



Scott Allen

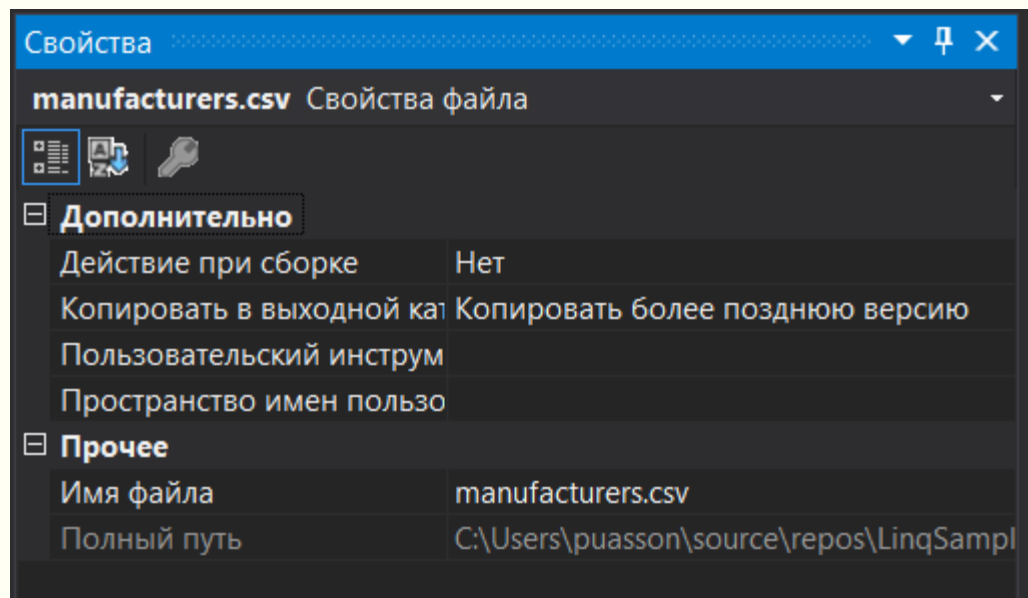
OdeToCode.com - @OdeToCode

З'ЄДНАННЯ, ГРУПУВАННЯ ТА АГРЕГУВАННЯ ДАНИХ

Питання 8.3.

Розглянемо інше джерело даних

- Файл `manufactures.csv` містить дані про виробників автомобілів: назву, країну зі штаб-квартирою та рік, у якому інформація актуальна.
 - У прикладі рік буде завжди 2016.



```
1 ALFA ROMEO,Italy,2016
2 Aston Martin Lagonda Ltd,UK,2016
3 Audi,Germany,2016
4 BMW,Germany,2016
5 Chevrolet,USA,2016
6 Dodge,USA,2016
7 Ferrari,Italy,2016
8 Honda,Japan,2016
9 Jaguar,UK,2016
10 Lamborghini,Italy,2016
11 MAZDA,Japan,2016
12 McLaren,UK,2016
13 Mercedes-Benz,Germany,2016
14 NISSAN,Japan,2016
15 Pagani Automobili S.p.A.,Italy,2016
16 Porsche,Germany,2016
17 FIAT,Italy,2016
18 Mini,Germany,2016
19 SCION,USA,2016
```

Визначимо клас Manufacturer для тримання даних

```
public class Manufacturer
{
    public string Name { get; set; }
    public string Headquarters { get; set; }
    public int Year { get; set; }
}
```

```
private static List<Manufacturer> ProcessManufacturers(string path)
{
    var query =
        File.ReadAllLines(path)
            .Where(l => l.Length > 1)
            .Select(l =>
            {
                var columns = l.Split(',');
                return new Manufacturer
                {
                    Name = columns[0],
                    Headquarters = columns[1],
                    Year = int.Parse(columns[2])
                };
            });
    return query.ToList();
}
```

■ Аналогічний статичний метод буде:

- Зчитувати рядки з файлу, відкидаючи порожні
- Проектувати кожний рядок на об'єкт типу Manufacturer.
- Повертати список об'єктів типу Manufacturer.

