



Linnéuniversitetet

Kalmar Vaxjö

Laborationsanvisning

Receptsamling med fil

Steg 3, laborationsuppgift 1



Författare: Mats Looch

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 Receptsamling med fil av Mats Looch, 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	5
Problem	5
Funktionalitet	5
Hämta recept	5
Spara recept	6
Ta bort recept	6
Visa recept	7
Visa alla recept	7
Format på textfil med recept	8
Algoritm för att läsa in recept	9
Klassdiagram	10
Strukturen Ingredient	10
Klassen Recipe	11
Klassen RecipeRepository och den uppräkningsbara typen RecipeReadStatus	13
Klassen RecipeView	13
Klassen Program	14
Krav	16
Läsvärt	16
Extrauppgifter	17
Lägga till recept	17
Receptets namn	17
Receptets ingredienser	17
Receptets instruktioner	18
Hitta recept med en eller flera ingredienser	18
Välja filnamn	19
Redigera recept	19

Uppgift

Problem

Du ska skriva en något större applikation för hantering av recept. Applikationen ska i grundutförandet via en enkelt utformad meny erbjuda användaren att hämta recept från en textfil, skriva recept till en textfil, ta bort recept och visa recept.



Figur 1.

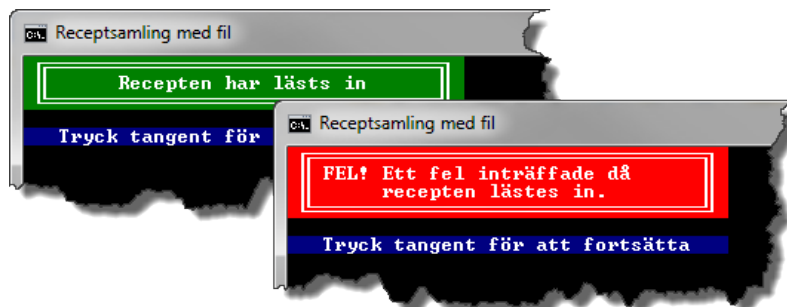
Applikationen ska delas upp i flera typer med klara ansvarsområden, t.ex. ska all hantering av recept i filer placeras i en klass medan presentation av recept placeras i en annan. Ett allmänt krav på applikationen är att den ska ge fel- och rättmeddelanden då användaren utför olika kommandon.

Funktionalitet

Hämta recept

Recept ska läsas in från textfilen `recipes.txt`. Väljer användaren menyalternativet '1. Öppna textfil med recept.' 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.

Ett rättmeddelande ska visas då applikationen lyckats läsa in recepten. Inträffar ett fel ska ett felmeddelanden visas. Oavsett om det gick att läsa in recepten från textfilen eller inte ska användaren återvända till menyn efter att ett rätt- eller felmeddelande visats.

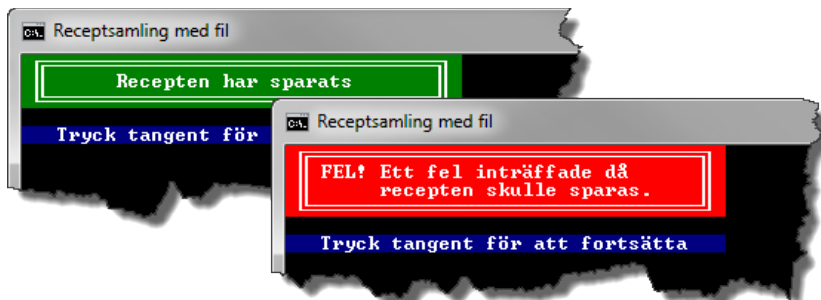


Figur 2. Rätt- och felmeddelanden efter att recept lästs in från textfilen `recipes.txt`.

Spara recept

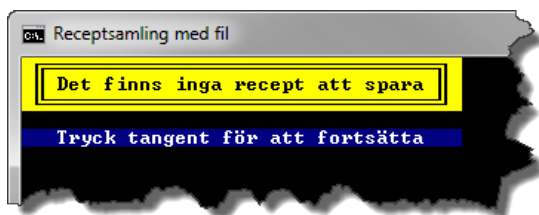
Recept ska sparas permanent i textfilen `recipes.txt`. Väljer användaren menyalternativet '2. Spara recept på textfil.' ska applikationen öppna textfilen och skriva recepten rad för rad till textfilen. Finns redan textfilen ska den skrivas över.

Ett rättmeddelande ska visas då applikationen lyckats spara recepten på textfilen. Inträffar ett fel ska ett felmeddelanden visas. Oavsett om det gick att spara recepten eller inte ska användaren återvända till menyn efter att ett rätt- eller felmeddelande visats.



Figur 3.

För att spara recept måste det finnas recept att spara. Saknas recept ska ett meddelande visas som informerar användaren att det inte finns några recept att spara.

