alltid börja med att skriva egen kod och verkligen förstå processen då det förbättrar både inlärningen, ansvarstagandet och problemlösningsförmågan.

Dessutom blir det både enklare, billigare och mer effektivt att återanvända kod från tidigare labbar och anpassa den till det aktuella projektet. I framtiden kan vi använda ChatGPT mer som en sökmotor för att ställa öppna frågor men vi kommer alltid att kritiskt granska svaren och jämföra dem med kurslitteratur eller andra källor. Vi föredrar att använda Stack Overflow eller Google direkt istället för att direkt ställa fråga till ChatGPT

# **Metod**

I detta projekt har vi använt av off Visual studio 2022, version 17.0, tillsammans med ramverket .NET 6.0 för att bygga och testa vår applikation.

För att skapa programmet började vi med att läsa igenom instruktionerna och referensmaterialet noggrant. Det tog oss några timmar att förstå vad som krävdes i uppgiften. Vi startade med att skriva pseudokod, skriv på papper klassar och skapade ett litet flödesschema för att planera loopar och switch-satser. För att göra arbetet effektivare och validera användarens input använde vi delar av koden från labb fyra men ansåg att programmet är större och har annan funktionalitet och därför redigerade och anpassade valideringskoderna löpande under utvecklingen. Datastrukturen var en viktig del i detta program. Vi lade därför mycket tid på att arbeta med den både genom att söka information på nätet och i kursboken samt genom att utveckla och förbättra vår programmet.

Vi delade upp arbetet men genomförde det i olika steg tillsammans:

1. Vi skapade ett en klass för ”BookClass”.
2. Det tog oss mycket tid både att söka efter information och utveckla kod för att bestämma mellan enum och klass för ”CartItem”. Det slut blev det klass då vi kommer utveckla mer programmet med mer funktionalitet. Cart
3. Vi skapade ett en klass för ” BookManager”.
4. Vi skapade ett en klass för ” Cart”.
5. Vi hade svårt att välja att skapa en gemensam lista för både ”Cart” och ”BookManager” eller två separata listor. Men det blev två separata utifrån funktionalitet och olika metoder.
6. Vi skapade en meny med hjälp av if-sats och while-loop. Vi undvikt använda switch då vi i tidigare uppgifter använde switch och i denna ville vi uppmuntra oss samt if-sats passade bästa best detta program.
7. Vi vill skriva koden kortare och därför vi utvecklade två olika metoder för administratör och köpares meny och sedan i huvudmeny anropar vi det baserad på användarens inmatning.
8. I varje steg tog vi tid och testade programmet för att snabbt lösa problemet.
9. Därefter testade vi programmet enskild och ren skrev kommentarerna i koden.

## **Design**

Vi tolkade uppgiftens beskrivning av (menydelen) på olika sätt men när vi diskuterade och resonerade kring lösningen insåg vi att vi egentligen menade samma sak. Vi visualiserade menyn med papper och penna inspirerade av exemplen som presenterades under projektintroduktionen. Till slut bestämde vi att skapa en huvudmeny med tre alternativ där användaren kan välja i vilken roll hen vill använda programmet eller avlusat programmet. Vi valde att inte inkludera användarnamn och lösenord eftersom vi trodde att uppgiften endast krävde fyra egna ”user story”. Men vi inser att det skulle vara en viktig funktion att utveckla vidare i framtida versioner av projektet vilket vi kommer göra.

Vi ville utmana oss själva så mycket som möjligt, och därför valde vi att skriva självständiga metoder där det var möjligt. Från tidigare labbar har vi lärt oss att metoder kan återanvändas i andra projekt vilket var en av anledningarna till att vi skrev totalt 17 metoder i detta projekt. En annan anledning var att vi stötte på nya problem hela tiden som delvis berodde på att vi hoppade mellan olika delar av lösningen i stället för att fokusera på ett problem i taget. Detta tog tid, men vi insåg att det var en ineffektiv strategi. Därför beslutade vi oss för att hålla oss till en sak i taget när vi löste problem. Vi använde oss mycket av pseudokod och kontinuerlig problemlösning och testning för att strukturera arbetet.

