1. Visual 2022

create new project->consol app (.net core)

2. Solution Explorer-> jobb Klikk → Add → Class-> Berles.cs

vagy a program.cs-be felviszem az osztályokat

class Car

{

public int Id { get; set; }

public string Model { get; set; }

public string Brand { get; set; }

public string LicencePlate { get; set; }

public int Year { get; set; }

public int DailyPrice { get; set; }

}

class Booking

{

public int Id { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public int CarId { get; set; }

public int TotalPrice { get; set; }

public string UserUID { get; set; }

}

**3. Fájlbeolvasás**

Bemásolom bin\Debug\

**Main metódus vázba**

static void Main(string[] args)

{

**var cars = ReadCars("cars.csv");**

**var bookings = ReadBookings("bookings.csv");**

**Console.WriteLine($"Beolvasott autók: {cars.Count} db");**

**Console.WriteLine($"Beolvasott foglalások: {bookings.Count} db");**

}

**4. class Programba**

class Program

{

static void Main(string[] args)

{

var cars = ReadCars("cars.csv");

var bookings = ReadBookings("bookings.csv");

// További kódok...

}

static List<Car> ReadCars(string path)

{

return File.ReadAllLines(path)

.Skip(1)

.Select(line => line.Split(','))

.Select(parts => new Car

{

Id = int.Parse(parts[0]),

Brand = parts[1],

Model = parts[2],

LicensePlate = parts[3],

Year = int.Parse(parts[4]),

DailyPrice = int.Parse(parts[5])

})

.ToList();

}

static List<Booking> ReadBookings(string path)

{

return File.ReadAllLines(path)

.Skip(1)

.Select(line => line.Split(','))

.Select(parts => new Booking

{

Id = int.Parse(parts[0]),

StartDate = DateTime.Parse(parts[1]),

EndDate = DateTime.Parse(parts[2]),

CarId = int.Parse(parts[3]),

TotalPrice = int.Parse(parts[4]),

UserUID = parts[5]

})

.ToList();

}

}

**5. using System.IO importál**

**6. Feladatok main-be:**

**Console.WriteLine("\n1) Autók napi bérleti díj szerint csökkenően:");**

var sortedCars = cars.OrderByDescending(c => c.DailyPrice).ToList();

foreach (var car in sortedCars)

{

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate}) - {car.DailyPrice} Ft/nap");

}

**Console.WriteLine("\n2) Foglalások autó márkával és bérleti díjjal:");**

//Szűrés: csak létező autókhoz tartozó foglalások

var validBookings = bookings.Where(b => cars.Any(c => c.Id == b.CarId)).ToList();

foreach (var booking in validBookings)

{

var car = cars.First(c => c.Id == booking.CarId);

Console.WriteLine(

$"{car.Brand} {car.Model} ({car.LicensePlate}) - {car.DailyPrice} Ft/nap | " +

$"Foglalás: {booking.StartDate:yyyy-MM-dd} - {booking.EndDate:yyyy-MM-dd} | " +

$"Teljes ár: {booking.TotalPrice} Ft");

}

**Console.WriteLine("\n3) Legtöbbször lefoglalt autó:");**

var mostBookedCarId = validBookings

.GroupBy(b => b.CarId)

.OrderByDescending(g => g.Count())

.Select(g => new { CarId = g.Key, Count = g.Count() })

.FirstOrDefault();

if (mostBookedCarId != null)

{

var car = cars.First(c => c.Id == mostBookedCarId.CarId);

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate}) - {mostBookedCarId.Count} foglalás");

}

else

{

Console.WriteLine("Nincs foglalás.");

}

**Console.WriteLine("\n4) Legtöbb bevételt hozó autó:");**

var topEarning = validBookings

.GroupBy(b => b.CarId)

.Select(g => new

{

CarId = g.Key,

TotalRevenue = g.Sum(b => b.TotalPrice)

})

.OrderByDescending(x => x.TotalRevenue)

.FirstOrDefault();

if (topEarning != null)

{

var car = cars.First(c => c.Id == topEarning.CarId);

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate}) - {topEarning.TotalRevenue} Ft összbevétel");

