

Sovelto

Tiedostoharjoituksia

Hakemistojen ja tiedostojen käsittely

Hakemistot ja tiedostot 0

- Aloita uusi projekti (Console Application) ja nimeksi sopii vaikka RiviLkm
- Komentoriviparametrina lue Main-metodissa käsiteltävä hakemisto
`c:\work\viikko1\ke>RiviLkm c:\work\viikko1\program.cs`
- Tulosta määritellyn tiedoston rivien lukumäärä
- Testaa että toimii oikein
- Lisää toiminnallisuutta niin että sovellus laskee myös montako tyhjää riviä on tiedostossa
- Lisätehtävät:
 - Jos tiedosto on c#-lähdekooditiedosto niin laske myös kommenttirien lukumäärä
 - Huomaa että kommentoinnin voin tehdä myös käyttämällä merkkejä `/* */`

Hakemistot ja tiedostot 1

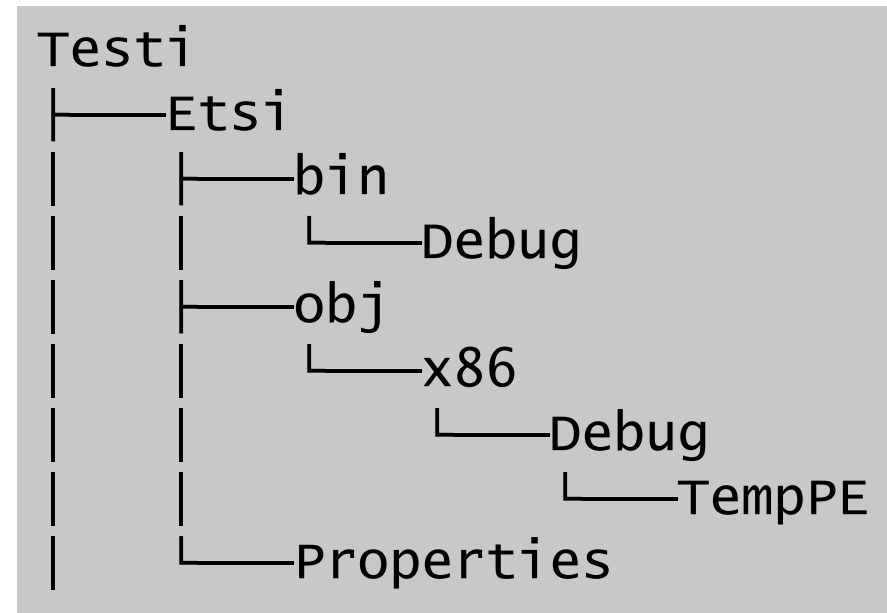
- Aloita uusi projekti (Console Application) ja nimeksi sopii vaikka TiedostoHaku
- Komentoriviparametrina lue Main-metodissa käsiteltävä hakemisto
`c:\work>TiedostoHaku c:\temp`
- Tulosta määritellyn hakemisto kaikki tiedostot (nimi ja koko)
- Jos komentoriviparametreja on kaksi niin toinen olkoon tiedostopääte jonka mukaiset tiedostot tulostetaan konsolille (nimi, koko, pvm)
- Jos komentoriviparametreja on kolme niin silloin kolmas parametri on teksti mitä etsit hakemiston tiedostoista (alihakemistoinen!). Luet siis tiedoston sisällön ja tarkistat löytyykö annettu merkkijono. Tulostukseen ne tiedostot (nimet!) mistä merkkijono löytyy.
`c:\work> TiedostoHaku c:\temp cs double`
- Lisätehtävät:
 - tulosta myös positio mistä ensimmäinen löydös sijaitsee tiedoston sisällä.
 - miten voisit vaikuttaa tulostukseen, siis mitä tulostetaan?

Hakemistot ja tiedostot 2

- Tee uusi konsolisovellus ja nimeksi Arkistoi
- Sovelluksen pitäisi 'arkistoida' komentoriviparametrina annetun hakemiston kaikki .cs-päätteiset tiedostot
- Arkistointihakemisto on käyttäjän profiilissa Arkisto-hakemisto. Jos hakemisto puuttuu sovelluksen pitäisi tehdä kyseinen hakemisto. Vaihtoehtona olkoon että arkistohakemisto on kovakoodattu c:\temp\arkisto
- Toimintalogiikka:
 - kopioi arkistoitavan hakemiston kaikki .cs-tiedostot arkistohakemistoon
 - jos saman niminen hakemisto on jo olemassa niin lisää kopioitavan tiedoston nimeen järjestysnumero , esimerkiksi Koodi.cs tiedosto on arkistossa Koodi.cs, Koodi.1.cs, Koodi.2.cs jne
- Vinkkejä: Path-luokan avulla onnistuu helpoiten tiedostonimien käsittely, File.Exists – metodilla saat selville onko tiedosto olemassa ja File.Copy suorittaa kopioinnin
- Testaa

