GIK299: Objektorienterad Programmering – Labb 4

Laborationsrapport

Författare: Ariz Biswas Islam
Datum: 2025-04-02
Kursnamn: GIK299 – Objektorienterad Programmering
Examinator: Elin Ekman och Ulrika Artursson Wissa

Innehållsförteckning

1. Introduktion

I denna laboration har jag genomfört programmeringsuppgiften för Labb 4. Syftet med labben var att simulera hur persondata kan samlas in i ett system med hjälp av objektorienterade principer. Arbetet var uppdelat i två sprintar. I Sprint 1 skapade jag grunden för att lagra information om personer. I

Sprint 2 byggde jag vidare programmet med inmatning av flera personer, menyhantering och felhantering.

2. Metod

2.1 Verktyg

- Visual Studio 2022, version 17.0
- .NET 6.0

2.2 Stegvis beskrivning av tillvägagångssätt

I Sprint 1 skapade jag en enum Gender som innehåller olika könsalternativ. Jag skapade även en struct Hair som innehöll information om hårfärg och hårlängd. Därefter skapade jag en klass Person som innehöll information om namn, kön, hår, födelsedatum och ögonfärg. Jag lade till en ToString()-metod i klassen som presenterade personens information. I Main-metoden skapade jag ett hår- och ett person-objekt med hårdkodade värden och skrev ut informationen.

I Sprint 2 utvecklade jag programmet vidare genom att skapa en meny med tre alternativ:

-Lägga till en person
-Visa alla personer
-Avsluta programmet
Programmet kördes i en while-loop tills användaren valde att avsluta. Jag skapade två metoder: LäggTillPerson() och VisaPersoner(). Jag lade också till validering av inmatning med hjälp av if-satser, try-catch och TryParse() för att hantera felaktig data.

2.3 Förutsättningar för att göra labben

För att genomföra labben behövde jag ha Visual Studio 2022 installerat och skapa ett konsolapplikationsprojekt med .NET 6.0 som ramverk.

2.4 Testning av koden

I Sprint 1 testade jag koden genom att köra programmet och kontrollera att personens information skrevs ut korrekt.
I Sprint 2 testade jag programmet genom att lägga till flera personer, skriva in både korrekta och felaktiga inmatningar och kontrollera att programmet visade felmeddelanden när något blev fel. Jag testade också att avsluta programmet via menyn.

3. Resultat

Resultatet av Sprint 1 blev att jag kunde skapa ett person-objekt och skriva ut informationen korrekt till konsolen.

I Sprint 2 fungerade menyhanteringen som den skulle. Programmet tillät användaren att mata in flera personer, visade alla personer när användaren valde det alternativet och avslutades korrekt. Felaktig inmatning hanterades med felmeddelanden och användaren fick prova igen.

4. Diskussion och reflektion

4.1 Reflektion kring sprint 1

Under Sprint 1 lärde jag mig hur man kan strukturera kod med hjälp av enum, struct och class. Det var lite nytt för mig men jag fick en tydlig förståelse för hur man skapar olika datatyper och hur man använder dem tillsammans. Jag lärde mig också att använda ToString()-metoden för att skriva ut information om ett objekt. Det var bra träning i hur man bygger upp ett program steg för steg. Eftersom jag arbetade ensam under denna sprint, blev det extra viktigt för mig att ta ansvar för alla delar av uppgiften, från att skapa en enum och struct till att implementera klassen Person. Jag lärde mig att tänka självständigt kring problem och lösningar och att fördela min tid för att säkerställa att varje del fungerade som den skulle.

4.2 Reflektion kring sprint 2

Sprint 2 var mer utmanande eftersom jag skulle lägga till meny och inmatningsvalidering. Jag lärde mig hur man kan använda try-catch-satser och if-satser för att kontrollera att användaren skrev in korrekt information. Det tog lite tid att förstå hur jag skulle använda Enum.TryParse() och DateTime.TryParse(), men efter att ha testat flera gånger fick jag det att fungera. Jag fick också träna på hur man använder listor och loopar. I Sprint 2, när jag implementerade menyhantering och felhantering, stötte jag på flera utmaningar med att säkerställa att alla användarinmatningar hanterades korrekt. Eftersom jag arbetade ensam, tvingades jag att noggrant testa varje del av programmet för att säkerställa att det fungerade som avsett. Detta gav mig en djupare förståelse för hur viktigt det är med noggrann testning och hur man delar upp ett program i hanterbara delar

