**Media Shop**

Av: Fabian Fröding

# 1. Antaganden

Ange alla antaganden ni gjort under projektets gång. Antaganden är sådant som inte står i labbspecen men ändå måste bestämmas, och inte saker som "användaren har datorvana". Dessa bör skrivas upp kontinuerligt eftersom de annars lätt glöms bort. Exempelvis om specen anger att data ska sparas till fil så kan ett antagande vara hur datan sparas (format på text, binärt o.s.v...)

* Varunummer representeras av fältet ”Id”.

# 2. Översikt

En beskrivning av programmet från ett användarperspektiv. Beskriv vad programmet kan göra och hur det används. Exempelvis för ett spel bör dess regler beskrivas.

## Exempel:

Programmet består av ett fönster som visar den nuvarande månaden samt har en knapp för att byta till nästa månad. Programmet justerar automatiskt månad vid årsbyte.

(Mycket kortfattat exempel p.g.a. mycket enkelt program.)

# 3. Detaljerad beskrivning

En beskrivning av programmet från ett implementationsperspektiv (kod). Beskriv programmets struktur och funktionalitet. Vilka klasser finns, vad representerar de, vad innehåller de (data och funktioner/metoder), och hur relaterar de till varandra. Beskriv kortfattat vad variabler lagrar för data samt vad funktionerna gör. Använd diagram för att illustrera programmets struktur (klassdiagram och eventuellt flödesdiagram eller liknande).

Programmet använder sig av MVC-arkitekturen.

* Models består av domän-enheter som t.ex. Product.
* Views består av Windows Forms.
* Controller-lagret är uppdelat i tre ytterligare lager:
  + Controller: Hanterar vilka vyer som ska visas baserat på vilka enheter som efterfrågas.
  + Service: Hanterar logik mellan Controller och Repository.
  + Repository: Hanterar lagring och hämtning av data.

## Exempel:

Programmet är uppbyggt enligt MVC mönstret och består av klasserna Model, View och Controller (se klassdiagram).

[KLASSDIAGRAM HÄR]

* Grundläggande funktioner som getters/setters och liknande är inte inkluderat i klassdiagrammet. Detta för att hålla diagrammet minimalistiskt och begripligt.
* Metoder som ärvs från parent-klasser eller interfaces är inte representerade i den ärvande klassen. Detta för att reduca redundans.
* Klasserna Product och User är POCOs (Plain Old C-Object), som är klasser endast håller information.
* Somliga metoder som återger en bool gör detta för att indikera om operationen lyckades eller inte. T.ex. om metoden ”Add(Product)” återger *true* indikerar det att tiläget av produkten lyckades medans *false* hade indikerat att tilläget misslyckades. Denna sorts funtionalitet kan komma till nytta senare i utvecklingen.

### Klass Model:

En klass för hanteringen av en kalenders logik. Lagrar den nuvarande månaden och tillhandahåller metoder för att hämta och ändra månad.

Variabler:

* int current\_month – Ett heltal som representerar den nuvarande månaden

Metoder:

* public Model(int month) – Konstruktor. Sätter den nuvarande månaden i programmet till *month*. Giltiga värden: 1 <= month <= 12
* public int getMonth() – Returnerar den nuvarande månaden.
* public void nextMonth() – Byter till nästa månad (med hänsyn till årsbyte).

### Klass View:

...

o.s.v.

# 4. Problem

Beskriv problem som uppstått under projektets gång. Beskriv hur ni har löst dem och vad ni lärt er av dem. Anteckna kontinuerligt eftersom det kan vara svårt att komma ihåg varför man gjorde på ett visst sätt. Problem avser övergripande svårigheter, inte mindre misstag i koden.

# 5. Sammanfattning

Sammanfatta ert projekt med hur det har gått och vad ni har lärt er. Beskriv alternativa lösningar samt fördelar och/eller nackdelar med dessa och varför ni valt den lösning ni har. Ta också med ungefärlig tidsåtgång för de olika momenten.

# 6. Referenslista

Författare, titel, förlag och år. Endast nödvändig om du använt andra böcker än kurslitteraturen.

# Bilagor

Innehållet är beroende av vilken kurs och vilken labb som utförs men innehåller ofta programkod och diagram.