|  |
| --- |
| Laborationsanvisning |

Digital väckarklocka

Steg 2, laborationsuppgift 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 Digital väckarklocka 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 samt fotografier 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**

[A. Uppgift 3](#_Toc304742270)

[Problem 3](#_Toc304742271)

[Hantering av timmar och minuter 3](#_Toc304742272)

[Test av klassen 3](#_Toc304742273)

[Klassen AlarmClock 5](#_Toc304742274)

[Klassen Program 6](#_Toc304742275)

[A-krav 7](#_Toc304742276)

[Läsvärt 8](#_Toc304742277)

[B. Uppgift 9](#_Toc304742278)

[Problem 9](#_Toc304742279)

[Väckarklockans delar 9](#_Toc304742280)

[Hantering av timmar och minuter 9](#_Toc304742281)

[Test av klassen 10](#_Toc304742282)

[Klassdiagram över AlarmClock, ClockDisplay och NumberDisplay 12](#_Toc304742283)

[Klassen AlarmClock 12](#_Toc304742284)

[Klassen ClockDislay 13](#_Toc304742285)

[Klassen NumberDisplay 13](#_Toc304742286)

[Klassen Program 14](#_Toc304742287)

[B-krav 15](#_Toc304742288)

[Läsvärt 16](#_Toc304742289)

[C. Uppgift 19](#_Toc304742290)

[Problem 19](#_Toc304742291)

[Väckarklockans delar 19](#_Toc304742292)

[Hantering av timmar och minuter 19](#_Toc304742293)

[Test av klassen 20](#_Toc304742294)

[Klassdiagram över AlarmClock, ClockDisplay och NumberDisplay 22](#_Toc304742295)

[Klassen AlarmClock 22](#_Toc304742296)

[Klassen ClockDislay 24](#_Toc304742297)

[Klassen NumberDisplay 24](#_Toc304742298)

[Klassen Program 26](#_Toc304742299)

[C-krav 26](#_Toc304742300)

[Läsvärt 28](#_Toc304742301)

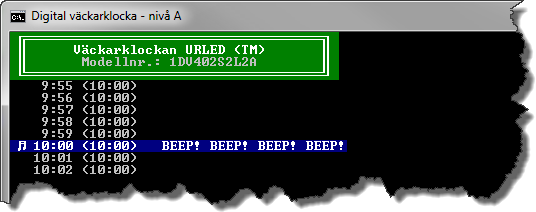
# Uppgift

## Problem

Digitala klockor visar tiden med siffror. Två sorters visning förekommer:

* 24-timmars som visar 0:00 – 23:59
* 12 timmars som visar 1:00 – 12:59

Du ska skriva ett program som simulerar en 24-timmars digital klockdisplay. För att simulera den digitala klockdisplayen ska du skriva och använda dig av klassen AlarmClock. Förutom att hålla ordning på aktuell tid ska även klassen kunna hantera en alarmtid. Du ska även testa att klassen fungerar som det är tänkt genom att skriva ett mindre test.



Figur A.1. Exempel på resultat av test av klassen AlarmClock.

### Hantering av timmar och minuter

Den digitala klockdisplayen ska presentera tiden i timmar och minuter, t.ex. 9:57. Värdet till vänster om kolonet är timmarna, som går från 0 till och med 23. Passeras 23 ska värdet sättas till 0. Värdet till höger är minuterna. Giltiga värden för det högra värdet är 0 till och med 59. Passeras 59 ska värdet sättas till 0.

I princip handlar det om att låta en tidpunkt utgöras två värden som presenteras åtskilda av ett kolon (:). Med andra ord kan en tidpunkt hanteras med hjälp av två heltal i form av två privata fält i klassen AlarmClock. För att säkerställa att fälten verkligen inte tilldelas felaktiga värden måste dessa kapslas in av publika egenskaper vars set-metoder kastar undantag om försök görs att tilldela egenskaperna värden som inte klarar valideringen.

### Test av klassen

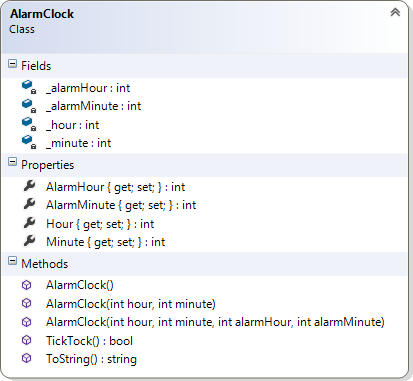
För att säkerställa att klassen AlarmClock uppfyller ställda krav ska ett enklare test skrivas som visar detta. Testet ska innehålla kod som verifierar att konstruktorer, egenskaper och metoder fungerar. Efter att klassen AlarmClock har implementerats ska testkoden skrivs i metoden Main() i klassen Program. Testet ska bestå av sju deltester:

#### Testlista

|  |  |
| --- | --- |
|  | Test av standardkonstruktor.  För att kontroller det nya objektet status ska strängen representerande värdet av objektet skrivas ut. I konsolfönstret ska strängen 0:00 (0:00). Skrivas ut om standardkonstruktorn fungerar som den ska.    Figur A.2. |
|  | Test av konstruktorn med två parametrar.  Argumenten 9 och 42 ska användas då ett nytt objekt skapas. Tiden som ska skrivas ut är 9:42 (0:00).    Figur A.3. |
|  | Test av konstruktorn med fyra parametrar.  Argumenten 13, 24, 7 och 35 ska användas då ett nytt objekt skapas. Tiden som ska skrivas ut är 13:24 (7:35).    Figur A.4. |
|  | Test av metoden TickTock() som ska låta klockan gå en minut.  Ställ ett befintligt AlarmClock-objekt till 23:58 och låter den gå 13 minuter. Lista med 13 tider ska skrivas ut där minuterna ökar med en minut för varje tid.  Timmar ska gå från 23 till 0 och då timmar utgörs av ett ental ska timmen beskrivas att en siffra. Då minuterna är ental ska beskrivningen av minuten inledas med 0.    Figur A.5. |
|  | Ställer befintligt AlarmClock-objekt till tiden 6:12 och alarmtiden till 6:15 och låter den gå 6 minuter. Testkoden ska på lämpligt sätt indikera när ett alarm går.    Figur A.6. |
|  | Test av egenskaperna så att undantag kastas då tid och alarmtid tilldelas felaktiga värden.    Figur A.7. |
|  | Test av konstruktorer så att undantag kastas då tid och alarmtid tilldelas felaktiga värden.    Figur A.8. |

### Klassen AlarmClock

Klassen AlarmClock måste implementeras så den minst innehåller medlemmarna enligt klassdiagrammet i Figur A.9 och har den funktionalitet som beskrivs för respektive medlem.



Figur A.9. Klassdiagram för klassen AlarmClock.

#### Fältet \_alarmHour

Privat fält som innehåller värdet för den alarmtimmen. Kapslas in av egenskapen AlarmHour.

#### Fältet \_alarmMinute

Privat fält som innehåller värdet för den alarmminuten. Kapslas in av egenskapen AlarmMinute.

#### Fältet \_hour

Privat fält som innehåller värdet för timmen för aktuellt klockslag. Kapslas in av egenskapen Hour.

#### Fältet \_minute

Privat fält som innehåller värdet för minuten för aktuellt klockslag. Kapslas in av egenskapen Minute.

#### Egenskapen AlarmHour

Egenskap, som kapslar in det privata fältet \_alarmHour. set-metoden måste validera att värdet som ska tilldelas \_alarmHour är i det slutna intervallet mellan 0 och 23. Är värdet inte i intervallet ska ett undantag av typen ArgumentException kastas.

#### Egenskapen AlarmMinute

Egenskap, som kapslar in det privata fältet \_alarmMinute. set-metoden måste validera att värdet som ska tilldelas \_alarmMinute är i det slutna intervallet mellan 0 och 59. Är värdet inte i intervallet ska ett undantag av typen ArgumentException kastas.