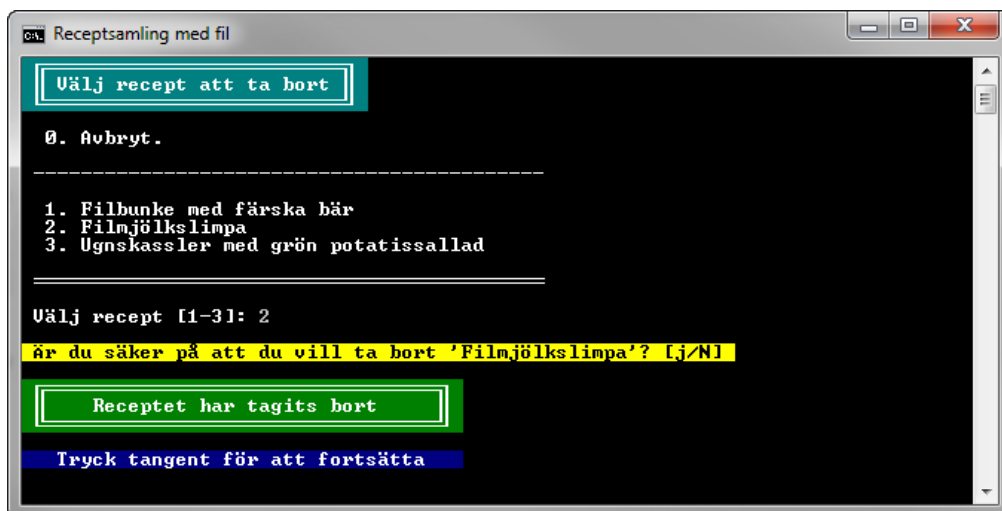


Figur 4.

Ta bort recept

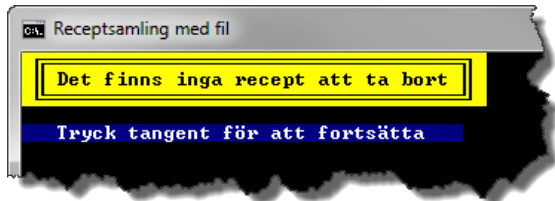
Menykommandot '3. Ta bort recept.' ska presentera en lista med samtliga recepts namn där användaren kan välja ett recept som ska tas bort. Innan ett recept tas bort måste användaren bekräfta att det är rätt recept som kommer att tas bort. Användaren ska kunna välja att avbryta kommandot och återvända till menyn utan att ta bort ett recept.

Ett rättmeddelande ska visas då applikationen lyckats ta bort ett recept. Inträffar ett fel ska ett felmeddelanden visas. Oavsett om det gick att ta bort ett recept eller inte ska användaren kunna välja ett nytt recept att ta bort. Användaren ska kunna välja att avbryta borttagning av recept och då återvända till menyn.



Figur 5. Exempel där ett recept tas bort.

För att ta bort recept måste det finnas recept att ta bort. Saknas recept ska ett meddelande visas som informerar användaren att det inte finns några recept att ta bort.

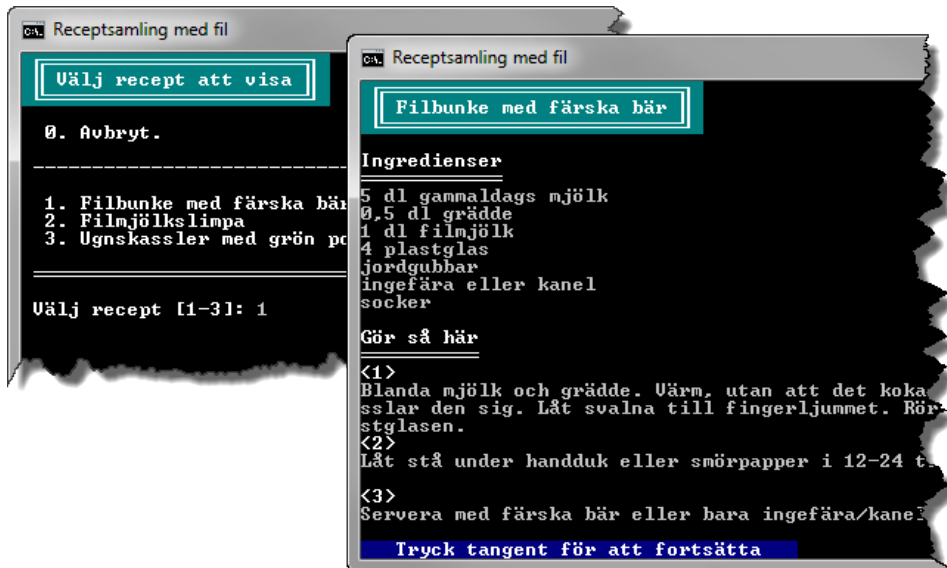


Figur 6.

Visa recept

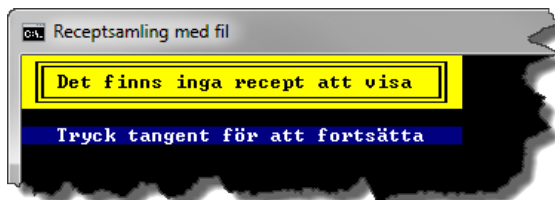
Då användaren väljer menykommandot '4. Visa recept.' ska en lista med samtliga recepts namn presenteras varefter användaren ska kunna välja det recept som ska visas.

Efter att ett recept visats ska användaren kunna välja ett nytt recept att visa. Användaren ska kunna välja att avbryta visningen av recept och då återvända till menyn.



Figur 7. Användaren har valt att visa receptet med index 1.

För att visa recept måste det finnas recept att visa. Saknas recept ska ett meddelande visas som informerar användaren att det inte finns några recept att visa.