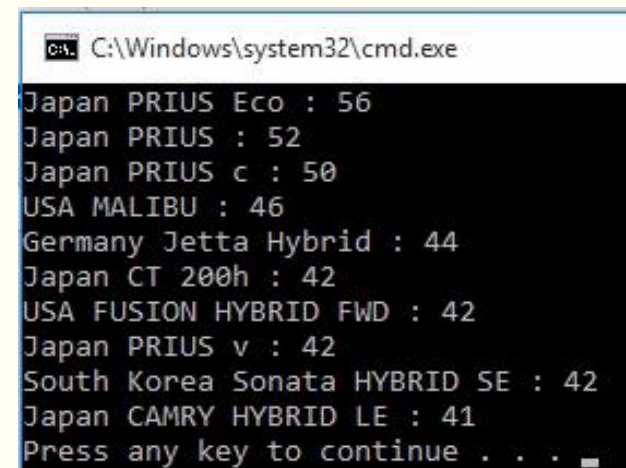
З'єднання даних

- LINQ-оператор Join() дозволяє поєднувати дані з кількох джерел.
 - Подібний до оператора INNER JOIN з SQL.
 - З точки зору синтаксису подібний до from.
- Нехай потрібно вивести найефективніші автомобілі та штаб-квартири їх виробників.
 - Оператор join буде здійснювати з'єднання за спільним значенням (car.Manufacturer == manufacturer.Name).
 - Доступне тільки еквіз'єднання, тому порівняння відбувається за допомогою оператора equals.

```
static void Main(string[] args)
{
    var cars = ProcessCars("fuel.csv");
    var manufacturers = ProcessManufacturers("manufacturers.csv");

    var query =
        from car in cars
        join manufacturer in manufacturers
          on car.Manufacturer equals manufacturer.Name
        orderby car.Combined descending, car.Name ascending
        select new
        {
            manufacturer.Headquarters,
            car.Name,
            car.Combined
        };
}
```

```
foreach (var car in query.Take(10))
{
    Console.WriteLine($"{car.Headquarters} {car.Name} : {car.Combined}");
}
```



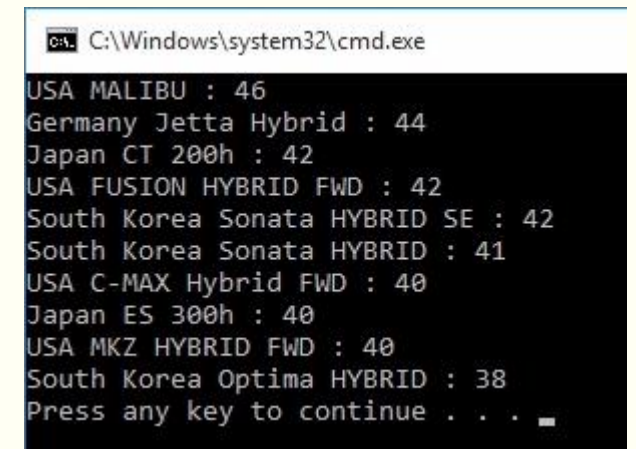
```
C:\Windows\system32\cmd.exe
Japan PRIUS Eco : 56
Japan PRIUS : 52
Japan PRIUS c : 50
USA MALIBU : 46
Germany Jetta Hybrid : 44
Japan CT 200h : 42
USA FUSION HYBRID FWD : 42
Japan PRIUS v : 42
South Korea Sonata HYBRID SE : 42
Japan CAMRY HYBRID LE : 41
Press any key to continue . . .
```

З'єднання даних

- Зверніть увагу, що еквіз'єднання відкидатиме виробника, для якого немає еквівалента в manufactures.csv.
 - Наприклад, видаливши виробника TOYOTA, отримаємо інший вивід:
 - Немає інформації для зв'язування, щоб отримати дані щодо Пріусів.