#### Egenskapen Hour

Egenskap, som kapslar in det privata fältet \_hour. set-metoden måste validera att värdet som ska tilldelas \_hour är i det slutna intervallet mellan 0 och 23. Är värdet inte i intervallet ska ett undantag av typen ArgumentException kastas.

#### Egenskapen Minute

Egenskap, som kapslar in det privata fältet \_minute. set-metoden måste validera att värdet som ska tilldelas \_minute är i det slutna intervallet mellan 0 och 59. Är värdet inte i intervallet ska ett undantag av typen ArgumentException kastas.

#### Konstruktorerna

Konstruktorerna, som är tre till antalet, ska se till att ett AlarmClock-objekt blir korrekt initierat. Det innebär att fälten ska tilldelas lämpliga värden.

Standardkonstruktorn AlarmClock() ska initiera fälten till deras standardvärden. Ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har två parametrar.

Med konstruktorn AlarmClock(int hour, int minute) ska ett objekt kunna initieras så att alarmklockan ställs på den tid som parametrarna för timme respektive minut anger. Ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har fyra parametrar.

Med konstruktorn AlarmClock(int hour, int minute , int alarmHour, int alarmMinute) ska ett objekt kunna initieras så att alarmklockan ställs på den tid och alarmtid som parametrarna anger. Detta är den enda av konstruktorerna som får innehålla kod som leder till att fält i klassen tilldelas värden.

#### Metoden TickTock

Publik metod som anropas för att få klockan att gå en minut. Om den nya tiden överensstämmer med alarmtiden ska metoden returnera true, annars false. Inga utskrifter till konsolfönstret får göras av metoden.

Metoden ansvarar för att öka minuternas värde med 1. Värdet för minuterna måste vara i det slutna intervallet mellan 0 och 59. Då värdet för minuterna sätts till 0 ökas lämpligen timmarnas värde med 1. Värdet för timmarna måste vara i det slutna intervallet mellan 0 och 23.

#### Metoden ToString

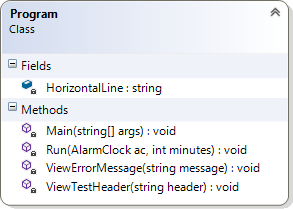
Publik metod som har som uppgift att returnera en sträng representerande värdet av en instans av klassen AlarmClock. Inga utskrifter till konsolfönstret får göras av metoden.

Är timmen ett ental ska timmen presenteras utan att inledas med 0, medan då minuterna är ental ska de inledas med 0. Är klocka fem över elva på kvällen ska tiden presenteras som 23:05. Är tiden åtta minuter över sju på morgonen ska tiden presenteras som 7:08.

### Klassen Program

Klassen program ska innehålla koden som testar klassen AlarmClock.

Figur A.10 visar ett förslag på medlemmar som kan användas för att skapa de sju testerna som listas under rubriken ”Testlista” på sidan 3. Klassen Program behöver inte på något sätt följa förslaget som enbart ska ses som en rekommendation. Det finns bara ett krav som måste uppfyllas och det är att metoden Main() måste se till att de sju deltesterna genomförs på avsett sätt.



Figur A.10. Förslag på medlemmar av klassen Program.

#### Metoden Main

Metoden ska instansiera objekt av klassen AlarmClock och testa konstruktorerna, egenskaperna och metoderna.

#### Metoden Run

Privat statisk metod som har två parametrar. Den första parametern är en referens till AlarmClock-objekt. Den andra parametern är antalet minuter som AlarmClock-objektet ska gå (vilket lämpligen görs genom att låta ett AlarmClock-objekt göra upprepade anrop av metoden TickTock()).

#### Metoden ViewErrorMessage

Privat statisk metoden som tar ett felmeddelande som argument och presenterar det.



Figur A.11. Exempel presentation av två felmeddelanden.

#### Metoden ViewTestHeader

Privat statisk metod som tar en sträng som argument och presenterar strängen.



Figur A.12. Exempel på ett tests rubrik inklusive horisontell linje ovan testrubriken.

## A-krav

|  |  |
| --- | --- |
|  | Klassen AlarmClock ska implementeras enligt klassdiagrammet i Figur A.9. |
|  | Inga av medlemmarna i klassen AlarmClock får skriva ut något i konsolfönstret. |
|  | Fältet \_hour ska kapslas in av egenskapen Hour som validerar att värdet är i det slutna intervallet mellan 0 och 23 innan värdet tilldelas fältet \_hour. Är inte värdet i det slutna intervallet ska ett undantag av typen ArgumentException kastas. |
|  | Fältet \_minute ska kapslas in av egenskapen Minute som validerar att värdet är i det slutna intervallet mellan 0 och 59 innan värdet tilldelas fältet \_minute. Är inte värdet i det slutna intervallet ska ett undantag av typen ArgumentException kastas. |
|  | Fältet \_alarmHour ska kapslas in av egenskapen AlarmHour som validerar att värdet är i det slutna intervallet mellan 0 och 23 innan värdet tilldelas fältet \_alarmHour. Är inte värdet i det slutna intervallet ska ett undantag av typen ArgumentException kastas. |
|  | Fältet \_alarmTimeMinute ska kapslas in av egenskapen AlarmMinute som validerar att värdet är i det slutna intervallet mellan 0 och 59 innan värdet tilldelas fältet \_alarmMinute. Är inte värdet i det slutna intervallet ska ett undantag av typen ArgumentException kastas. |
|  | Då ett nytt AlarmClock-objekt instansieras ska undantag kastas av klassen om försök görs att initiera objektet med värden som inte ligger inom de slutna intervallen för timmar och minuter. |
|  | Ett anrop av metoden TickTock() ska leda till att klockan går en minut, t.ex. från 9:42 till 9:43. |
|  | Metoden TickTock() ska returnera true om den nya tiden är lika med alarmtiden, annars false. |
|  | Metoden ToString() ska beskriva aktuellt AlarmClock-objekt i form av en sträng innehållande aktuell tid samt alarmtiden inom parenteser.  Tiderna ska presenteras på formatet HH:mm, d.v.s. timmar och minuter separerade av kolon (:). Är timmen mindre än 10 ska enbart entalet visas. Är minuten mindre än 10 ska presentationen av minuten inledas med 0. |
|  | Klassen Program måste implementeras så att de sju deltesterna under rubriken ”Testlista” på sidan 3 körs då programmet exekveras.  Var och en av testerna måste lyckas vilket ska kunna verifieras genom lämpliga utskrifter i konsolfönstret. |

## Läsvärt

* Klasser
  + Essential C# 5.0, 209-220.
  + <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
* Åtkomstmodifierare (*”Access Modifiers”*)
  + Essential C# 5.0, 227-229.
  + <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
* Egenskaper
  + Essential C# 5.0, 229-235.
  + <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.
* Konstruktorer
  + Essential C# 5.0, 244-248, 250-253.
  + <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>

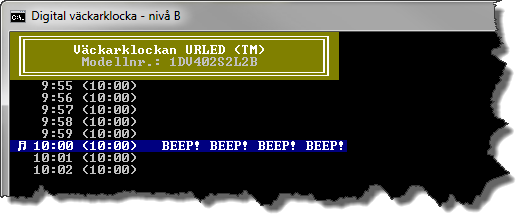
# Uppgift

## Problem

Digitala klockor visar tiden med siffror. Två sorters visning förekommer:

* 24-timmars som visar 0:00 – 23:59
* 12 timmars som visar 1:00 – 12:59

Du ska skriva ett program som simulerar en 24-timmars digital klockdisplay. För att simulera den digitala klockdisplayen ska du skriva och använda dig av klasserna AlarmClock, ClockDisplay och NumberDisplay. Förutom att hålla ordning på aktuell tid ska även klassen AlarmClock kunna hantera en alarmtid. Du ska även testa att klasserna fungerar som det är tänkt genom att skriva ett mindre test.



Figur B.1. Exempel på resultat av test av den digitala väckarklockan.

