Mikael Karlgren

Design för projekt 1 – nätbutik i C#

De viktigaste designbesluten rörde sig mestadels runt den visuella GUI:n. Då jag visste att jag ville ha nån typ av varukorg-lista så var DataGridView perfekt att börja med, vilket i faktum tog mest tid i projektet. Jag tog mycket tid på mig att testa vad jag tyckte de optimala egenskaperna var för denna objekt-typ. För att namnge några egenskaper så var de;

- full-rad selektion istället för cell-för-cell selektion så att hela produkten markeras
- `RowHeadersVisible = false`, för att ta bort den första tomma raden längst till vänster
- en DataGridViewCheckBoxColumn för att snyggt visa bool-värden
- specifik font för rubriker, för att snyggt separera rubrikerna till produkterna
- användarens oförmåga att förändra storlekar och innehållet i rutnätet, så att det visuella GUI:t ser mer simpelt och ”clean” ut, samt så att användaren inte
    - skapar problem i programmet
    - inte blir förvirrad av att kunna redigera saker hen inte borde kunna

När jag såg att detta såg och funkade väldigt bra så valde jag av att även ha en DataGridView inte bara till min kundvagn utan även mitt sortiment. När jag hade två DataGridViews på ett fönster märkte jag dock att gränssnittet såg väldigt kladdigt ut, programmet var inte så ”clean” längre. Jag valde därför att använda mig av två flikar, en för sortiment och en för kundvagn, vilket gjorde det hela ”clean” igen, det såg snyggt ut och var mycket mer strukturerat. Jag valde då att inte använda mig av någon typ av FlowLayoutPanel med bilder för sortimentet. Jag bestämde att skippa bild-stöd helt, vilket hade vart en bra funktion att ha på sortiment-listan. Detta skippade jag då jag istället fokuserade på att göra ett bra projekt färdigt i tid.

Angående datan så tycker jag att det blev bra, jag sparade all viktig data (rabattkuponger, sortiment & kundvagn) i huvud-programmet, och den andra datan i de klasserna de användes i. De svåraste design-besluten rörde sig mer om hur klass-filerna skulle se ut. Till en början så hade jag väldigt mycket kod i min GUI-klass, t.ex. kod som rörde sig runt event-händelser. Jag valde därför att göra en enskild klass till varje liten ”del”, så att t.ex. GUI-klassen `ShopWindow` och Event-klassen `EventHandling` hade sin egna fil. Detta gjorde klasserna mycket snyggare, men i de scenarion data var tvungen att förmedlas mellan de två klasser så skapade de ibland design-besvär. Jag försökte t.ex. undvika statisk kod med GUI-koden, och kunde använda `EventHandling`-klassens metoder o-statiskt genom att skapa ett objekt av klassen. Jag kunde hursomhelst inte göra det samma baklänges så att `EventHandling.cs` använde sig utav `ShopWindow.cs` o-statiskt på ett snyggt sätt. Jag experimenterade med det ett tag men gav sedan upp för att låta event-klassen nå några av GUI-klassens variabler och metoder på ett statiskt sätt. Det skapade ju inga problem utan det vara bara jag som ville att det skulle se bättre ut.

För att slutligen göra projekt-strukturen snyggare så sorterade jag in de flesta av klasserna in i mappar. Jag skapade en ”Data” mapp för .csv filerna, samt en ”Util” mapp för all kod som skulle kunna användas igen i andra projekt. Resten av materialet som är ”hårdkodat” just till denna applikation, så som `Program.cs`, `EventHandling.cs` och `ShopWindow.cs` fick ligga fritt utan mapp.