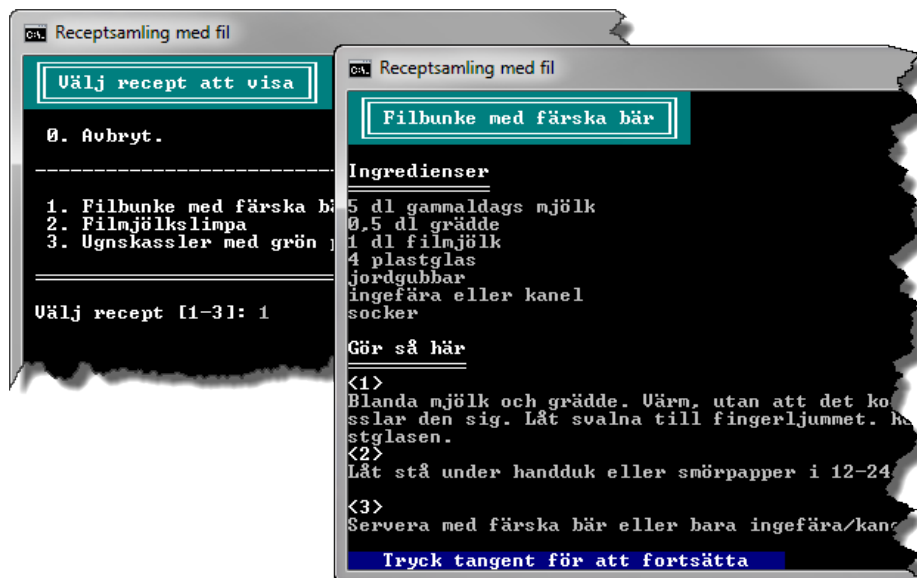


Figur 8.

Visa alla recept

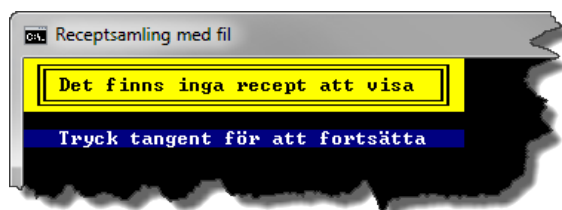
Då användaren väljer menykommandot '5. Visa alla recept.' ska alla recept visas sorterade efter receptens namn.

Efter att recepten visats ska användaren kunna trycka på en tangent för att återvända till menyn.



Figur 9. Användaren har valt att visa samtliga recept.

För att visa recept måste det finnas recept att visa. Saknas recept ska ett meddelande visas som informerar användaren att det inte finns några recept att visa.

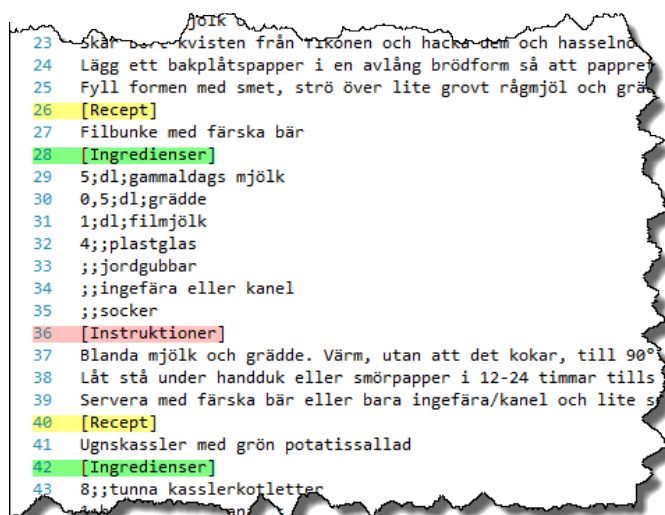


Figur 10.

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 11. 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.



```
30 0;;dl;
31 1;dl;filmjolk
32 4;;plastglas
33 ;;jordgubbar
34 ;;inget
```

Figur 12. 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 filmjolk (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 av olika typer representerande recepten i textfilen kan upplevas vara en stor utmaning varför en beskrivning av en algoritm kan underlätta.

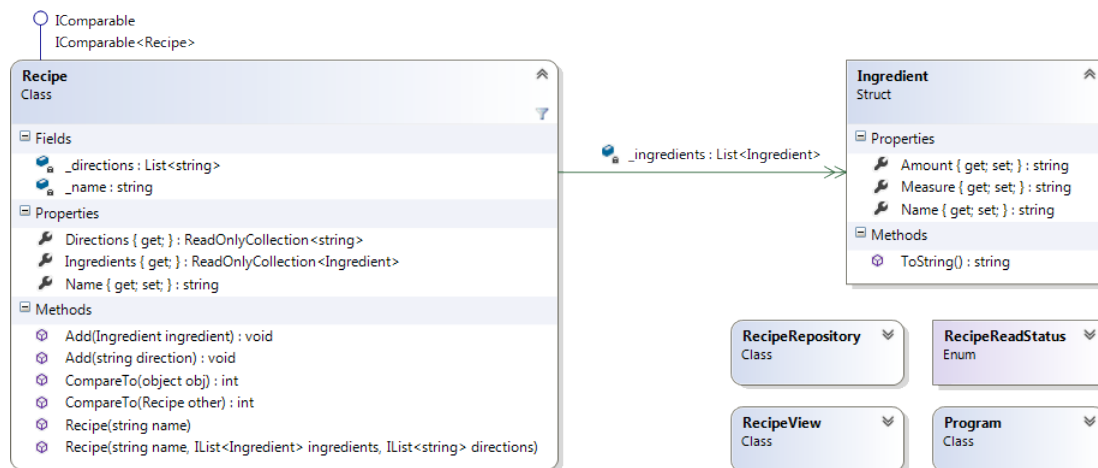
Algoritmen beskriver innehållet i metoden `RecipeRepository.Load()` som returnerar en lista med referenser till `Recipe`-objekt.

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 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 ska kastas.
4. Sortera listan med recept med avseende på receptens namn.
5. Returnera en referens till listan.

Klassdiagram

Applikationen ska delas upp i flera typer. Typerna *Recipe*, *Ingredient*, *RecipeRepository*, *RecipeReadStatus*, *RecipeView* och *Program* ansvara var och en för sin del av applikationen.