- Застосовуючи синтаксис методів розширення, матимемо:

```
var query2 =
    cars.Join(manufacturers,
        c => c.Manufacturer,
        m => m.Name, (c, m) => new
        {
            m.Headquarters,
            c.Name,
            c.Combined
        })
    .OrderByDescending(c => c.Combined)
    .ThenBy(c => c.Name);
```



```
C:\Windows\system32\cmd.exe
USA MALIBU : 46
Germany Jetta Hybrid : 44
Japan CT 200h : 42
USA FUSION HYBRID FWD : 42
South Korea Sonata HYBRID SE : 42
South Korea Sonata HYBRID : 41
USA C-MAX Hybrid FWD : 40
Japan ES 300h : 40
USA MKZ HYBRID FWD : 40
South Korea Optima HYBRID : 38
Press any key to continue . . . _
```

З'єднання даних

```
var query2 =
    cars.Join(manufacturers,
        c => c.Manufacturer,
        m => m.Name,
        (c, m) => new
        {
            Car = c,
            Manufacturer = m
        })
    .OrderByDescending(c => c.Car.Combined)
    .ThenBy(c => c.Car.Name)
    .Select(c => new
    {
        c.Manufacturer.Headquarters,
        c.Car.Name,
        c.Car.Combined
    });

foreach (var car in query2.Take(10))
{
    Console.WriteLine($"{car.Headquarters} {car.Name} : {car.Combined}");
}
```

- Якщо є потреба в повному з'єднанні інформації, можна здійснити проєкцію на об'єкти типів Car і Manufacturer.
 - Дещо зміниться лямбда для оператора OrderByDescending().
 - Додамо оператор Select() для відображення даних у нові об'єкти.
 - Таким чином, існуватиме доступ до всіх з'єднаних даних.

З'єднання даних за кількома атрибутами

- Додамо як критерій з'єднання ще й рік.
 - Для цього доведеться створювати об'єкти анонімного типу.
 - Зверніть увагу, що назви властивостей повинні бути однаковими.
 - Таким чином, відбувається з'єднання за складеним (composite) ключем.

```
var query =  
    from car in cars  
    join manufacturer in manufacturers  
        on new { car.Manufacturer, car.Year }  
            equals  
            new { Manufacturer = manufacturer.Name, manufacturer.Year }  
    orderby car.Combined descending, car.Name ascending  
    select new  
    {  
        manufacturer.Headquarters,  
        car.Name,  
        car.Combined  
    };
```

```
var query2 =  
    cars.Join(manufacturers,  
        c => new { c.Manufacturer, c.Year },  
        m => new { Manufacturer = m.Name, m.Year },  
        (c, m) => new  
        {  
            m.Headquarters,  
            c.Name,  
            c.Combined  
        });  
    foreach (var car in query2.Take(10))  
    {  
        Console.WriteLine($"{car.Headquarters} {car.Name} : {car.Combined}");  
    }
```

Групування даних

- Спробуємо згрупувати автомобілі за їх виробниками та показати по топ-2 автомобілі з кожної групи.
 - Запишемо частину умов за допомогою синтаксису запитів.
 - Групування сформує «кишені» (buckets) з даними згідно з назвою виробника автомобілів (Key):

```
static void Main(string[] args)
{
    var cars = ProcessCars("fuel.csv");
    var manufacturers = ProcessManufacturers("manufacturers.csv");

    var query =
        from car in cars
        group car by car.Manufacturer;

    foreach (var result in query)
    {
        Console.WriteLine($"{result.Key} has {result.Count()} cars");
    }
}
```

