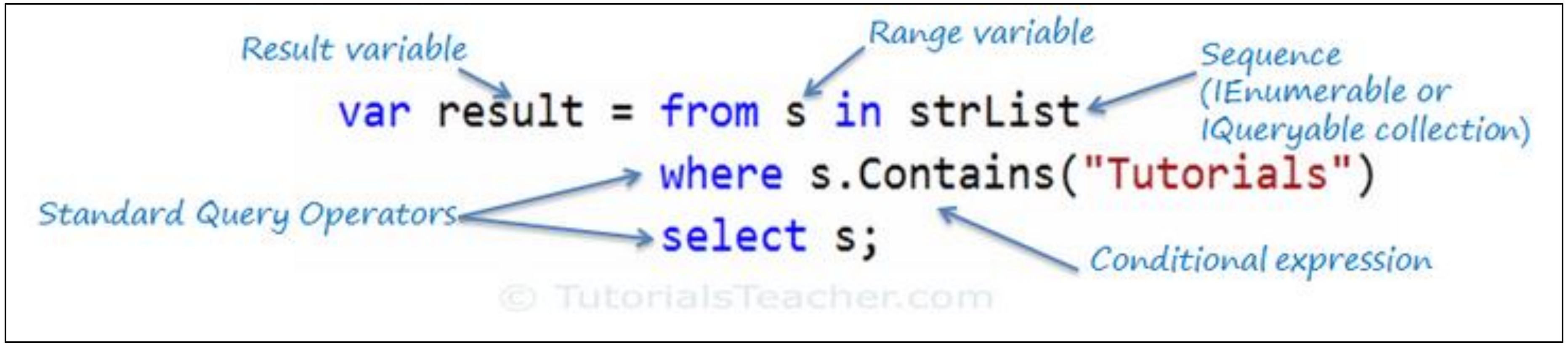
Linq

(bron : https://www.tutorialsteacher.com/linq/linq-tutorials)

Query syntax

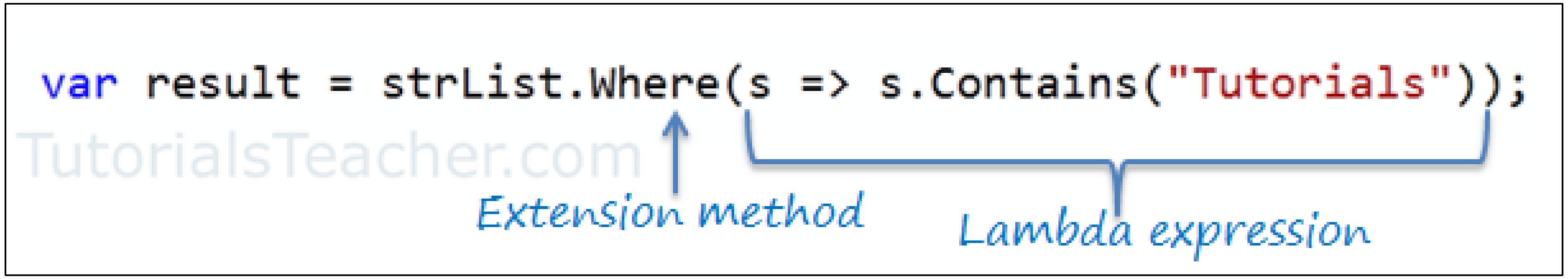


```
var result = from s in strList
             where s.Contains("Tutorials")
             select s;
```

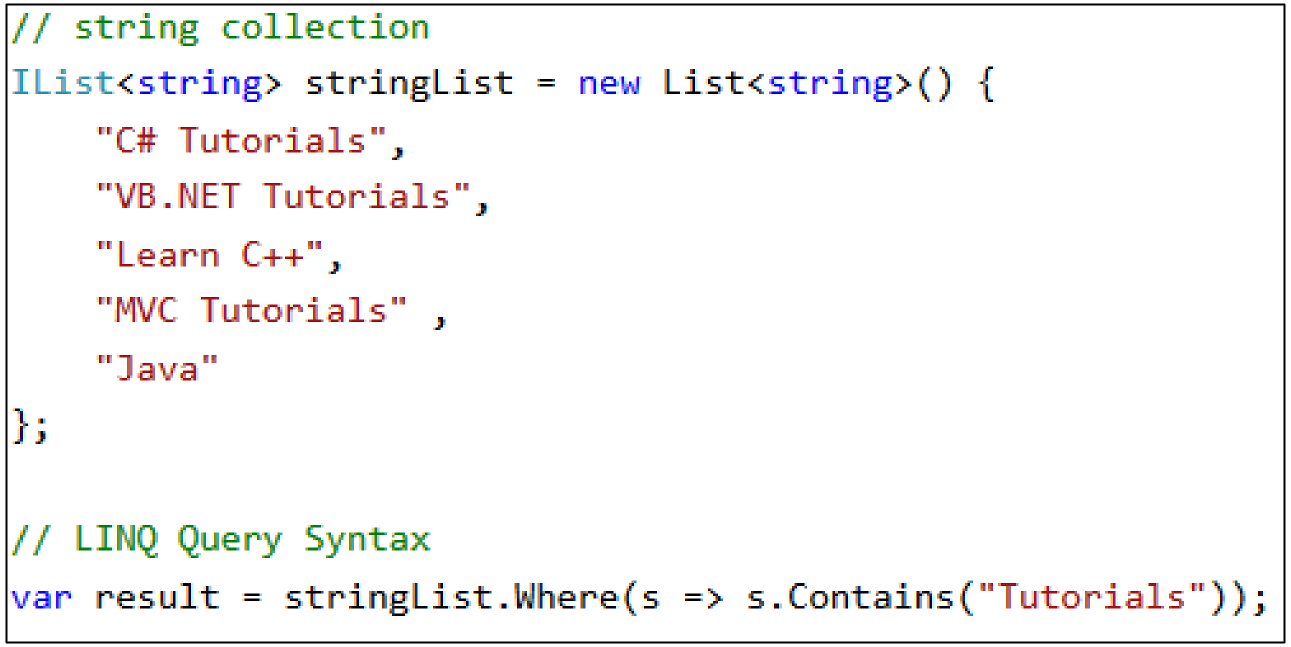
```
// string collection
IList<string> stringList = new List<string>() {
    "C# Tutorials",
    "VB.NET Tutorials",
    "Learn C++",
    "MVC Tutorials" ,
    "Java"
};

// LINQ Query Syntax
var result = from s in stringList
            where s.Contains("Tutorials")
            select s;
```

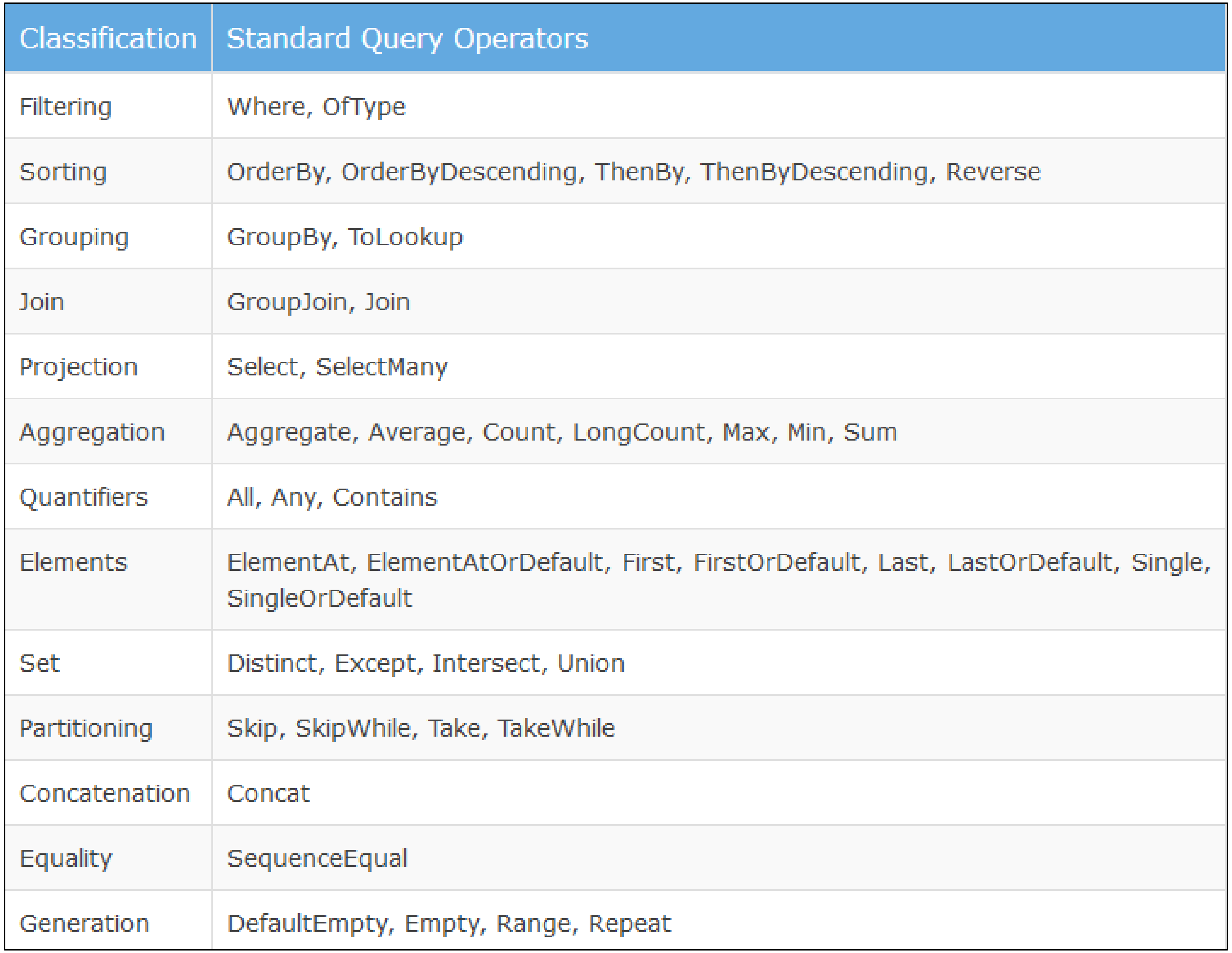
Method syntax (fluent syntax)


```
var result = strList.Where(s => s.Contains("Tutorials"));
```

