|  |
| --- |
| Övningsuppgift |

Bankkonton

Steg 2

|  |
| --- |
| *Författare:* Mats Loock  *Kurs:* Inledande programmering med C#  *Kurskod:*1DV402 |

**Upphovsrätt för detta verk**

Detta verk är framtaget i anslutning till kursen Inledande programmering med C# vid Linnéuniversitetet.

**Du får använda detta verk så här:**

Allt innehåll i verket Bankkonton av Mats Loock, förutom Linnéuniversitetets logotyp, symbol och kopparstick, är licensierad under:

|  |  |
| --- | --- |
|  | Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens. <http://creativecommons.org/licenses/by-nc-sa/2.5/se/> |

**Det betyder att du i icke-kommersiella syften får:**

* kopiera hela eller delar av innehållet
* sprida hela eller delar av innehållet
* visa hela eller delar av innehållet offentligt och digitalt
* konvertera innehållet till annat format
* du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp, symbol och/eller kopparstick i din nya version!

Vid all användning måste du ange källan: ”Linnéuniversitetet – Inledande programmering med C#” och en länk till <https://coursepress.lnu.se/kurs/inledande-programmering-med-csharp> och till Creative Common-licensen här ovan.

**Innehåll**

[Uppgift 3](#_Toc332705354)

[Problem 3](#_Toc332705355)

[Deposit 3](#_Toc332705356)

[Withdraw 3](#_Toc332705357)

[AddInterest 4](#_Toc332705358)

[DisplayAccount 4](#_Toc332705359)

[Testa programmet 4](#_Toc332705360)

[Mål 4](#_Toc332705361)

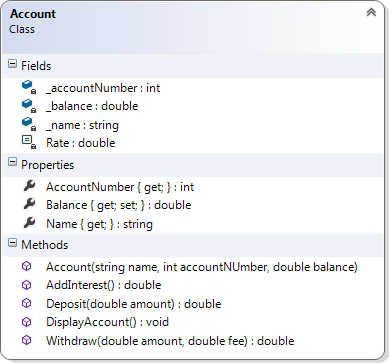
[Tips 4](#_Toc332705362)

[Lösning 5](#_Toc332705363)

# Uppgift

## Problem

Klassen Account representerar ett enkelt bankkonto. Klassen innehåller fält som representerar bankkontots ägare, nummer och saldo. Räntan lagras i en symbolisk konstant, Rate.



Konstanten Rate ger du värdet 3,5 % eller snarare 0.035.

Figur 1. Klassdiagram över klassen Account.

Klassen Account har en konstruktor som tar tre parametrar som används till att initiera fälten då ett nytt objekt instansieras av klassen. De övriga metoder i klassen används för att kunna göra olika saker som att sätta och ta ut pengar. Metoderna måste undersöka datat som skickas in till metoderna så att en transaktion kan utföras korrekt. Till exempel ska metoden Withdraw() hindra ett negativt uttag (som ju då skulle bli en insättning). Metoden AddInterest() finns också och används till att uppdatera saldot med räntan. Dessa publika metoder används för att modifiera saldot (\_balance). Egenskapen Balance, som både har en get- och set-metod, är kopplad till fältet \_balance men det är bara get-metoden som ska vara publik; set-metoden ska vara privat.

Till uppgiften finns ett projekt där klassen Account saknas. Du ska skapa och implementera klassen Account så att Main-metoden i klassen Program fungerar. **OBS! Du får inte ändra på någon kod i Main-metoden**. Ingen ny kod får läggas till i klassen Program. Läs koden i Main-metoden, titta på klassdiagrammet, titta på skärmdumpen lite längre fram i detta dokument och lista ut vad klassen Account ska innehålla.

(Kod ska alltid kommenteras, men i detta fall är det med avsikt som det inte finns några kommentarer i koden. Du ska kunna förstå det du behöver av koden ändå.)

### Deposit

För att sätta in pengar på kontot anropar du metoden Deposit(). Du måste kontrollera att beloppet som ska sättas in verkligen är större än 0. Är beloppet inte större än 0 kastar du ett undantag med meddelandet "The amount can not be less than 0.".

### Withdraw

För att ta ut pengar från bankkontot anropar du metoden Withdraw(). Du måste förvissa dig om att det finns pengar på kontot så att de räcker både till uttaget och till avgiften för uttaget innan uttaget genomförs. Saknas pengar kastar du ett undantag med meddelandet "Manage your account wisely so you do not overdraw." .

### AddInterest

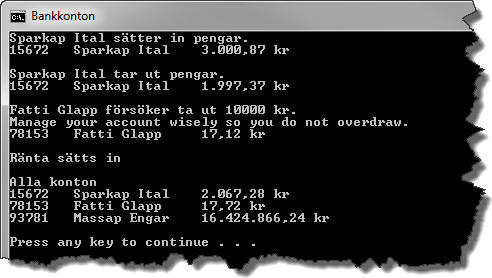
Du anropar AddInterest() för att lägga till ränta till bankkontot. Multiplicera saldot med räntan och addera resultatet av multiplikationen (produkten) till saldot.

### DisplayAccount

Då du vill presentera ett bankonto anropar du metoden DisplayAccount(). Bankkontonummer, innehavarens namn samt saldot ska presenteras. Använd tabbtecken (’\t’) för att separera de olika värden åt. För att presentera ett tal som en valuta använder du dig av formatspecificeraren c, t.ex. {0:c}.

