

Labbrapport: Labb 4

Författare: Jonathan Klinratree & Daniel Stadler

Datum: 2025-01-09

Kursnamn: GIK299 – Objektorienterad Programmering

Examinator: Elin Ekman och Ulrika Artursson Wissa

Innehållsförteckning

1. Introduktion	2
2. Metod	2
2.1. Verktyg	2
2.2. Stegvis beskrivning av tillvägagångssätt	2
2.3. Förutsättningar för att göra labben	3
2.4. Testning av koden	3
2.5. Etiska överväganden	3
3. Resultat	3
4. Diskussion och reflektion	4
4.1. Diskussion kring resultat	4
4.2. Reflektion kring sprint 1	4
4.3. Reflektion kring sprint 2	4
4.4. Reflektion kring alternativa lösningar	4
Frågor till AI-verktyg	4
Referenser	5

1. Introduktion

I den här labben har vi genomfört programmeringsuppgifter för labb 4 som i korthet gick ut på att skapa ett program som kan lägga till personer med deras respektive egenskaper. Syftet med labbrapporten är att beskriva hur vi genomförde labb 4 och hur arbetet fördelas.

2. Metod

2.1. Verktyg

- Visual Studio 2026, version 18.1.1, tillsammans med ramverket .NET 10.0 användes för att bygga och testa vår applikation på Windows.
- Rider 2025.2.4, tillsammans med ramverket .NET 9.0 användes för att bygga och testa vår applikation på macOS.
- Kodexempel baserat på kursmaterialet av Elin Ekman

2.2. Stegvis beskrivning av tillvägagångssätt

Labb 4 löstes genom att noggrant läsa igenom uppgiften för att förstå vad som efterfrågades. Eftersom vi fick en tydlig bild på vad programmet skulle utföra arbetade vi sedan stegvis baklänges från målet och implementerade funktionerna. Denna metod gjorde det enkelt för oss att strukturera arbetet.

Vi fördelade arbetet genom att göra våra egna program. Detta var för att båda skulle kunna vara 100% aktiv och samtidigt lära sig så mycket som möjligt. Därefter delade vi våra program med varandra och gjorde en gemensam för att få en bredare syn på hur uppgiften kunde lösas. Då alla har olika kunskaper och erfarenheter kan det bli att den ena som är mer kunnig gör allt, men i detta fall löstes det och båda kände sig nöjda.

2.3. Förutsättningar för att göra labben

För att kunna göra labb 4 krävdes det tillgång till en dator med installerad IDE för C#. För att kunna använda C# på macOS behövdes IDE:n JetBrains Rider användas och på windows användes Visual Studio Code.

2.4. Testning av koden

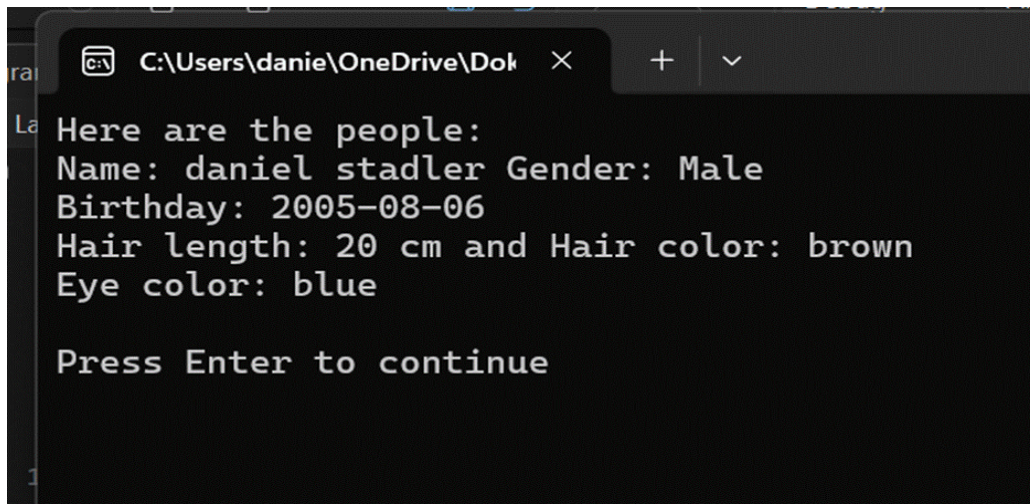
Båda IDE:erna har inbyggda debug funktioner där man kan sätta breakpoints. Detta användes tillsammans med IDE:ernas inbyggda felsökningsfunktion för att kunna identifiera fel i koden.'