4.3 Reflektion kring alternativa lösningar

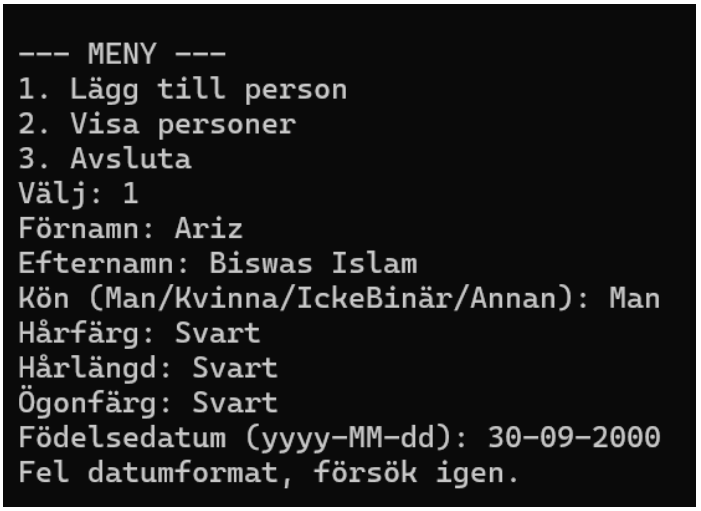
Om jag skulle göra om labben skulle jag kunna lägga till fler alternativ i menyn, till exempel att ta bort en person eller ändra informationen. Jag hade också kunnat skapa en separat klass för menyhanteringen för att göra koden ännu tydligare. Dessutom skulle jag kunna förbättra felhanteringen med mer detaljerade meddelanden till användaren. Om jag hade haft möjlighet att samarbeta med någon, hade jag kanske delat upp uppgiften mer och haft möjlighet att diskutera lösningar och struktur i förväg. Det hade sannolikt lett till en mer effektiv arbetsprocess. Men jag tycker att arbetet självständigt gav mig bra insikter i att tänka igenom alla delar av uppgiften och ta eget ansvar för att få allt att fungera.

5. Problem och lösningar

Under arbetet med labben stötte jag på flera problem:

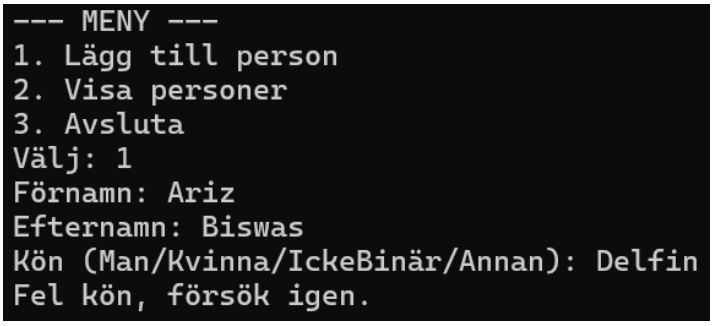
Problem 1 – Validering av datumformat
När jag först implementerade inmatningen av födelsedatum upptäckte jag att programmet kraschade om användaren skrev in datumet i fel format.

Lösning:
Jag använde DateTime.TryParse() för att kontrollera om inmatningen var korrekt innan den sparades. Om användaren skrev fel datum visades ett felmeddelande och användaren fick försöka igen.



Problem 2 – Ogiltigt kön
Användaren kunde skriva in ett kön som inte fanns i enum Gender, vilket orsakade problem.

Lösning:
Jag använde Enum.TryParse() tillsammans med en kontroll av om könet fanns i enum-listan. Vid fel visades ett meddelande.



Problem 3 – Otydlig programstruktur
När funktionaliteten växte blev Main-metoden lång och svår att läsa.

Lösning:
Jag flyttade logik till separata metoder (LäggTillPerson() och VisaPersoner()) vilket gjorde koden lättare att förstå och hantera.

6. Frågor till AI-verktyg

Under arbetet med labben använde jag AI-verktyget ChatGPT som stöd för att förstå specifika begrepp i C#. Jag använde det inte för att skriva koden, utan för att få hjälp med förklaringar och förståelse.

Verktyg: ChatGPT

Fråga: Vad är skillnaden mellan struct och class i C#?

På vilket sätt svaret användes: Hjälpte mig att förstå varför struct är lämpligt för Hair och class för Person.

Verktyg: ChatGPT

Fråga: Hur fungerar ToString() i C#?

På vilket sätt svaret användes: För att förstå hur jag kunde skriva ut information om personens data.

7. Referenser

• Land, R. (2024). Objektorienterad programmering [Föreläsning].

• Microsoft. (n.d.). .NET documentation.https://learn.microsoft .com/en-us/dotnet/