Figur 13. Övergripande klassdiagram med focus på typerna *Recipe* och *Ingredient*.

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.

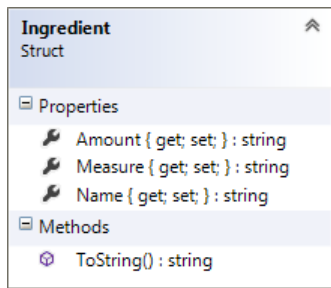
RecipeRepository ansvarar för allt som har med persistent lagring av recept, d.v.s. klassen har metoder för att läsa recept från en textfil och skriva recept till en textfil. Klassen använder i samband med inläsning av recept lämpligen den uppräkningsbara typen *RecipeReadStatus* för att hålla ordningen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

Då recept ska visas ska en instans av klassen *RecipeView* användas, som till skillnad mot klassen *Recipe* vet hur ett, eller flera, recept skrivs ut i ett konsolfönster.

Klassen *Program* har huvudansvaret för exekveringen av applikationen och erbjuder användaren med hjälp av en meny ett antal kommandon som kan användas för att hantera recept.

Strukturen *Ingredient*

För att lagra information om en ingrediens ska en struktur användas. Strukturen ska vara enkelt utformad och bara ha autoimplementerade egenskaper, vilket innebär att varken mängd, mått eller namn på något sätt ska valideras. Den ska dock överskugga metoden `ToString()` så att en textbeskrivning av en ingrediens kan fås på ett enkelt sätt.



Figur 14. Strukturen Ingredient.

Egenskapen Amount

Publik autoimplementerad egenskap av typen String representerande mängden det ska vara av en ingrediens.

Egenskapen Measure

Publik autoimplementerad egenskap av typen String representerande vilket mått ingrediensen ska använda.

Egenskapen Name

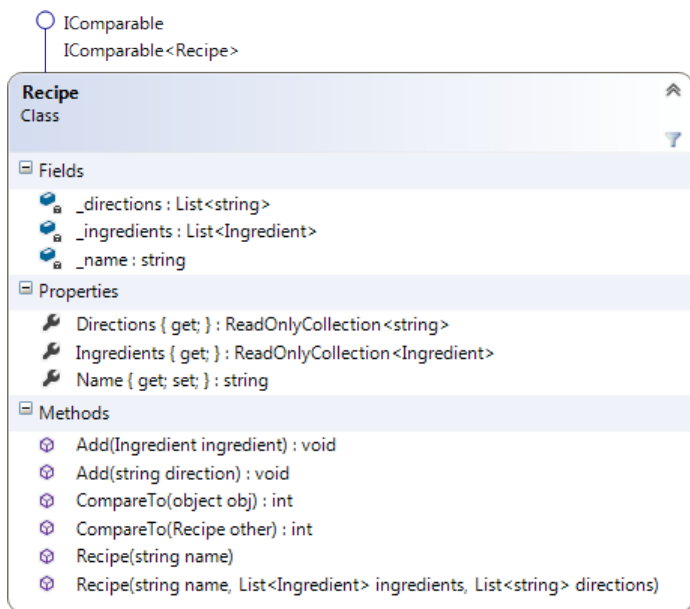
Publik autoimplementerad egenskap av typen String representerande ingrediensen namn.

Metoden ToString

Metoden ska returnera en sträng som beskriver en ingrediens. Strängen ska vara väl formaterad och inte innehålla onödiga mellanslag.

Klassen Recipe

Instanser av klassen används för att representera recept. Klassen implementerar IComparable och IComparable<T> för att det ska vara möjligt att sortera instanser med avseende på receptens namn.



Figur 15.

Fältet _directions

Privat fält av typen List<string> som används för att representera ett recepts instruktioner.

Fältet _ingredients

Privat fält av typen List<Ingredient> som används för att representera ett recepts ingredienser.

Fältet _name

Privat fält av typen String som används för att representera ett recepts namn.

Egenskapen Directions

Publik egenskap av typen `ReadOnlyCollection<string>` som ger en "read-only"-referens till fältet `_directions`. I och med att referensen är av typen `ReadOnlyCollection<string>` och klassen `String` är "immutable" sker ingen "privacy leak".

Egenskapen Ingredients

Publik egenskap av typen `ReadOnlyCollection<Ingredient>` som ger en "read-only"-referens till fältet `_ingredients`. I och med att referensen är av typen `ReadOnlyCollection<Ingredient>` och strukturen `Ingredient` är en värdetyp sker ingen "privacy leak".

Egenskapen Name

Publik egenskap av typen `string` som ger eller sätter namnet på receptet. `set`-metoden ska validera så att namnet inte refererar till `null` eller är en tom sträng.

Konstruktörerna

De två konstruktörerna ska se till att ett `Recipe`-objekt blir korrekt initierat. Det innebär att fälten ska initieras med lämpliga värden.

Konstruktorn `Recipe(string name)` ska initiera fälten så att de refererar till objekt.

Med konstruktorn `Recipe(string name, List<Ingredient> ingredients, List<string> directions)` ska ett objekt kunna initieras med ett recepts namn, ingredienser och instruktioner. För att undvika "privacy leak" får inte fälten tilldelas referenserna till `List`-objekten rakt av utan det måste skapas kopior av argumenten som skickades med då konstruktorn anropades.

Metoderna Add

`Add()` ska överlagras, d.v.s. det ska finnas två metoder med samma namn men med olika parameterlistor.

Metoden `Add(Ingredient ingredient)` används för att lägga till en ny ingrediens till ett recept. Det är ingen risk för "privacy leak" då typen `Ingredient` är en struktur och alltså är en värdetyp.