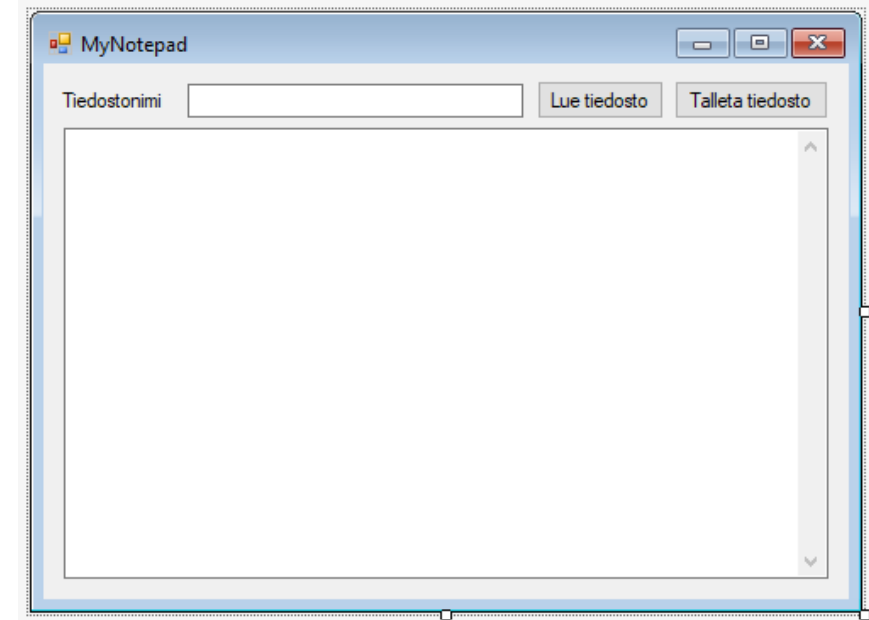
Lisäharjoitus (nopeimmille)

- Tee uusi luokka nimeltä Hakemistorakenne (joko edelliseen projektiin tai tee uusi projekti, miten haluat tätä testata)
- Lisää luokkaan konstruktori ja sille parametri string hakemisto
- Lisää readonly-ominaisuudet HakemistoLkm, TiedostoLkm ja Koko (kaikki long-tyyppisiä), nimet varmaankin jo kertovat mitä ne palauttavat ja Koko on sitten kaikkien tiedostojen yhteenlaskettu koko konstruktorille annetussa hakemistopuussa
- Tee vielä metodi Tulostarakenne() joka tulostaa hakemistorakenteen ilman tiedostoja konsolille, esimerkki vieressä. HUOM: tulosta soveltaen viivat, älä käytä paljon aikaa näiden esimerkin merkkien etsimiseen.
- Luultavasti tarvitset rekursiivisen algoritmin



MyNotepad (ylimääräinen lisäharjoitus)

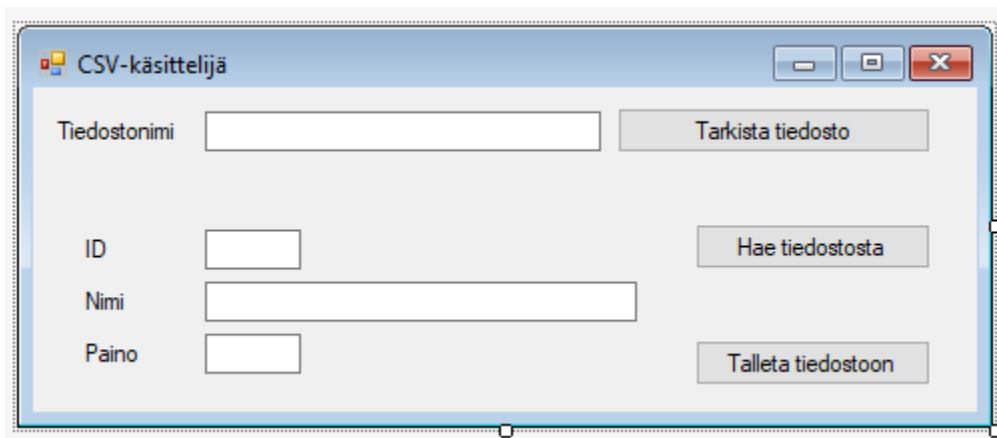
- Tee Windows Forms App –projekti, nimi MyNotepad
- Lisää lomakkeelle Label kaksi TextBox:ia ja kaksi Button-kontrollia. Alempi TextBox, MultiLine == true, ScrollBars == Both ja lisäksi voit kokeilla mitä Anchor-ominaisuus tekee
- Toiminto: kirjoita tiedostonimi ylempään TextBox:iin, avaa ja lue tiedoston sisältö ja aseta se alemman TextBox:in Text-ominaisuudeksi. Samalla lisää ikkunan otsikkopalkkiin (Caption) lisää tiedoston nimi ilman polkua (MyNotepad – demo.txt)
- Muokkaa tekstiä
- Kirjoita takaisin tiedostoon muokattu teksti Talleta tiedosto buttonilla
- Luku ja kirjoitus StreamReader/Writer –luokilla tai onnistuu myös File ja FileInfo-luokkienkin avulla
- Jos tiedostoa ei ole olemassa niin mieti mitä siinä tilanteessa sovelluksen pitäisi tehdä?