TutorialsTeacher.com


```
// string collection
IList<string> stringList = new List<string>() {
    "C# Tutorials",
    "VB.NET Tutorials",
    "Learn C++",
    "MVC Tutorials" ,
    "Java"
};

// LINQ Query Syntax
var result = stringList.Where(s => s.Contains("Tutorials"));
```



Classification	Standard Query Operators
Filtering	Where, OfType
Sorting	OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse
Grouping	GroupBy, ToLookup
Join	GroupJoin, Join
Projection	Select, SelectMany
Aggregation	Aggregate, Average, Count, LongCount, Max, Min, Sum
Quantifiers	All, Any, Contains
Elements	ElementAt, ElementAtOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
Set	Distinct, Except, Intersect, Union
Partitioning	Skip, SkipWhile, Take, TakeWhile
Concatenation	Concat
Equality	SequenceEqual
Generation	DefaultEmpty, Empty, Range, Repeat

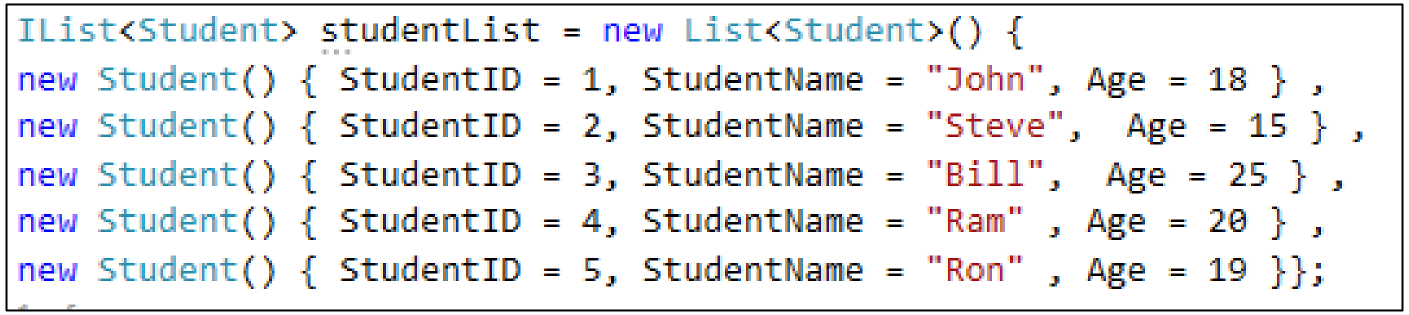
Where

- Gebruikt om te filteren op basis van een expressie – deze expressie kan zowel een lambda expressie zijn als een Func delegate.
- Overloads :

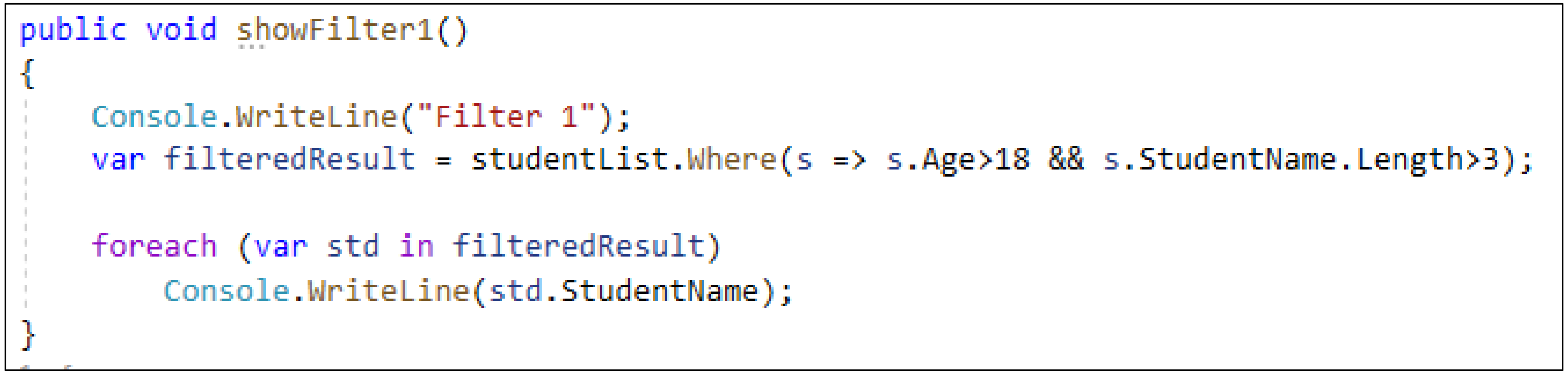
```
public static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source,
Func<TSource, bool> predicate);

public static IEnumerable<TSource>
Where<TSource>(this IEnumerable<TSource> source, Func<TSource, int, bool> predicate);
```

Where

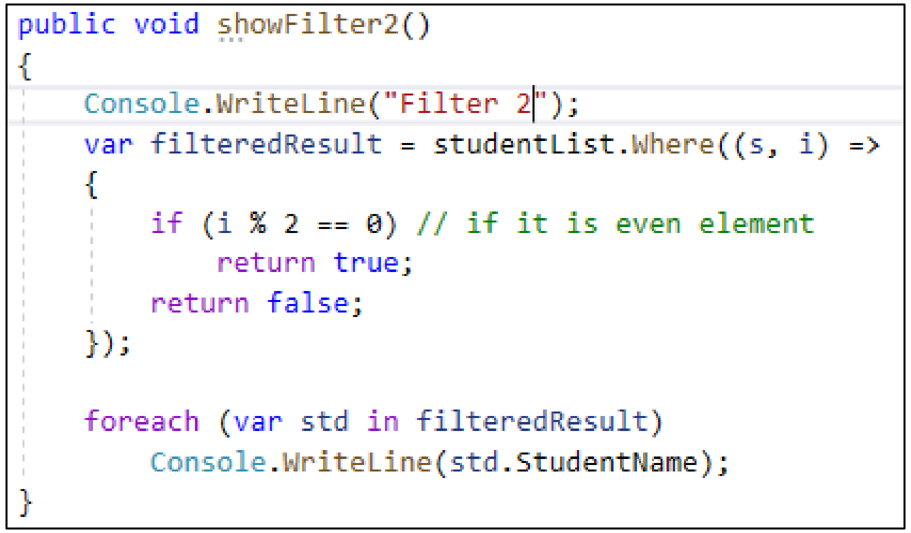


```
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 }};
```



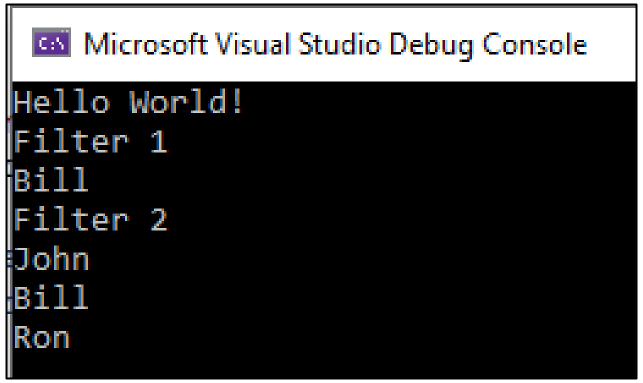
```
public void showFilter1()
{
    Console.WriteLine("Filter 1");
    var filteredResult = studentList.Where(s => s.Age>18 && s.StudentName.Length>3);

    foreach (var std in filteredResult)
        Console.WriteLine(std.StudentName);
}
```



