**Rapport inlämningsuppgift 2**

**Objektorienterad prorammering i C#**

**Hur jag bröt ner problemet till mindre delar.**

Som med alla uppgifter är det bra att bryta ned ett problem i mindre delar. Jag utgick mer eller mindre från instruktionerna och gjorde punkt för punkt. Vissa punkter gick även att dela in i ytterligare mindre delar. Jag började med inloggningsfunktionen för att sedan skapa menyn. Efter detta var gjort tog jag mig an att skapa en en klass (Member) som fungerar som en mall för varje medlem i basgruppen. När jag var klar med den skapade jag en lista som innehöll varje medlem. Sedan kom jag fram till implementeringen av menyn. Det vill säga det som ska ske när användaren gör ett val från menyn. Detta steg delade jag upp så att varje valmöjlighet blev en egen del. Sedan gick jag mer eller mindre uppifrån och ned och tog menypunkt för menypunkt och skapade upp implementeringen. Jag började med att skriva all kod i Main-metoden för att sedan undan för undan refoktorera koden in i olika metoder. Jag

försökte hålla mig till att varje metod bara skulle göra en sak vilket ledde till ett flertal metoder. Jag skapade en textfil där all information om alla medlemmar sparas. Texten från textfilen läses sedan in med hjälp av en metoden Read(). Till sist skapade jag tre klasser (Print, CRUD och Logic) där jag placerade de olika metoderna.

**Inloggningsfunktionen**

Här får användaren skriva in ett lösenord som sedan jämförs med strängen "Bästkusten". Om strängarna är lika får användaren tillträde till programmet och metoden Run() anropas som sedan körs tills användaren avslutar programmet. Stämmer inte användarens inmating överens med lösenordet skrivs ett felmeddelande ut och användaren får försöka igen med hjälp av en do/while-loop. När jag var klar med det tyckte jag att det skulle vara kul att försöka göra så att lösenordet som användaren matar in döljs av asterisker likt ett riktigt lösenord, samtidigt som knapptryckningarna sparas till en sträng i bakgrunden. Jag implementerade detta med hjälp av ConsoleKeyInfo och Console.ReadKey(true), jag använder 'true' så att kanpptrycken registreras men inte syns. Trycker användaren endast Enter och inget annat skrivs ett lösenordstips ut till skärmen. Matar användaren in fel lösenord tre gånger i rad avslutas programmet.

**Menyn**

För att skapa upp menyn använde jag flera Console.WriteLine(). Varje alternativ fick en siffra kopplad till sig för att underlätta för användaren. Koden för menyn extraherade jag till en egen metod som jag döpte till MainMenu() och som sedan lades i klassen Print.

**Mallen för medlemmarna**

Jag skapade en klass som jag döpte till Member eftersom den skulle fungera som en mall för alla medlemmar i basgruppen. Jag gav Member 12 privata fält (jag lade till ett fält för efternamn och ett extra fält som jag döpte till gender för att kunna använda pronomen (han/hon) för varje medlem). Sedan skapade jag upp properties för varje fält. Jag skapade två konstruktrer, en tom (just in case) och en som tog emot argument för varje fält vilket gjorde att jag kunde instansiera varje medlem på ett enkelt sätt. Till sist skapade jag en metod, DescribeMember(), som använder alla fält för att beskriva medlemmen och skriva ut det till skärmen. Jag gjorde också en override av metoden ToString() vilken nu skriver ut medlemmens för- och efternamn.

**Läsa från en textfil**

Jag skapade metoden ReadFromTextFile() som läser in information från en textfil och sparar den i en lista. Informationen delas sedan upp, med hjälp av en for-loop, och sparas i olika variabler som skickas in som argument till metoden AddMemberToList() som för varje iteration lägger till en ny medlem i en lista. ReadFromTextFile() anropas varje gång användaren återgår till huvudmenyn för att få en korrekt version av informationen i textfilen.

**Implementeringen av menyn**

Detta steg delade jag upp i flera mindre steg där jag valde att tackla varje

moment för sig.

**Visa alla medlemmar**

Första delen är att lista alla medlemmar i basgruppen separerat med ett kommatecken. Här använde jag en for-loop vilket gav mig mer kontroll än en foreach. Detta gjorde att jag kunde bestämma om det skulle va ett kommatecken efter namnet eller ett &-tecken innan sista namnet. Jag valde även att göra så att programmet byter till en ny rad efter att 5 namn skrivits ut så att texten inte sticker iväg alldeles för långt åt höger.