### Väckarklockans delar

För att undvika att kod upprepas (bryta mot principen DRY, *”Don’t Repeat Your Self”*) ska väckarklockan utgöras av flera klasser. Där en klass representerande en väckarklocka byggs upp med hjälp av aggregat, d.v.s. en klass innehåller fält med referenser till objekt av andra klasser.

Den egentliga väckarklockan ska representeras av klassen AlarmClock. För att undvika upprepning av kod ska AlarmClock innehålla två referenser till objekt av typen ClockDisplay. Den ena instansen av ClockDisplay ska ansvara för den aktuella tiden; den andra ska ansvara för alarmtiden.

Objekt av typen ClockDisplay används för att hålla reda på timmar och minuter, och presentationen av dem. För att undvika upprepning av kod som har med hantering av timmar och minuter att göra ska klassen ClockDisplay innehålla två referenser till objekt av typen NumberDisplay. Den ena instansen av NumberDisplay ska ansvara för timmarna; den andra ska ansvara för minuterna.

Instanser av klassen NumberDisplay ska ansvara för att ett värde håller sig inom ett givet intervall, som sträcker sig från 0 till ett givet maximalt värde. Beroende på om det är timmar eller minuter ser intervallet olika ut. I det slutna intervallet mellan 0 och 23 om det gäller timmar. I det slutna intervallet mellan 0 och 59 om det gäller minuter.

### Hantering av timmar och minuter

Den digitala klockdisplayen ska presentera tiden i timmar och minuter, t.ex. 9:57. Värdet till vänster om kolonet är timmarna, som går från 0 till och med 23. Passeras 23 ska värdet sättas till 0. Värdet till höger är minuterna. Giltiga värden för det högra värdet är 0 till och med 59. Passeras 59 ska värdet sättas till 0.

Det är ClockDipsplay tillsammans med NumberDisplay som ansvara för att en korrekt textbeskrivning av en tid finns att tillgå. Exempelvis ska en fem över sju på morgonen ge textbeskrivningen 7:05, medan fem minuter i nio på kvällen ska ge 20:55.

I princip handlar det om att låta en tidpunkt utgöras två värden, representerade av NumberDisplay-objekt, som presenteras åtskilda av ett kolon (:). Med andra ord kan en tidpunkt representeras med hjälp av två privata fält med referenser till NumberDisplay-objekt i klassen ClockDisplay.

Används ett NumberDisplay-objekt till att representera minuter ska textbeskrivningen av värdet inledas med 0 om värdet är mindre än 10. Används ett NumberDisplay-objekt till att representera timmar ska inte textbeskrivningen inledas med 0.

För att säkerställa att fält inte tilldelas felaktiga värden måste fält i klasserna kapslas in av lämpliga publika egenskaper. Även om klasserna AlarmClock och ClockDisplay innehåller egenskaper med set-metoder räcker det om validering sker i NumberDisplay, eftersom NumberDisplay-objektet känner till vilket som är dess maximala värde. En set-metod i klassen NumberDisplay ska därför kasta ett undantag om försök görs att tilldela en egenskap ett värde som inte ligger i det angivna intervallet för objektet.

### Test av klassen

För att säkerställa att klasserna AlarmClock, ClockDisplay och NumberDisplay uppfyller ställda krav ska ett enklare test skrivas som visar detta. Testet ska innehålla kod som verifierar att konstruktorer, egenskaper och metoder fungerar. Efter att klassen AlarmClock, ClockDisplay och NumberDisplay har implementerats ska testkoden skrivs i metoden Main() i klassen Program. Testet ska bestå av sju deltester:

#### Testlista

|  |  |
| --- | --- |
|  | Test av standardkonstruktor.  För att kontroller det nya objektet status ska strängen representerande värdet av objektet skrivas ut. I konsolfönstret ska strängen 0:00 (0:00). Skrivas ut om standardkonstruktorn fungerar som den ska.    Figur B.2. |
|  | Test av konstruktorn med två parametrar.  Argumenten 9 och 42 ska användas då ett nytt objekt skapas. Tiden som ska skrivas ut är 9:42 (0:00).    Figur B.3. |
|  | Test av konstruktorn med fyra parametrar.  Argumenten 13, 24, 7 och 35 ska användas då ett nytt objekt skapas. Tiden som ska skrivas ut är 13:24 (7:35).    Figur B.4. |
|  | Test av metoden TickTock() som ska låta klockan gå en minut.  Ställ ett befintligt AlarmClock-objekt till 23:58 och låter den gå 13 minuter. Lista med 13 tider ska skrivas ut där minuterna ökar med en minut för varje tid.  Timmar ska gå från 23 till 0 och då timmar utgörs av ett ental ska timmen beskrivas att en siffra. Då minuterna är ental ska beskrivningen av minuten inledas med 0.    Figur B.5. |
|  | Ställer befintligt AlarmClock-objekt till tiden 6:12 och alarmtiden till 6:15 och låter den gå 6 minuter. Testkoden ska på lämpligt sätt indikera när ett alarm går.    Figur B.6. |
|  | Test av egenskaperna så att undantag kastas då tid och alarmtid tilldelas felaktiga värden.    Figur B.7.  (Av felmeddelandena framgår inte att det första och andra meddelandet berörs väckarklockans tid och det tredje och fjärde felmeddelandet berör väckarklockans alarmtid.) |
|  | Test av konstruktorer så att undantag kastas då tid och alarmtid tilldelas felaktiga värden.    Figur B.8.  (Av felmeddelandena framgår inte att det första meddelandet berörs väckarklockans tid och det andra felmeddelandet berör väckarklockans alarmtid.) |

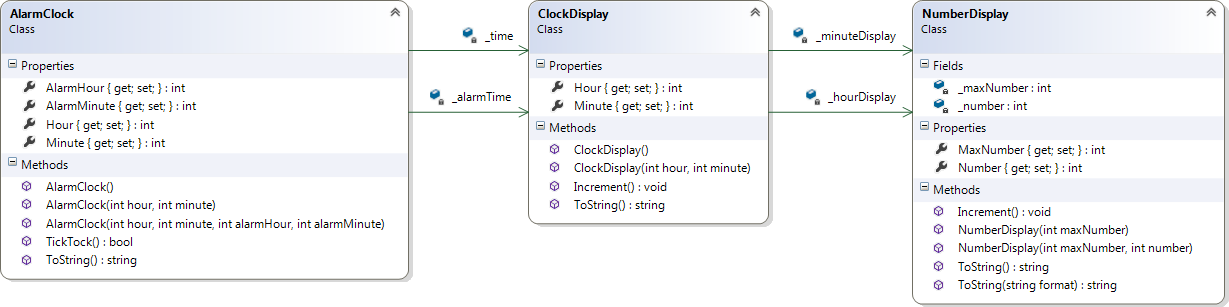
### Klassdiagram över AlarmClock, ClockDisplay och NumberDisplay

Klasserna AlarmClock, ClockDisplay och NumberDisplay måste implementeras så den minst innehåller medlemmarna enligt klassdiagrammet i Figur B.9 och har den funktionalitet som beskrivs för respektive klass och medlem.

Klasserna AlarmClock och ClockDisplay ser ut att inte ha några privata fält då avdelningen Fields saknas för klasserna i klassdiagrammet. Klasserna har dock fält och dessa symboliseras med så kallade aggregat.

Mellan AlarmClock och ClockDisplay finns två aggregat, \_alarmTime och \_time. Dessa ska implementeras som privata fält som ska kunna referera till objekt av typen ClockDipslay.

På samma sätt ska aggregaten \_minuteDisplay och \_hourDisplay mellan ClockDisplay och NumberDisplay implementeras i klassen ClockDisplay som privata fält som ska kunna referera till objekt av typen NumberDisplay.



Figur B.9. Klassdiagram för klasserna AlarmClock, ClockDisplay och NumberDisplay.

### Klassen AlarmClock

#### Fältet \_alarmTime

Privat fält som i innehåller referens till ett ClockDisplay-objekt som ansvarar för alarmtiden. Kapslas in av egenskaperna AlarmHour och AlarmMinute.

