Sovelto

ACADEMY
A PART OF ACADEMIC WORK

Viikko 1, ensimmäinen harjoitus ja samalla kertausta esiopiskeluun

1. Komentotulkki

- Käytä harjoituksessa editoria, esimerkiksi Notepad tai Notepad++ ja komentotulkkia (cmd.exe), nyt et siis edes käynnistä Visual Studiota! Huomaan että koneellasi on myös Developer Command Prompt joka on muuten 'tavallinen' komentotulkki mutta siinä polkuun on lisätty kaikki SDK:n hakemistot (Software Development Kit)
- Luo harjoitusta varten uusi hakemisto (esim. c:\work\v1ti) , lisää sinne Eka.cs tiedosto (tai Program.cs jos se tuntuu tutummalta)
- Tee luokkaan Main-metodi joka tulostaa 5 kertaa "Tämä on C#-kurssin eka harjoitus"
- Käännä ohjelma komentoriviltä (csc) c:\work\v1ti> csc Eka.cs
- Korjaa syntaksivirheet, talleta tiedosto ja käännä sekä toista kunnes ei enää virheitä
- Suorita ohjelma komentoriviltä c:\work\v1ti> Eka

- Jos ä-kirjain ei tulostu ihan oikein niin se on korjattavissa mutta ei välitetä siitä toistaiseksi

2. Komentoriviparametrit

- Muuta Main-metodia niin että sovelluksen komentoriviparametrina voi määrittää montako kertaa tulostus tapahtuu
- muista tarkistaa myös että toimii ilman komentoriviparametriakin, olkoon silloin tulostuslukumäärä 5 kuten harjoituksessa 1

```
c:\work\v1ti> Eka 3
```

- Lisää vielä toinen parametri joka määrittää sisennystason, esim: `Eka 2 10` tulostaa teksti kaksi kertaa ja lisää rivin alkuun 10 välilyöntiä
- Lisää kolmas parametri joka on tulostettava merkkijono

- Ohjaa tulostus tiedostoon output.txt (ja tämä siis komentotulkissa, ei sovelluskoodissa!)

3. Dotnet CLI ja .NET 5 (.NET Core) sekä Visual Studio

- Tarkista ensin onko .NET 5 jo asennettu ja asenna tarvittaessa: https://dotnet.microsoft.com/download
- Tee uusi .NET 5 projekti antamalla komentotulkissa komento: c:\work\v1ti>dotnet new console –n Eka –o Eka
- Siirry muodostuneeseen Eka-hakemistoon, poista Program.cs ja kopioi edellisen harjoituksen Eka.cs tiedosto tähän hakemisoon
- Käännä komennolla: dotnet build
- Suorita sovellus: dotnet run 7
- Etsi Eka.exe –tiedosto projektihakemiston alihakemistoista ja käynnistä se
- Palaa takaisin c:\work\v1ti\Eka –hakemistoon ja käynnistä Visual Studio komennolla c:\work\v1ti> start Eka.csproj

3.1 IDE (Visual Studio) vs. komentorivi

- Osaat nyt siis tehdä projektin komentoriviltä käsin ja avata sen Visual Studioon
- Samoin voit tehdä täysin samanlaisen projektin Visual Studiosta
- Lisäksi voit kokeilla Visual Studio **Code**:lla editointia jos se on asennettu, Code on käytössä viikoilla 4 ja 5 jolloin viimeistään asennat sen. Nyt ei tarvitse vielä välttämättä asentaa Visual Studio Code:a.
- Suorita sovellus (Start debugging) Visual Studiosta käsin siten että saat annettua komentoriviparametrit Visual Studion kautta (Project Properties ja etsi oikea välilehti sekä sitten Application arguments)
- Nyt ei enempää kannata editoida Notepadilla vaan jatka harjoitusta Visual Studiolla
- Jatkossa ellei toisin mainita, ovat kaikki projektit Core/.NET 5 -projekteja