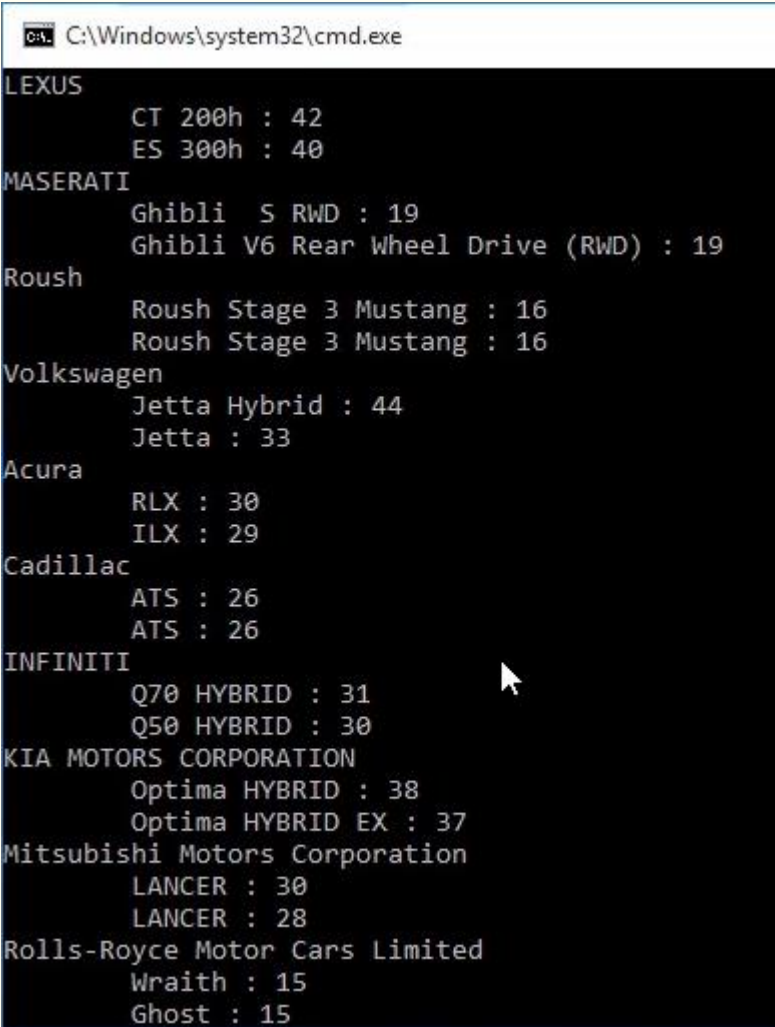
```
C:\Windows\system32\cmd.exe
Pagani Automobili S.p.A. has 1 cars
Porsche has 68 cars
FIAT has 11 cars
Mini has 34 cars
SCION has 8 cars
Subaru has 21 cars
Bentley has 7 cars
Buick has 20 cars
Ford has 86 cars
HYUNDAI MOTOR COMPANY has 43 cars
LEXUS has 35 cars
MASERATI has 8 cars
Roush has 4 cars
Volkswagen has 33 cars
Acura has 12 cars
Cadillac has 26 cars
INFINITI has 25 cars
KIA MOTORS CORPORATION has 41 cars
Mitsubishi Motors Corporation has 13 cars
Rolls-Royce Motor Cars Limited has 8 cars
TOYOTA has 50 cars
Volvo has 26 cars
Chrysler has 11 cars
Lincoln has 23 cars
GMC has 43 cars
RAM has 11 cars
CHEVROLET has 1 cars
Jeep has 38 cars
Land Rover has 11 cars
Press any key to continue . . .
```


Групування даних

- Реалізуємо повну умову:

```
foreach (var group in query)
{
    Console.WriteLine(group.Key);
    foreach (var car in group.OrderByDescending(c => c.Combined).Take(2))
    {
        Console.WriteLine($"{car.Name} : {car.Combined}");
    }
}
```

- Назви виробників з'являються неупорядковано.
- Крім того, для марки Chevrolet існує 2 записи назви: «CHEVROLET» та «Chevrolet», що потрібно буде виправити.



```
C:\Windows\system32\cmd.exe
LEXUS
    CT 200h : 42
    ES 300h : 40
MASERATI
    Ghibli S RWD : 19
    Ghibli V6 Rear Wheel Drive (RWD) : 19
Roush
    Roush Stage 3 Mustang : 16
    Roush Stage 3 Mustang : 16
Volkswagen
    Jetta Hybrid : 44
    Jetta : 33
Acura
    RLX : 30
    ILX : 29
Cadillac
    ATS : 26
    ATS : 26
INFINITI
    Q70 HYBRID : 31
    Q50 HYBRID : 30
KIA MOTORS CORPORATION
    Optima HYBRID : 38
    Optima HYBRID EX : 37
Mitsubishi Motors Corporation
    LANCER : 30
    LANCER : 28
Rolls-Royce Motor Cars Limited
    Wraith : 15
    Ghost : 15
```

Групування даних

- Для усунення дублів Chevrolet будемо зводити всі назви до великих літер.
 - У той же час впорядкування ускладнюється, оскільки єдиний набір даних розбився на групи.
 - Потрібно зібрати погруповані дані в змінну, відносно якої можна буде робити сортування.

```
var query =  
    from car in cars  
    group car by car.Manufacturer.ToUpper() into manufacturer  
    orderby manufacturer.Key  
    select manufacturer;
```

