Mikael Karlgren

Översikt för projekt 1 – nätbutik i C#

Detta dokument beskriver hur C# koden i detta projekt fungerar. Alla småsaker är inte listade, utan detta dokument beskriver endast projektet översiktligt. Programmet startar i ”`Program.cs`”-klassen. Denna klass har fyra användningsområden; starta programmet, hämta och lagra lokala data, lagra kundvagnens varor och slutligen att starta programmets GUI.

Den lokala lagrade datan som programmet inhämtar på datorn är produkt-data och rabatt-data, vilket är sparat i .csv-format. Denna lokala data finns sparad i ”Data” mappen i filerna ”Produkter.csv” samt ”Rabattkoder.csv”, och dessa filers innehåll kan redigeras av t.ex. butikspersonal för att ändra sortimentet samt modifiera butikens rabattkoder.

Text-datan som hämtas från dessa två lokala filer konverteras till arrayer av objekt-typerna ”`Product`” och ”`Discount`”, och lagras sedan i programmet. Denna objekt-array konvertering sker med hjälp av objekt-klasserna samt metoder från ”`Data.cs`”-klassen, som finns placerad i ”Util” mappen. `Data.cs`-klassen har också en metod som används för att hämta data (i string[] format) från de lokala filerna. Konverteringen och data-inhämtningen använder sig av felhantering och kraschar programmet ifall nån data är formulerad på fel sätt eller om filerna inte finns, för att undvika problem som kan orsakas i följd. För övrigt så är har projektets andra klasser också modifierats för att vara felsäkra. Vidare så är alla klasser som är placerade i ”Util” mappen skapade utan förbindelse av projektets andra klasser, så att de kan användas i andra projekt.

Programmets GUI och dess kod som `Program.cs` startar är ”`ShopWindow.cs`”. Detta är klassen som visar upp applikationens grafiska fönster, och låter användaren se information samt kommunicera med programmet. Denna grafiska klass har även uppgiften att skapa ett objekt av `EventHandling.cs`, där programmets kod som angår GUI-events finns. GUI:n har två flikar (två TabPage-objekt i en TabControl) som användaren kan växla mellan; en sortiment-flik och en kundvagn-flik. Flikarna ser rätt lika ut men har några skillnader. Båda flikar har en rubrik (Label) uppritad där det står ”Sortiment” eller ”Kundvagn”, i följd av en DataGridView (`productGrid` & `cartGrid`). Dessa två DataGridView-objekt skapas med `#CreateUnmodifiableDataGridView` metoden som finns i `Util#Forms` klassen. Metoden skapar och returnerar ett DataGridView-objekt med egenskaper som gör objektet snyggare, samt inaktiverar användarens förmåga att direkt redigera rutnätets innehåll.

I sortiment-flikens DataGridView så läggs den inhämtade produkt-array datan till direkt efter objektet har skapats. Användaren ser sortimentets produkt-data enkelt i rad-för-rad form. För att lägga till produkten i kundvagnen så klickar användaren på ”Lägg till”-knappen (`addCartItemButton`) medans produkten som ska läggas till i kundvagnen är markerad. När programmet märker av knappens tryck-event (Click) så körs ”`#AddCartItem`” metoden i `EventHandling.cs`-filen. Den första gången användaren klickar på denna knapp så visas lite information till användaren för att veta att produkten successivt blev tillagd till varukorgen.

Knapparna i kundvagn-fliken blir även aktiverade så att man kan klicka på de. Förutom detta så läggs produkten självklart till i kundvagnen.

Kundvagns-tilläggningen görs genom att lägga till produkten i ”Cart” Dictionary:t som finns i huvudfilen Program.cs. För att veta antalet av produkten som ska köpas så används ett nytt objekt, ”ShopItem.cs”, vilket är väldigt likt Product-objektet då detta objekt ärvs. Skillnaden är att ShopItem.cs inkluderar en sammanfattning-metod vilket returnerar en sträng om varan, samt att objektet har en int ”Amount” instansvariabel för att veta produktens antal i sortiment eller kundvagn-syfte (i detta fall endast för kundvagn). Ifall produkten som ska läggas till i varukorgen redan är listad i Cart-variabeln så läggs inte ett nytt objekt till, utan ShopItem-objektet som redan finns i Cart-Dictionary:t modifieras för att höja ”Amount”-instansvariabeln med ett (1). Efter att ShopItem-objektet lagts till i varukorgen så läggs dess data till i cartGrid-objektet för att göra det visuellt för användaren.

För att beställa/rensa produkterna i varukorgen eller för att ta bort en enskild produkt så går man till varukorg-fliken. Denna flik är lik sortiment-fliken, skillnaden är att produkt-datan som visas endast är produkter som valts och lagts till av användaren, vilket är data från ShopItem-objekt istället för Product-objekt. Denna vy har även tre knappar, ”Beställ”, ”Ta bort” (enskild markerad produkt) och ”Rensa” (hela varukorgen) där varje knapp såklart har kopplat sin ”Click”-händelse till en metod i EventHandling.cs-klassen för att genomföra dess funktion. ”Ta bort” och ”Rensa” Click-metoder är relativt lik AddCartItem-metoden, metoderna modifierar datan i Cart-variabeln samt innehållet i cartGrid-objektet. De tre kundvagn-knapparna blir även inaktiverade (tills nästa produkt är tillagd) ifall varukorgen blir tom.

”Beställ”-knappens (finalizeOrderButton) ”Click”-händelse som är kopplad till EventHandling.cs metoden #FinalizeOrder beter sig lite annorlunda. Metoden börjar med att anropa #GetDiscount metoden som finns i Util#Input.cs-klassen. Denna metod öppnar en dialogruta som frågar om användaren vill ange en rabattkod, genom att kalla Microsoft#VisualBasic#Interaction#InputBox metoden; en gammal metod skapad av Microsoft som är gjord för att hantera inmatning från en dialogruta. Om användaren inmatar en giltig rabatt-kod så returneras Discount-objektet som är kopplat till den rabattkoden, och om användaren matar in en felaktig rabattkod så får hen försöka igen. Ifall användaren inte matar in något eller klickar på ”Cancel” så returneras ett Discount-objekt som är null.

#FinalizeOrder metoden fortsätter sedan med att loopa igenom elementen i Cart-Dictionary:t för att räkna ihop varornas alla kostnader och dess sammanfattningar. Efter detta så appliceras rabattkodens procentsats för att minska total-kostnaden, så länge Discount-objektet inte är null. Till slut så visas beställning-informationen ut i form av ett kvitto till användaren med en MessageBox#Show dialogruta, och ShowWindow.cs metoden #ClearCart kallas; vilket rensar Cart-variabeln, rensar innehållet i cartGrid samt inaktiverar de tre knapparna på kundvagn-fliken (tills en ny produkt har lagts till).