

Objektorienterad Programmering med C# del1

KURS • BED25GB_OOP1

[Innehåll](#) [Nyheter](#) [Kanaler](#) [Personal](#) [Om](#)Ej klar 

Inlämningsuppgift 1

Kunskapskontroll · Inlämningsuppgift · IG - VG · Senast klar 19 okt 23:59

Innehåll

Uppgift/Betyg







INLÄMNINGSUPPGIFT

Personal Budget Tracker (OOP, Collections & GitHub)



Syfte

Syftet med denna uppgift är att du ska visa att du:





-  Förstår grunderna i **objektorienterad programmering** (klasser, objekt, metoder).
-  Kan använda **List** och **kontrollflöde** (loopar, if, switch) för att hantera data.
-  Kan bygga ett **menybaserat program** i C#.
-  Förstår hur man arbetar professionellt med **Git & GitHub** (branches, pull requests och branch-skydd).



Scenario

Du ska bygga ett **Personal Budget Tracker** – ett konsolprogram som hjälper användaren att hålla koll på sina inkomster och utgifter.

Användaren ska kunna:

-  Lägga till inkomster och utgifter.
-  Visa en lista över alla transaktioner.
-  Se sin aktuella balans.
-  Ta bort poster.

Programmet ska byggas med **klasser, listor och metoder** och versionshanteras med **GitHub**.

Instruktioner


- 1 Skapa ett nytt **C# Console Project**.
- 2 Skapa två klasser:
 - **Transaction** – representerar en inkomst eller utgift.
 - **BudgetManager** – ansvarar för listan av transaktioner och logiken.
- 3 I Program.cs ska du skapa en **meny i en while-loop** med switch för användarens val.
- 4 Använd en **List<Transaction>** för att spara alla poster i minnet.
- 5 Koden ska vara **välstrukturerad, kommenterad och lätt att läsa**.
- 6 Projektet ska hanteras via **GitHub** enligt instruktionen nedan.

Kravspecifikation

Steg 0. Klass Diagram är ett MÅSTE !

Steg 0.1 Flödesdiagram - FlowChart - Flödesschema hur ni vill kalla den, ska också vara med !!!

Klass: Transaction

 Innehåller egenskaper för:

- Description (t.ex. "Lön", "Matinköp")
- Amount (decimal, positivt = inkomst, negativt = utgift)
- Category (t.ex. "Mat", "Transport", "Hyra", "Inkomst")
- Date (skrivs som text, t.ex. "2025-10-10")

 Metod:

- ShowInfo() – skriver ut all information om transaktionen.

Klass: BudgetManager






 Innehåller en lista av Transaction-objekt.

 Ska ha metoder för:

- AddTransaction() – lägger till en ny post.
- ShowAll() – visar alla transaktioner.
- CalculateBalance() – räknar ut total balans.
- DeleteTransaction() – tar bort en post.

Menykrav

Programmet ska ha minst följande menyval:

- 1  Lägg till transaktion
- 2  Visa alla transaktioner
- 3  Visa total balans
- 4  Ta bort transaktion
- 5  Avsluta programmet

(Bonus: skapa en extra meny för att visa transaktioner per kategori.)





GitHub & Versionshantering

Projektet ska hanteras i ett **GitHub Repository** med följande krav:

- ◆ Minst **3 grenar (branches)**, till exempel: main, development, feature-addtransaction.
- ◆ Minst **3 Pull Requests (PRs)**.
- ◆ **Branch Protection Rules** aktiverade för main (ingen direkt push till main).
- ◆ Tydliga **PR-beskrivningar** som kort beskriver ändringarna.
- ◆ Minst **5 commits** med bra meddelanden (ex: "Added BudgetManager class", "Created menu system").
- ◆ En **README.md** i repot med:
 - Kort projektbeskrivning.
 - Hur man kör programmet.
 - Ditt namn och datum och svar på reflektionsfrågor som finns längre ner här...

Bonusnivå (VG)

För extra poäng eller högre betyg kan du:

-  Lägga till färg i konsolen (grön för inkomster, röd för utgifter).
-  Filtrera eller sortera poster efter kategori eller datum.
-  Visa statistik: antal transaktioner, total inkomst, total utgift.
-  Validera inmatning (t.ex. kontrollera att belopp är numeriskt).

Reflektionsfrågor

Besvara kort i en separat textfil:

- 1** Hur hjälpte klasser och metoder dig att organisera programmet?
- 3** Vilken del av projektet var mest utmanande?





Inlämning

 Lämna in länken till ditt GitHub Repository i Learnpoint.

 Kontrollera att du har:

- Minst 3 branches.
- Minst 3 Pull Requests.
- Branch protection aktiv.
- Klassdiagram + flödesschema
- En [README.md](#) med projektinfo.

Tips

-  Skriv tydliga kommentarer i koden – låtsas att någon annan ska läsa ditt program.
-  Håll Main() kort och ren, lägg logiken i metoder.
-  Push ofta till GitHub – små steg visar bra arbetsflöde.
-  Testa programmet flera gånger innan du lämnar in.