- Запис за допомогою методів розширення дещо простіший:

```
var query2 =  
    cars.GroupBy(c => c.Manufacturer.ToUpper())  
        .OrderBy(g => g.Key);
```

```
C:\Windows\system32\cmd.exe  
PAGANI AUTOMOBILI S.P.A.  
    Huayra Coupe : 13  
PORSCHE  
    Boxster : 26  
    Cayman : 26  
RAM  
    1500 HFE 4X2 : 24  
    Promaster City : 24  
ROLLS-ROYCE MOTOR CARS LIMITED  
    Wraith : 15  
    Ghost : 15  
ROUSH  
    Roush Stage 3 Mustang : 16  
    Roush Stage 3 Mustang : 16  
SCION  
    iA : 37  
    iA : 35  
SUBARU  
    IMPREZA : 31  
    IMPREZA SPORT : 31  
TOYOTA  
    PRIUS Eco : 56  
    PRIUS : 52  
VOLKSWAGEN  
    Jetta Hybrid : 44  
    Jetta : 33  
VOLVO  
    S60 FWD : 30  
    S60 Inscription FWD : 29  
Press any key to continue . . .
```

Групування ієрархічних даних за допомогою GroupJoin()

- Оператор пропонує відразу з'єднання та групування.
 - Починатимемо запис запиту з того об'єкта, за яким буде відбуватись групування, - manufacturers.
 - Оператор GroupJoin() все ще застосовує ключове слово join, проте в парі з ключовим словом into.

```
var query =
    from manufacturer in manufacturers
    join car in cars on manufacturer.Name equals car.Manufacturer
    into carGroup
    orderby manufacturer.Name
    select new
    {
        Manufacturer = manufacturer,
        Cars = carGroup
    };

foreach (var group in query)
{
    Console.WriteLine($"{group.Manufacturer.Name}:{group.Manufacturer.Headquarters}");
    foreach (var car in group.Cars.OrderByDescending(c => c.Combined).Take(2))
    {
        Console.WriteLine($"{car.Name} : {car.Combined}");
    }
}
```

```
C:\Windows\system32\cmd.exe
Pagani Automobili S.p.A.:Italy
  Huayra Coupe : 13
Porsche:Germany
  Boxster : 26
  Cayman : 26
RAM:USA
  1500 HFE 4X2 : 24
  Promaster City : 24
Rolls-Royce Motor Cars Limited:UK
  Wraith : 15
  Ghost : 15
Roush:USA
  Roush Stage 3 Mustang : 16
  Roush Stage 3 Mustang : 16
SCION:USA
  iA : 37
  iA : 35
Subaru:Japan
  IMPREZA : 31
  IMPREZA SPORT : 31
TOYOTA:Japan
  PRIUS Eco : 56
  PRIUS : 52
Volkswagen:Germany
  Jetta Hybrid : 44
  Jetta : 33
Volvo:Sweden
  S60 FWD : 30
  S60 Inscription FWD : 29
Press any key to continue . . .
```

Групування ієрархічних даних за допомогою GroupJoin()

- Синтаксис методів розширення:

```
var query2 =
    manufacturers.GroupJoin(cars, m => m.Name, c => c.Manufacturer, )
```

▲ 1 of 2 ▼ (extension) IEnumerable<TResult> IEnumerable<Manufacturer>.GroupJoin<Manufacturer, TInner, TKey, TResult>(IEnumerable<TInner> inner, Func<Manufacturer, TKey> outerKeySelector, Func<TInner, TKey> innerKeySelector, Func<Manufacturer, IEnumerable<TInner>, TResult> resultSelector)

Correlates the elements of two sequences based on equality of keys and groups the results. The default equality comparer is used to compare keys.
resultSelector: A function to create a result element from an element from the first sequence and a collection of matching elements from the second sequence.

```
var query2 =
    manufacturers.GroupJoin(cars, m => m.Name, c => c.Manufacturer,
        (m, g) =>
            new
            {
                Manufacturer = m,
                Cars = g
            })
    .OrderBy(m => m.Manufacturer.Name);

foreach (var group in query2)
{
    Console.WriteLine($"{group.Manufacturer.Name}:{group.Manufacturer.Headquarters}");
    foreach (var car in group.Cars.OrderByDescending(c => c.Combined).Take(2))
    {
        Console.WriteLine($"{car.Name} : {car.Combined}");
    }
}
```