Metoden `Add(string direction)` används för att lägga till en ny instruktion till ett recept. Det är ingen risk för "privacy leak" då typen `string` är "immutable" och fungerar som en värdetyp trots att det är en referenstyp.

Metoderna CompareTo

`CompareTo()` ska överlagras, d.v.s. det ska finnas två metoder med samma namn men med olika parameterlistor.

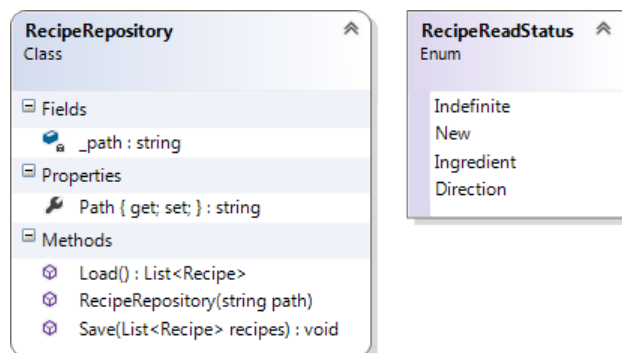
Metoderna anropas i regel inte direkt av kod utvecklare skriver utan det sker automatiskt av ramverket. Metoden `CompareTo(object obj)` används t.ex. av metoden `Array.Sort()` då instanser av typen `Recipe` ska sorteras. Metoden `CompareTo(Recipe other)` används av metoden `List.Sort()` då instanser av typen `Recipe` ska sorteras.

Metoderna ska jämföra två objekt med avseende på fältet för receptets namn.

- Refererar parametern till `null` ska ett heltal större än 0 returneras.
- Refererar parametern till ett objekt som inte är av typen `Recipe` ska ett undantag av typen `ArgumentException` kastas.
- Refererar parametern till ett objekt vars namn ska sorteras efter det anropande objektets namn ska ett heltal mindre än 0 returneras.
- Refererar parametern till ett objekt vars namn ska sorteras före det anropande objektets namn ska ett heltal större än 0 returneras.
- Refererar parametern till ett objekt vars namn är samma som det anropande objektets namn ska heltalet 0 returneras.

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.



Figur 16.

Fältet `_path`

Privat fält av typen `string` innehållande sökvägen till den fil en instans av `RecipeRepository` arbetar mot.

Egenskapen `Path`

Publik egenskap av typen `string` som kapslar in fältet `_path`. `set`-metoden ska validera sökvägen så att den inte refererar till `null`, är tom eller bara innehåller vita tecken ("*white spaces*").

Konstruktorn

Konstruktorn ska initiera fältet `_path`, via egenskapen `Path`, så att det instansierade objektet innehåller en sökväg.

Metoden `Load`

Den publika metoden `Load()` ska läsa in textfilen och tolka den för att skapa en lista med referenser till `Recipe`-objekt som returneras.

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.

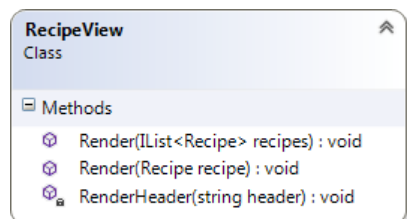
Då textfilen tolkas används lämpligen en instans av typen `RecipeReadStatus` så metoden vet hur den aktuella raden som lästs in ska tolkas. Uppstår fel under inläsningen eller tolkningen, t.ex. på grund av att textfilen inte är korrekt formaterad, ska metoden kasta ett undantag.

Metoden `Save`

Den publika metoden `Save()` ska spara de recept som skickas med som argument vid anrop av metoden på en textfil. Recepten ska spara enligt det format som beskrivs under rubriken 'Format på textfil med recept'.

Klassen `RecipeView`

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



Figur 17.

Metoderna `Render`

`Render()` ska överlagras, d.v.s. det ska finnas två metoder med samma namn men med olika parameterlistor.

Metoden `Render(List<Recipe> recipes)` ska skriva ut samtliga recept i samlingen som skickades med som argument vid anropet av metoden.

Metoden `Render(Recipe recipe)` ska skriva ut receptet som skickades med som argument vid anropet av metoden.

Metoden *RenderHeader*

Den privata metoden `RenderHeader(string header)` används för att skriva ut ett recepts rubrik innehållande receptets namn. Metoden anropas av metoderna `Render()`.

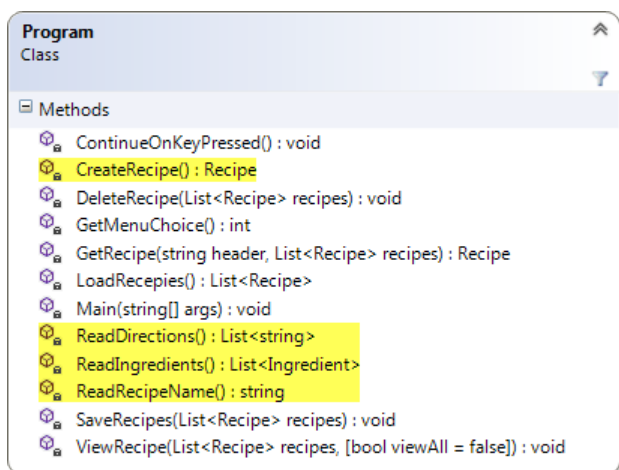


Figur 18. Metoderna `Render()` anropar `RenderHeader()` för att skriva ut receptets rubrik.

Klassen Program

Klassen `Program` ska användas för att skriva den menystyrda delen av applikationen där användaren väljer olika menykommandon för att hantera recept på olika sätt.

