Linnéuniversitetet
Kalmar Växjö

Laborationsanvisning

Recept på fil

Steg 3, laborationsuppgift 1

Författare: Mats Loock

Kurs: Inledande programmering med C#

*Kurskod:*1DV402

Linnéuniversitetet
Kalmar Växjö

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:

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.

Linnéuniversitetet
Kalmar Växjö

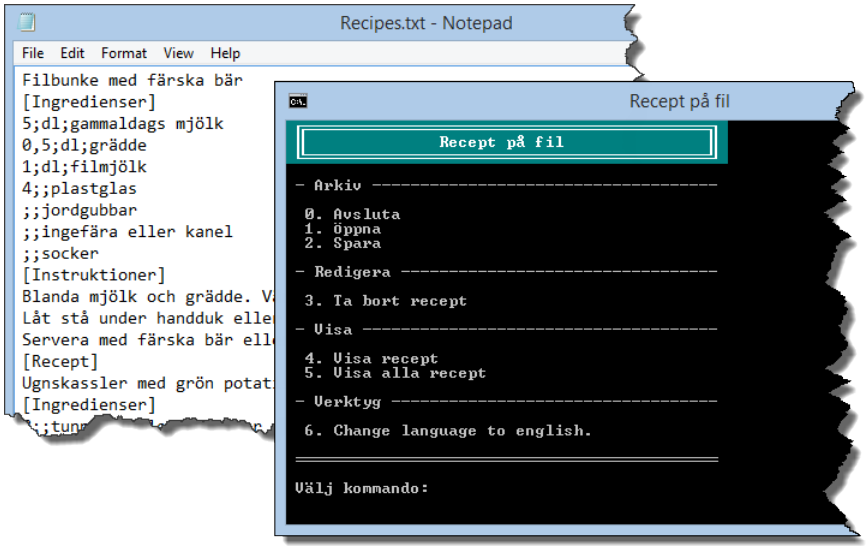
Innehåll

Uppgift 5
- Problem 5
- Funktionalitet som saknas 5
  - Hämta recept 5
  - Spara recept 5
  - Visa recept 5
  - Visa alla recept 6
- Format på textfil med recept 6
- Algoritm för att läsa in recept 7
- Arkitektur 8
  - Domänlagret 8
  - Presentationslagret 9
- Krav 9
- Läsvärt 9

Uppgift

Problem

Du ska komplettera en något större applikation för hantering av recept. Applikationen är i stort färdig men behöver kompletteras med fyra metoder för att det ska gå att visa recept, hämta recept från och skriva recept till en textfil.



Figur 1.

Funktionalitet som saknas

Hämta recept

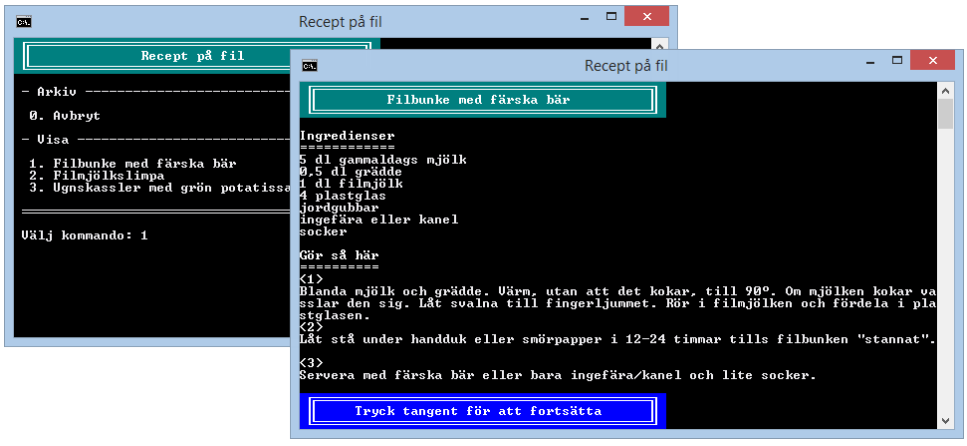
Recept ska läsas in från textfilen `recipes.txt`. Väljer användaren menyalternativet `'1.Öppna'` ska applikationen öppna textfilen, läsa och tolka den rad för rad för att skapa en lista med recept som användaren sedan ska kunna välja att via menykommandon hantera på olika sätt.

Spara recept

Recept ska sparas permanent i textfilen `recipes.txt`. Väljer användaren menyalternativet `'2. Spara'` ska applikationen öppna textfilen och skriva recepten rad för rad till textfilen. Finns redan textfilen ska den skrivas över.