#### Fältet \_time

Privat fält som i innehåller referens till ett ClockDisplay-objekt som ansvarar för väckarklockans aktuella tid. Kapslas in av egenskaperna Hour och Minute.

#### Egenskapen AlarmHour

Publik egenskap, som kapslar in det privata fältet \_alarmTime och dess egenskap Hour.

#### Egenskapen AlarmMinute

Publik egenskap, som kapslar in det privata fältet \_alarmTime och dess egenskap Minute.

#### Egenskapen Hour

Publik egenskap, som kapslar in det privata fältet \_time och dess egenskap Hour.

#### Egenskapen Minute

Publik egenskap, som kapslar in det privata fältet \_time och dess egenskap Minute.

#### Konstruktorerna

Konstruktorerna, som är tre till antalet, ska se till att ett AlarmClock-objekt blir korrekt initierat. Det innebär att fälten ska initieras med lämpliga värden.

Standardkonstruktorn AlarmClock() ska initiera fälten till deras standardvärden. Ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har två parametrar.

Med konstruktorn AlarmClock(int hour, int minute) ska ett objekt kunna initieras så att alarmklockan ställs på den tid som parametrarna för timme respektive minut anger. Ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har fyra parametrar.

Med konstruktorn AlarmClock(int hour, int minute , int alarmHour, int alarmMinute) ska ett objekt kunna initieras så att alarmklockan ställs på den tid och alarmtid som parametrarna anger. Detta är den enda av konstruktorerna som får innehålla kod som leder till att fält i klassen tilldelas värden.

#### Metoden TickTock

Publik metod som anropas för att få klockan att gå en minut. Om den nya tiden överensstämmer med alarmtiden ska metoden returnera true, annars false. Inga utskrifter till konsolfönstret får göras av metoden.

#### Metoden ToString

Publik metod som har som uppgift att returnera en sträng representerande värdet av en instans av klassen AlarmClock. Strängen ska innehålla aktuell tid samt alarmtiden inom parenteser. Inga utskrifter till konsolfönstret får göras av metoden.

### Klassen ClockDislay

#### Fältet \_hourDisplay

Privat fält som i innehåller referens till ett NumberDisplay-objekt som ansvarar för timmarna. Kapslas in av egenskapen Hour.

#### Fältet \_minuteDisplay

Privat fält som i innehåller referens till ett NumberDisplay-objekt som ansvarar för minuterna. Kapslas in av egenskapen Minute.

#### Egenskapen Hour

Publik egenskap, som kapslar in det privata fältet \_hourDisplay och dess egenskap Number.

#### Egenskapen Minute

Publik egenskap, som kapslar in det privata fältet \_minuteDisplay och dess egenskap Number.

#### Konstruktorerna

Konstruktorerna, som är två till antalet, ska se till att ett ClockDisplay-objekt blir korrekt initierat. Det innebär att fälten ska initieras med lämpliga värden.

Standardkonstruktorn ClockDisplay() ska se till att fälten initieras så de refererar till NumberDisplay-objekt men ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har två parametrar.

Med konstruktorn ClockDisplay(int hour, int minute) ska ett objekt initieras så att objektet ställs på den tid som parametrarna anger. Detta är den enda av konstruktorerna som får innehålla kod som leder till att fält i klassen tilldelas värden.

#### Metoden Increment

Publik metod som anropas för att få ClockDisplay-objektet att gå en minut. Metoden ansvarar för att öka minuternas värde med 1. Då värdet för minuterna blir 0 ökas lämpligen timmarnas värde med 1. Inga utskrifter till konsolfönstret får göras av metoden.

#### Metoden ToString

Publik metod som har som uppgift att returnera en sträng representerande värdet av en instans av klassen ClockDisplay. Strängen ska innehålla tiden på formatet HH:mm. Inga utskrifter till konsolfönstret får göras av metoden.

### Klassen NumberDisplay

#### Fältet \_maxNumber

Privat fält som innehåller maxvärdet numret kan anta. Kapslas in av egenskapen MaxNumber.

#### Fältet \_number

Privat fält som innehåller själva numrets värde. Kapslas in av egenskapen Number.

#### Egenskapen MaxNumber

Publik egenskap, som kapslar in det privata fältet \_maxNumber. set-metoden måste validera att värdet som ska tilldelas \_maxNumber är större än 0. Är värdet inte det ska ett undantag av typen ArgumentException kastas.

#### Egenskapen Number

Publik egenskap, som kapslar in det privata fältet \_number. set-metoden måste validera att värdet som ska tilldelas \_number är i det slutna intervallet mellan 0 och maxvärdet. Är värdet inte i intervallet ska ett undantag av typen ArgumentException kastas.

#### Konstruktorerna

Konstruktorerna, som är två till antalet, ska se till att ett NumberDisplay-objekt blir korrekt initierat. Det innebär att fälten ska initieras med lämpliga värden.

Konstruktorn NumberDisplay(int maxNumber) ska se till att fälten initieras så de refererar till NumberDisplay-obejkt men ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har två parametrar.

Med konstruktorn NumberDisplay(int maxNumber, int number) ska ett objekt initieras så att objektets fält tilldelas värdena parametrarna har. Detta är den enda av konstruktorerna som får innehålla kod som tilldelar fält i klassen värden.

#### Metoden Increment

Publik metod som anropas för att få NumberDisplay-objektet att öka numret med 1. Då värdet fältet \_number har ska passera värdet fältet \_maxNumber har ska fältet \_number tilldelas värdet 0. Inga utskrifter till konsolfönstret får göras av metoden.

#### Metoderna ToString

ToString() ska överlagras, d.v.s. det ska finnas två metoder med samma namn men med olika paramterlistor.

Den publika metoden ToString() har som uppgift att returnera en sträng representerande värdet av en instans av klassen NumberDisplay. Strängen ska innehålla numret, utan att nummer mindre än tio inleds med 0. Inga utskrifter till konsolfönstret får göras av metoden.

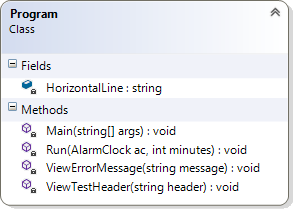
Den publika metoden ToString(string format) har som uppgift att returnera en sträng representerande värdet av en instans av klassen NumberDisplay. Formatsträngen ska bestämma om nummer mindre än tio ska inledas med 0.

Är formatsträngen ”0” eller ”G” ska numret inte inledas med 0. Är formatsträngen ”00” ska numret inleda med 0 om numret är mindre än tio. Alla övriga värden på formatsträngen ska leda till att ett undantag av typen FormatException kastas.

Inga utskrifter till konsolfönstret får göras av någon av metoderna.

### Klassen Program

Klassen program ska innehålla koden som testar klassen AlarmClock. Figur B.10 visar ett förslag på medlemmar som kan användas för att skapa de sju testerna som listas under rubriken ”Testlista” på sidan 10. Klassen Program behöver inte på något sätt följa förslaget som enbart ska ses som en rekommendation. Det finns bara ett krav som måste uppfyllas och det är att metoden Main() måste se till att de sju deltesterna genomförs på avsett sätt.



Figur B.10. Förslag på medlemmar av klassen Program.

#### Metoden Main

Metoden ska instansiera objekt av klassen AlarmClock och testa konstruktorerna, egenskaperna och metoderna.

#### Metoden Run

Privat statisk metod som har två parametrar. Den första parametern är en referens till AlarmClock-objekt. Den andra parametern är antalet minuter som AlarmClock-objektet ska gå (vilket lämpligen görs genom att låta ett AlarmClock-objekt göra upprepade anrop av metoden TickTock()).

#### Metoden ViewErrorMessage

Privat statisk metoden som tar ett felmeddelande som argument och presenterar det.



Figur B.11. Exempel presentation av två felmeddelanden.

#### Metoden ViewTestHeader

Privat statisk metod som tar en sträng som argument och presenterar strängen.