4. Laskin

- Komentoriviparametrien antaminen saattaa olla hieman työlästä, vaihdetaan tiedon lukemiseen konsolilta, käytä siis Console-luokkaa lukemiseen
- Tee uusi konsoliprojekti Laskin (Console Application mutta **EI** Framework, luithan edellisen sivun loppuun saakka!).
  Sovellus lukee kaksi numeroa ja tulostaa niiden summan
- Lisätehtävä: Kun ohjelma toimii, niin lisää myös muita laskutoimituksia, eli voit laskea esim. 2 + 3 tai 12 / 5
- Tässä siis luet kolme parametria joista keskimmäinen on operaattori ja muut ovat operandeja. Tarkista että parametrejä on todella kolme kappaletta ja keskimmäinen on joku peruslaskutoimitus (+ - / *). Lisäksi tarkistus että operandit ovat numeerisia.
- Huomaa että 2+3 ei oikein toimi mutta 2 + 3 on ok, siis välissä pitää olla vähintään yksi välilyönti jotta komentotulkki osaa erottaa parametrit toisistaan.
- Lisätehtävänä voit miettiä mitä pitää muuttaa, jotta myös syöte 2+3 toimii oikein. Älä kuitenkaan käytä tähän liikaa aikaa vaan siirry seuraavaan tehtävään jos tuntuu vaikealta.

4.1 Binäärilaskin (vaikea!)

- Toteuta harjoituksen 4-laskin, mutta laske binääriluvuilla - eli lue ja tulosta binäärilukuja
- Binaariluvut ovat siis ykkösiä ja nollia, eli luvut
  [1-9]: 1, 10, 11, 100, 101, 110, 111, 1000, 1001
- Näin siis: 10110 + 101 = 11011 eli 22 + 5 = 27
- Tee tulostus binäärisenä

- Jos tuntuu liian haasteelliselta niin ei hätää, binärilukuja tarvitset varsin harvoin C#-koodissa eli siirry seuraavaan harjoitukseen ja tämä olkoot 'puskurina' niille joilla C# sujuu jo vauhdikkaasti tai muuten vain haluaa kunnon haastetta.

5. Silmukat ja taulukot

- Tee ohjelma, joka käsittelee taulukossa olevia kokonaislukuja
- Tee ensin taulukko, jossa on 5 kokonaislukua ja laske luvuista (ilman valmiita Sum tai Average-metodeja), keksi ihan vapaasti luvut
  - Summa
  - Keskiarvo
- Tulosta lasketut arvot
- Kun olet saanut ohjelman toimimaan viidellä antamallasi luvulla, tee uusi versio, joka arpoo 100 satunnaista lukua väliltä 0 - 1000, tallettaa ne taulukkoon, ja sen jälkeen laskee vastaavasti summan ja keskiarvon
- Lisätehtävä: tulosta myös pienin ja suurin luku

6. Ryhmien arvonta

- Tee sovellus joka:
  - lukee konsulttien nimet tiedostosta Konsultit.txt (löytyy Canvaksesta)
  - tulostaa satunnaisessa järjestyksessä nimet konsolille (Random-luokka auttaa tässä)
- Lisää toimintaa niin että komentoriviparametrina voit antaa ryhmäkoon (olkoot oletuksena 3) ja sovellus tulostaa konsolille arvotut ryhmät:

```
c:\work\>TeeJako 3
Arvotut ryhmät:
Ryhmä 1:   Jaska Jokunen
           Aku Ankka
           Musta Naamio
Ryhmä 2:   Touho Ankka
           Mikki Hiiri  //
jne…
```

- Lisää vielä tulostus tiedostoon, toisena komentoriviparametrina olkoot arvonnan nimi ja tiedostonimi on lopulta muotoa <nimi>_vvvvkkpp.txt

```
c:\work\>TeeJako 3 Harjoitus6
➔ tulostaa tiedostoon "Harjoitus6_20180911.txt
```

- Lisää vielä toiminto: jos on vielä kolmas parametri, niin se on hakemisto mihin tiedosto tulostetaan. Muista tarkistaa että hakemisto on olemassa. Jos ei ole, niin anna käyttäjälle varoitus ja tulosta oletushakemistoon.