Visa recept

Då användaren väljer menykommandot `'4. Visa recept.'` ska en lista med samtliga recepts namn presenteras varefter användaren väljer det recept som ska visas.

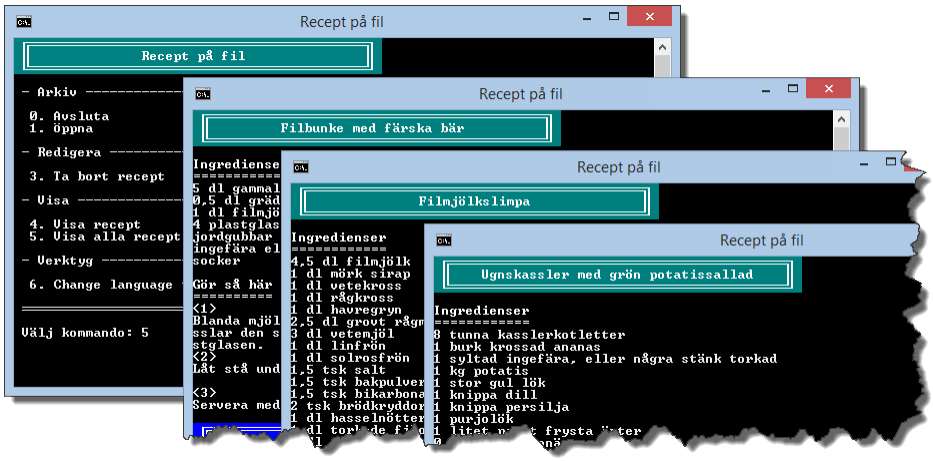


Figur 2. Användaren har valt att visa receptet med index 1.

Visa alla recept

Då användaren väljer menykommandot `’5. Visa alla recept.’` ska alla recept visas sorterade efter receptens namn.

Bara ett recept åt gången ska visas och användaren ska trycka på en tangent för att visa nästa recept. Efter att recepten visats ska användaren kunna trycka på en tangent för att återvända till menyn.

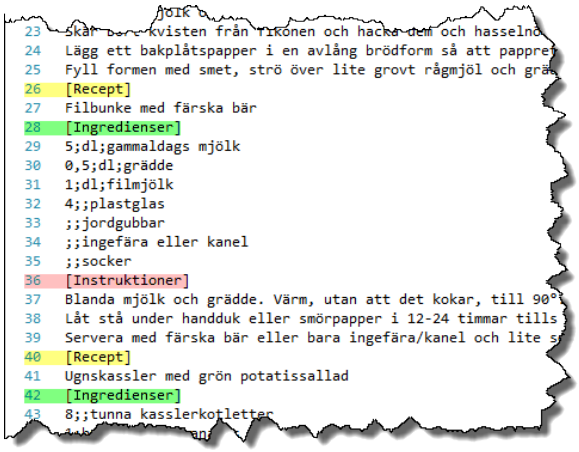


Figur 3. Användaren har valt att visa samtliga recept.

Format på textfil med recept

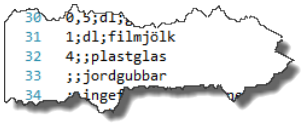
Recepten lagras permanent i en textfil. För att kunna skilja recepten åt, vad som är en ingrediens och vad som är en instruktion måste textfilen vara formaterad på ett förutbestämt sätt.

Varje recept består av avdelningarna `[Recept]`, `[Ingredienser]` och `[Instruktioner]`. Raden som följer efter `[Recept]` är receptets namn. Raderna som följer efter `[Ingredienser]`, fram till `[Instruktioner]` är receptets ingredienser där varje rad innehåller en ingrediens. Raderna som följer `[Instruktioner]`, fram till `[Recept]` eller slutet på filen, är receptets instruktioner där varje rad är en instruktion.



Figur 4. Textfil där varje recept består av tre avdelningar: recept, ingredienser och instruktioner.

En rad med en ingrediens består av tre delar separerade med semikolon (`;`). Första delen är mängden, andra delen är måttet och tredje delen är ingrediensens namn. Tredje delen får inte vara tom, d.v.s. ingrediensens namn måste alltid finnas. Övriga delar kan vara tomma men semikolon måste alltid finnas med.



Figur 5. Beskrivning av ingredienser behöver inte innehålla mängd och/eller mått. Ingrediensens namn är obligatoriskt.

Rad 31 i Figur 12 är ett exempel på en ingrediens där alla tre delarna används. Den består av delarna 1 (mängd), dl (mått) och filmjölk (namn).