Групування найефективніших автомобілів за країною

- Способів вирішити задачу багато. Наприклад, допишемо групування до існуючого запиту:
 - Сгруповані дані тепер не матимуть атрибутів Manufacturer і Car.
 - Тепер буде Key (критерій групування – назва країни).
 - Крім того, доведеться сплюснути (flatten) групи, щоб обрати трійки найефективніших автомобілів:

```
500 : 30
UK
  Range Rover Sport TDV6 : 25
  Range Rover TDV6 : 25
  XF : 24
Germany
  Jetta Hybrid : 44
  328d : 36
  Smart fortwo (COUPE) : 36
USA
  MALIBU : 46
  FUSION HYBRID FWD : 42
  C-MAX Hybrid FWD : 40
Japan
  PRIUS Eco : 56
  PRIUS : 52
  PRIUS c : 50
South Korea
  Sonata HYBRID SE : 42
  Sonata HYBRID : 41
  Optima HYBRID : 38
Hong Kong
  Q70 HYBRID : 31
  Q50 HYBRID : 30
  Q50 HYBRID AWD : 28
Sweden
  S60 FWD : 30
  S60 Inscription FWD : 29
  S80 FWD : 29
Press any key to continue . . .
```

```
var query =
    from manufacturer in manufacturers
    join car in cars on manufacturer.Name equals car.Manufacturer
    into carGroup
select new
{
    Manufacturer = manufacturer,
    Cars = carGroup
} into result
group result by result.Manufacturer.Headquarters;

foreach (var group in query)
{
    Console.WriteLine($"{group.Key}");
    foreach (var car in group.SelectMany(g => g.Cars)
        .OrderByDescending(c => c.Combined)
        .Take(3))
    {
        Console.WriteLine($"{car.Name} : {car.Combined}");
    }
}
```

Групування найефективніших автомобілів за країною

- Реалізація за допомогою синтаксису методів розширення:

```
var query2 =
    manufacturers.GroupJoin(cars, m => m.Name, c => c.Manufacturer,
        (m, g) =>
            new
            {
                Manufacturer = m,
                Cars = g
            })
    .GroupBy(m => m.Manufacturer.Headquarters);

foreach (var group in query2)
{
    Console.WriteLine($"{group.Key}");
    foreach (var car in group.SelectMany(g => g.Cars)
        .OrderByDescending(c => c.Combined)
        .Take(3))
    {
        Console.WriteLine($"  \t{car.Name} : {car.Combined}");
    }
}
```

```
C:\Windows\system32\cmd.exe
500 : 30
UK
  Range Rover Sport TDV6 : 25
  Range Rover TDV6 : 25
  XF : 24
Germany
  Jetta Hybrid : 44
  328d : 36
  Smart fortwo (COUPE) : 36
USA
  MALIBU : 46
  FUSION HYBRID FWD : 42
  C-MAX Hybrid FWD : 40
Japan
  PRIUS Eco : 56
  PRIUS : 52
  PRIUS c : 50
South Korea
  Sonata HYBRID SE : 42
  Sonata HYBRID : 41
  Optima HYBRID : 38
Hong Kong
  Q70 HYBRID : 31
  Q50 HYBRID : 30
  Q50 HYBRID AWD : 28
Sweden
  S60 FWD : 30
  S60 Inscription FWD : 29
  S80 FWD : 29
Press any key to continue . . .
```