}

else

{

Console.WriteLine("Nincs bevétel, mert nincs foglalás.");

}

**Console.WriteLine("\n5) Átlagos bérleti időtartam (napban):");**

var averageDays = validBookings

.Select(b => (b.EndDate - b.StartDate).TotalDays)

.DefaultIfEmpty(0)

.Average();

Console.WriteLine($"{averageDays:F1} nap");

**Console.WriteLine("\n6) Legutóbb lefoglalt autó:");**

var latestBooking = validBookings

.OrderByDescending(b => b.StartDate)

.FirstOrDefault();

if (latestBooking != null)

{

var car = cars.First(c => c.Id == latestBooking.CarId);

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate})");

Console.WriteLine($"Foglalás kezdete: {latestBooking.StartDate:yyyy-MM-dd}");

}

else

{

Console.WriteLine("Nincs foglalás.");

}

**Console.WriteLine("\n7) Összes foglalásból származó teljes bevétel:");**

var totalRevenue = validBookings.Sum(b => b.TotalPrice);

Console.WriteLine($"{totalRevenue} Ft");

**Console.WriteLine("\n8) Foglalások havi bontásban fájlba írva: foglalasok.csv");**

var groupedByMonth = validBookings

.GroupBy(b => new { b.StartDate.Year, b.StartDate.Month })

.OrderBy(g => g.Key.Year)

.ThenBy(g => g.Key.Month);

var lines = new List<string>();

lines.Add("Ev;Honap;Rendszam;Kezdet;Vege;Ar");

foreach (var group in groupedByMonth)

{

foreach (var booking in group)

{

var car = cars.First(c => c.Id == booking.CarId);

lines.Add($"{group.Key.Year};{group.Key.Month:00};{car.LicensePlate};{booking.StartDate:yyyy-MM-dd};{booking.EndDate:yyyy-MM-dd};{booking.TotalPrice}");

}

}

File.WriteAllLines("foglalasok.csv", lines, Encoding.UTF8);

Console.WriteLine("A 'foglalasok.csv' fájl elkészült.");

**Console.WriteLine("\n9) Bevételek autóként összesítve → bevetes.csv");**

var revenueByCar = validBookings

.GroupBy(b => b.CarId)

.Select(g => new

{

Car = cars.First(c => c.Id == g.Key),

Total = g.Sum(b => b.TotalPrice)

})

.OrderByDescending(x => x.Total);

var lines2 = new List<string>();

lines2.Add("Rendszam;Marka;Modell;Bevetel");

foreach (var item in revenueByCar)

{

lines2.Add($"{item.Car.LicensePlate};{item.Car.Brand};{item.Car.Model};{item.Total}");

}

File.WriteAllLines("bevetes.csv", lines2, Encoding.UTF8);

Console.WriteLine("A 'bevetes.csv' fájl elkészült.");

**Console.WriteLine("\n10) Nem használt autók listázása → nemhasznalt.csv");**

var usedCarIds = validBookings.Select(b => b.CarId).ToHashSet();

var unusedCars = cars

.Where(c => !usedCarIds.Contains(c.Id))

.OrderBy(c => c.Brand)

.ThenBy(c => c.Model);

var lines3 = new List<string>();

lines3.Add("Rendszam;Marka;Modell;Ev;NapiAr");

foreach (var car in unusedCars)

{

lines3.Add($"{car.LicensePlate};{car.Brand};{car.Model};{car.Year};{car.DailyPrice}");

}

File.WriteAllLines("nemhasznalt.csv", lines3, Encoding.UTF8);

Console.WriteLine("A 'nemhasznalt.csv' fájl elkészült.");

**Console.WriteLine("\n11) Átlagos napi bérleti díj márkánként → berkat.csv");**

var brandAvgPrices = cars

.GroupBy(c => c.Brand)

.Select(g => new

{

Brand = g.Key,

AvgPrice = g.Average(c => c.DailyPrice)

})

.OrderByDescending(x => x.AvgPrice);

var lines4 = new List<string>();

lines4.Add("Marka;AtlagosNapiAr");

foreach (var item in brandAvgPrices)