Tack vare att vi tidigare arbetat med klasser, enum och struct i labb fyra och dess uppvärmningsuppgifter, var det inte svårt att använda objektorienterade principer i projektet. Vi tog oss an projektet som en verklighetsnära lösning och tolkade det som en riktig bokhandel.

* För att hantera boklagret skapade vi klassen **"BookManager"**, där administratören har fler behörigheter.
* För köparen behövdes en plats att lagra och hantera sina valda böcker, som en kassa, vilket blev klassen **"Cart"**, där köparen har mer behörigheter.
* För att köparen skulle kunna se varje enskild bok i kundvagnen skapade vi klassen **"CartItem"**.
* Klassen **"Book"** blev en gemensam komponent som används i alla de andra klasserna och fungerar som en behållare för objekt.

## **Samarbete**

Vi hade inga problem med samarbetet. Vi är ambitiösa och vill lära oss så mycket som möjligt. Vi var alltid tillgängliga för varandra med respekt till ledigheter. Vi beslutade om varje lösning tillsammans, och det fungerade utmärkt.

## **Testning**

Vi testade koden och programmet flera gångar under utveckling och även efter utvecklingen. Vi matade in olika typer av data inklusive både giltig och ogiltig input. Vi fokuserade på att testa validering av användarens inmatning som att skriva in siffror och olika tecken där bokstäver förväntades och hantering av felaktiga värden. Vi har även testat null och sedan förbättrat programmet med null hantering.

Vi bad även vår familjemedlemmar att testa det för att det ska bli objektiv tolkning av vår program.

# **Arkitektur/Design**

Vårt program utvecklades som en konsolapplikation i C#. Det hanterar böcker och kundvagn. Programmet har 4 klassar (ärver från varandra) och 17 metoder exkl. Program klass och Main metoden. Programmet använder objektorienterings principer. Programmets huvuddel körs i en huvud-loop (while (running)) tills användaren avslutar. Inuti loopen visas huvudmenyn med tre alternativ: Administratör, Köpare eller Avsluta. Programmet läser användarens val via Console.ReadLine() och hanterar det med en if-else-struktur.

Designprinciperna inkluderar:

* Modularisering: Funktionaliteten i Main metoden är uppdelad i metoder som hanterar specifika uppgifter (t.ex. AdminMenu och BuyerMenu). Och varje metoder sedan uppdelat och hanterar specifika uppgifter i sin tur. Metoden DisplayBooks() används för både administratören och köparen när de vill se tillgängliga böcker i lagret.
* Användarvänlighet: Användargränssnittet är designat enkelt med tydlig textformatering och färg. Menyn noll ställs när både köparen och administratör kommer tillbaka till huvudmeny som är en fördel även för sekretessfrågor.
* Felhantering: Ogiltiga användarval hanteras med feedback och omdirigering till huvudmenyn.
* Loop och kontrollflöde: En evig loop styr interaktionen, och användaren kan fortsätta göra val tills programmet avslutas.

Flödesschema för programmets centrala del d.v.s. huvudmenyn är nedan.

En bild som visar diagram, text, skiss, Plan

Automatiskt genererad beskrivning

Klassdiagram över hela programmet.

En bild som visar text, skärmbild, diagram, Teckensnitt

Automatiskt genererad beskrivning

# **Resultat**

Programmet börjar med en välkommandemeddelande och information om företaget och applikationen. Därefter får användaren välja ett av alternativen beroende på sina preferenser. Om användaren väljer att avsluta programmet stängs det efter tre sekunder med en ljudsignal som indikerar avslutningen.

Vid alternativ 1 hämnar användaren i administratörsmenyn och får behörighet enligt nedan.

En bild som visar text, dator, skärmbild, programvara

Automatiskt genererad beskrivning

Varje alternativ aktiverar ett annat alternativ och funktionalitet.

Här användaren har valt Administratörsmeny.

En bild som visar text, skärmbild, Teckensnitt, programvara

Automatiskt genererad beskrivning

Här administratören har valt 1 för att se alla böcker i lagret

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Efter att administratören har sett samtliga böcker i lagret looper Administratörsmeny och administratören hämnar i sin meny och enlgit det kan välja ett alternativ. Vid felaktig matning får administratören meddelande om det och bes att ge rätt inmatning.