Rad 32 i Figur 12 är ett exempel på en ingrediens där två av tre delar används. Den består av delarna 4 (mängd), mått saknas och plastglas (namn).

Rad 33 i Figur 12 är ett exempel på en ingrediens där en av tre delar används. Den består av delen jordgubbar (namn) medan delarna mängd och mått saknas.

Algoritm för att läsa in recept

Textfilen med recept ska läsas rad för rad. Varje rad ska tolkas för att bestämma vad raden beskriver. Förfarandet att läsa in och tolka textfilen till en samling med objekt representerande recepten i textfilen kan upplevas vara en stor utmaning varför en beskrivning av en algoritm kan underlätta.

Algoritmen beskriver innehållet i metoden `RecipeRepository.Load()` som skapar en lista med referenser till `Recipe`-objekt.

1. Skapa lista som kan innehålla referenser till receptobjekt.
2. Öppna textfilen för läsning.
3. Läs rad från textfilen tills det är slut på filen.
    a. Om det är en tom rad…
        i. …fortsätt med att läsa in nästa rad.
    b. Om det är en avdelning för nytt recept…
        i. …sätt status till att nästa rad som läses in kommer att vara receptets namn.
    c. …eller om det är avdelningen för ingredienser…
        i. …sätt status till att kommande rader som läses in kommer att vara receptets ingredienser.
    d. …eller om det är avdelningen för instruktioner…
        i. …sätt status till att kommande rader som läses in kommer att vara receptets instruktioner.
    e. …annars är det ett namn, en ingrediens eller en instruktion
        i. Om status är satt att raden ska tolkas som ett recepts namn…
            1. Skapa nytt receptobjekt med receptets namn.
        ii. …eller om status är satt att raden ska tolkas som en ingrediens…
            1. Dela upp raden i delar genom att använda metoden `Split()` i klassen `String`. De olika delarna separeras åt med semikolon varför det alltid ska bli tre delar.
            2. Om antalet delar inte är tre…
                a. …är något fel varför ett undantag av typen `FileFormatException` ska kastas.
            3. Skapa ett ingrediensobjekt och initiera det med de tre delarna för mängd, mått och namn.
            4. Lägg till ingrediensen till receptets lista med ingredienser.
        iii. …eller om status är satt att raden ska tolkas som en instruktion…
            1. Lägg till raden till receptets lista med instruktioner.
        iv. …annars…

1. …är något fel varför ett undantag av typen `FileFormatException` ska kastas.

4. Sortera listan med recept med avseende på receptens namn.
5. Tilldela avsett fält i klassen, `_recipes`, en referens till listan.
6. Tilldela avsedd egenskap i klassen, `IsModified`, ett värde som indikerar att listan med recept är oförändrad.
7. Utlös händelse om att recept har lästs in genom att anropa metoden `OnRecipesChanged` och skicka med parametern `EventArgs.Empty`.

Arkitektur

Applikationen är uppdelad i ett domän- och presentationslager. Respektive lager finns i olika projekt. Projektet `FiledRecipes.Domain` innehåller domänlagret och `FiledRecipes` innehåller presentationslagret.

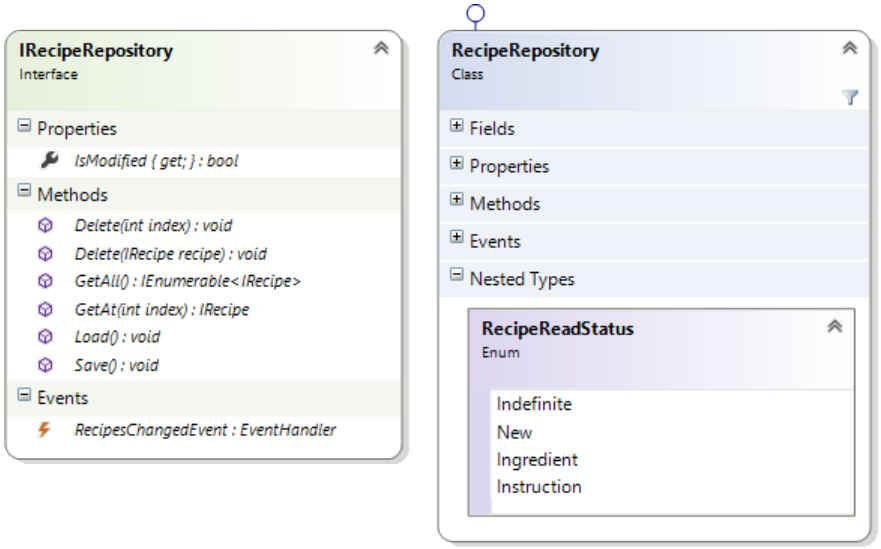
Domänlagret
Domänlagret innehåller ett flertal olika typer. Interfacen `IIngredient`, `IRecipe` och `IRecipeRepository` definierar funktionaliteten som de konkreta typerna `Ingredient`, `Recipe` och `RecipeRepository` ska ha. Dessutom finns den uppräkningsbara typen `RecipeReadStatus` nästlad i klassen `RecipeRepository`.