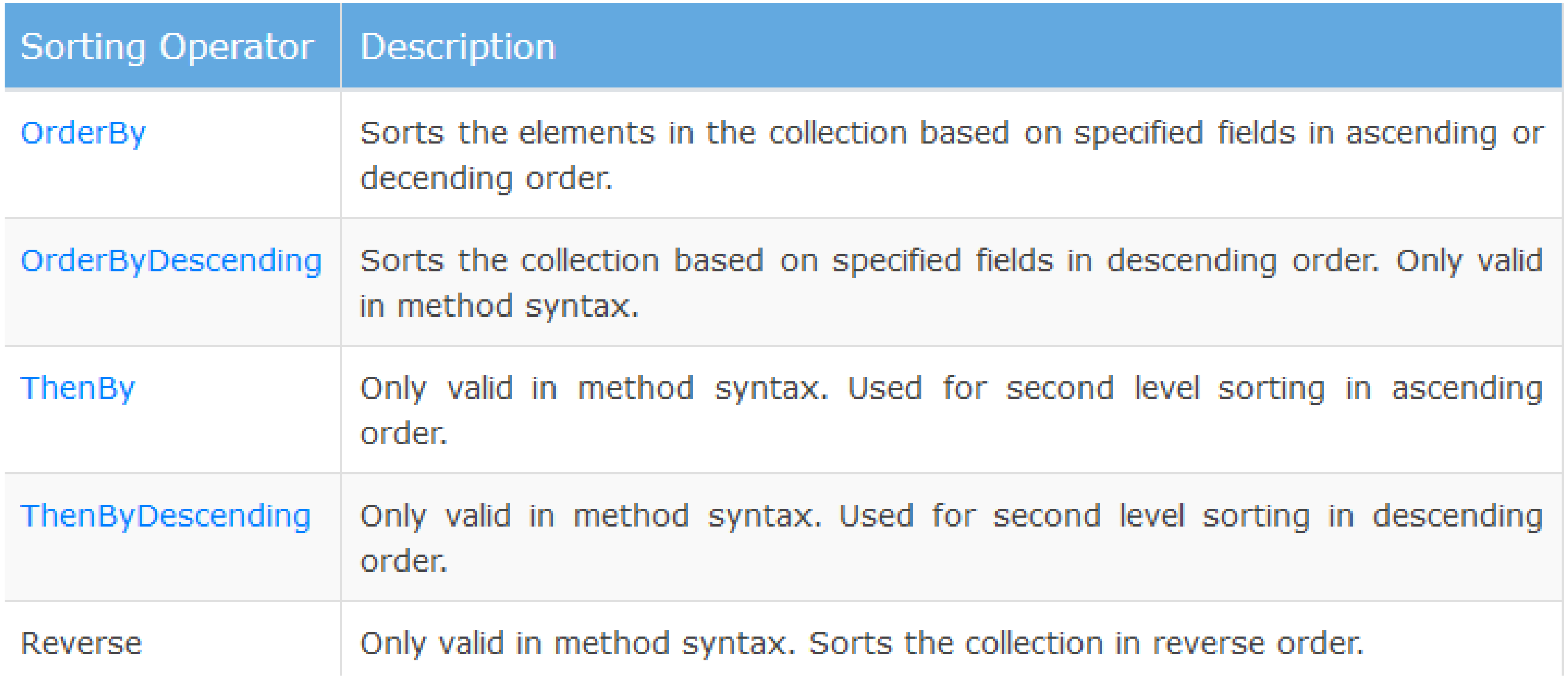
```
public void showFilter2()
{
    Console.WriteLine("Filter 2");
    var filteredResult = studentList.Where((s, i) =>
    {
        if (i % 2 == 0) // if it is even element
            return true;
        return false;
    });

    foreach (var std in filteredResult)
        Console.WriteLine(std.StudentName);
}
```



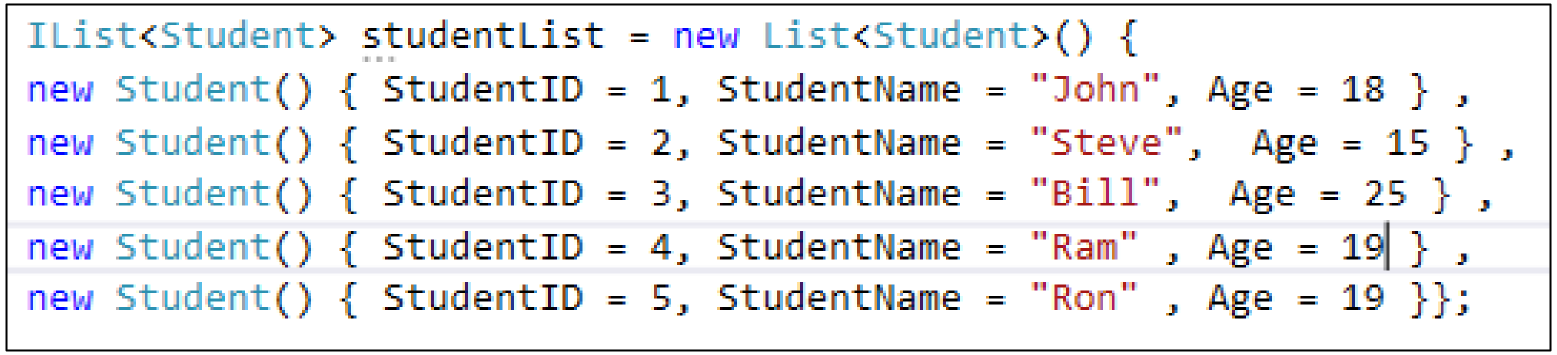
```
Hello World!
Filter 1
Bill
Filter 2
John
Bill
Ron
```

Ordering

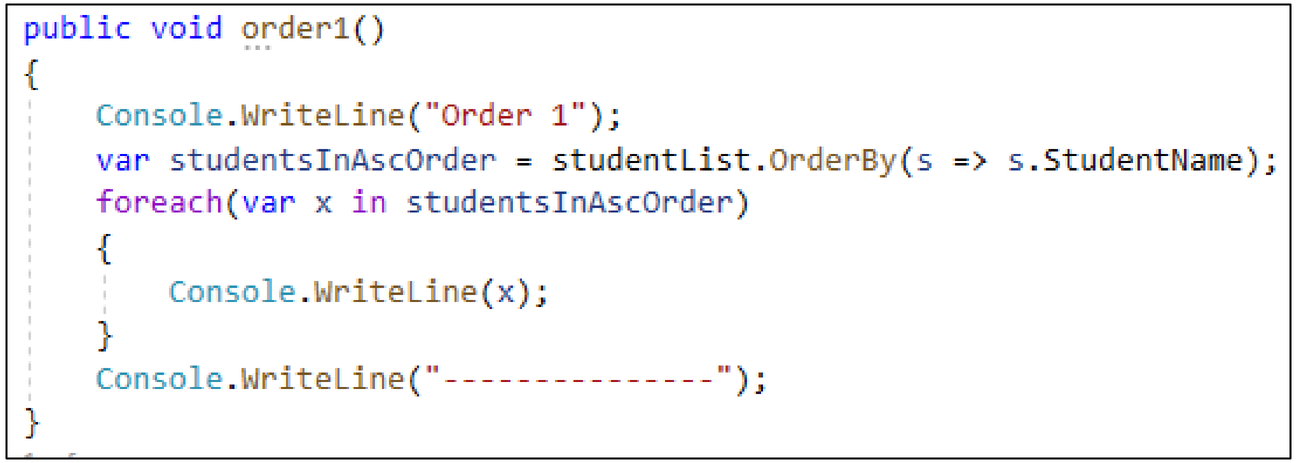


Sorting Operator	Description
OrderBy	Sorts the elements in the collection based on specified fields in ascending or decending order.
OrderByDescending	Sorts the collection based on specified fields in descending order. Only valid in method syntax.
ThenBy	Only valid in method syntax. Used for second level sorting in ascending order.
ThenByDescending	Only valid in method syntax. Used for second level sorting in descending order.
Reverse	Only valid in method syntax. Sorts the collection in reverse order.

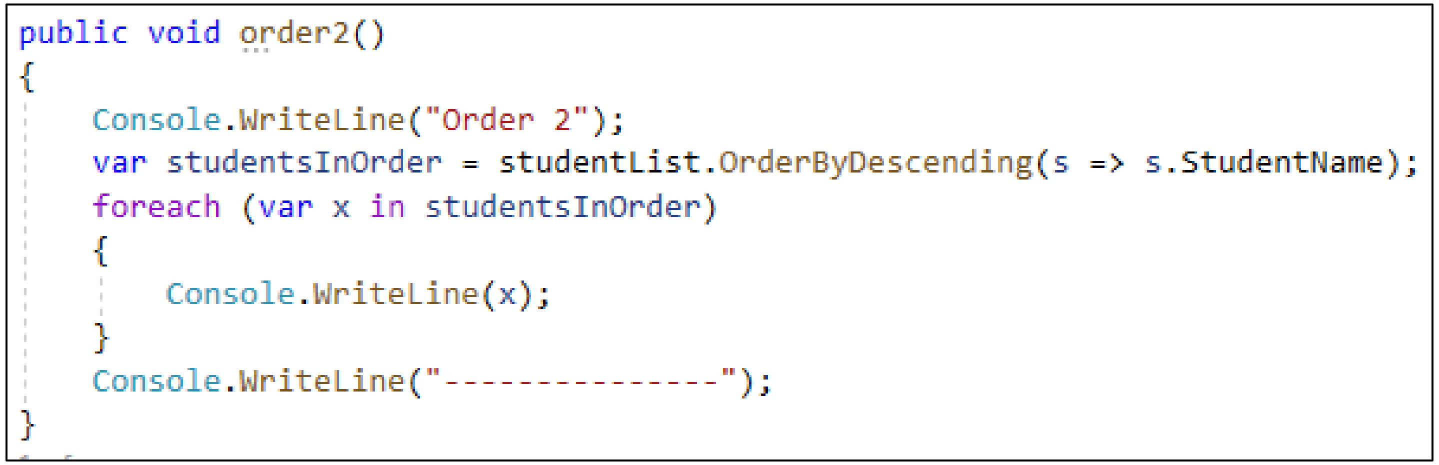
Ordering



```
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 19 } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 }};
```

