Tekniset tehtävät (aika yhteensä ~**4h 45min**)

Tehtävä 1: Koodikatselmointi (**Aika ~60min**, sis. Vastausten kirjoittamisen Wordiin.)

Turhat “*using*” -rivit kannattaa poistaa. Ei varsinaisesti paranna suorituskykyä, mutta koodin lukeminen on helpompaa, kun näkee suoraan mitä kirjastoja on käytetty.

Seuraavat rivit voi siis huoletta poistaa:

* *using System;* (tämä on kahteen kertaan, molemmat voi poistaa)
* *using System.Text;*
* *using System.Linq;*
* *using System.Threading.Tasks;*
* *using System.Threading;*

Luokan nimi on suomeksi **HaeRivit**, mutta muuten koodattu englanniksi. Ei mikään suuri ongelma, mutta luultavasti tällekin on jonkinlainen HiQ:n sisäinen standardi olemassa. Myöskin luokan (ja funktioiden) nimet olisi hyvä nimetä niin, että jonkinlainen yhteneväisyys säilyy. Vrt:  
*class* *haeRivit* => *class* *HaeRivit*  
*void* *GetRows* => *void GetRows*

*void* *writerows* => *void WriteRows*

Riippuen aina hieman tarkoituksesta, **ExecuteReader()** -funktiolle kanattaa antaa parametriksi *System.Data.CommandBehavior.CloseConnection* (tätä varten voisi myös lisätä uuden kirjaston *using System.Data;* ja parametriksi riittää *CommandBehavior.CloseConnection*. Vaihtoehtoisesti yhteyden voi sulkea käsin lisäämällä *con.Close();.* Vielä parempi vaihtoehto olisi lisätä koko roska using() -lausekkeen sisään. Karkea esimerkki alla:

*void GetRows(string param)*

*{*

*using (SqlConnection con = new SqlConnection(connection\_string))*

*{*

*con.Open();*

*using (SqlCommand command = new SqlCommand("SELECT \* FROM " +  
param, con))*

*using (SqlDataReader reader = command.ExecuteReader())*

*{*

*while (reader.Read())*

*{*

*result.Add(string.Format("{0} {1} {2}", reader.GetInt32(0),*

*reader.GetString(1), reader.GetString(2)));*

*}*

*}*

*}*

*}*

**WriteRows()** -funktiossa onkin hieman useampi ongelma.

Ensinnäkin, nyt **StreamWriter** writer luonti on *while*-loopin sisällä. Tämä muodostaa joka kerta uuden **StreamWriter** -objektin. Tämä kannattaa siirtää heti *int count =1;* -rivin jälkeen. Lisäksi *var writer = new StreamWriter(…);* voisi suosiolla muuttaa

* *StreamWriter writer = new StreamWriter(…)*
* Halutaanko **WriteRows()** -funktiossa aina ylikirjoittaa olemassaoleva tiedosto, jos sellainen löytyy? Tällä hetkellä näin tapahtuu. Mikäli ei haluta, StreamWriterin konstruktoriin voi syöttää toiseksi parametriksi arvon True. Tämä riippuu tarkoituksesta. Todennäköisesti halutaan kuitenkin ylikirjottaa, sillä halutaan varmaankin palauttaa vain uusin tulosjoukko.

**WriteRows()** -funktio saa parametriksi *string path*, jota ei kuitenkaan käytetä. Uskoisin, että tätä parametria haluttaisiin käyttää määrittämään tekstitiedoston luontipolku. Tämä on kuitenkin nyt kovakoodattu **StreamWriteriin**, eikä tuota polkua voi siis muuttaa. Tässä varmaan olisi tarkoitus syöttää **path** -parametri **StreamWriterin** konstruktoriin.

*while*-loopissa on myös muitakin ongelmia. Ensinnäkinn **count** -muuttujan arvo ei muutu, joten looppi pyörii ikuisesti. Tähän siis pitää lisätä count++;. w*hile* -loopin ehdossa *count < result.Count* ei tule koskaan tulostamaan kaikkia listan elementtejä, kun count alkaa arvosta 1. Ensimmäistä elementtiä ei siis tulosteta, koska listan alkioiden indeksointi alkaa nollasta.