## Testa programmet

Har du implementerat klassen Account korrekt ska du få följande resultat av en programkörning:



Figur 2.

## Mål

Efter att ha gjort uppgiften ska du:

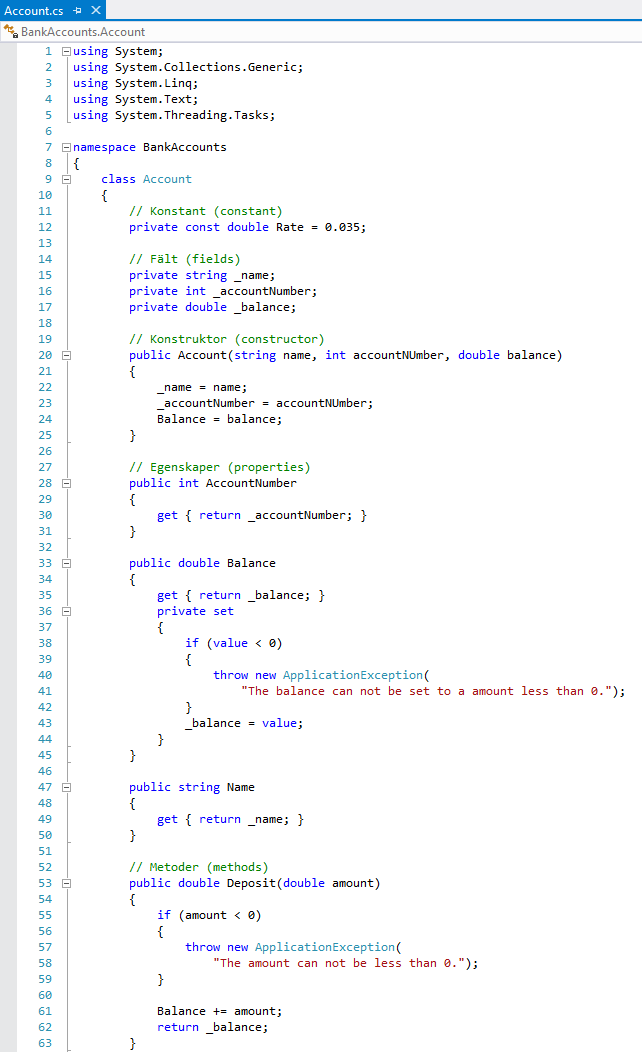
* Veta hur du skapar objekt och initierar ett objekts fält med hjälp av en konstruktor.
* Känna till att egenskapers get- och set-metoder kan vara en mix av public och private.
* Kunna använda en privat symbolisk konstant i en klass.

## Tips

Läs om:

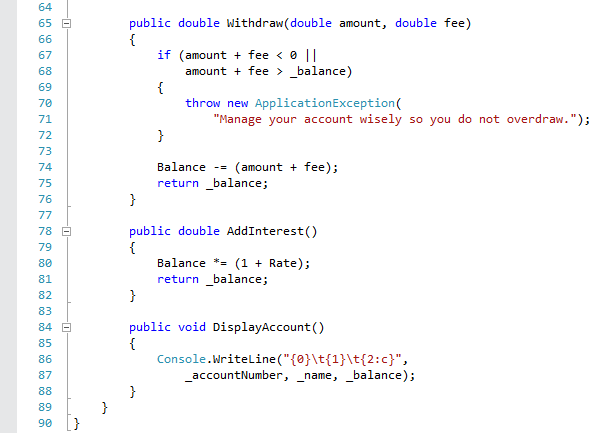
* Grunderna om klasser och objekt hittar du i inledningen av kapitel 5, som bl.a. tar upp saker som konstruktorer (*constructors*), accessmodifierare (*access modifiers*), fält (*fields*) och egenskaper (*properties*.
* Konstant som fält i kurslitteraturen, kapitel 5, under rubriken *”Encapsulating the Data”*.
* Grunderna i hur du kastar undantag hittar du i inledningen av kapitel 10.

# Lösning



*Lösningen forsätter på nästa sida!*

*Fortsättning på lösningen från föregående sida!*



Figur 3. Implementationen av klassen Account.

Fälten deklareras lämpligen som privata så dess värden inte kan manipuleras. Publika metoder får istället användas för att kontrollera så att t.ex. det inte går att ta ut mer pengar än vad som finns på kontot.

Den enda konstruktor initierar objektet med de värden som så önskas. Lägg märke till att konstruktorn använder sig av egenskapen Balance för att initiera det privata fältet \_balance. Varför? Därför att det inte ska vara möjligt att skapa ett Account-objekt med ett negativt tillgodohavande. Man vill ju inte börja med att vara skyldig banken pengar. Eller?

Kontonumret och namnet ska inte kunna ändras då ett Account-objekt väl blivit skapat, därför har egenskaperna AccountNumber och Name endast en get-metod. Data blir då vad man ibland kallar *read-only*, d.v.s. det går bara att läsa.

Egenskapen Balance har publik get-metod som returnerar tillgodohavandet. set-metoden är däremot privat och används av klassen ”internt” för att ändra värdet det privata fältet \_balance har.

Metoderna Deposit, Withdraw och AddInterest påverkar alla på något sätt det privata fältet \_balance. Diverse kontroller görs för att säkerställa att \_balance inte blir negativt.

Metoden DisplayAmount presenterar en textbeskrivning av ett Amount-objektets status. Lämpligare hade det varit att överskugga metoden ToString, men detta fungerar tillfredställande i denna uppgift.