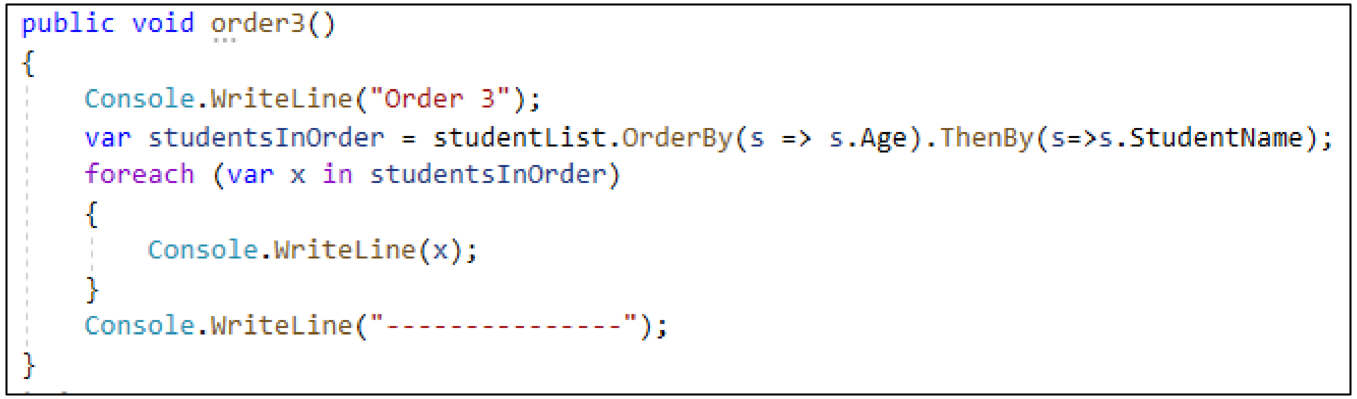


```
public void order1()
{
    Console.WriteLine("Order 1");
    var studentsInAscOrder = studentList.OrderBy(s => s.StudentName);
    foreach(var x in studentsInAscOrder)
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("---------------");
}
```

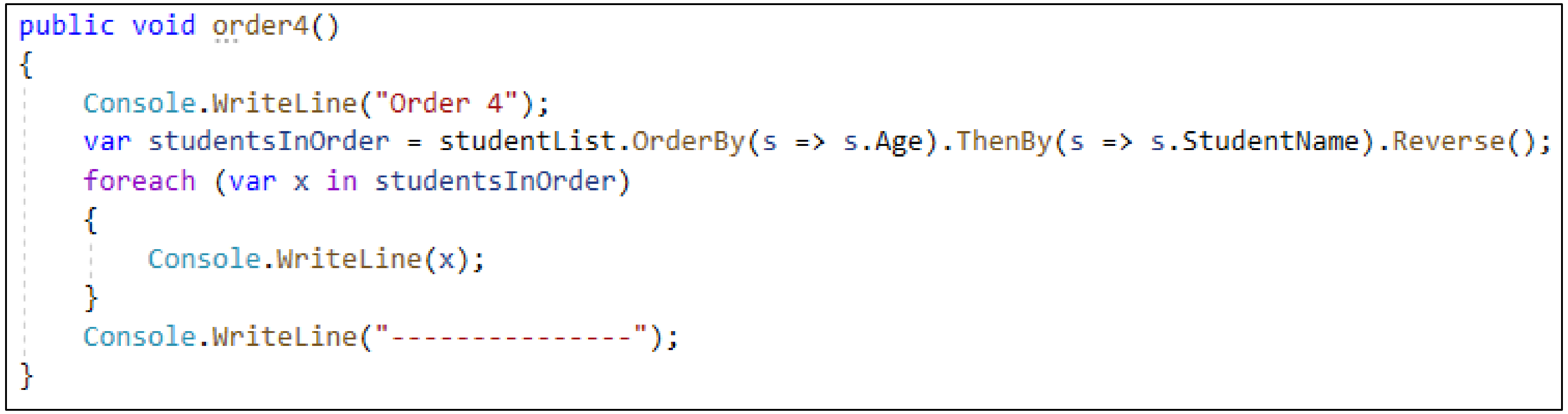


```
public void order2()
{
    Console.WriteLine("Order 2");
    var studentsInOrder = studentList.OrderByDescending(s => s.StudentName);
    foreach (var x in studentsInOrder)
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("---------------");
}
```

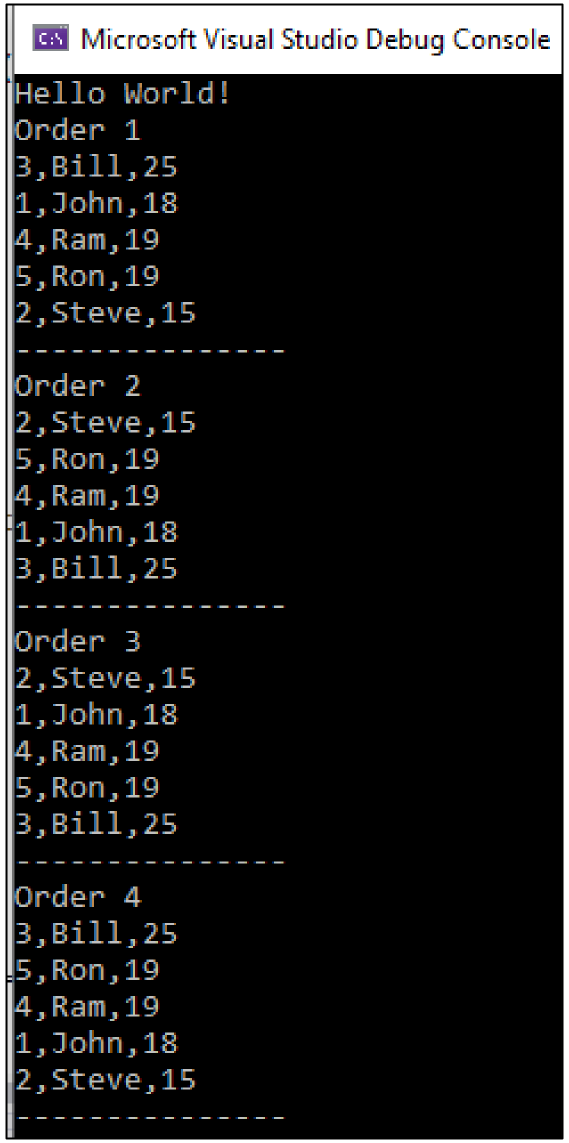
Ordering



```
public void order3()
{
    Console.WriteLine("Order 3");
    var studentsInOrder = studentList.OrderBy(s => s.Age).ThenBy(s=>s.StudentName);
    foreach (var x in studentsInOrder)
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("---------------");
}
```



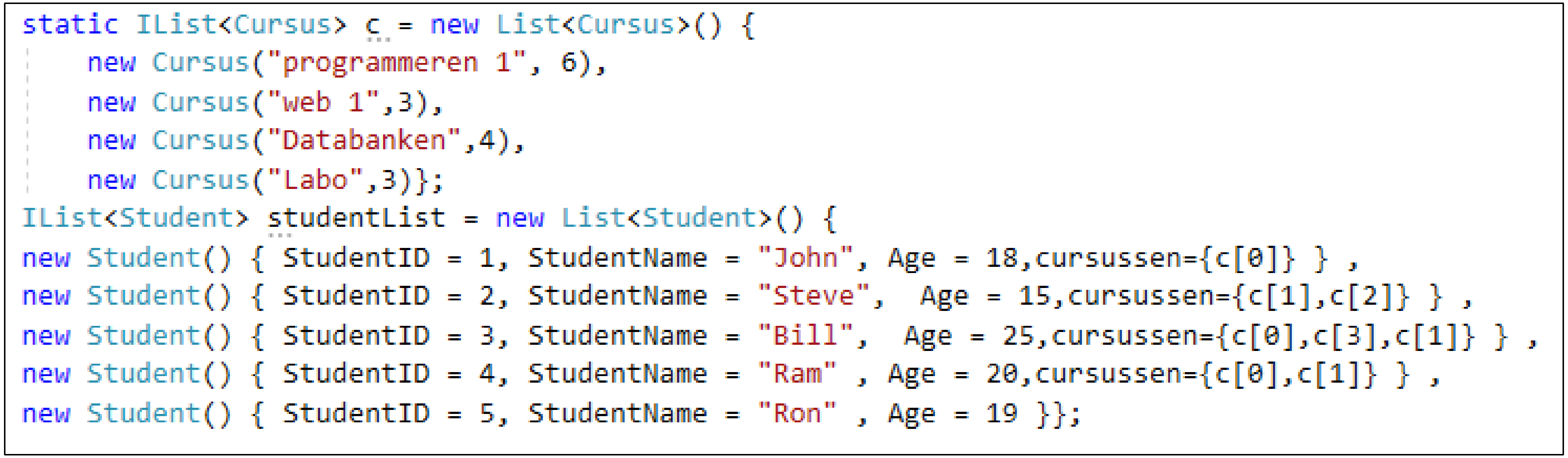
```
public void order4()
{
    Console.WriteLine("Order 4");
    var studentsInOrder = studentList.OrderBy(s => s.Age).ThenBy(s => s.StudentName).Reverse();
    foreach (var x in studentsInOrder)
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("---------------");
}
```



Microsoft Visual Studio Debug Console

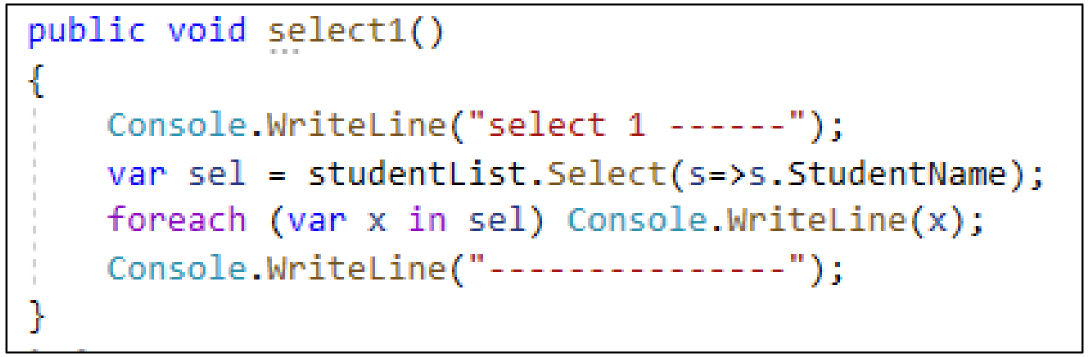
```
Hello World!
Order 1
3,Bill,25
1,John,18
4,Ram,19
5,Ron,19
2,Steve,15
---------------
Order 2
2,Steve,15
5,Ron,19
4,Ram,19
1,John,18
3,Bill,25
---------------
Order 3
2,Steve,15
1,John,18
4,Ram,19
5,Ron,19
3,Bill,25
---------------
Order 4
3,Bill,25
5,Ron,19
4,Ram,19
1,John,18
2,Steve,15
---------------
```