Så här ser ut köparensmenyn

En bild som visar text, skärmbild, Teckensnitt, design

Automatiskt genererad beskrivning

Varje alternativ aktiverar i detta menyn också ett alternativ och funktionalitet.

Här köparen har valt

1. För att visa en lista över samtliga böcker.
2. För att lägga till böcker i kundvagnen genom att mata in bok-ID och antal.
3. För att se rabatten och det totala beloppet att betala.
4. För att slutföra köpet.

Se bilden i nedan

En bild som visar text, skärmbild, meny

Automatiskt genererad beskrivning

## **Slutsats**

I detta projekt lärde vi oss mycket praktiska delar av objektorintering programmering i C#. Vi lärde oss mycket kring datastrukturer och klassar och arv. Vi hade initialt svårt att koppla klasser och välja mellan enum och klass vilket ledde till omskrivningar av koder och tidsförlust. Framöver planerar vi att lägga mer tid på pseudokod och djup förståelse av projekt beskrivning särskilt user story tolkning.

Vi uppnådde det resultat vi visualiserade i början av projektet men utvecklingen tog längre tid än vad vi planerade. Vi var tveksamma till att återanvända kod från tidigare laborationer men insåg att det var ett misstag. Om vi hade kopierat och anpassat tidigare valideringskod hade vi sparat tid och minskat arbetsbelastningen. Detta är något vi kommer att tänka på i framtida projekt för att effektivisera processen.

Testfasen var en viktig del av projektet eftersom vi ville säkerställa att programmet inte kraschar och att användarinmatning hanteras korrekt. En viktig lärdom var att ha god kodkvalitet från början för att undvika problem vid testning. Vi ville gärna utveckla projektet till en mer avancerad lösning, till exempel en Windows Forms-app men vi hade inte tillräcklig kunskap för att genomföra det. Något som vi kommer genomföra i framtiden.

# **Referenser**

* Troelsen, A., & Japikse, P. (2022). Pro C# 10 with .NET 6 : foundational principles and practices in programming (11 uppl.). New York: APress. ISBN: 9781484278680.
* Team draw.io. (n.d.). *Create UML class diagrams*. Retrieved December 30, 2024, from <https://www.drawio.com/blog/uml-class-diagrams>
* Lucid Software. (2023, August 10). *UML class diagrams* [Video]. YouTube. <https://www.youtube.com/watch?v=6XrL5jXmTwM>
* Panjuta, D. (n.d.). *Complete C# masterclass: Master C# programming from A to Z. Dive deep into .NET, OOP, Clean Code, LINQ, WPF, Generics, Unit Testing, and more*. Udemy. <https://www.udemy.com/course/complete-csharp-masterclass/?srsltid=AfmBOooEShZoQHB-VlqQzIskJ_I6ABITjVHKFpL30BFwis4JMN4R5T3O&couponCode=KEEPLEARNING>
* Caney, J. (2020, April 10). Why do we get possible dereference null reference warning, when null reference does not seem to be possible? *Stack Overflow*. <https://stackoverflow.com/questions/59524568/why-do-we-get-possible-dereference-null-reference-warning-when-null-reference-d>
* Wazupdanger. (2023, December 25). *Forgive me ……* Reddit. <https://www.reddit.com/r/csharp/comments/1682yux/forgive_me_if_im_like_new_or_something_but_for>
* Kodexempel: Ahmad, S., & Lindfors, L. (2024). *WeatherSenseApplication console application in C#* (Lab 3, students of the Department of Computer Science at Dalarna University.).

# **Appendix**

## **Parprogrammeringslogg**

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Tid i timmar** | **Ensam eller i par** | **% fördelat i att sitta vid tangentbordet** |
| 2024-12-27 | 8 | par | 50%, 50% |
| 2024-12-29 | 7 | par | 50%, 50% |
| 2024-12-30 | 8 | par | 50%, 50% |
| 2024-12-31 | 4 | par | 50%, 50% |
| 2024-01-02 | 6 | par | 50%, 50% |