Figur B.12. Exempel på ett tests rubrik inklusive horisontell linje ovan testrubriken.

## B-krav

|  |  |
| --- | --- |
|  | Klasserna AlarmClock, ClockDisplay och NumberDisplay ska implementeras enligt klassdiagrammet i Figur B.9. |
|  | Inga av medlemmarna i klasserna AlarmClock, ClockDisplay och NumberDisplay får skriva ut något i konsolfönstret. |
|  | Det är bara den konstruktor i klasserna AlarmClock, ClockDisplay och NumberDisplay som har flest parametrar som får tilldela fält, eller egenskaper, värden. |
|  | Då ett nytt AlarmClock-objekt instansieras ska undantag kastas av klassen om försök görs att initiera objektet med värden som inte ligger inom de slutna intervallen för timmar och minuter. |
|  | Klassen AlarmClock ska innehålla två aggregat i form av två privata fält av typen ClockDisplay. |
|  | Klassen AlarmClock får inte innehålla någon kod som validerar parametrars värden. Validering ska ske genom aggregerade klasser. |
|  | Klassen AlarmClock ska ha fyra publika egenskaper för väckarklockans tid och alarmtid. Två egenskaper för hantering av timmar och två egenskaper för hantering av minuter. |
|  | Ett anrop av metoden TickTock() ska leda till att väckarklockan går en minut, t.ex. från 9:42 till 9:43. |
|  | Metoden TickTock() ska returnera true om den nya tiden är lika med alarmtiden, annars false. |
|  | Metoden ToString() ska beskriva aktuellt AlarmClock-objekt i form av en sträng innehållande aktuell tid samt alarmtiden inom parenteser. |
|  | Då ett nytt ClockDisplay-objekt instansieras ska undantag kastas av klassen om försök görs att initiera objektet med värden som inte ligger inom de slutna intervallen för timmar och minuter. |
|  | Klassen ClockDisplay ska innehålla två aggregat i form av två privata fält av typen NumberDisplay. Det ena aggregatet ska representera timmar och dess maxvärde ska därför sättas till 23. Det andra aggregatet ska representera minuter varför dess maxvärde ska sättas till 59. |
|  | Klassen ClockDisplay får inte innehålla någon kod som validerar parametrars värden. Validering ska ske i aggregerad klass. |
|  | Ett anrop av metoden Increment() i klassen ClockDisplay ska leda till att NumberDisplay-objektet som representerar minuter går en minut, t.ex. från 9:42 till 9:43. Får detta NumberDisplay-objekt värdet 0 ska det andra NumberDisplay-objektet, som representerar timmar, öka sitt värde. |
|  | Metoden ToString() ska beskriva aktuellt ClockDisplay-objekt i form av en sträng innehållande aktuell tid. Tiden ska presenteras på formatet HH:mm, där minuterna ska inledas med 0 om minuterna är mindre än tio. Timmarna ska inte inledas med 0 om de är mindre än tio. |
|  | Då ett nytt NumberDisplay-objekt instansieras ska undantag kastas av klassen om försök görs att initiera objektet med värden som inte ligger inom det slutna intervallen mellan 0 och maxvärdet. |
|  | I klassen NumberDisplay ska fältet \_maxNumber kapslas in av egenskapen MaxNumber som validerar att värdet är större än 0. Är inte värdet inte större än 0 ska ett undantag av typen ArgumentException kastas. |
|  | I klassen NumberDisplay ska fältet \_number kapslas in av egenskapen Number som validerar att värdet är i det slutna intervallet mellan 0 och maxvärdet innan värdet tilldelas fältet \_number. Är inte värdet i det slutna intervallet ska ett undantag av typen ArgumentException kastas. |
|  | Ett anrop av metoden Increment() i klassen NumberDisplay ska leda till att egenskapen Number, och därmed fältet \_number, ökar sitt värde med 1. Om värdet skulle bli större än maxvärdet ska värdet istället sättas till 0. |
|  | Klassen NumberDisplay ska två överlagrade versioner av metoden ToString(). Den ena versionen ska överskugga ToString(), som ärvs från basklassen Object. Den andra versionen ska ha en sträng i parameterlistan som bestämmer formatet som textbeskrivningen av anropande NumberDisplay-objekt ska ha. Skickas formatsträngen ”0” eller ”G” med ska textbeskrivningen av numret inte inledas med 0 om numret är mindre än 10. Skickas formatsträngen ”00” med ska textbeskrivningen inledas med 0 om numret är mindre än tio. |
|  | Klassen Program måste implementeras så att de sju deltesterna under rubriken ”Testlista” på sidan 10 körs då programmet exekveras.  Var och en av testerna måste lyckas vilket ska kunna verifieras genom lämpliga utskrifter i konsolfönstret. |

## Läsvärt

* Klasser
  + Essential C# 5.0, 209-220.
  + <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
* Åtkomstmodifierare (*”Access Modifiers”*)
  + Essential C# 5.0, 227-229.
  + <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
* Egenskaper
  + Essential C# 5.0, 229-235.
  + <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.
* Konstruktorer
  + Essential C# 5.0, 244-248, 250-253.
  + <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>

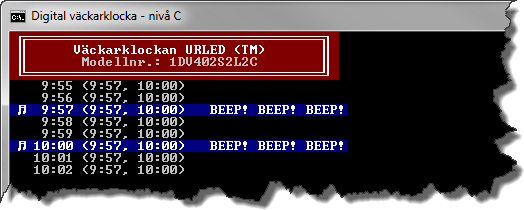
# Uppgift

## Problem

Digitala klockor visar tiden med siffror. Två sorters visning förekommer:

* 24-timmars som visar 0:00 – 23:59
* 12 timmars som visar 1:00 – 12:59

Du ska skriva ett program som simulerar en 24-timmars digital klockdisplay. För att simulera den digitala klockdisplayen ska du skriva och använda dig av klasserna AlarmClock, ClockDisplay och NumberDisplay. Förutom att hålla ordning på aktuell tid ska även klassen AlarmClock kunna hantera flera alarmtider. Du ska även testa att klasserna fungerar som det är tänkt genom att skriva ett mindre test.



Figur C.1. Exempel på resultat av test av den digitala väckarklockan.

### Väckarklockans delar

För att undvika att kod upprepas (bryta mot principen DRY, *”Don’t Repeat Your Self”*) ska väckarklockan utgöras av flera klasser. Där en klass representerande en väckarklocka byggs upp med hjälp av aggregat, d.v.s. en klass innehåller fält med referenser till objekt av andra klasser.

Den egentliga väckarklockan ska representeras av klassen AlarmClock. För att undvika upprepning av kod ska AlarmClock innehålla två referenser; en till ett objekt av typen ClockDisplay. Representerande väckarklockans aktuella tid, och en array med referenser till ClockDisplay-objekt representerande alarmtiderna.

Objekt av typen ClockDisplay används för att hålla reda på timmar och minuter, och presentationen av dem. För att undvika upprepning av kod som har med hantering av timmar och minuter att göra ska klassen ClockDisplay innehålla två referenser till objekt av typen NumberDisplay. Den ena instansen av NumberDisplay ska ansvara för timmarna; den andra ska ansvara för minuterna.

Instanser av klassen NumberDisplay ska ansvara för att ett värde håller sig inom ett givet intervall, som sträcker sig från 0 till ett givet maximalt värde. Beroende på om det är timmar eller minuter ser intervallet olika ut. I det slutna intervallet mellan 0 och 23 om det gäller timmar. I det slutna intervallet mellan 0 och 59 om det gäller minuter.

### Hantering av timmar och minuter

Den digitala klockdisplayen ska presentera tiden i timmar och minuter, t.ex. 9:57. Värdet till vänster om kolonet är timmarna, som går från 0 till och med 23. Passeras 23 ska värdet sättas till 0. Värdet till höger är minuterna. Giltiga värden för det högra värdet är 0 till och med 59. Passeras 59 ska värdet sättas till 0.