Lopuksi koko StreamWriterin luonnissa ja käytössä on sama ongelma kuin ylempänä **SqlConnection**, **SqlCommand** ja **SqlDataReader** -objektien kanssa. Tuo koko funktio kannattaisi siis mahdollisesti muuttaa muotoon:

*public void WriteRows(List<string> result, string path)*

*{*

*int count = 0;*

*using (StreamWriter writer = new StreamWriter(path, true))*

*{*

*while (count < result.Count)*

*{*

*writer.WriteLine(result[count]);*

*count++;*

*}*

*}*

*}*

Halutaanko luokan muuttujat (**connection\_string**, **param** ja **result** pitää julkisina niin, että millä tahansa koodilla on mahdollisuus näitä muokata?

Tehtävä 2: XML ja ohjelmointi

1. XSD-tiedosto sähköpostin liitteenä **(~45min**)
2. Pseudokoodi perättäistiedoston luontiin (**~1h**):

**Luokka Parser**

{

Globaalit muuttujat järjestysnumero (int), nykyisen\_elementin\_nimi, vanhemman\_nimi, lista johon lisätään tiedostoon kirjoitettavat rivit;

**Suoritettava\_funktio\_main()**

{

Lataa xml-tiedosto (System.Xml.XmlDocument);

Kutsu rekursiivista funktiota, jolla käydään elementit yksitellen läpi (parametreina xml-tiedoston ChildNodet ja tyhjä merkkijono);

While(dokumentilla on NextSibling)  
{

Kutsu samaa rekursiivista funktiota(parametreina dokumentin NextSiblingin Child Nodet ja tyhjä merkkijono)

}

Looppaa lista läpi, ja kirjoita tiedostoon

}

**Rekursiivinen\_funktio(nodes,ylimmän\_tason\_vanhimman\_nimi)**

{

For each node in nodes

{

Jos noden nimi == “name”

{

Lisää listaan (lukumäärä + ; + elementin\_teksti + ; + ylimmän\_tason\_vanhimman\_nimi + ; + vanhemman\_nimi)

Jos ylimmän\_tason\_vanhin ==””, ota se talteen

Jos nodella on lapsia, ota nykyisen noden nimi talteen vanhemman\_nimi

Lisää järjestysnumeroa +1

}

Kutsu tätä rekursiivista funktiota uudestaan(parametrit tämän noden childnodet ja ylimmän\_tason\_vanhin)

}

}

}

1. Pseudokoodi item -elementin lisäämiseen (karkeammin kuin b) -kohta) **(~30min**)

using System.Xml.Linq;

Lataa xml-tiedosto

Käy rekursiivisesti elementit läpi, kunnes löydetään elementti, jonka nimi on name ja arvo (ilmeisesti parametrinä syötetyn) merkkijonon arvo.

Jos nykyisellä elementillä on jo lapsia

Luo elementti (item, elementti (name, syötettävän itemin nimi)

Else

Luo elementti(children, elementti(item, elementti(name, syötettävän itemin nimi)

Lopeta rekursiivinen looppi

Tallenna muokattu xml-tiedosto vanhan päälle.

Tehtävä 3: SQL ja OpenAPI **(yht ~90min**)

1. YAML-tiedosto liitteenä.
2. Tarvittavat kyselyt:
   1. GET / event

SELECT t.\*,  
h.Nimi

FROM Tapahtumat t JOIN Henkilo h  
ON t.HenkiloID = h.HenkiloID  
WHERE t.Tyyppi = @tapahtumaTyyppi  
OR t.Aika = @tapahtumaAika

* 1. GET /person

SELECT h.\*,   
t.TapahtumaID

FROM Henkilo h JOIN Tapahtumat t

ON t.HenkiloID = h.HenkiloID

WHERE h.Nimi = @henkiloNimi

AND t.Tyyppi =@tapahtumaTyyppi

* 1. POST /event

INSERT INTO Tapahtumat

VALUES (@tapahtumaID, @henkiloID, @tapahtumaTyyppi, @tapahtumaAika)

* 1. DELETE /event

DELETE FROM Tapahtumat t

WHERE t.TapahtumaID=@tapahtumaID

AND t.HenkiloID = @henkiloID

1. **-**
2. Muuttaisin Tapahtumat -taulun Primary Key constraintin sisältämään sarakkeet TapahtumaID ja HenkiloID (indeksiin 2 saraketta)
3. Lisäisin indeksit:
   1. Tapahtumat -tauluu NONCLUSTERED indexin (Tyyppi, Aika) (toki tässä voidaan pohtia voiko samalla sekunnin tuhannesosalla (datetime) tulla 2 samantyyppistä tapahtumaa. Kaipa se on mahdollista, jos ei, indeksi voi myös olla clustered..)
   2. Henkilo -tauluun NONCLUSTERED indexin Nimi.