Select

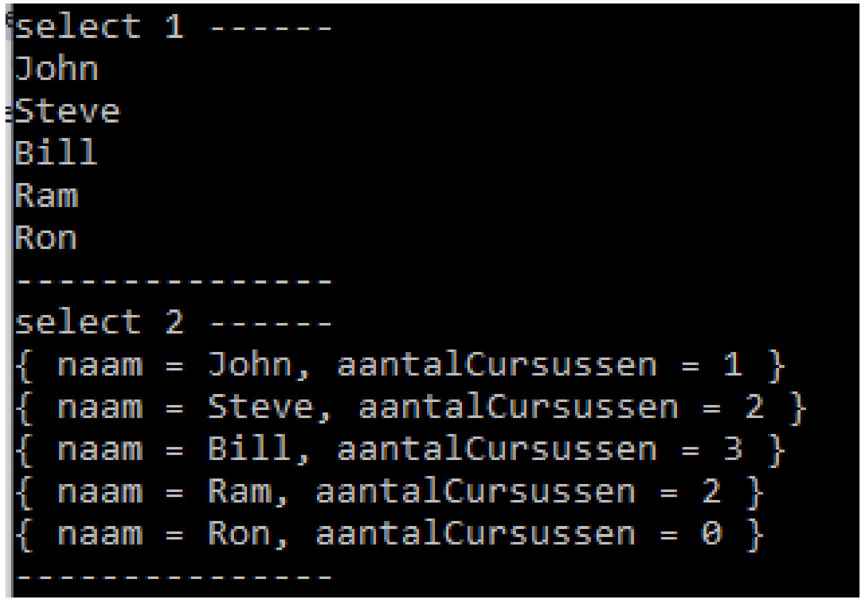


```
static IList<Cursus> c = new List<Cursus>() {
    new Cursus("programmeren 1", 6),
    new Cursus("web 1",3),
    new Cursus("Databanken",4),
    new Cursus("Labo",3)};
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18,cursussen={c[0]} } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15,cursussen={c[1],c[2]} } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25,cursussen={c[0],c[3],c[1]} } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20,cursussen={c[0],c[1]} } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 }};
```

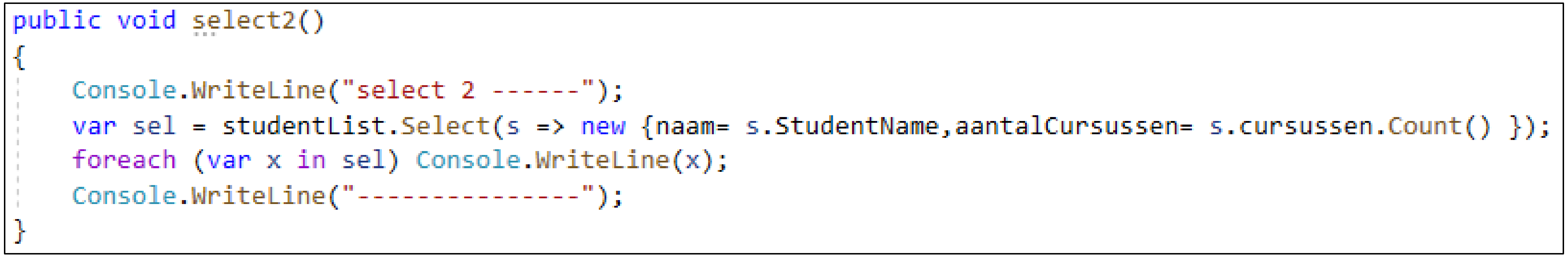
Select



```
public void select1()
{
    Console.WriteLine("select 1 ------");
    var sel = studentList.Select(s=>s.StudentName);
    foreach (var x in sel) Console.WriteLine(x);
    Console.WriteLine("---------------");
}
```

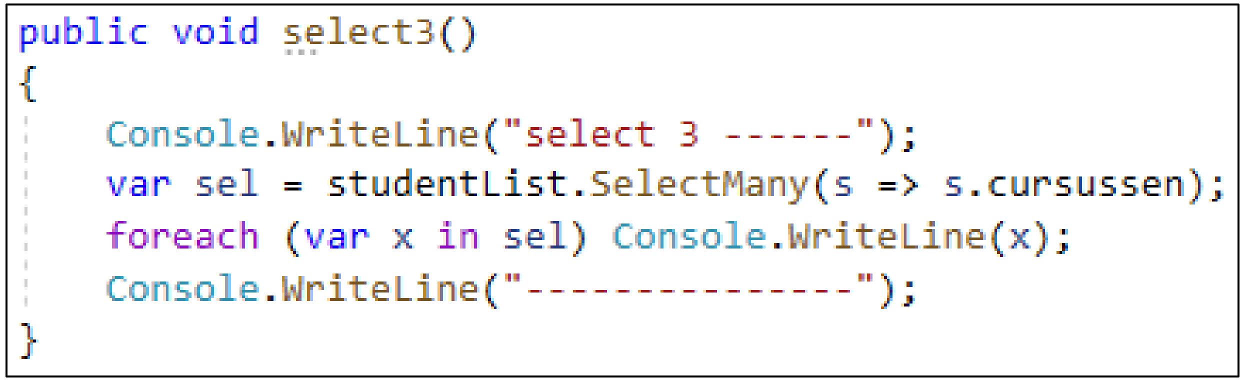


```
select 1 ------
John
Steve
Bill
Ram
Ron
---------------
select 2 ------
{ naam = John, aantalCursussen = 1 }
{ naam = Steve, aantalCursussen = 2 }
{ naam = Bill, aantalCursussen = 3 }
{ naam = Ram, aantalCursussen = 2 }
{ naam = Ron, aantalCursussen = 0 }
---------------
```

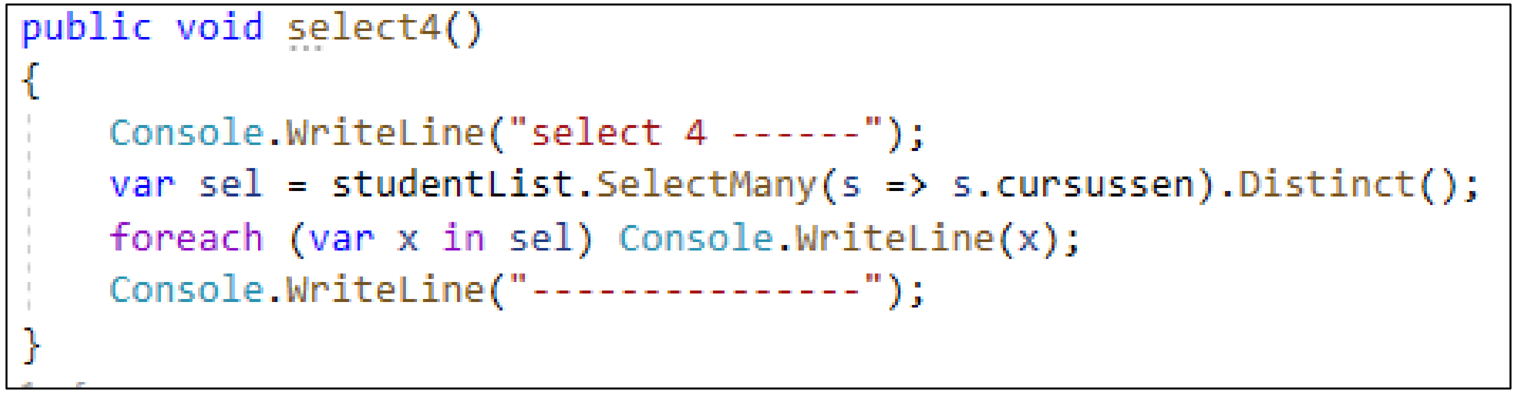


```
public void select2()
{
    Console.WriteLine("select 2 ------");
    var sel = studentList.Select(s => new {naam= s.StudentName,aantalCursussen= s.cursussen.Count() });
    foreach (var x in sel) Console.WriteLine(x);
    Console.WriteLine("---------------");
}
```

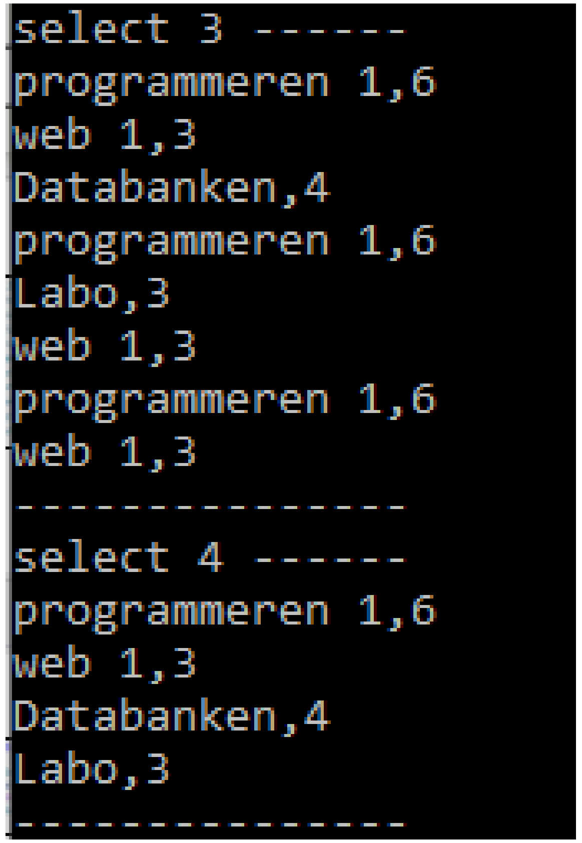
Select



```
public void select3()
{
    Console.WriteLine("select 3 ------");
    var sel = studentList.SelectMany(s => s.cursussen);
    foreach (var x in sel) Console.WriteLine(x);
    Console.WriteLine("---------------");
}
```