Агрегування даних

- Здійснюється за допомогою операторів, які здійснюють обчислення: `Count()`, `Sum()`, `Min()`, `Max()`, `Average()`.
 - Навіть групування є формою агрегування даних.
 - Потрібні для виводу певної статистики, резюме щодо даних.
- Виведемо мінімум, максимум та середнє значення показника ефективності `Combined` для кожного з виробників.

```
var query =  
    from car in cars  
    group car by car.Manufacturer into carGroup  
    select new  
    {  
        Name = carGroup.Key,  
        Max = carGroup.Max(c => c.Combined),  
        Min = carGroup.Min(c => c.Combined),  
        Avg = carGroup.Average(c => c.Combined)  
    };
```

```
foreach (var result in query)  
{  
    Console.WriteLine($"{result.Name}");  
    Console.WriteLine($" \t Max: {result.Max}");  
    Console.WriteLine($" \t Min: {result.Min}");  
    Console.WriteLine($" \t Avg: {result.Avg}");  
}
```

```
C:\Windows\system32\cmd.exe  
Avg: 24  
Chrysler  
    Max: 28  
    Min: 19  
    Avg: 22.545454545454545  
Lincoln  
    Max: 40  
    Min: 16  
    Avg: 20.8260869565217  
GMC  
    Max: 26  
    Min: 12  
    Avg: 18.7906976744186  
RAM  
    Max: 24  
    Min: 15  
    Avg: 19.818181818181818  
CHEVROLET  
    Max: 25  
    Min: 25  
    Avg: 25  
Jeep  
    Max: 27  
    Min: 15  
    Avg: 22.7105263157895  
Land Rover  
    Max: 25  
    Min: 16  
    Avg: 19.7272727272727  
Press any key to continue . . .
```

Агрегування даних

- Будемо впорядковувати виробників за найефективнішими автомобілями.
 - Зверніть увагу, що LINQ буде тричі проходити набір даних у carGroup, щоб знайти максимум, мінімум та середнє значення.

```
var query =  
    from car in cars  
    group car by car.Manufacturer into carGroup  
    select new  
    {  
        Name = carGroup.Key,  
        Max = carGroup.Max(c => c.Combined),  
        Min = carGroup.Min(c => c.Combined),  
        Avg = carGroup.Average(c => c.Combined)  
    } into result  
    orderby result.Max descending  
    select result;
```

```
C:\Windows\system32\cmd.exe  
TOYOTA  
    Max: 56  
    Min: 14  
    Avg: 25.92  
Chevrolet  
    Max: 46  
    Min: 12  
    Avg: 23.3953488372093  
Volkswagen  
    Max: 44  
    Min: 19  
    Avg: 27.151515151515152  
Ford  
    Max: 42  
    Min: 16  
    Avg: 22.8604651162791  
HYUNDAI MOTOR COMPANY  
    Max: 42  
    Min: 18  
    Avg: 25.9302325581395  
LEXUS  
    Max: 42  
    Min: 15  
    Avg: 24.0285714285714  
Lincoln  
    Max: 40  
    Min: 16  
    Avg: 20.8260869565217  
KIA MOTORS CORPORATION  
    Max: 38
```

Агрегування даних

- Відповідний синтаксис методів розширення може бути більш ефективним, оскільки пропонує єдиний метод `Aggregate()` для обчислення максимуму, мінімуму та середнього значення. Візьмемо його версію з наступними параметрами:
 - Початковий стан акумулятора (збирача даних для статистичних обчислень).
 - Лямбда-вираз, який визначатиме взаємодію акумулятора з об'єктом `Car` та надання акумулятору даних для оновлення статистики. Викликається один раз для кожного елемента послідовності.
 - Лямбда-вираз, який повертатиме результат на основі даних акумулятора.

```
var query2 =  
    cars.GroupBy(c => c.Manufacturer)  
        .Select(g =>  
            {  
                var results = g.Aggregate()            })
```

▲ 3 of 3 ▼ (extension) `TResult IEnumerable<Car>.Aggregate<Car, TAccumulate, TResult>(TAccumulate seed, Func<TAccumulate, Car, TAccumulate> func, Func<TAccumulate, TResult> resultSelector)`
Applies an accumulator function over a sequence. The specified seed value is used as the initial accumulator value, and the specified function is used to select the result value.
seed: The initial accumulator value.