**Visa detaljerad medlemsinformation**

Andra delen är att låta användaren få ut detaljerad information om en specifik gruppmedlem. Detta görs med hjälp av en metod, MemberDetails(), som först anropar ListOfMembers() som listar alla medlemmar. Sedan anropas metoden DescribeMember() via den gruppmedlem som valts som i sin tur skriver ut den detaljerade informationen. Denna del körs sedan ända tills användaren väljer att återgå till huvudmenyn igen.

**Skapa en ny medlem**

Sedan valde jag att lägga till en egen del som ger användaren möjligheten att lägga till en medlem i gruppen. Detta görs genom att användaren får mata in alla värden för medlemmen. Sedan sparas detta till textfilen tillsammans med alla andra medlemmar så att den nya medlemmen finns kvar även om programmet stängs ned och startas om. Detta görs med hjälp av File.WriteAllLines().

**Ändra en medlem**

Fjärde delen är också det en egen tillagd del som låter användaren ändra på information om en gruppmedlem. Först listas medlemmarna med ListOfMembers() och sedan får användaren välja vilken medlem som ska ändras. Efter det får användaren välja vilken del av informationen som ska ändras. Om användaren väljer att endast trycka Enter eller om användaren matar in bokstäver där det förväntas siffror avbryts åtgärden annars byts den gamla informationen ut mot den nya. Detta sparas sedan till textfilen med File.WriteAllLines().

**Ta bort en medlem**

Femte delen är att ta bort en medlem. Detta görs med metoden DeleteMember() som först visar alla medlemmar med hjälp av ListOfMembers(). Sedan får användaren välja vilken medlem som ska tas bort och den medlemmen tas sedan bort från listan med hjälp av metoden RemoveAt(). Medlemmen tas även från textfilen med File.WriteAllLines().

**Avsluta programmet**

Sista delen är att avsluta programmet och detta görs med metoden ExitProgram(). Här skrivs en avslutande text ut till skärmen och sedan sätts bool exit till true vilket gör att villkoret i do/while-loopen inte längre är sant och programmet avslutas.

**Namngivning**

När det kommer till namngivning använde jag mig av camelCase för alla variabler och PascalCase för metoder och klasser. Jag använde enbart engelska variabelnamn och jag försökte vara så beskrivande som jag bara kunde så att namnen tydligt skulle indikera vad de var till för.

**Saker jag kunde gjort annorlunda**

Min felhantering i metoden CreateNewMember() är något bristfällig. Det enda jag har gjort är att åtgärden avslutas om användaren trycker Enter utan att mata in något. Detsamma sker också om användaren matar in något annat än det som förväntas (tex bokstäver till de fälten som ska spara ett heltal) men annars kan användaren mer eller mindre skriva in lite vad som helst på varje fält. Detta skulle jag kunnat gjort mer utförligt. Koden för hur metoden avbryts om användaren väljer att avbryta är inte heller den bästa med mycket liknande kod vilket gör att jag kanske inte är helt DRY där. Det är också något jag kunde gjort bättre även om jag försökte skapa två metoder, en för att tilldela ett värde och en för validering, för att minska ner på all upprepad kod.

**Övrigt**

När det gäller for- och foreach-loopar valde jag att använda båda för att göra mer eller mindre samma sak fast på olika sätt. Till exempel använder jag en for-loop för att skriva ut alla medlemmars namn på ett ställe men en foreach-loop på ett annat. En foreach-loop är väldigt smidig när man bara vill skriva ut allt ur en lista från början till slut, men när man behöver lite mer kontroll på var i listan man befinner sig är en for-loop så mycket bättre.

Jag lade till två metoder som bara är till för ett estetiskt syfte. Ena metoden Print.TextInGreen() skriver ut ett meddelande som skickades med som argument i grönt och den andra Print.TextInRed() skriver ut ett meddelande i rött.

Till sist lade jag till lite rubriker med hjälp av en ASCII Art Generator vilken gör programmet lite roligare att titta på. Jag hade säkert kunnat sparat rubrikerna till en textfil och sedan läst in dem i programmet likt jag gör med medlemmarna i Read()-metoden istället för att hårdkoda dem med Console.WriteLines, men jag var var inte helt säker på hur jag skulle göra det på ett smidigt sätt så jag lät dem vara hårdkodade direkt i programmet.