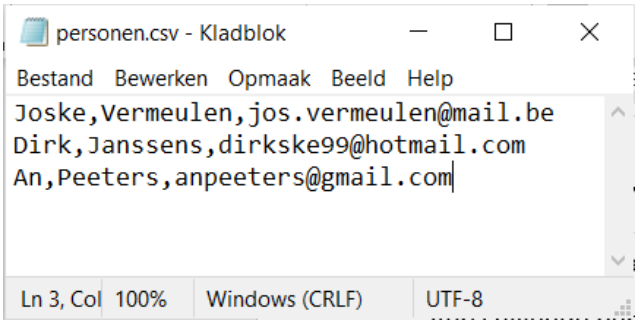
C# 2 – Les 2: Van tekstbestanden naar objecten

In deze les gaan we objecten inlezen en opslaan met tekstbestanden. Elke regel in het tekstbestand bevat de gegevens van 1 object. Per regel worden de gegevens gescheiden door een komma (of een ander speciaal teken).

Verschillende persoonobjecten met als properties Voornaam, Achternaam en Email zien er dan zo uit in ons tekstbestand:



We gaan van elke regel een instantie maken van de Persoon class:

```
public class Persoon
{
    public string Voornaam { get; set; }
    public string Achternaam { get; set; }
    public string Email { get; set; }

    public Persoon(string voornaam, string achternaam, string email)
    {
        Voornaam = voornaam;
        Achternaam = achternaam;
        Email = email;
    }
}
```

Een bestand waar de gegevens zijn gescheiden door komma's wordt een csv-bestand (comma seperated values) genoemd. Het wordt doorgaans opgeslagen met extensie .csv.

Net als in de vorige les gaan we de regels inlezen met File.ReadAllLines. Zoals elke progammeertaal heeft C# methodes om strings te splitsen. Na het inlezen splitsen we elke regel op met de Split methode en maken we een nieuw persoon object.

```
List<string> lines = File.ReadAllLines("personen.csv").ToList();

foreach(var l in lines)
{
```

```
    string[] parts = l.Split(",");
    Klanten.Add(new Persoon(parts[0], parts[1], parts[2]));
}
```

De documentatie van Split (met voorbeelden!) vind je hier: https://docs.microsoft.com/en-us/dotnet/api/system.string.split?view=net-5.0 (Bekijk ook eens aantal van de andere methodes, ze komen vaak goed van pas.)

Objecten bewaren in een bestand

De properties van een object kunnen we, nadat we het object hebben aangepast, ook weer gaan bewaren als regels in het bestand. We kunnen hiervoor eventueel een static methode maken om een object om te zetten in een regel met komma's (in het voorbeeld is dat de methode NaarCSVRegel).

```
class Program
{
    static List<Persoon> Klanten = new List<Persoon> {};
    static void Main(string[] args)
    {
        List<string> lines = File.ReadAllLines("personen.csv").ToList();

        foreach(var l in lines)
        {
            string[] parts = l.Split(",");
            Klanten.Add(new Persoon(parts[0], parts[1], parts[2]));
        }

        Console.Write("Geef voornaam: ");
        string vn = Console.ReadLine();
        Console.Write("Geef achternaam: ");
        string an = Console.ReadLine();
        Console.Write("Geef email: ");
        string email = Console.ReadLine();

        if (Klanten.Exists(klant => klant.Voornaam == vn
                                    && klant.Achternaam == an))
        {
            Console.WriteLine("Klant bestaat reeds.");
        }
        else
        {
            Klanten.Add(new Persoon(vn, an, email));
        }

        lines.Clear();

        foreach(var k in Klanten)
```

```
        {
            Console.WriteLine(k.ToString());
            lines.Add(NaarCSVRegel(k));
        }

        Console.WriteLine("Naar bestand schrijven...");
        File.WriteAllLines("personen.csv", lines);
        Console.WriteLine("Bestand geschreven.");
    }

    static string NaarCSVRegel(Persoon p)
    {
        return $"{p.Voornaam},{p.Achternaam},{p.Email}";
    }
}
```

Oefening 1: HighScoreManager

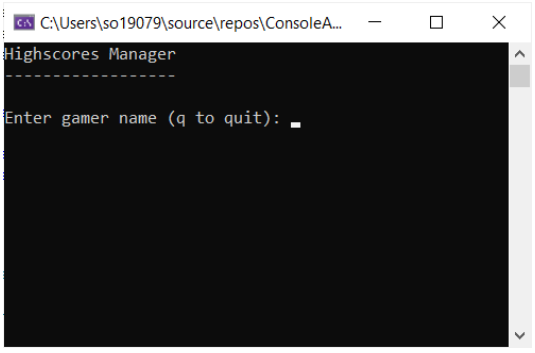
Maak een programma *HighScoreManager* voor het bijhouden van de 10 hoogste score van een game. Het programma bewaart de highscores in een bestand. Een highscore object bevat een naam van een speler en de score.