Klassen `Recipe` beskriver ett recept med ett namn, en lista med ingredienser och en lista med instruktioner. Strukturen `Ingredient` beskriver en ingrediens med mängd, mått och ingrediensens namn.

`RecipeRepository` ansvarar för allt som har med persistent lagring av recept, d.v.s. klassen har metoder för att läsa recept från en textfil och skriva recept till en textfil. Klassen använder i samband med inläsning av recept lämpligen den uppräkningsbara typen `RecipeReadStatus` för att hålla ordningen på vilken typ av data som lästs in från textfilen.

Klassen RecipeRepository och den uppräkningsbara typen RecipeReadStatus
En instans av klassen `RecipeRepository` används för att hantera persistent lagrade recept. Den uppräkningsbara typen `RecipeReadStatus` används för att hålla ordning på innebörden av en rad med data som lästs från en textfil.

Klassen `RecipeRepository` är inte komplett utan behöver kompletteras. Vilka metoder som saknas och dess signaturer definieras av det interface klassen implementerar.



Figur 6.

Under rubriken ’Format på textfil med recept’ finns information om textfilen format. Under rubriken ’Algoritm för att läsa in recept’ finns en algoritm som kan användas för att läsa in och tolka textfilen.

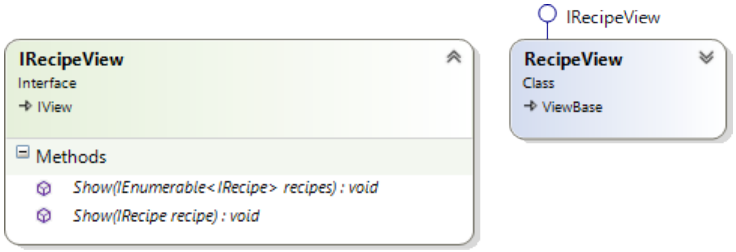
Då textfilen tolkas används lämpligen en lokal variabel av typen `RecipeReadStatus` så metoden vet hur den aktuella raden som lästs in, och kommande rader, ska tolkas. Uppstår fel under inläsningen eller tolkningen, t.ex. på grund av att textfilen inte är korrekt formaterad, ska metoden kasta ett undantag av typen `FileFormatException`.

Recepten ska spara enligt det format som beskrivs under rubriken ’Format på textfil med recept’.

Presentationslagret
Presentationslagret innehåller ett stort antal typer av vilka typerna `IRecipeView` och `RecipeView` är intressantast. Interfacet `IRecipeView` definierar den funktionalitet den konkreta klassen `RecipeView` ska ha. Basklassen `ViewBase` innehåller intressanta medlemmar som `Header`, `ShowHeaderPanel` och `ContinueOnKeyPressed`, som kommer till användning vid implementation av metoderna `IRecipeView` deklarerar.

Klassen RecipeView
En instans av klassen `RecipeView` används för att skriva ut recept i ett konsolfönster.

Klassen `RecipeView` är inte komplett utan behöver kompletteras. Vilka metoder som saknas och dess signaturer definieras av det interface klassen implementerar. Se Figur 2 och Figur 3 för information om hur recept ska presenteras.



Figur 7.

Krav

Befintlig källkod i det påbörjade projektet får under några omständigheter inte ändras.

Endast de metoder som interfacen `IRecipeRepository` och `IRecipeView` definierar, och som saknas, får läggas till.

Samtliga krav som ställs under rubrikerna ovan ska vara uppfyllda.

Läsvärt

- Sortera med OrderBy()
  - Essential C# 5.0, 590-592.
  - http://msdn.microsoft.com/en-us/library/6sh2ey19.aspx
- Klassen List<T>
  - Essential C# 5.0, 638-644.
  - http://msdn.microsoft.com/en-us/library/6sh2ey19.aspx
- Läsa och skriva till textfiler
  - StreamReader, http://msdn.microsoft.com/en-us/library/6aetdk20.aspx.
  - StreamWriter, http://msdn.microsoft.com/en-us/library/3ssew6tk.aspx