

TDDD49/725G66 - Programmering C# och .NET

LABORATIONER

Linköpings universitet - 2016



Inledning

Temat för denna laborationsserie är brädspel. En lämplig storlek är schack eller poker - tic-tac-toe är för litet. Du kan implementera ett existerande brädspel med regler och allt som en övning i kursen - men förstås inte publicera sådant copyright-skyddat material.

Tänk på att implementera ett lagom stort brädspel. Kursen är på 107 timmars eget arbete. Om du vill göra ett för stort spel - lämna då in något tekniskt och volymmässigt tillräckligt och jobba vidare efter kursen. Om du vill göra ett väldigt, väldigt litet spel - gör inte det för då kan du inte bli godkänd.

Tänk också på att labbarna hänger ihop. När du löser en labb, gör det med de andra labbarna i åtanke. Detta gäller skärskilt den första labben som också inkluderar mycket planering.



Spelet ska ha 2 eller flera spelare. Du KAN också använda AI, INTE obligatoriskt, för att göra det möjligt för spelaren att spela mot datorn.

Några exemplar...

Backgammon

<http://games.aarp.org/games/backgammon/backgammon.aspx>

Schack

<http://www.mdgx.com/chess.htm>

Poker

<http://www.thepokerpractice.com/>

Ultimate Tic Tac Toe

<http://ultimatetictactoe.creativitygames.net/>



Spelet kan implementeras som en desktop-app (i WPF), webbapp eller mobilapplikation. Det är upp till dig att bestämma vilket.

LABB ①


SPELETS MOTOR

Syfte

I slutet av labben förväntas du ha avslutat spelmotorn, en av de tre befintliga lagren i programmet. Spelmotorn ska åtminstone ha tre komponenter som är ansvariga för olika uppgifter.

Krav

- Följande ska finnas:
 - Datatyper som du behöver
 - Regelmotor som validerar förändringar av spelets datatyper (tillåtet, icke tillåtet)
 - Spellogik
 - Datorspelare med artificiell intelligens (frivilligt)
- Spellogik, datatyper och regelmotorn ska vara väl separerade. Det finns många olika sätt. Hitta ett som ser till att man med minimal insats kan byta ut någon del.
- Ett sista krav här är att spelmotorn kan hantera flera spelare och det måste vara irrelevant till spelmotorn huruvida ett spelare är faktiskt en AI eller en mänsklig spelare (via GUI).
- Felhantering

 **Läs de övergripande kraven (finns i slutet av dokumentet) innan du börjar med labben. Det är jätteviktigt att alla labbar följer dessa.**

Redovisning

Läs Redovisningsdelen vidare i dokumentet.

LABB 2

GRAPHICAL USER INTERFACE

Syfte

I slutet av labben förväntas du ha konstruerat ett grafiskt användargränssnitt, ett av de tre befintliga lagren i programmet. GUI har ansvar för att kommunicera med användaren, hantera inmatning, och visa återkoppling och resultat.

Krav

- Lämplig återkoppling till användare: Användaren ska få återkoppling för misslyckande eller framgång
- Logiska och lämpliga GUI-komponenter ska användas.
- C # Ritning Kapacitet ska användas eller någon grafisk ersättning.
- Om spelet startas upp senare ska spelet återta det tillstånd det hade senast.
- Spelet ska ha ett tydligt start och slut, och resultatet ska presenteras tydligt för användarna.
- Felhantering.

Redovisning

Läs Redovisningsdelen vidare i dokumentet.



Kolla de övergripande kraven (sist i dokumentet) innan du börjar med labben. Det är jätteviktigt att alla labbar följer dem.



Vissa krav ska involvera flera lagren för att implementeras. Till exempel, "Om spelet startas upp senare ska spelet återta det tillstånd det hade senast."

LABB 3

DATA LAGRING OCH LINQ

Syfte

Detta lager handlar om att lagra information om spelet i ett kvarstående sätt.

Krav

- Du kan välja att hantera lagring med en databas, XML, JSON ... eller något annat format men det ska gå att tolka utanför C# och inte direkt bygga på C#-serialisering.
- LINQ ska i alla lägen användas för att jobba med din lagring.
- Implementera stöd för att lagra spelets tillstånd på ett oberoende format. Detta ska ske hela tiden, d.v.s. en ändring i spelts tillstånd medför att det lagrade datat uppdateras.
- Spelet ska hela tiden hålla ett och samma tillstånd. Spelet behöver dock inte klara att hantera mer än en spelomgång samtidigt.
- Felhantering

Redovisning

Läs Redovisningsdelen vidare i dokumentet.



Kolla de övergripande kraven (sist i dokumentet) innan du börjar med labben. Det är jätteviktigt att alla labbar följer dem.

Övergripande krav

Ditt spel ska uppfylla följande krav:

- Ha en genomtänkt **3-tier arkitektur**.
[https://msdn.microsoft.com/en-us/library/windows/desktop/ms685068\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms685068(v=vs.85).aspx)
- **Dependency Injection**.
<https://docs.asp.net/en/latest/fundamentals/dependency-injection.html#what-is-dependency-injection>
- **Program to Interface, Loosely coupled layers**.
<http://www.codeproject.com/Articles/702246/Program-to-Interface-not-Implementation-Beginners>
- **Felhantering: Bra användning av "custom designed exception classes"**.
[https://msdn.microsoft.com/en-us/library/87cdya3t\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/87cdya3t(v=vs.110).aspx)
- **Att använda ett "Uniform approach" för att överföra data mellan lagren**.
<http://www.codeproject.com/Articles/493389/Four-ways-of-passing-data-between-layers>

Du ska vid redovisningstillfället kunna motivera och förklara dina (tekniska) designval - varför klassindelningen ser ut som den gör, varför delarna av din applikation interagerar med varandra på de sätt de gör med mera.

Litteratur

Det finns ingen obligatorisk litteratur i kursen. Ett förslag på bra bok är senaste upplagan av Deitels C# for programmers-bok.

<http://www.deitel.com/Books/C/C6forProgrammersSixthEdition/tabid/3682/Default.aspx>

Notera att datorerna i labbsalarna klarar C# 5, så nya C# 6-features stöds inte.

Redovisning

Labbar ska redovisas separat eller i klump, personligen av samtliga (1 eller 2) studenter som lämnar in labben. Labbar ska även lämnas in elektroniskt i slutet. Inlämningssätt och redovisningstillfällen tillkännages på kurshemsidan, och av handledare.



Laborationerna utförs i par, eller undantagsvis själv (om handledare godkänner det - tala med din handledare!).

Handledning

Kolla [timeedit](#) för att få veta vilka [assistenter](#) ska vara med i vilket labbtillfälle. [Camedin](#), ett elektroniskt kö system, ska användas för att assistera studenterna under labbtillfällena. Använd följande länken för att få hjälp när du befinner dig i de bokade labbsalarna:

<http://www.camedin.com/live/34b402d0054a4ca4a6125b31e3f5026c>

Lösenord: TDDD49

Användbara länkar

Create a Simple Application with Visual C#

<https://msdn.microsoft.com/en-us/library/jj153219.aspx>

Walkthrough: Creating a Web Application Using Visual C# or Visual Basic

[https://msdn.microsoft.com/en-us/library/aa302124\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa302124(v=vs.71).aspx)

.NET Framework

<https://www.microsoft.com/net>

C# Programming Guide

<https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx>

Introduction to the C# Language and the .NET Framework

<https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx>

Visual Studio Downloads

<https://www.visualstudio.com/downloads/>

Installing Visual Studio

<https://msdn.microsoft.com/en-us/library/e2h7fzkw.aspx>