Samtliga metoder i klassen `Program` måste vara statiska. Klassen får inte ha några fält varför ett `List`-objekt med referenser måste instansieras i metoden `Main()` och skickas som argument till övriga statiska metoder i klassen.



Figur 19. Samtliga metoder i klassen `Program` måste vara statiska. Gulmarkerade medlemmar behövs bara om du gör extrauppgiften "Lägg till recept".

Metoden *Main*

Metoden `Main` ska anropa metoden `GetMenuChoice()` för att visa en meny. Så länge som användaren inte väljer att avsluta applikationen, genom att välja menyalternativet 0, ska menyn visas på nytt efter att något av övriga menykommandon utförts.

Beroende på vilket menykommando användaren väljer ska metoderna `LoadRecipes()`, `SaveRecipes()`, `CreateRecipe()`, `DeleteRecipe()` eller `ViewRecipe()` anropas.

Metoden *ContinueOnKeyPressed*

I flera situationer kan det vara lämpligt att användaren uppmanas att trycka på en tangent innan konsolfönstrets innehåll rensas och ersätts. Istället för att upprepa koden som uppmanar användaren att trycka på en tangent för att fortsätta placeras lämpligen den koden i metoden `ContinueOnKeyPressed()`, som enkelt kan anropas vid behov.

Metoden *CreateRecipe* (OBS! Krävs endast om du gör extrauppgiften "Lägg till recept")

Då användaren vill skapa ett nytt recept ska metoden `CreateRecipe()` användas. Metoden anropar metoderna `ReadRecipeName()`, `ReadIngredients()` och `ReadDirections()` för att läsa in det användaren matar in för att skapa ett nytt recept i form av ett `Recipe`-objekt.

Väljer användaren att avbryta skapandet av ett nytt recept ska metoden returnera null.

Metoden DeleteRecipe

Metoden DeleteRecipe() ska ta bort ett recept ur listan med recept som skickas som argument till metoden.

Metoden GetMenuChoice

Metoden GetMenuChoice() ska presentera en meny, läsa in menyalternativet användaren väljer och returnera det heltal som symboliserar menyvalet. Metoden ska validera det användaren matar in så att endast heltal knutna till menykommandon godtas. Matar användaren in något som inte kan tolkas som ett heltal knutet till ett menykommando ska ett felmeddelande visas.

Metoden GetRecipe

Metoden presenterar en indexerad lista med samtliga recepts namn. Användaren ska bara kunna välja bland de index som är knutna till recept. Metoden ska returnera en referens till det recept som blivit valt. Metoderna DeleteRecipe() och ViewRecipe() anropar denna metod för att få reda på vilket recept som ska tas bort respektive visas.

Värdet 0 ska användaren kunna välja för att avbryta förfarandet att välja ett recept. I så fall ska metoden returnera värdet null.

Se Figur 5 på sidan 6, eller Figur 7 på sidan 7, för exempel på hur en indexerad lista med recepts namn kan utformas.

Metoden LoadRecipes

Metoden LoadRecipes() läser in recepten från en textfil genom att använda en instans av klassen RecipeRepository.

Då recepten lästs in utan problem ska ett rättmeddelande visas och en referens till ett List-objekt innehållande referenser till Recipe-objekt ska returneras.

Inträffar ett fel i samband med att recepten läses in ska ett felmeddelande visas och metoden returnera värdet null.

Metoden ReadDirections (OBS! Krävs endast om du gör extrauppgiften "Lägg till recept")

Metoden ska på lämpligt sätt läsa in ett recepts instruktioner i form av strängar, en sträng för varje instruktion. Strängarna användaren matar in lagras i ett List-objekt som metoden returnerar en referens till. Se Figur 25 på sidan 18 för exempel på hur inläsningen av instruktioner kan utformas.

Väljer användaren att avsluta inmatningen av instruktioner ska användaren varnas och få en möjlighet att bekräfta att inga fler instruktioner ska matas in.

Matar inte användaren in några instruktioner ska ett felmeddelande visas varefter användaren ska kunna välja att mata in instruktioner eller att avbryta inmatningen. Avbryts inmatningen utan att några instruktioner matats in ska värdet null returneras.

Metoden ReadIngredients (OBS! Krävs endast om du gör extrauppgiften "Lägg till recept")

Metoden ska på lämpligt sätt läsa in ett recepts ingredienser. Ingredienserna användaren matar in lagras i ett List-objekt som metoden returnerar en referens till.

En ingrediens består av ingrediensens namn, mängd och mått. Namnet är obligatoriskt medan mängd och mått är frivilliga uppgifter. Se Figur 22 på sidan 17 för exempel på hur inläsningen av en ingrediens kan utformas.

Väljer användaren att avsluta inmatningen av ingredienser ska användaren varnas och få en möjlighet att bekräfta att inga fler ingredienser ska matas in.

Matar inte användaren in några ingredienser ska ett felmeddelande visas varefter användaren ska kunna välja att mata in ingredienser eller att avbryta inmatningen. Avbryts inmatningen utan att några ingredienser matats in ska värdet null returneras.

Metoden ReadRecipeName (OBS! Krävs endast om du gör extrauppgiften "Lägg till recept")

Metoden anropas för att läsa in ett recepts namn. Väljer användaren att inte ange något namn ska ett felmeddelande visas och efter användaren ska kunna välja att avbryta inläsningen. Avbryter användaren inläsningen ska metoden returnera värdet null.

Metoden SaveRecipes

Metoden SaveRecipes() sparar recepten på en textfil genom att använda en instans av klassen RecipeRepository.

