## Programmeringsuppgift 1.1

Kenneth Sörling

[kennethsorling@mail.com](mailto:kennethsorling@mail.com)

Tel: 073-8914630

## Om Prioriteringsordningen:

Föreställningen att arbete obetingat går före sociala aktiviteter och att träning kommer i sista hand är en fråga om personlig prioritering, och det förefaller mig oklokt att kallt förutsätta att alla användare har denna inställning. Om de skulle ändra prioritering på sin livsföring, har vi bakbundit våra händer. Dessutom finns det tillfällen, som bröllop, begravning, etc. där en aktivitet från en typ av schema oaktat går före de flesta andra aktiviteter. Exempelvis har jag inte mött en förälder hittills som ansett att ett personalmöte går före att hämta dottern från dagis på eftermiddagen.

Icke desto mindre anger beskrivningen att vi ska prioritera arbete framför sociala aktiviteter och att träning kommer i sista hand. Så jag ska försöka tillgodose det kravet, samtidigt som jag lämnar dörren öppen för andra ordningar.

## Om uppgiften

Vi har att identifiera schemakonflikter där vi måste prioritera vissa aktiviteter och utesluta andra beroende på aktivitetstyp, och leverera ett schema som användaren har en chans att hinna med.

Arbete går i första hand, sociala aktiviteter såsom dejting, i andra hand, och träning i tredje hand. Vad avser träningen är vissa pass att prioritera före andra. Vi bör lämna modellen öppen för andra aktivitetstyper, och, anser jag, möjligheten att användaren omprioriterar dessa.

## Lösningen:

Vi inhämtar tre scheman från olika källor, och låtsas att de avser i tur och ordning Arbete, Fritid och träning. Vi låtsas att de kommer från olika källor utanför vår kontroll och gör informationen homogen med transformationer. I själva verket är de JSON-filer som läses in från disk.

Vi sorterar och prioriterar aktiviteterna, identifiera konflikter, och löser dessa genom att förkasta mindre viktiga aktiviteter. Vi tar i beaktande tidsmarginaler som behövs före och efter aktiviteten.

För att lättare tackla problemet, skapar vi en interim-modell, halvvägs mellan Model och ViewModel, vars enda syfte är att hjälpa oss lösa konflikter. Den omfattar ursprunglig information, plus kalkylerade fält som tidsåtgång och dylikt.

Uppgiften bör gå att lösa genom tvånivåsortering med LINQ, och utgallring av lågprioriterade aktiviteter som inte hinns med.

## Identifiering av konflikter

En aktivitet kan sägas krocka med en annan om följande gäller:

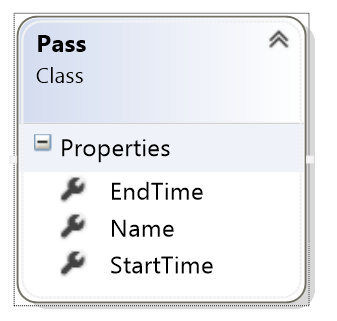
* Den börjar innan föregående aktivitet avslutats, eller
* Den börjar samtidigt som en annan aktivitet, eller
* Den börjar och slutar inom ramen för en annan aktivitet (omsluten tidsmässigt), eller
* En efterkommande aktivitet av högre prioritet börjar innan den aktuella aktiviteten avslutats.

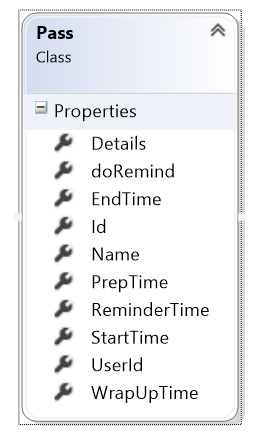
Konflikten löses genom att den lägre prioriterade aktiviteten plockas ur schemat, oavsett när den börjar och slutar.

I detta läge är det lämpligt att använda sig av en interim-modell, som omberäknar start- och slutttid för aktiviteten så att nödvändiga tidsmarginaler före och efteråt tas i beaktande. Det gör matematiken kring tider och tidsåtgång enklare. För enkelhets skull kombinerar jag interim- och ViewModell i samma klass.

## Modeller:

Den allra enklaste modellen av ett ”Pass” för att följa er nomenklatur, ser troligtvis ut så här (ursäkta den röriga fältordningen) :

Om ni inte har kontroll över formatet på alla de heterogena datakällorna för scheman, så är det förmodligen de enda termer olika versioner kan komma överens om. För att lösa programmeringsuppgiften, är det också den enda information vi egentligen behöver. **StartTime** och **Endtime** är **DateTime**-fält, innehållande exakt datum och klockslag då aktiviteten börjar och slutar. **Name** är ett textfält för att beskriva vad det egentligen handlar om, t.ex ”Spinning på SATS”. Annat, som tidsförbrukningen, kan vi beräkna. T.ex är ”Duration” något som vi kan beräkna som skillnaden mellan EndTime och StartTime.

Mera realistiskt är att entiteten kommer att se ut något i den här stilen:

Återigen, ursäkta fält-röran; jag kan inte få Visual Studio att låta bli att sortera medlemmarna alfabetiskt. Omvänt är det svårt att få Word att formatera källkden för klassen på ett någorlunda läsligt sätt.

Nämnas bör här, att vi inkluderat medlemmarna **PrepTime** och **WrapUpTime**, som anger tiden, i minuter, som vi behöver före och efter aktiviteten och som måste tas i beaktande. Saker som transporttid, tid för omklädning och dusch, etc.

Ett **Id**-fält är oundvikligt om man vill spara aktiviteten i egen datakälla. Nämnas bör att den inte avser Id:n från separata scheman från andra leverantörer. Dessa har föga mening för oss.

**doRemind** är ett enkelt ja/nej-fält som anger om vi vill ha förhandslarm. **ReminderTime** är tiden, i minuter, innan aktiviteten som användaren vill bli underrättad.

**UserId** är självklart kopplingen till ett användarkonto vars scheman vi administrerar. Jag har svårt att föreställa mig att vi gör allt detta jobb för en enda användares skull, så vi behöver en mekanism att skilja flera användare åt.

Många av dessa fält är irrelevanta för problemet jag fått att lösa, men jag tyckte att en realistisk modell borde innefatta några vanliga praktika såsom **Details**, för anteckningar runt en aktivitet, som t.ex. var, med vem, om vad, och varför.

Själva arbetet utförs dock i en klass kallad **PassVM**, som är förvillande lik ovanstående modellklass.

## Projektet

Jag skapar en enkel kontroller kring modellen **Pass** och Inför metoden **Consolidated**(), där allt händer.

Tre scheman inhämtas från fingerade datakällor med avseende på arbete, fritid och träning.

De omdanas och kombineras till en **List<PassVm>**.

Denna lista sorteras med LINQ efter Starttid och prioritet(fallande), så att viktigare aktiviteter kommer före lägre prioriterade samtidiga aktiviteter.

Listan gallras efter de kriterier som anges ovan.

Slutligen presenteras resultatet för användaren.