Det är ClockDipsplay tillsammans med NumberDisplay som ansvara för att en korrekt textbeskrivning av en tid finns att tillgå. Exempelvis ska en fem över sju på morgonen ge textbeskrivningen 7:05, medan fem minuter i nio på kvällen ska ge 20:55.

I princip handlar det om att låta en tidpunkt utgöras två värden, representerade av NumberDisplay-objekt, som presenteras åtskilda av ett kolon (:). Med andra ord kan en tidpunkt representeras med hjälp av två privata fält med referenser till NumberDisplay-objekt i klassen ClockDisplay.

Används ett NumberDisplay-objekt till att representera minuter ska textbeskrivningen av värdet inledas med 0 om värdet är mindre än 10. Används ett NumberDisplay-objekt till att representera timmar ska inte textbeskrivningen inledas med 0.

För att säkerställa att fält inte tilldelas felaktiga värden måste fält i klasserna kapslas in av lämpliga publika egenskaper. Även om klasserna AlarmClock och ClockDisplay innehåller egenskaper med set-metoder räcker det om validering sker i NumberDisplay, eftersom NumberDisplay-objektet känner till vilket som är dess maximala värde. En set-metod i klassen NumberDisplay ska därför kasta ett undantag om försök görs att tilldela en egenskap ett värde som inte ligger i det angivna intervallet för objektet.

### Test av klassen

För att säkerställa att klasserna AlarmClock, ClockDisplay och NumberDisplay uppfyller ställda krav ska ett enklare test skrivas som visar detta. Testet ska innehålla kod som verifierar att konstruktorer, egenskaper och metoder fungerar. Efter att klassen AlarmClock, ClockDisplay och NumberDisplay har implementerats ska testkoden skrivs i metoden Main() i klassen Program. Testet ska bestå av åtta deltester definierade under rubriken ”Testlista”.

#### Testlista

|  |  |
| --- | --- |
|  | Test av standardkonstruktor.  För att kontroller det nya objektet status ska strängen representerande värdet av objektet skrivas ut. I konsolfönstret ska strängen 0:00 (0:00). Skrivas ut om standardkonstruktorn fungerar som den ska.    Figur C.2. |
|  | Test av konstruktorn med två parametrar av typen int.  Argumenten 9 och 42 ska användas då ett nytt objekt skapas. Tiden som ska skrivas ut är 9:42 (0:00). Anges ingen alarmtid ska alltså alarmtiden sättas till 0:00.    Figur C.3. |
|  | Test av konstruktorn med fyra parametrar.  Argumenten 13, 24, 7 och 35 ska användas då ett nytt objekt skapas. Tiden som ska skrivas ut är 13:24 (7:35).    Figur C.4. |
|  | Test av konstruktorn med variabelt antal, dock minst två, parametrar av typen string.    Figur C.5. |
|  | Test av metoden TickTock() som ska låta klockan gå en minut.  Ställ ett befintligt AlarmClock-objekt till 23:58 och låter den gå 13 minuter. Lista med 13 tider ska skrivas ut där minuterna ökar med en minut för varje tid.  Timmar ska gå från 23 till 0 och då timmar utgörs av ett ental ska timmen beskrivas att en siffra. Då minuterna är ental ska beskrivningen av minuten inledas med 0.    Figur C.6. |
|  | Ställer befintligt AlarmClock-objekt till tiden 6:12 och alarmtiden till 6:15 och låter den gå 6 minuter. Testkoden ska på lämpligt sätt indikera när ett alarm går.    Figur C.7. |
|  | Test av egenskaperna så att undantag kastas då tid och alarmtid tilldelas felaktiga värden.    Figur C.8.  (Av felmeddelandena framgår inte att det första meddelandet berör väckarklockans tid och det andra felmeddelandet berör väckarklockans alarmtid.) |
|  | Test av konstruktorer så att undantag kastas då tid och alarmtid tilldelas felaktiga värden.    Figur C.9.  (Av felmeddelandena framgår inte att det första meddelandet berörs väckarklockans tid och det andra felmeddelandet berör väckarklockans alarmtid.) |

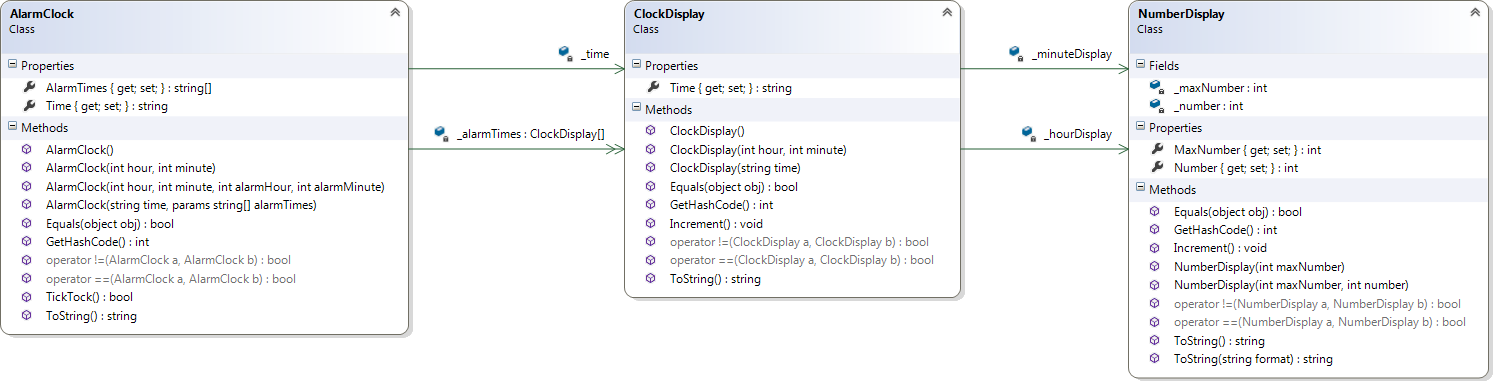
### Klassdiagram över AlarmClock, ClockDisplay och NumberDisplay

Klasserna AlarmClock, ClockDisplay och NumberDisplay måste implementeras så den minst innehåller medlemmarna enligt klassdiagrammet i Figur C.10 och har den funktionalitet som beskrivs för respektive klass och medlem. Samtliga typer ska uppföra sig väl vilket bl.a. innebär att metoderna Equals(), GetHashCode() och ToString(), som deklareras i basklassen Object, ska överskuggas. Dessutom ska operatorerna == och != överlagras av samtliga klasser.

Klasserna AlarmClock och ClockDisplay ser ut att inte ha några privata fält då avdelningen Fields saknas för klasserna. Klasserna har dock fält och dessa symboliseras med hjälp så kallade aggregat.

Mellan AlarmClock och ClockDisplay finns två aggregat, \_alarmTimes och \_time. Dessa ska implementeras som privata fält som ska kunna referera till objekt av typen ClockDipslay[] respektive ClockDisplay.

På samma sätt ska aggregaten \_minuteDisplay och \_hourDisplay mellan ClockDisplay och NumberDisplay implementeras i klassen ClockDisplay som privata fält som ska kunna referera till objekt av typen NumberDisplay.



Figur C.10. Klassdiagram för klasserna AlarmClock, ClockDisplay och NumberDisplay.

### Klassen AlarmClock

#### Fältet \_alarmTimes

Privat fält som i innehåller referens till ett ClockDisplay[]-objekt som ansvarar för alarmtiderna. Kapslas in av egenskaperna AlarmTimes.

#### Fältet \_time

Privat fält som i innehåller referens till ett ClockDisplay-objekt som ansvarar för väckarklockans aktuella tid. Kapslas in av egenskapen Time.

#### Egenskapen AlarmTimes

Publik egenskap som av typen string[] som kapslar in fältet \_alarmTimes som är av typen ClockDisplay[].

get-metoden ska ge en array innehållande alarmtider i form av strängar. Egenskapen konverterar alltså referenser till ClockDisplay-objekt till strängar. Vid ändring av en sträng i arrayen ska inte underliggande ClockDisplay-objekt ändras.

set-metoden ska konvertera varje alarmtid, i form av en sträng, till ett ClockDisplay-objekt.