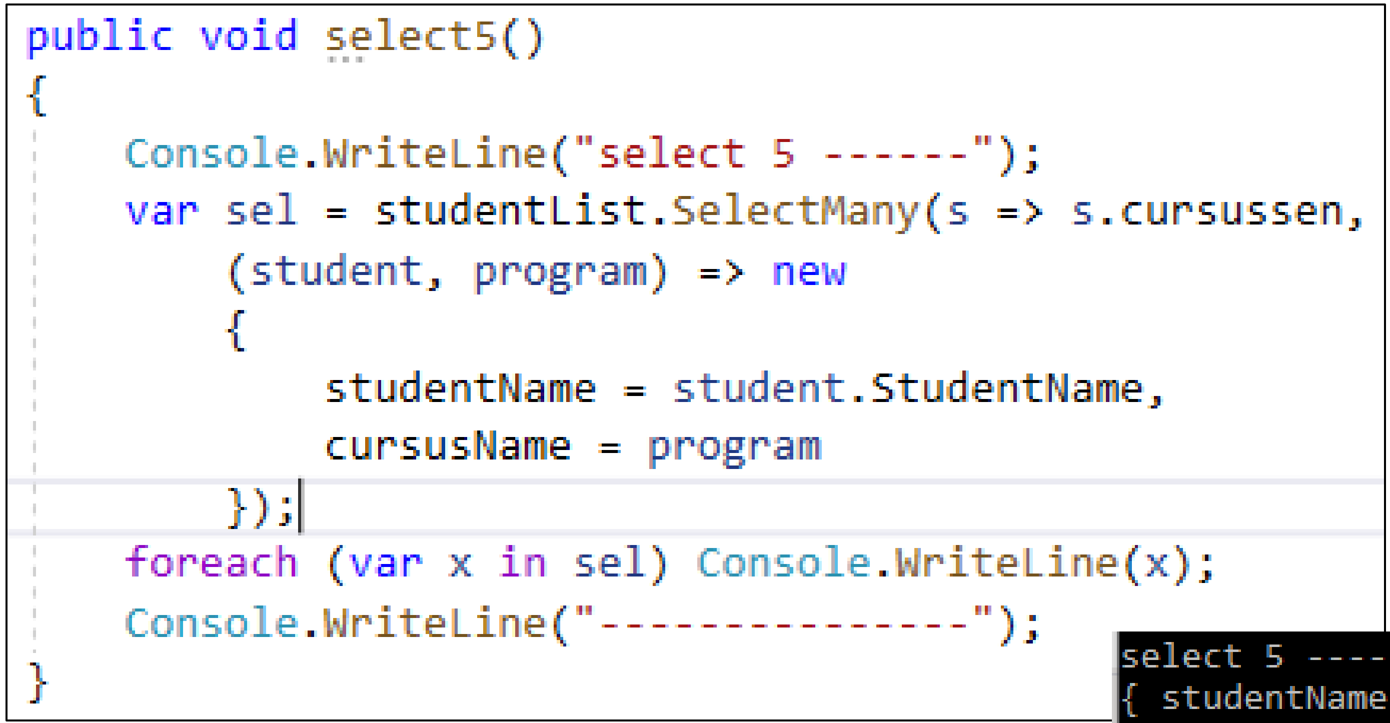


```
public void select4()
{
    Console.WriteLine("select 4 ------");
    var sel = studentList.SelectMany(s => s.cursussen).Distinct();
    foreach (var x in sel) Console.WriteLine(x);
    Console.WriteLine("---------------");
}
```

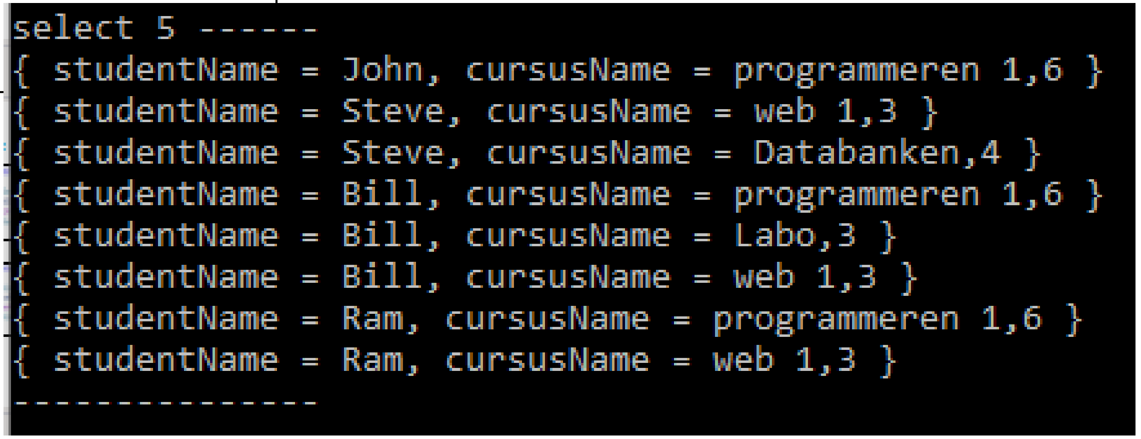


```
select 3 ------
programmeren 1,6
web 1,3
Databanken,4
programmeren 1,6
Labo,3
web 1,3
programmeren 1,6
web 1,3
---------------
select 4 ------
programmeren 1,6
web 1,3
Databanken,4
Labo,3
---------------
```

Select

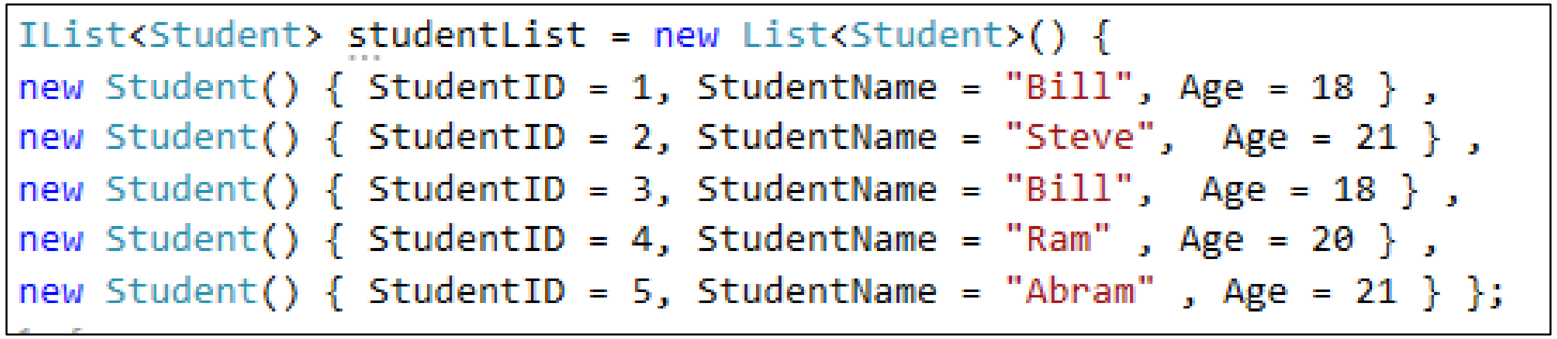


```
public void select5()
{
    Console.WriteLine("select 5 ------");
    var sel = studentList.SelectMany(s => s.cursussen,
        (student, program) => new
        {
            studentName = student.StudentName,
            cursusName = program
        });
    foreach (var x in sel) Console.WriteLine(x);
    Console.WriteLine("---------------");
}
```