CSVKäsittelijä

- CSV-tiedosto on määrämuotoinen tapa tallettaa ja siirtää tietoa, lyhenne tulee sanoista Comma Separated Values mutta suomalainen versio käyttää puolipistettä erottimena
- Tee uusi projekti jolla voit käsitellä oheisen esimerkin mukaisia CSV-tiedostoja
- Tehdään konsolisovelluksena, toki näihin tämän tyyppisiin harjoituksiin sopisi ikkunoituva UI ihan hyvin (Windows Forms tai WPF). WinForms ja WPF eivät kuulu varsinaisesti kurssin sisältöön joten niitä käytetään harjoituksissa todella vähän tai ei ollenkaan. Saa toki käyttää ja opetella myös nämä käyttöliittymätekniikat

```
Id;Nimi;Paino  
1001;Hiirimatto;10  
1002;Laturi;560  
2011;Jatkojohto;1200
```



The screenshot shows a Windows Forms application window titled "CSV-käsittelijä". The window contains a form with the following elements:

- A text box labeled "Tiedostonimi" (Filename) with a "Tarkista tiedosto" (Check file) button next to it.
- A text box labeled "ID" with a "Hae tiedostosta" (Search from file) button next to it.
- A text box labeled "Nimi" (Name) with a "Talleta tiedostoon" (Save to file) button next to it.
- A text box labeled "Paino" (Weight) with a "Talleta tiedostoon" (Save to file) button next to it.

CSVKäsittelijä - toiminnot

- Tarkista tiedosto
 - tarkista onko tiedosto olemassa ja että se on sopiva CSV-tiedosto, lue ensimmäinen rivi ja tarkista että siellä on kolme saraketta joiden nimet ovat id, nimi ja paino (muista tehdä niin että kirjaimien koolla ei ole väliä)
 - jos tiedosto puuttuu tai ei ole oikean muotoinen niin anna virheilmoitus `MessageBox.Show`-metodilla
- Hae tiedostosta
 - Id-kontrolliin kirjoitetaan ensin ID-numero ja sitten tarkistetaan löytyykö tuolla ID:llä riviä tiedostosta, jos löytyy niin sen tiedot tuodaan käyttöliittymään
- Talleta tiedostoon
 - jos tiedostossa on id:n mukainen rivi niin päivitetään sen tiedot
 - muutoin kirjoitetaan uusi rivi tiedoston loppuun ja sisältöhän on se mitä käyttöliittymässä on kirjoitettuna
- Tämä on jo hieman vaativampi tehtävä

CSVKäsittelijä – Vinkit

- Tee ensiksi luokka `class Tuote { }` jossa on ominaisuudet `Id (int)` , `Nimi (string)` ja `Paino (int)` ja vielä sopiva konstruktori (myös oletuskonstruktori)
- Jos tiedosto on ok, niin luo `List<Tuote>`, avaa tiedosto ja kun luet rivin niin muodosta sen perusteella uusi `Tuote`-olio ja lisää `List<Tuote>`-kokoelmaan. Nyt haku ja päivitys pitäisi onnistua helposti kokoelmasta.
- Luultavasti kannattaa tehdä oma luokka `CSVKäsittelijä` joka sisältää edellä mainitun kokoelmaluokan ja muut tarvittavat toiminnot (`Haku`, `Päivitys`, `Talletus`) niin saadaan siististi paketoitua toiminnallisuus yhteen luokkaan.
- Aina kun päivität jotain tuotetta niin kirjoita kaikki tuotteet uudelleen tiedostoon
- Tiedostoon kirjoitus `StreamWriter`-luokan instanssin avulla ja tee joko sopiva `ToString()` – metodi `Tuote`-luokkaan tai tee uusi metodi `ToCSV()`.
- muista sulkea tiedosto kun kirjoitus on valmis ja avaa tiedosto aina uudelleen kun sinne pitää kirjoittaa muutokset.