{

lines4.Add($"{item.Brand};{item.AvgPrice:F2}");

}

File.WriteAllLines("berkat.csv", lines4, Encoding.UTF8);

Console.WriteLine("A 'berkat.csv' fájl elkészült.");

TELJES

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace luxCar\_console

{

internal class Program

{

static void Main(string[] args)

{

var cars = ReadCars("cars.csv");

var bookings = ReadBookings("bookings.csv");

Console.WriteLine($"Beolvasott autók: {cars.Count} db");

Console.WriteLine($"Beolvasott foglalások: {bookings.Count} db");

Console.WriteLine("\n1) Autók napi bérleti díj szerint csökkenően:");

var sortedCars = cars.OrderByDescending(c => c.DailyPrice).ToList();

foreach (var car in sortedCars)

{

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate}) - {car.DailyPrice} Ft/nap");

}

Console.WriteLine("\n2) Foglalások autó márkával és bérleti díjjal:");

//Szűrés: csak létező autókhoz tartozó foglalások

var validBookings = bookings.Where(b => cars.Any(c => c.Id == b.CarId)).ToList();

foreach (var booking in validBookings)

{

var car = cars.First(c => c.Id == booking.CarId);

Console.WriteLine(

$"{car.Brand} {car.Model} ({car.LicensePlate}) - {car.DailyPrice} Ft/nap | " +

$"Foglalás: {booking.StartDate:yyyy-MM-dd} - {booking.EndDate:yyyy-MM-dd} | " +

$"Teljes ár: {booking.TotalPrice} Ft");

}

Console.WriteLine("\n3) Legtöbbször lefoglalt autó:");

var mostBookedCarId = validBookings

.GroupBy(b => b.CarId)

.OrderByDescending(g => g.Count())

.Select(g => new { CarId = g.Key, Count = g.Count() })

.FirstOrDefault();

if (mostBookedCarId != null)

{

var car = cars.First(c => c.Id == mostBookedCarId.CarId);

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate}) - {mostBookedCarId.Count} foglalás");

}

else

{

Console.WriteLine("Nincs foglalás.");

}

Console.WriteLine("\n4) Legtöbb bevételt hozó autó:");

var topEarning = validBookings

.GroupBy(b => b.CarId)

.Select(g => new

{

CarId = g.Key,

TotalRevenue = g.Sum(b => b.TotalPrice)

})

.OrderByDescending(x => x.TotalRevenue)

.FirstOrDefault();

if (topEarning != null)

{

var car = cars.First(c => c.Id == topEarning.CarId);

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate}) - {topEarning.TotalRevenue} Ft összbevétel");

}

else

{

Console.WriteLine("Nincs bevétel, mert nincs foglalás.");

}

Console.WriteLine("\n5) Átlagos bérleti időtartam (napban):");

var averageDays = validBookings

.Select(b => (b.EndDate - b.StartDate).TotalDays)

.DefaultIfEmpty(0)

.Average();

Console.WriteLine($"{averageDays:F1} nap");

Console.WriteLine("\n6) Legutóbb lefoglalt autó:");

var latestBooking = validBookings

.OrderByDescending(b => b.StartDate)

.FirstOrDefault();

if (latestBooking != null)

{

var car = cars.First(c => c.Id == latestBooking.CarId);

Console.WriteLine($"{car.Brand} {car.Model} ({car.LicensePlate})");

Console.WriteLine($"Foglalás kezdete: {latestBooking.StartDate:yyyy-MM-dd}");

}

else

{

Console.WriteLine("Nincs foglalás.");

}

Console.WriteLine("\n7) Összes foglalásból származó teljes bevétel:");

var totalRevenue = validBookings.Sum(b => b.TotalPrice);

Console.WriteLine($"{totalRevenue} Ft");

Console.WriteLine("\n8) Foglalások havi bontásban fájlba írva: foglalasok.csv");

var groupedByMonth = validBookings

.GroupBy(b => new { b.StartDate.Year, b.StartDate.Month })

.OrderBy(g => g.Key.Year)

