Rapport Multifabriken. Av Jonatan Hellqvist

Under hela uppgiften har jag försökt tänka att det ska vara så enkelt som möjligt att lägga till framtida produkter men samtidigt hålla en personlig dialog med användaren för de enskilda produkterna då de har väldigt olika variabler.

För uppgiften som sådan kändes det onödigt med någon superklass eller interface men har tänkt att det kanske skulle vara att föredra om man faktiskt skulle lägga till fler produkter till programmet. Då mest för att det skulle kunna bli ännu tydligare att man behöver ha en arraylist för varje produkt, ett produktnamn och personliga metoder för ”printInfo” och skapande av nya objekt.

Försökte göra så många gemensamma metoder som möjligt och hålla filer och klasser sorterade och separerade så att man lätt ska kunna förstå vad som händer, vart man hittar allt och var det skall placeras.

Valde att lägga alla metoder som rörde skapande av orders och utskrift i en separat klass ”Orders”, möjligtvis att man kunde haft skapandet av objekten i respektive klassfil men personligen så kändes det smidigare att ha dem på samma plats för lättare åtkomst om man skulle ändra på flera metoder samtidigt.

I ”Menu” klassen lade jag de listor som skulle använda sig av printmetoderna för menyn. Kändes vettigt att ha alla dessa på samma ställe för eventuella framtida ändringar.

Till variabler som behövde låsa användaren till ett antal val har jag använt mig av switch satser och listor för att på ett enkelt sätt hantera detta i en gemensam metod ”printSwitch” med cases. Då det annars blev väldigt rörigt med en ny switchsats i main cases och sedan if satser på det så kändes detta som den bästa lösningen.

Här stötte jag dock på en del problem, dels behövde listorna vara lika långa (hittade inget bra sätt att skapa cases efter längden på listan, dels kunde man inte på ett smidigt sätt skicka tillbaka användaren till main menyn med en ”continue” vid back val eller felaktig input. För att lösa continue biten gjorde jag helt enkelt så att den ingående variabeln returnerar ”backOrError” vid back val eller vid felaktig input och sedan en if sats som hanterar detta värde.

Vill man ha olika längd på dessa listor så är det nog en bättre lösning med en if sats som hanterar olika indexvärden.

För att kolla om ett värde är int eller double gjorde jag metoder för detta, som också returnerar ett minusvärde om ett felaktigt val anges för att på ett enkelt sätt felhantera med default i main switchen. Dock gick dessa inte använda till de inputs där objekten skapas men sparar några rader kod.

Till de inputval som användaren anger har jag valt att inte använda några metoder, alla produkter har så unika attribut och det blev väldigt annorlunda uttryck för dess if satser och felmeddelanden osv att det kändes meningslöst. Hade man velat ha en lite mer clean main fil hade man kunnat skapa dessa som metoder i Menu filen men då känns det som det går emot DRY principen om de ändå bara används 1 gång.

Lade också dessa inputs i while loops så man kan guidea användaren genom olika fel som kan uppstå utan att man behöver börja om med sin order från början.

Slutligen så skrev jag en liten förklaring för skapandet av nya produkter som kommentarer längst upp i programmet för extra tydlighet.