Multifabriken

Reflektioner

Jag bestämde mig ganska tidigt att använda en superklass och sedan köra vidare med arv på mina subklasser. Detta för att jag ansåg att det blev en mer clean kod då jag kan skapa en lista av superklassen och sedan loopa igenom den. I och med superklassen skapar jag en metod (showOrder) som går att skriva över i subklasserna, där jag kan specificera properties per klass i Console.WriteLine. Sedan när jag instansierar klasserna och kallar på metoden så får jag ett unikt meddelande per klass.

Jag valde att skapa en klass per produkt (substantiv) för att det helt enkelt var best practise av det vi lärt oss i denna modul.

Valde även att lägga in validering på datan som är av int och detta gör jag via en metod som heter ValidateNum i klassen Menu, detta för att programmet inte ska krascha ifall användare inte fyller i en siffra när programmet ber om det.

Det som ligger i klass Menu låg initialt i Program men valde att flytta över detta så att program blev mer clean och för att jag ansåg att det blir enklare om någon utvecklare skulle ta över koden.

Beskrivning

När programmet startar så instansierar jag klassen Menu. Jag skapar en lista utanför Do While loopen för att inte skapa en ny lista varje gång programmet ”loopar om”. Sedan skapar jag en variabel av typen bool som jag anger till true, sedan när användaren avslutar programmet ändrar jag denna till false.

Anledningen till att jag använder mig av en Do While loop är för att printa ut menyn en gång innan loopen börjar användas.

Jag skriver ut de olika meny valen och sedan fångar jag input från användaren.

Beroende på vad användaren anger så går koden in i switch och respektive case.

Case 1-4 är relativt lika, jag rensar konsolen och skriver ut diverse olika frågor för respektive klass, fångar input och sedan instansierar jag klasserna med variablerna som parametrar. Till sist så lägger jag till objektet i listan Products.

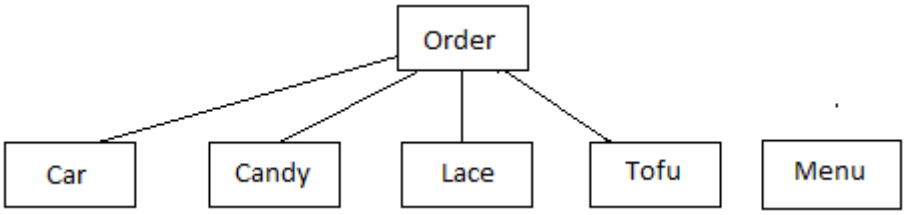
Det som skiljer case 1-4 åt är de produkter man kan beställa som har nummer/int, då har jag skrivit en metod ValidateNum. I denna metod validerar jag så att användaren har fyllt i en siffra. Försöker parsa input från användaren till en int och om det inte går så gör jag en bool(!) while loop som skriver ut ”Vänligen ange en siffra (heltal)” tills användaren fyller i en siffra och till sist returnerar jag number.

Case 5 så loopar jag igenom listan jag skapade med hjälp av superklassen och kallar sedan på metoden showOrder().

Case 6 så skriver jag ut ett avskedsmeddelande samt ändrar variabeln restart till false så att programmet ska sluta loopa.

Beskrivning Klasser

Har strukturerat mina klasser följande:



Order är min superklass som jag satte som abstrakt då den ej ska gå att instansiera. Sedan skapar jag en metod även den som abstract så går ej ha något kodblock där, utan den är till för att skrivas över i subklasserna. Har satt properties inputColor och inputPcs som protected i superklassen då dessa går att använda över flera subklasser.

Subklasser: Anger de properties som är unika för klassen och anger dessa till private. Skapar sedan en konstruktor (samma namn som klassen) där jag fyller i parametrar med respektive typ (string, int).

Sedan skapar jag metoden showOrder() med en override för att skriva över metoden i superklassen samt void för att den inte ska returnera något. Skriver sedan ut ett unikt meddelande per klass till konsolen med korrekt field.

Menu: Menu var den klassen jag skapade sist, till en början låg det i program. Valde att göra så för att få program mer clean samt att alla menyval är i Menu.