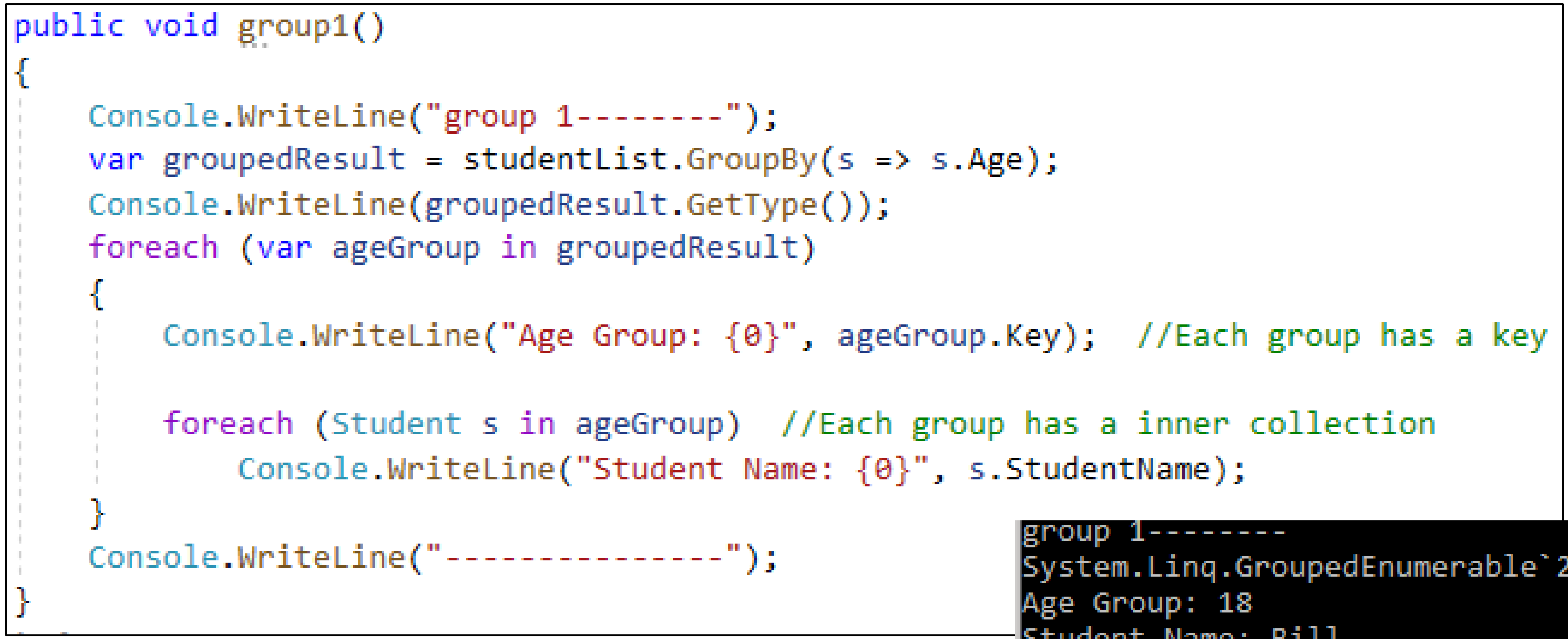


```
select 5 ------
{ studentName = John, cursusName = programmeren 1,6 }
{ studentName = Steve, cursusName = web 1,3 }
{ studentName = Steve, cursusName = Databanken,4 }
{ studentName = Bill, cursusName = programmeren 1,6 }
{ studentName = Bill, cursusName = Labo,3 }
{ studentName = Bill, cursusName = web 1,3 }
{ studentName = Ram, cursusName = programmeren 1,6 }
{ studentName = Ram, cursusName = web 1,3 }
---------------
```

GroupBy / ToLookup

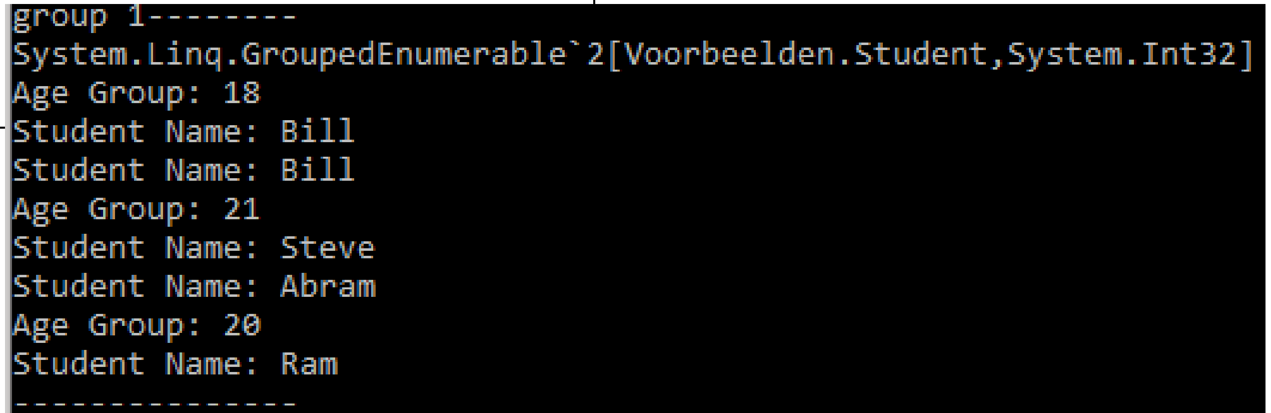


```
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "Bill", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 21 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 18 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
new Student() { StudentID = 5, StudentName = "Abram" , Age = 21 } };
```



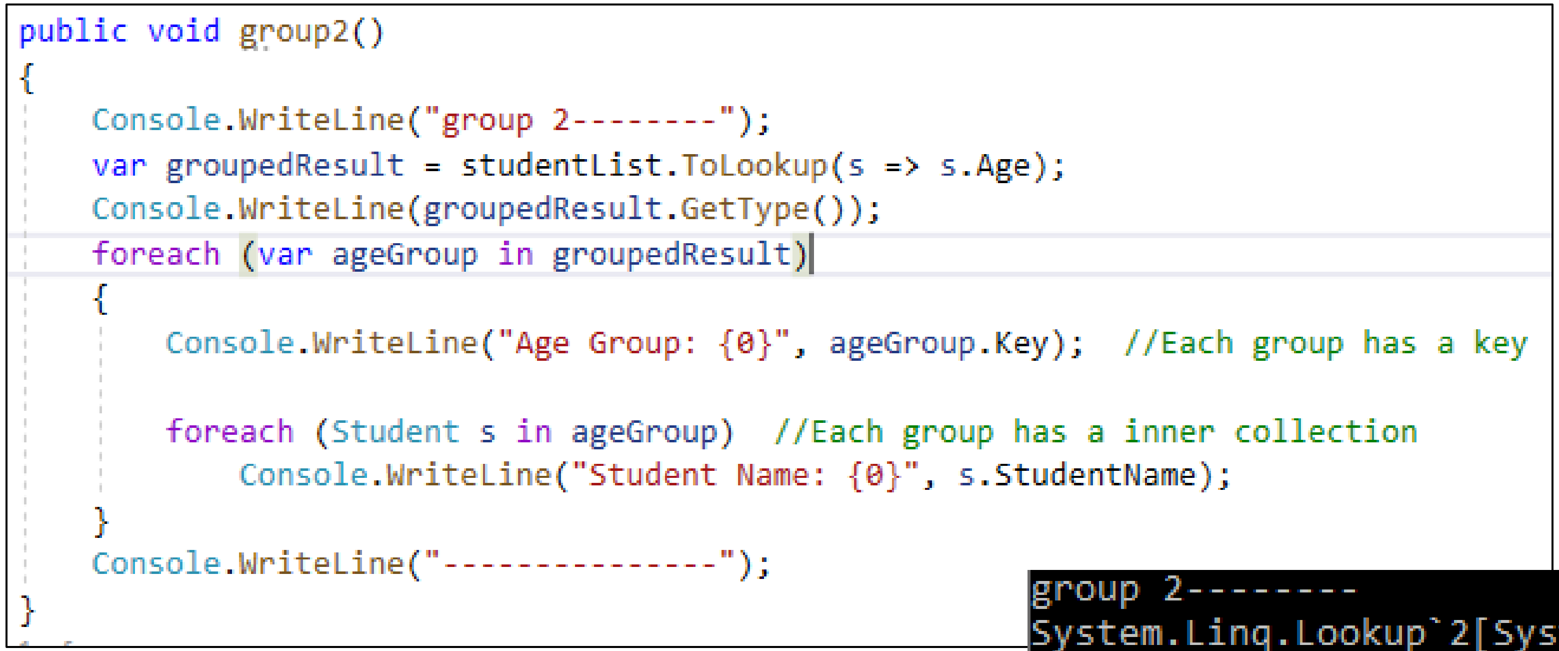
```
public void group1()
{
    Console.WriteLine("group 1--------");
    var groupedResult = studentList.GroupBy(s => s.Age);
    Console.WriteLine(groupedResult.GetType());
    foreach (var ageGroup in groupedResult)
    {
        Console.WriteLine("Age Group: {0}", ageGroup.Key);  //Each group has a key

        foreach (Student s in ageGroup)  //Each group has a inner collection
            Console.WriteLine("Student Name: {0}", s.StudentName);
    }
    Console.WriteLine("---------------");
}
```



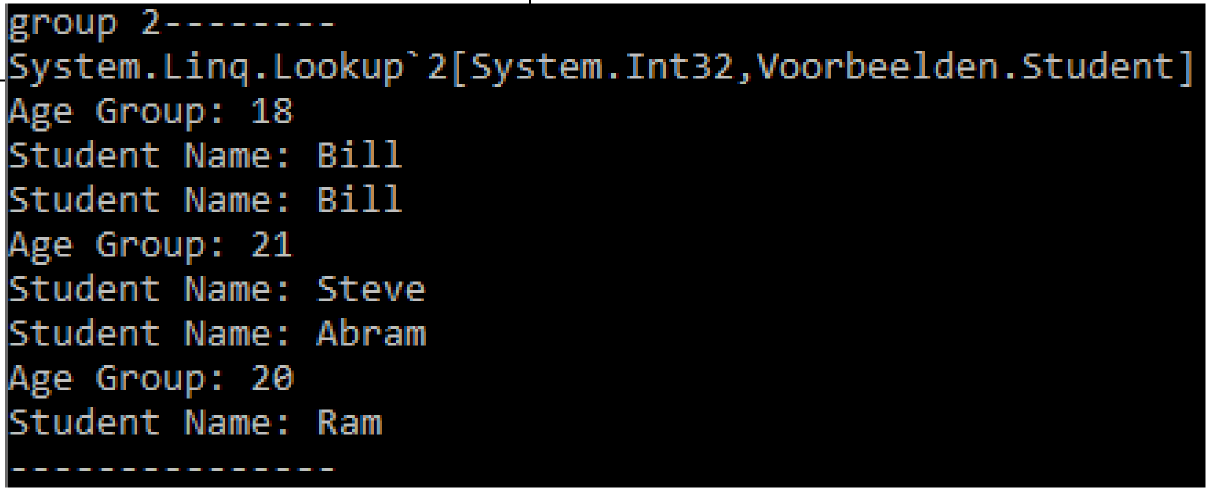
```
group 1--------
System.Linq.GroupedEnumerable`2[Voorbeelden.Student,System.Int32]
Age Group: 18
Student Name: Bill
Student Name: Bill
Age Group: 21
Student Name: Steve
Student Name: Abram
Age Group: 20
Student Name: Ram
---------------
```