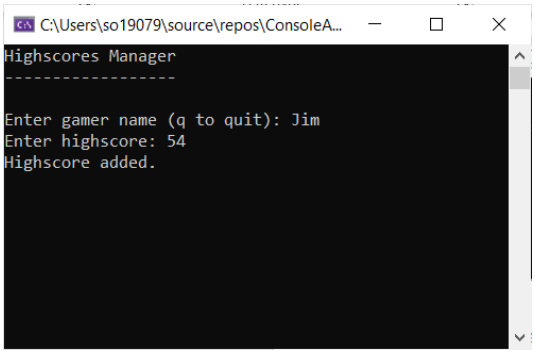
Het is de bedoeling dat de HighScoreManager wordt gebruikt in een game, maar om te testen zorg je dat je via de Console een nieuwe highscore kan ingeven door een naam en een score in te geven.

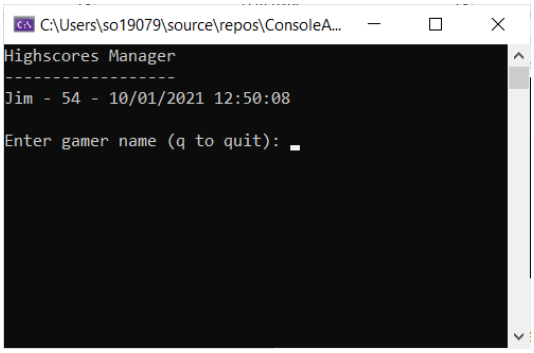
Sorteer de highscores van hoog naar laag en bewaar enkel de laatste 10 scores in het bestand.

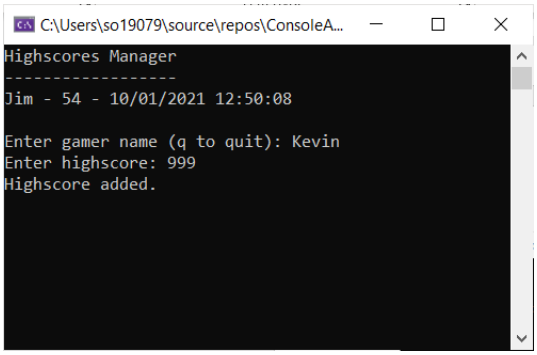
Extra 1: HighScores met datum

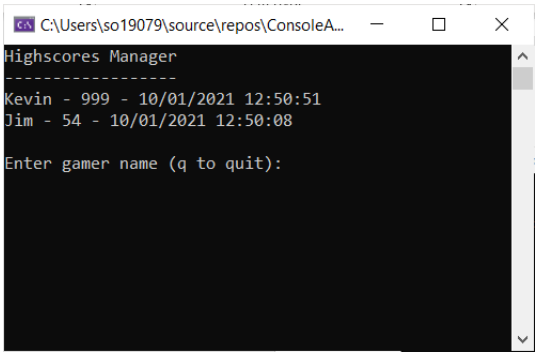
Voeg een datum toe aan het highscore object. Zorg er voor dat de datum automatisch wordt ingevuld naar de huidige datum bij het aanmaken van een HighScore object.











TIP: Een datum is van het type DateTime en kan je zonder problemen wegschrijven naar een csv-bestand. De datum wordt automatisch omgezet in een string (volgens de Belgische schrijfwijze). Om de string terug in te lezen gebruik je DateTime.Parse(…).

Om dit correct te laten werken op computers met andere datumschrijfwijzen is er eigenlijk nog meer werk nodig. (Om dit bijvoorbeeld zowel in Japan als in de USA te laten werken.) Je hoeft hier geen rekening mee te houden in deze oefening. Meer info om een datumschrijfwijze cultuuronafhankelijk vast te leggen vind je hier: https://docs.microsoft.com/en-us/dotnet/api/system.datetime.parseexact?view=net-5.0

Extra 2: Hoger/lager met HighScores

Breidt het programma hoger/lager uit met highscores. De 10 beste scores worden opgeslagen in een bestand.