.ThenBy(g => g.Key.Month);

var lines = new List<string>();

lines.Add("Ev;Honap;Rendszam;Kezdet;Vege;Ar");

foreach (var group in groupedByMonth)

{

foreach (var booking in group)

{

var car = cars.First(c => c.Id == booking.CarId);

lines.Add($"{group.Key.Year};{group.Key.Month:00};{car.LicensePlate};{booking.StartDate:yyyy-MM-dd};{booking.EndDate:yyyy-MM-dd};{booking.TotalPrice}");

}

}

File.WriteAllLines("foglalasok.csv", lines, Encoding.UTF8);

Console.WriteLine("A 'foglalasok.csv' fájl elkészült.");

Console.WriteLine("\n9) Bevételek autóként összesítve → bevetes.csv");

var revenueByCar = validBookings

.GroupBy(b => b.CarId)

.Select(g => new

{

Car = cars.First(c => c.Id == g.Key),

Total = g.Sum(b => b.TotalPrice)

})

.OrderByDescending(x => x.Total);

var lines2 = new List<string>();

lines2.Add("Rendszam;Marka;Modell;Bevetel");

foreach (var item in revenueByCar)

{

lines2.Add($"{item.Car.LicensePlate};{item.Car.Brand};{item.Car.Model};{item.Total}");

}

File.WriteAllLines("bevetes.csv", lines2, Encoding.UTF8);

Console.WriteLine("A 'bevetes.csv' fájl elkészült.");

Console.WriteLine("\n10) Nem használt autók listázása → nemhasznalt.csv");

var usedCarIds = validBookings.Select(b => b.CarId).ToHashSet();

var unusedCars = cars

.Where(c => !usedCarIds.Contains(c.Id))

.OrderBy(c => c.Brand)

.ThenBy(c => c.Model);

var lines3 = new List<string>();

lines3.Add("Rendszam;Marka;Modell;Ev;NapiAr");

foreach (var car in unusedCars)

{

lines3.Add($"{car.LicensePlate};{car.Brand};{car.Model};{car.Year};{car.DailyPrice}");

}

File.WriteAllLines("nemhasznalt.csv", lines3, Encoding.UTF8);

Console.WriteLine("A 'nemhasznalt.csv' fájl elkészült.");

Console.WriteLine("\n11) Átlagos napi bérleti díj márkánként → berkat.csv");

var brandAvgPrices = cars

.GroupBy(c => c.Brand)

.Select(g => new

{

Brand = g.Key,

AvgPrice = g.Average(c => c.DailyPrice)

})

.OrderByDescending(x => x.AvgPrice);

var lines4 = new List<string>();

lines4.Add("Marka;AtlagosNapiAr");

foreach (var item in brandAvgPrices)

{

lines4.Add($"{item.Brand};{item.AvgPrice:F2}");

}

File.WriteAllLines("berkat.csv", lines4, Encoding.UTF8);

Console.WriteLine("A 'berkat.csv' fájl elkészült.");

}

static List<Car> ReadCars(string path)

{

return File.ReadAllLines(path)

.Skip(1)

.Select(line => line.Split(','))

.Select(parts => new Car

{

Id = int.Parse(parts[0]),

Brand = parts[1],

Model = parts[2],

LicensePlate = parts[3],

Year = int.Parse(parts[4]),

DailyPrice = int.Parse(parts[5])

})

.ToList();

}

static List<Booking> ReadBookings(string path)

{

return File.ReadAllLines(path)

.Skip(1)

.Select(line => line.Split(','))

.Select(parts => new Booking

{

Id = int.Parse(parts[0]),

StartDate = DateTime.Parse(parts[1]),

EndDate = DateTime.Parse(parts[2]),

CarId = int.Parse(parts[3]),

TotalPrice = int.Parse(parts[4]),

UserUID = parts[5]

})

.ToList();

}

}

class Car

{

public int Id { get; set; }

public string Model { get; set; }

public string Brand { get; set; }

public string LicensePlate { get; set; }

public int Year { get; set; }

public int DailyPrice { get; set; }

}

class Booking

{

public int Id { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public int CarId { get; set; }

public int TotalPrice { get; set; }

public string UserUID { get; set; }

}

}