#### Egenskapen Time

Publik egenskap av typen string som kapslar in fältet \_time som är av typen ClockDisplay.

#### Konstruktorerna

Konstruktorerna, som är fyra till antalet, ska se till att ett AlarmClock-objekt blir korrekt initierat. Det innebär att fälten ska initieras med lämpliga värden.

Standardkonstruktorn AlarmClock() ska initiera fälten så att de refererar till objekt. Ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har två parametrar.

Med konstruktorn AlarmClock(int hour, int minute) ska ett objekt kunna initieras så att väckarklockan ställs på den tid som parametrarna för timme respektive minut anger. Ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har fyra parametrar.

Med konstruktorn AlarmClock(int hour, int minute , int alarmHour, int alarmMinute) ska ett objekt kunna initieras så att väckarklockan ställs på den tid och alarmtid som parametrarna anger. Detta är en konstruktor som får innehålla kod som leder till att fält i klassen tilldelas värden.

Med konstruktorn AlarmClock(string time, params string[] alarmTimes) ska ett objekt kunna initieras så att väckarklockan ställs på den tid och ett godtyckligt antal, minst en dock, alarmtider som parametrarna anger. Detta är en konstruktor som får innehålla kod som leder till att fält i klassen tilldelas värden.

#### Metoden Equals

Överskuggar metoden Equals() i basklassen Object och returnerar ett värde som anger om den anropande instansen är lika med ett angivet objekt beträffande textbeskrivningarna innehållande aktuell tid samt alarmtider.

#### Metoden GetHashCode

Överskuggar metoden GetHashCode() i basklassen Object och returnerar ett heltal av typen int som beskriver det anropande objektet. Lämpligen returnerar metoden hashkoden för textbeskrivningen, d.v.s. det som metoden ToString() returnerar.

#### Metoden TickTock

Publik metod som anropas för att få klockan att gå en minut. Om den nya tiden överensstämmer med någon av alarmtiderna ska metoden returnera true, annars false. Inga utskrifter till konsolfönstret får göras av metoden.

#### Metoden ToString

Publik metod som har som uppgift att returnera en sträng representerande värdet av en instans av klassen AlarmClock. Strängen ska innehålla aktuell tid samt alarmtiderna inom parenteser. Inga utskrifter till konsolfönstret får göras av metoden.

#### Operatorn ==

Avgör om två instanser av AlarmClock är lika. Enkel att implementera då metoden Equals() finns att tillgå.

#### Operatorn !=

Avgör om två instanser av AlarmClock är olika. Enkel att implementera då metoden Equals() finns att tillgå.

### Klassen ClockDislay

#### Fältet \_hourDisplay

Privat fält som i innehåller referens till ett NumberDisplay-objekt som ansvarar för timmarna. Kapslas delvis in av egenskapen Time, som är av typen string.

#### Fältet \_minuteDisplay

Privat fält som i innehåller referens till ett NumberDisplay-objekt som ansvarar för minuterna. Kapslas delvis in av egenskapen Time, som är av typen string.

#### Egenskapen Time

Publik egenskap av typen string, som kapslar in de privata fälten \_hourDisplay och \_minuteDisplay. Strängens format ska vara HH:mm, d.v.s. timmar och minuter åtskilda av ett kolon.

set-metoden ska kasta ett undantag av typen FormatException om strängen som tilldelas egenskapen inte har rätt format. Använd det reguljära uttrycket "^(([0-1]?[0-9])|([2][0-3])):([0-5][0-9])$" för att validera tiden. När väl tiden är validerad delas den enkelt upp i timmar och minuter med hjälp av metoden String.Split().

#### Konstruktorerna

Konstruktorerna, som är tre till antalet, ska se till att ett ClockDisplay-objekt blir korrekt initierat. Det innebär att fälten ska initieras med lämpliga värden.

Standardkonstruktorn ClockDisplay() ska se till att fälten initieras så de refererar till NumberDisplay-objekt men ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har två parametrar.

Med konstruktorn ClockDisplay(int hour, int minute) ska ett objekt initieras så att objektet ställs på den tid som parametrarna anger. Detta är den ena av konstruktorerna som får innehålla kod som leder till att fält i klassen tilldelas värden.

Med konstruktorn ClockDisplay(string time) ska ett objekt initieras så att objektet ställs på den tid som parametern, på formatet HH:mm, anger. Detta är den ena av konstruktorerna som får innehålla kod som leder till att fält i klassen tilldelas värden.

#### Metoden Equals

Överskuggar metoden Equals() i basklassen Object och returnerar ett värde som anger om den anropande instansen är lika med ett angivet objekt beträffande fälten \_hourDisplay och \_minuteDisplay.

#### Metoden GetHashCode

Överskuggar metoden GetHashCode() i basklassen Object och returnerar ett heltal av typen int som beskriver det anropande objektet. Lämpligen returnerar metoden hashkoden för textbeskrivningen, d.v.s. det som metoden ToString() returnerar.

#### Metoden Increment

Publik metod som anropas för att få ClockDisplay-objektet att gå en minut. Metoden ansvarar för att öka minuternas värde med 1. Då värdet för minuterna blir 0 ökas lämpligen timmarnas värde med 1. Inga utskrifter till konsolfönstret får göras av metoden.

#### Metoden ToString

Publik metod som har som uppgift att returnera en sträng representerande värdet av en instans av klassen ClockDisplay. Strängen ska innehålla tiden på formatet HH:mm. Inga utskrifter till konsolfönstret får göras av metoden.

#### Operatorn ==

Avgör om två instanser av ClockDisplay är lika. Enkel att implementera då metoden Equals() finns att tillgå.

#### Operatorn !=

Avgör om två instanser av ClockDisplay är lika. Enkel att implementera då metoden Equals() finns att tillgå.

### Klassen NumberDisplay

#### Fältet \_maxNumber

Privat fält som innehåller maxvärdet numret kan anta. Kapslas in av egenskapen MaxNumber.

#### Fältet \_number

Privat fält som innehåller själva numrets värde. Kapslas in av egenskapen Number.

#### Egenskapen MaxNumber

Publik egenskap, som kapslar in det privata fältet \_maxNumber. set-metoden måste validera att värdet som ska tilldelas \_maxNumber är större än 0. Är värdet inte det ska ett undantag av typen ArgumentException kastas.

#### Egenskapen Number

Publik egenskap, som kapslar in det privata fältet \_number. set-metoden måste validera att värdet som ska tilldelas \_number är i det slutna intervallet mellan 0 och maxvärdet. Är värdet inte i intervallet ska ett undantag av typen ArgumentException kastas.

#### Konstruktorerna

Konstruktorerna, som är två till antalet, ska se till att ett NumberDisplay-objekt blir korrekt initierat. Det innebär att fälten ska initieras med lämpliga värden.

Konstruktorn NumberDisplay(int maxNumber) ska se till att fälten initieras så de refererar till NumberDisplay-obejkt men ingen tilldelning får ske i konstruktorns kropp, som måste vara tom. Denna konstruktor måste därför anropa den konstruktor i klassen som har två parametrar.

Med konstruktorn NumberDisplay(int maxNumber, int number) ska ett objekt initieras så att objektets fält tilldelas värdena parametrarna har. Detta är den enda av konstruktorerna som får innehålla kod som tilldelar fält i klassen värden.

#### Metoden Equals

Överskuggar metoden Equals() i basklassen Object och returnerar ett värde som anger om den anropande instansen är lika med ett angivet objekt beträffande fälten \_maxNumber och \_number.

#### Metoden GetHashCode

Överskuggar metoden GetHashCode() i basklassen Object och returnerar ett heltal av typen int som beskriver det anropande objektet. Lämpligen returnerar metoden hashkoden för t.ex. textbeskrivningen av fälten.