2.5. Etiska överväganden

Ingen etiska överväganden var nödvändiga eftersom inga verkliga personuppgifter eller känslig data hanterades i denna laboration. Labben handlade inte om att få tag på en verklig persons personuppgifter, utan skapa fiktiva egenskaper.

3. Resultat

Vi använde oss av de erfarenheter som de tidigare labben lärde oss, för att genomföra uppgiften. Ett problem vi upptäckte var när användaren printade ut listan av personer försvann listan för snabbt. Detta var på grund av att vi använde `Console.Clear()`; efter printen, så för att lösa detta problem så la vi till ett extra steg där användaren behöver klicka enter för att gå vidare efter den printat listan.



The screenshot shows a Windows command prompt window with a dark background and white text. The title bar at the top indicates the path `C:\Users\danie\OneDrive\Doc`. The output of the program is as follows:

```
Here are the people:  
Name: daniel stadler Gender: Male  
Birthday: 2005-08-06  
Hair length: 20 cm and Hair color: brown  
Eye color: blue  
  
Press Enter to continue
```

4. Diskussion och reflektion

Under arbetet lärde vi oss att förbättra användarens upplevelse genom justeringar. Där ett exempel var att lägga till en extra kommando instruktion efter "List" skrivits ut, där användaren får trycka på "Enter" när den är klar med att läsa listan. Det gjorde att användarens upplevelse blev bättre och inte hastig då det nu fanns en möjlighet att läsa informationen i lugn och ro.

4.1. Diskussion kring resultat

I resultatdelen tar vi upp problemet med att `Console.Clear()`; tar bort all text och gör att listan blir oläslig, hur vi löste det problemet var med att lägga till ett paus mellan print och clear som gör att användaren kan läsa listan. Detta gjorde vi med hjälp av `Console.ReadLine()`; med en `Console.WriteLine("Press enter to continue");` över för att förklara för användaren hur de ska fortsätta.

4.2. Reflektion kring sprint 1

Vi lärde oss om hur konstruktörer fungerar i C# då, båda två av oss aldrig hade använt det utanför skolan.

4.3. Reflektion kring sprint 2

Vi hade inga stora förändringar i vårt arbetssätt då vi gjorde båda sprintarna under samma tillfälle, eftersom vi båda gjorde uppgiften själva först och sen satte ihop våra erfarenheter till samma program.

4.4. Reflektion kring alternativa lösningar

Om vi fick full kontroll över uppgiften hade vi valt att göra uppgiften i python då vi aldrig använd C# tidigare.

Frågor till AI-verktyg

Verktyg: CoPilot

Fråga/prompt: Autokorrekt och kompatibilitet. Frågan var "I have a C# program that I want to run on VSCode, what do I do"

På vilket sätt svaret användes: Svaret hjälpte oss att skriva saker som switch-cases mycket snabbare. När man skriver så ger CoPilot förslag på hur meningen ska avslutas, så vi använde det. Daniel ville testa om koden fungerade i VSCode, då använde vi CoPilot för att få förslag på hur man kan fixa det. Medan vi programmerade märkte vi att man kan använda `Console.ReadLine();` som en paus i koden, hur vi använde det var att efter användaren fråga efter att printa ut listan, utlöser vi en `Console.ReadLine();` med meddelandet "Press Enter to continue". Då behöver användaren trycka enter för att komma vidare, det gör att vi kan visa listan på skärmen tills användaren trycker enter. I framtida projekt skulle vi kunna använda det igen då det är en ganska lätt men komplett lösning.

Referenser

Vi hade inga referenser