Finns det inga recept att spara ska användaren informeras med ett meddelande. Finns det recept ska alla recept sparas och ett rättmeddelande visas om allt gått bra. Inträffar ett fel i samband med att recepten sparas ska ett felmeddelande visas.

Metoden ViewRecipe

Metoden ViewRecipe() ska kunna visa ett enskilt recept eller samtliga recept. Metoden har två parametrar. Den första parametern recipes är en referens till listan med referenser till recept. Den andra parametern viewAll, med standardvärdet false, bestämmer om ett eller flera recept ska visas.

Om ett recept ska visas ska metoden GetRecipe() anropas för att erhålla en referens till receptet. Oavsett om ett eller flera recept ska visas ska en instans av typen RecipeView sköta presentationen.

Krav

Samtliga krav som ställs under rubrikerna ovan ska vara uppfyllda.

Läsvärt

- Strukturer
 - Essential C# 5.0, 340-347.
 - <http://msdn.microsoft.com/en-us/library/ah19swz4.aspx>
- 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>
- Klassen ReadOnlyCollection<T>
 - <http://msdn.microsoft.com/en-us/library/ms132474.aspx>
- Läs och skriv till textfiler
 - StreamReader, <http://msdn.microsoft.com/en-us/library/6aetdk20.aspx>.
 - StreamWriter, <http://msdn.microsoft.com/en-us/library/3srew6tk.aspx>



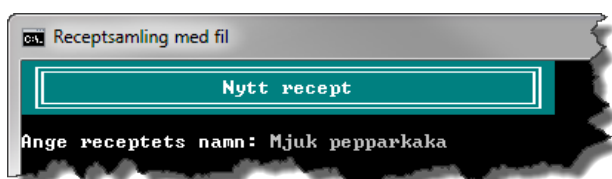
Extrauppgifter

Lägga till recept

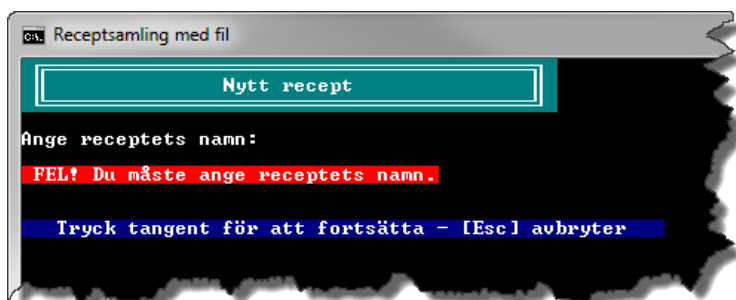
Användaren ska kunna lägga till ett nytt recept genom att välja menyalternativet '6. Lägg till nytt recept.'. Ett nytt recept skapas genom att användaren anger receptets namn, minst en ingrediens och minst en instruktion. Det nya receptet läggs där efter till samlingen med recept.

Receptets namn

Ett recept måste ha ett namn. Anger inte användaren ett namn ska ett felmeddelande visas. Användaren ska då kunna välja att göra ett nytt försök att ange ett namn eller att avbryta skapandet av receptet.



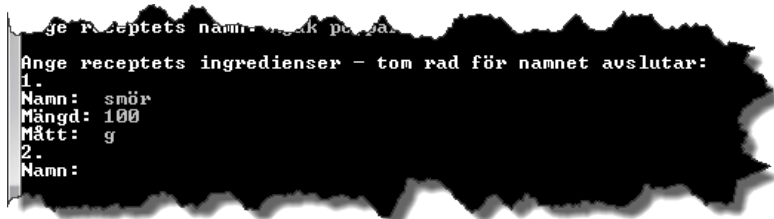
Figur 20. Användaren anger ett receptets namn.



Figur 21. Användaren har inte angivit ett namn, ett felmeddelande visas och användaren kan då välja att avbryta.

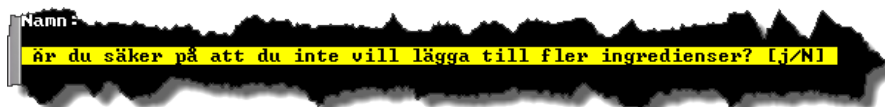
Receptets ingredienser

Efter att användaren angett ett namn ska ingredienserna anges. Ett recept måste minst ha en ingrediens. En ingrediens ska bestå av tre delar, ingrediensens namn, mängd och mått, där ingrediensens namn måste anges medan mängd och mått är frivilliga.



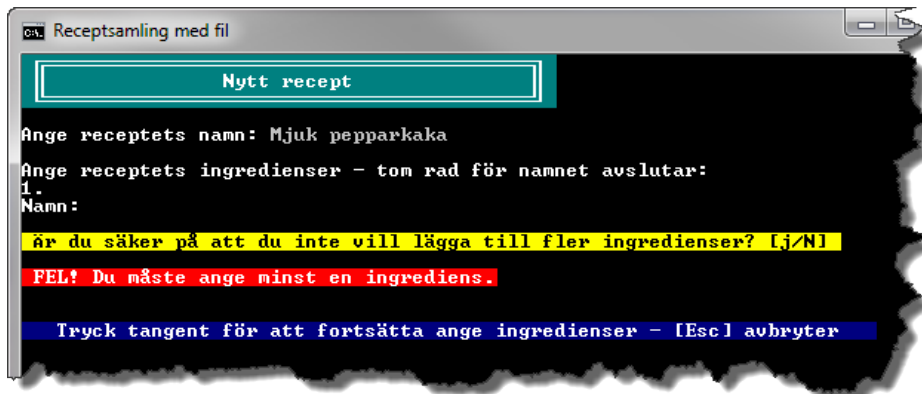
Figur 22. Ingrediensens namn är obligatorisk, mängd och mått behöver inte anges.

