Development Assessment Test - Intelligent Systems

Development Assessment Test - Intelligent Systems

Opgave:

- Programming
  - 1.1 Thread safe class and instance variable access
  - 1.2 Network socket programming
  - 1.3 Persistence and database programming
  - 1.4 Library system - simple system including UI
  - 1.5 Unit tests
- 2 System design and analysis
  - 2.1 UML class diagram
  - 2.2 UML State diagram (state-event diagram)
  - 2.3 Extend UML state diagram
  - 2.4 UML Sequence diagram
- 3 Programming concepts
  - 3.1 Object vs. class
  - 3.2 Global variables
  - 3.3 Thread safe
  - 3.4 Encapsulation
  - 3.5 Asynchronous vs. synchronous programming
- 4 Design patterns
  - 4.1 Observer pattern
  - 4.2 Singleton pattern
  - 4.3 Your favorite pattern
- 5 Algorithmic Task

Opgave:

1. Programming

Til disse kodnings opgaver brugte jeg Microsoft Visual Studio 2019 til at programmere i, og jeg har brugt Microsoft SQL Server Management Studio til at oprette databaser i.
Alt kode til følgende opgaver kan findes via følgende link:
https://github.com/leonwrathMA/DeveloperAssesmentTest

1.1 Thread safe class and instance variable access

Denne opgave tog ca. 1 time.

1.2 Network socket programming

Denne opgave tog ca. 2 timer.

1.3 Persistence and database programming

Denne opgave tog ca. 3 timer.

For at oprette den database der er blevet brugt her, så skal følgende query køres:

```
USE [IntelligentSystemsTest]
GO

/****** Object:  Table [dbo].[TestTable]    Script Date: 24/01/2021 17:17:38 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[TestTable](
	[_id] [int] IDENTITY(1,1) NOT NULL,
	[Integer] [int] NULL,
	[String] [nvarchar](max) NULL,
	[DateTime] [datetime] NULL,
 CONSTRAINT [PK_TestTable] PRIMARY KEY CLUSTERED
(
	[_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

1.4 Library system - simple system including UI

Denne opgave tog ca. 4 timer.

For at oprette den database der er blevet brugt her, så skal følgende queries køres:

QUERY 1:

```
USE [IntelligentSystemsLibraryTest]
GO

/****** Object:  Table [dbo].[BookInfoTable]    Script Date: 24/01/2021 17:19:07 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[BookInfoTable](
	[title] [nvarchar](max) NULL,
	[author] [nvarchar](max) NULL,
	[ISBNNumber] [int] NOT NULL,
 CONSTRAINT [PK_BookInfoTable] PRIMARY KEY CLUSTERED
(
	[ISBNNumber] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

QUERY 2:

```
USE [IntelligentSystemsLibraryTest]
GO

/****** Object:  Table [dbo].[BookTable]    Script Date: 24/01/2021 17:19:36 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[BookTable](
	[_id] [int] IDENTITY(1,1) NOT NULL,
	[LibraryUserNumber] [nchar](10) NULL,
	[ISBNNumber] [int] NULL,
 CONSTRAINT [PK_BookTable] PRIMARY KEY CLUSTERED
(
	[_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Query 3:

```
USE [IntelligentSystemsLibraryTest]
GO

/****** Object:  Table [dbo].[UserTable]    Script Date: 24/01/2021 17:19:49 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[UserTable](
	[name] [nvarchar](max) NULL,
	[libUserNumber] [int] NOT NULL,
 CONSTRAINT [PK_UserTable] PRIMARY KEY CLUSTERED
(
	[libUserNumber] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

1.5 Unit tests

Denne opgave tog ca. 1 timer.

2 System design and analysis

Følgende diagram opgaver blev løst via følgende website:
https://online.visual-paradigm.com/
Alle diagrammer kan findes under linket:
https://github.com/leonwrathMA/DeveloperAssesmentTest

2.1 UML class diagram

Denne opgave blev færdig på under en time, jeg ved stort set intet om biler, så vidste ikke lige hvad jeg burde/kunne smide på.

2.2 UML State diagram (state-event diagram)

Denne opgave tog mig ca. 30min

2.3 Extend UML state diagram

Denne opgave tog mig ca. 30min

2.4 UML Sequence diagram

Denne opgave tog mig ca. 1 time

3 Programming concepts

3.1 Object vs. class

En klasse er en skabelon for et objekt.
Et objekt er en klasse der er blevet initialiseret.

3.2 Global variables

Globale variabler kan påvirkes fra alle steder i programmet. Dette kan selvfølgelig være en positiv ting, men det kan også være negativt. Hvis den globale variable skal skiftes ud, eller slettes på et senere tidspunkt, så kan det skabe problemer alle steder i programmet, hvor hvis variablen er mere begrænset, så vil det den kan ødelægge hvis den mangler, også være mere begrænset.

3.3 Thread safe

At være “thread safe” er når flere forskellige tråde af programmet ikke skaber fejl ved at skrive/læse a den samme ting på samme tid, et eksempel på dette kunne være at flere tråde prøver at skrive til den samme variable, og det kan godt skabe problemer, da de alle gerne vil skrive deres egen værdi i den.
Der er flere måder at være “thread safe” en af disse måder er vist i opgave 1.1 hvor man låser et objekt/variable så når først en tråd tager fat i den, så låser den objektet/variablen, og hvis der så er flere tråde der gerne vil skrive til samme fil, så bliver de sat til at vente indtil at den lås bliver frigivet.
En anden måde at være “thread safe” på er at lave en klasse immutable, dette gør at den ikke kan ændres efter klassen er oprettet, dog skal man huske at bare fordi en klasse er immutable, så betyder det ikke at dens referencer er.

3.4 Encapsulation

Encapsulating betyder at div. Variabler i en klasse ikke kan tilgås uden at bruge klassens metoder.
Dette kan gøres ved at gøre alle variabler i en klasse private.

3.5 Asynchronous vs. synchronous programming

Synkron: programmet køre 1 tråd, og løser opgaver 1 af gangen.

Asynkron: programmet køre X antal tråde, og hver tråd løser deres egne opgaver.

4 Design patterns

4.1 Observer pattern

Et observer pattern er når en opdatering af et objekt skal informere andre objekter om denne ændring.
Et eksempel på dette kan være youtube og deres subscription, hvis en person klikker på subscribe så melder den persons bruger sig ind som en observer til den kanal. Når så den kanal opdatere noget, så vil alle observerne, som i dette tilfælde være deres subscribers, blive informeret omkring dette.

4.2 Singleton pattern

Singleton pattern er når en klasse kun kan laves 1 gang i programmet. Jeg har ingen ide om hvornår man burde bruge sådan en klasse.

4.3 Your favorite pattern

Jeg har aldrig rigtigt tænkt over mit yndlings pattern, så jeg må nok sige at jeg ikke har et.

5 Algorithmic Task

Jeg har aldrig arbejdet med checksums, så jeg har ikke den store forstand på hvad der er bedst at gøre for at gemme dem, ville tro at de burde få deres egen database, så når man skal checke en ny værdi imod alle checksums, så belaster det ikke resten af systemet, da jeg godt kan forestille mig at på noget så som et bibliotek godt kan være mange nye bøger, og hvis de alle skal checkes, så kunne det nok godt tage noget tid.