Агрегування даних

```
public class CarStatistics
{
    public CarStatistics()
    {
        Max = Int32.MinValue;
        Min = Int32.MaxValue;
    }
    public CarStatistics Accumulate(Car car)
    {
        Count += 1;
        Total += car.Combined;
        Max = Math.Max(Max, car.Combined);
        Min = Math.Min(Min, car.Combined);

        return this;
    }

    public CarStatistics Compute()
    {
        Avg = (double)Total / Count;
        return this;
    }

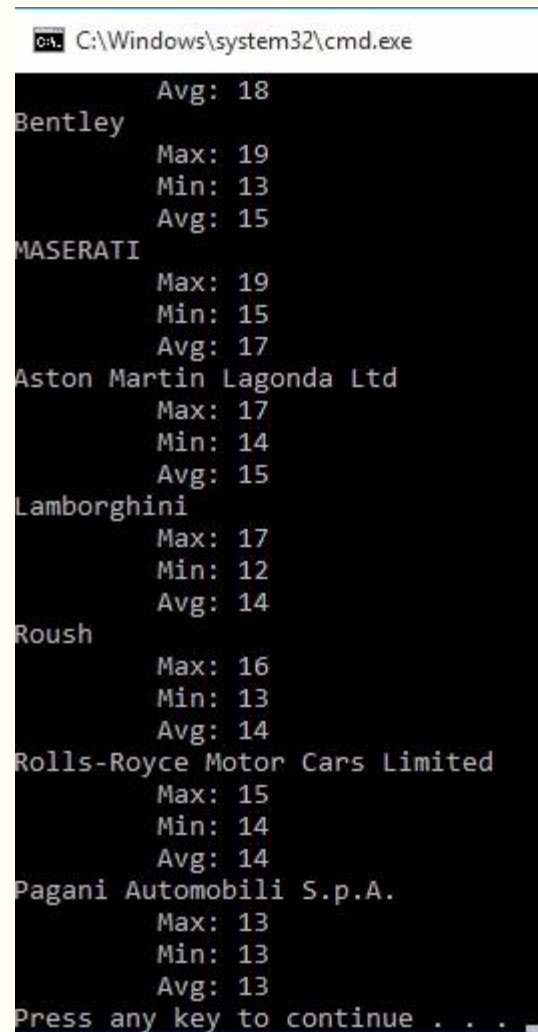
    public int Max { get; set; }
    public int Min { get; set; }
    public int Total { get; set; }
    public int Count { get; set; }
    public double Avg { get; set; }
}
24.11.2020
```

- Для спрощення роботи з методом розширення `Aggregate()` можна створити агрегуючий клас, тут назовемо його `CarStatistics`:
 - Другий параметр методу потребуватиме іншого методу, що пов'яже акумулятор з об'єктами типу `Car`.
 - Створимо в нашому класі спеціальний метод `Accumulate()` та введемо 2 властивості для проміжних обчислень: `Total` і `Count`.
 - На початку максимум проініціалізуємо найменшим для типу значенням, щоб решта значень були завжди більшими.
 - Мінімум проініціалізуємо найбільшим для типу значенням, щоб решта значень були меншими.
 - У методі `Accumulate` оновлюємо статистичні дані та проміжні значення.
 - Третій параметр методу `Aggregate()` повинен надавати результат обчислень, тому в нашому класі введемо метод `Compute()`, який сформує результат усереднення даних.

Агрегування даних (вигляд запиту)

```
var query2 =
    cars.GroupBy(c => c.Manufacturer)
        .Select(g =>
            {
                var results = g.Aggregate(new CarStatistics(),
                    (acc, c) => acc.Accumulate(c),
                    acc => acc.Compute());

                return new
                {
                    Name = g.Key,
                    Avg = results.Average,
                    Min = results.Min,
                    Max = results.Max
                };
            })
        .OrderByDescending(r => r.Max);
```



```
C:\Windows\system32\cmd.exe
Avg: 18
Bentley
Max: 19
Min: 13
Avg: 15
MASERATI
Max: 19
Min: 15
Avg: 17
Aston Martin Lagonda Ltd
Max: 17
Min: 14
Avg: 15
Lamborghini
Max: 17
Min: 12
Avg: 14
Roush
Max: 16
Min: 13
Avg: 14
Rolls-Royce Motor Cars Limited
Max: 15
Min: 14
Avg: 14
Pagani Automobili S.p.A.
Max: 13
Min: 13
Avg: 13
Press any key to continue . . .
```



ДЯКУЮ ЗА УВАГУ!

Наступна тема: Основи модульного тестування об'єктно-орієнтованого коду