#### Metoden Increment

Publik metod som anropas för att få NumberDisplay-objektet att öka numret med 1. Då värdet fältet \_number har ska passera värdet fältet \_maxNumber har ska fältet \_number tilldelas värdet 0. Inga utskrifter till konsolfönstret får göras av metoden.

#### Metoderna ToString

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

Den publika metoden ToString() har som uppgift att returnera en sträng representerande värdet av en instans av klassen NumberDisplay. Strängen ska innehålla numret, utan att nummer mindre än tio inleds med 0. Inga utskrifter till konsolfönstret får göras av metoden.

Den publika metoden ToString(string format) har som uppgift att returnera en sträng representerande värdet av en instans av klassen NumberDisplay. Formatsträngen ska bestämma om nummer mindre än tio ska inledas med 0.

Är formatsträngen ”0” eller ”G” ska numret inte inledas med 0. Är formatsträngen ”00” ska numret inleda med 0 om numret är mindre än tio. Alla övriga värden på formatsträngen ska leda till att ett undantag av typen FormatException kastas.

Inga utskrifter till konsolfönstret får göras av någon av metoderna.

#### Operatorn ==

Avgör om två instanser av NumberDisplay är lika. Enkel att implementera då metoden Equals() finns att tillgå.

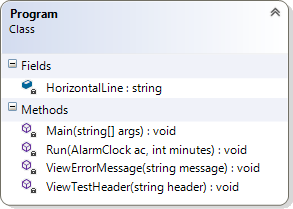
#### Operatorn !=

Avgör om två instanser av NumberDisplay är lika. Enkel att implementera då metoden Equals() finns att tillgå.

### Klassen Program

Klassen program ska innehålla koden som testar klassen AlarmClock.

Figur C.11. Förslag på medlemmar av klassen Program. visar ett förslag på medlemmar som kan användas för att skapa de åtta testerna som listas under rubriken ”Testlista” på sidan 20. Klassen Program behöver inte på något sätt följa förslaget som enbart ska ses som en rekommendation. Det finns bara ett krav som måste uppfyllas och det är att metoden Main() måste se till att de åtta deltesterna genomförs på avsett sätt.



Figur C.11. Förslag på medlemmar av klassen Program.

#### Metoden Main

Metoden ska instansiera objekt av klassen AlarmClock och testa konstruktorerna, egenskaperna och metoderna.

#### Metoden Run

Privat statisk metod som har två parametrar. Den första parametern är en referens till AlarmClock-objekt. Den andra parametern är antalet minuter som AlarmClock-objektet ska gå (vilket lämpligen görs genom att låta ett AlarmClock-objekt göra upprepade anrop av metoden TickTock()).

#### Metoden ViewErrorMessage

Privat statisk metoden som tar ett felmeddelande som argument och presenterar det.



Figur C.12. Exempel presentation av två felmeddelanden.

#### Metoden ViewTestHeader

Privat statisk metod som tar en sträng som argument och presenterar strängen.



Figur C.13. Exempel på ett tests rubrik inklusive horisontell linje ovan testrubriken.

## C-krav

|  |  |
| --- | --- |
|  | Klasserna AlarmClock, ClockDisplay och NumberDisplay ska implementeras enligt klassdiagrammet i Figur C.10. |
|  | Inga av medlemmarna i klasserna AlarmClock, ClockDisplay och NumberDisplay får skriva ut något i konsolfönstret. |
|  | Klasserna AlarmClock, ClockDisplay och NumberDisplay måste överskugga Equals(), GetHashCode() och ToString() samt överlagar operatorerna == och !=. |
|  | Då ett nytt AlarmClock-objekt instansieras ska undantag kastas av klassen om försök görs att initiera objektet med värden som inte ligger inom de slutna intervallen för timmar och minuter. |
|  | Klassen AlarmClock ska innehålla två aggregat i form av två privata fält av typen ClockDisplay[] Respektive ClockDisplay. |
|  | Klassen AlarmClock får inte innehålla någon kod som validerar parametrars värden. Validering ska ske genom aggregerade klasser. |
|  | Klassen AlarmClock ska ha två publika egenskaper för väckarklockans tid och alarmtid. |
|  | Ett anrop av metoden TickTock() ska leda till att väckarklockan går en minut, t.ex. från 9:42 till 9:43. |
|  | Metoden TickTock() ska returnera true om den nya tiden är lika med alarmtiden, annars false. |
|  | Metoden ToString() ska beskriva aktuellt AlarmClock-objekt i form av en sträng innehållande aktuell tid samt alarmtiden inom parenteser. |
|  | Då ett nytt ClockDisplay-objekt instansieras ska undantag kastas av klassen om försök görs att initiera objektet med värden som inte ligger inom de slutna intervallen för timmar och minuter. |
|  | Klassen ClockDisplay ska innehålla två aggregat i form av två privata fält av typen NumberDisplay. Det ena aggregatet ska representera timmar och dess maxvärde ska därför sättas till 23. Det andra aggregatet ska representera minuter varför dess maxvärde ska sättas till 59. |
|  | Klassen ClockDisplay får inte innehålla någon kod som validerar parametrars värden. Validering ska ske i aggregerad klass. |
|  | Ett anrop av metoden Increment() i klassen ClockDisplay ska leda till att NumberDisplay-objektet som representerar minuter går en minut, t.ex. från 9:42 till 9:43. Får detta NumberDisplay-objekt värdet 0 ska det andra NumberDisplay-objektet, som representerar timmar, öka sitt värde. |
|  | Metoden ToString() ska beskriva aktuellt ClockDisplay-objekt i form av en sträng innehållande aktuell tid. Tiden ska presenteras på formatet HH:mm, där minuterna ska inledas med 0 om minuterna är mindre än tio. Timmarna ska inte inledas med 0 om de är mindre än tio. |
|  | Då ett nytt NumberDisplay-objekt instansieras ska undantag kastas av klassen om försök görs att initiera objektet med värden som inte ligger inom det slutna intervallen mellan 0 och maxvärdet. |
|  | I klassen NumberDisplay ska fältet \_maxNumber kapslas in av egenskapen MaxNumber som validerar att värdet är större än 0. Är inte värdet inte större än 0 ska ett undantag av typen ArgumentException kastas. |
|  | I klassen NumberDisplay ska fältet \_number kapslas in av egenskapen Number som validerar att värdet är i det slutna intervallet mellan 0 och maxvärdet innan värdet tilldelas fältet \_number. Är inte värdet i det slutna intervallet ska ett undantag av typen ArgumentException kastas. |
|  | Ett anrop av metoden Increment() i klassen NumberDisplay ska leda till att egenskapen Number, och därmed fältet \_number, ökar sitt värde med 1. Om värdet skulle bli större än maxvärdet ska värdet istället sättas till 0. |
|  | Klassen NumberDisplay ska två överlagrade versioner av metoden ToString(). Den ena versionen ska överskugga ToString(), som ärvs från basklassen Object. Den andra versionen ska ha en sträng i parameterlistan som bestämmer formatet som textbeskrivningen av anropande NumberDisplay-objekt ska ha. Skickas formatsträngen ”0” eller ”G” med ska textbeskrivningen av numret inte inledas med 0 om numret är mindre än 10. Skickas formatsträngen ”00” med ska textbeskrivningen inledas med 0 om numret är mindre än tio. |
|  | Klassen Program måste implementeras så att deltesterna under rubriken ” Testlista” på sidan 20 körs då programmet exekveras.  Var och en av testerna måste lyckas vilket ska kunna verifieras genom lämpliga utskrifter i konsolfönstret. |

## Läsvärt

* Klasser
  + Essential C# 5.0, 209-220.
  + <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
* Åtkomstmodifierare (*”Access Modifiers”*)
  + Essential C# 5.0, 227-229.
  + <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
* Egenskaper
  + Essential C# 5.0, 229-235.
  + <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.
* Konstruktorer
  + Essential C# 5.0, 244-248, 250-253.
  + <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>
* *”Well-formed Types”*
  + Essential C# 5.0, 371-421.