7. Henkilö-luokka

- Toteuta uusi luokka Henkilö
- Lisää luokkaan ominaisuudet Etunimi ja Sukunimi sekä Ikä
- Toteuta myös sopivat konstruktorit, ainakin sellainen jolla on parametrit etunimi, sukunimi ja ikä
- Toteuta luokalle ToString-metodi (override) ja mieti mikä voisi olla sopiva tulostusmuoto
- Tee toinen luokka HenkilöTesteri jolla testaat että Henkilö-luokka toimii oikein

```
Henkilö[] hlot = { new Henkilö ("Jaska", "Jokunen", 21),
    new Henkilö("Maija", "Meikäläinen", 28),
    new Henkilö("John", "Doe", 55),
    new Henkilö("John", "Deere", 21),
    new Henkilö("Musta", "Naamio", 355)
};
foreach (var h in hlot)
{
    Console.WriteLine(h);
}
```

7. Henkilö ja lisätehtävä

- Muokkaa Henkilö -luokkaa siten, että lisäät luokalle kentän readonly int _id;
- Tee ominaisuus jolla luokan ulkopuolelta voi kysyä Id:n arvon
- Tee toteutus siten, että jokaisella henkilöllä on oma yksilöllinen id-arvonsa. Helppo toteutus: lisää luokkaan staattinen kenttä private static int seuraavaVapaaId;
- Konstruktorissa aseta Id-kenttä ja päivitä seuraavaVapaaId yhtä suuremmaksi
- Muokkaa myös ToString() metodia siten että palautat myös Id:n osana merkkijonoa
- Testaa että Id:n arvo päivittyy oikein

7. Luokkasuunnittelu - Osoite

- Henkilöllä pitää olla mahdollisuus tallettaa osoite ja jos vaikka kuvitellaan lehtimyyntijärjestelmä jossa henkilöön ja/tai tilaukseen voi liittyä sekä toimitus- että laskutusosoite, niin ei liene järkevää että osoitteen osat laitetaan suoraan Henkilöluokkaan vaan tehdään oma luokka *Osoite*
- Mietittävää ja toteuttavaa:
  - Mieti mikä voisi olla sopiva nimiavaruuden nimi
  - mitä tietoa pitäisi luokassa olla?
  - mitä konstruktoreita tarvitset?
  - mitä käsittelysääntöjä tarvitset ja mitä ominaisuuksia olisi hyvä olla luokassa?
  - tee ToString() metodi

7. Jatkoa edelliseen

- Muuta Henkilö-luokkaa siten että lisäät ominaisuuden Kotiosoite ja sen tietotyyppi on tietenkin Osoite
- Muuta/lisää konstruktoreita osoitteen osalta
- Tulosta henkilöiden nimet ja osoitteet konsolille
- Lisätehtävä: Tulosta henkilöt postinumeron mukaisessa järjestyksessä

8. Virhekäsittelyä eli poikkeuksia

- Korjaa tai muuta Laskin –harjoitusta siten että jos laskennassa tulee virhe (esimerkiksi nollalla jako) niin otat sen kiinni ja annat hallitusti käyttäjän kannalta mielekkään virheilmoituksen
- Lisää samaan tehtävään vielä tarkistus tuleeko ylivuotoa, esimerkiksi onnistuuko laskutoimitus 1000000000 * 4242424242

- Lisää Osoite-luokkaan tarkistus, että annettu postinumero on validi eli sisältää 5 **numero**merkkiä
- Jos postinumero on virheellinen, niin heitä poikkeus ArgumentOutOfRangeException. Mitä muita tarkistuksia pitäisi tai voisi liittää osoitteeseen?
- Lisätehtävä: Tee oma poikkeusluokka OsoiteException ja käytä sitä. Mitä tietoa OsoiteException-luokassa pitäisi olla?