Användaren ska kunna avsluta inmatningen av ingredienser på lämpligt sätt, t.ex. genom att ange en tom rad vid ingrediensens namn. Användaren måste bekräfta att inga fler ingredienser ska läggas till.



Figur 23. Användaren måste bekräfta att inga fler ingredienser ska läggas till.

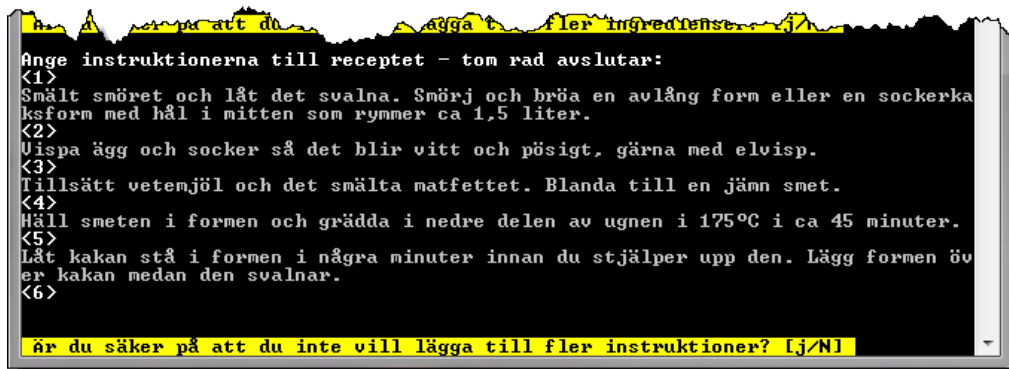
Väljer användaren att inte lägga till några ingredienser alls ska ett felmeddelande visas varefter användaren ska kunna välja att avbryta skapandet av ett nytt recept och återvända till menyn.



Figur 24. Inga ingredienser har lagts till varför ett felmeddelande visas.

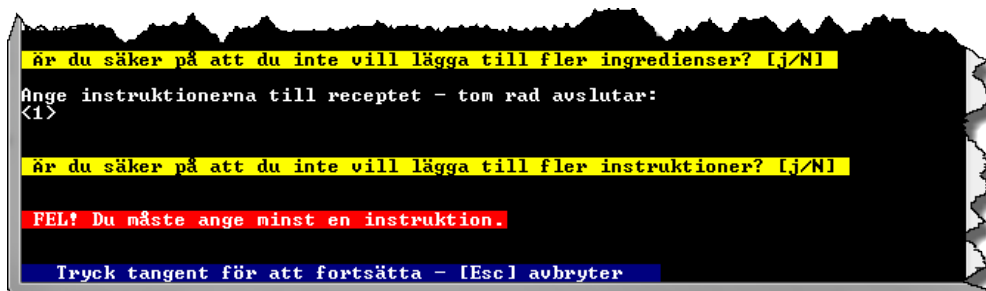
Receptets instruktioner

Efter att ingredienserna matats in ska instruktionerna anges. Ett recept måste minst ha en instruktion. Användaren ska kunna avsluta inmatningen av instruktioner på lämpligt sätt, t.ex. genom att ange en tom rad för en instruktion.



Figur 25. Fem instruktioner har lagts till. Den sjätte är en tom rad varför användaren måste bekräfta att inga fler instruktioner ska läggas till.

Väljer användaren att inte lägga till några instruktioner alls ska ett felmeddelande visas varefter användaren ska kunna välja att avbryta skapandet av ett nytt recept, namn och ingredienser går då förlorade, och återvända till menyn.



Figur 26. Inga instruktioner har lagts till varför ett felmeddelande visas. Väljer användaren att trycka på Esc-tangenten går inmatat namn och ingredienser förlorade.

Hitta recept med en eller flera ingredienser

Väljer användaren menyalternativet '7. Hitta recept med ingredienser.' ska applikationen uppmana att ange en eller flera ingredienser ett recept ska innehålla. Recepten som innehåller ingrediensen eller ingredienserna ska presenteras i en lista där användaren ska kunna välja att visa ett av recepten.

För att avgränsa uppgiften något behöver inte fritextsökning implementeras. Detta innebär att vid filtrering av recept så behöver inte t.ex. "lök" ge träffar på recept som innehåller "purjolök" eller "gul lök".

Görs en sökning av recept innehållande t.ex. "dill filmjolk" ska bara recept innehållande båda ingredienserna ge träffar.

Välja filnamn

Väljer användaren menyalternativet '1. Öppna textfil med recept.' ska applikationen be användaren ange namnet på textfilen som ska öppnas. Anger inte användaren något namn ska `recipes.txt` öppnas.

Väljer användaren menyalternativet '2. Spara recept på textfil.' ska applikationen be användaren ange namnet på textfilen som recepten ska skrivas till. Anger inte användaren något namn ska `recipes.txt` öppnas. Existerar filen som recepten ska skrivas till ska användaren informeras om detta och bekräfta att den existerande textfilen kan skrivas över. Om en fil finns eller inte kan undersökas med klassen `File` och den statiska metoden `Exists()`.

Redigera recept

Väljer användaren menyalternativet '8. Ändra recept.' ska användaren kunna redigera ett recepts ingredienser och instruktioner. Både ingredienser och instruktioner ska kunna tas bort. Nya ingredienser och instruktioner ska kunna läggas till ett befintligt recept. Förutom att själva ingredienserna och instruktionerna ska kunna ändras ska ordningen också kunna ändras.