GroupBy / ToLookup



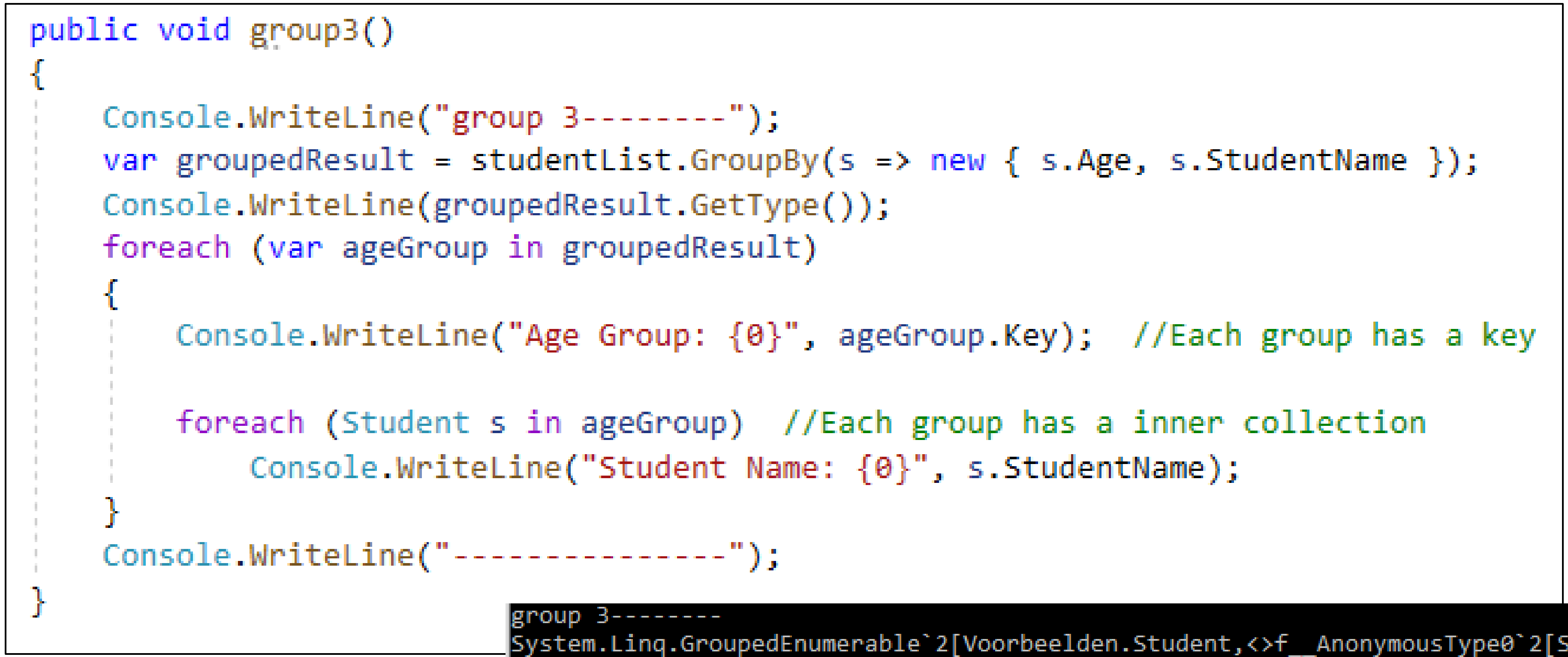
```
public void group2()
{
    Console.WriteLine("group 2--------");
    var groupedResult = studentList.ToLookup(s => s.Age);
    Console.WriteLine(groupedResult.GetType());
    foreach (var ageGroup in groupedResult)
    {
        Console.WriteLine("Age Group: {0}", ageGroup.Key);  //Each group has a key

        foreach (Student s in ageGroup)  //Each group has a inner collection
            Console.WriteLine("Student Name: {0}", s.StudentName);
    }
    Console.WriteLine("---------------");
}
```



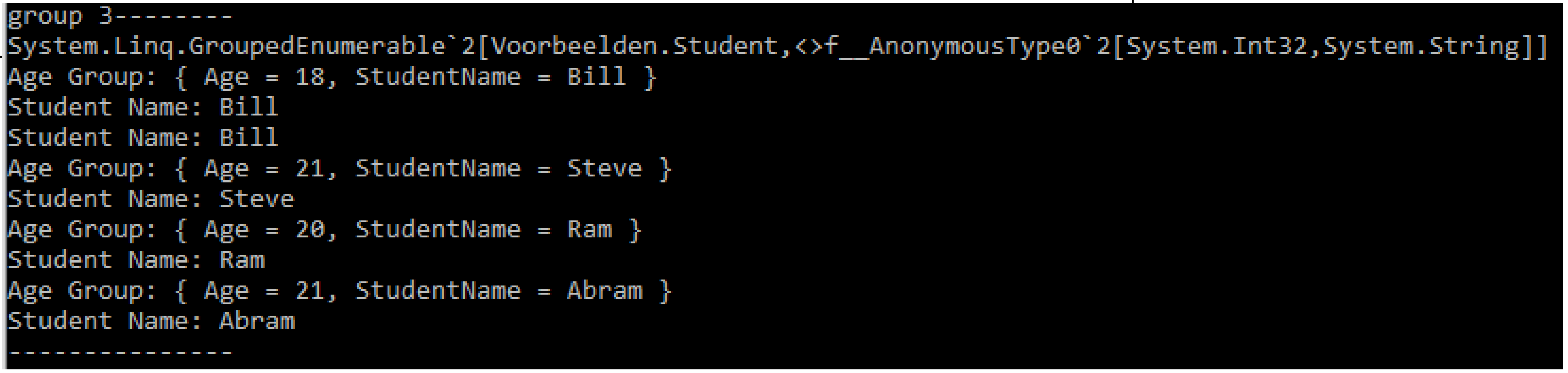
```
group 2--------
System.Linq.Lookup`2[System.Int32,Voorbeelden.Student]
Age Group: 18
Student Name: Bill
Student Name: Bill
Age Group: 21
Student Name: Steve
Student Name: Abram
Age Group: 20
Student Name: Ram
---------------
```

GroupBy / ToLookup



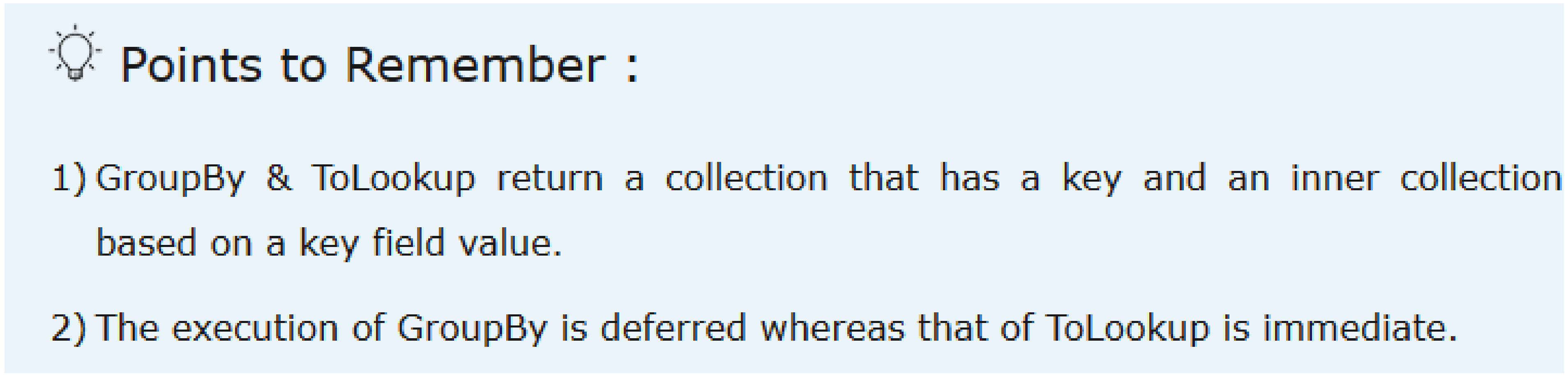
```
public void group3()
{
    Console.WriteLine("group 3--------");
    var groupedResult = studentList.GroupBy(s => new { s.Age, s.StudentName });
    Console.WriteLine(groupedResult.GetType());
    foreach (var ageGroup in groupedResult)
    {
        Console.WriteLine("Age Group: {0}", ageGroup.Key);  //Each group has a key

        foreach (Student s in ageGroup)  //Each group has a inner collection
            Console.WriteLine("Student Name: {0}", s.StudentName);
    }
    Console.WriteLine("---------------");
}
```



```
group 3--------
System.Linq.GroupedEnumerable`2[Voorbeelden.Student,<>f__AnonymousType0`2[System.Int32,System.String]]
Age Group: { Age = 18, StudentName = Bill }
Student Name: Bill
Student Name: Bill
Age Group: { Age = 21, StudentName = Steve }
Student Name: Steve
Age Group: { Age = 20, StudentName = Ram }
Student Name: Ram
Age Group: { Age = 21, StudentName = Abram }
Student Name: Abram
---------------
```

GroupBy / ToLookup

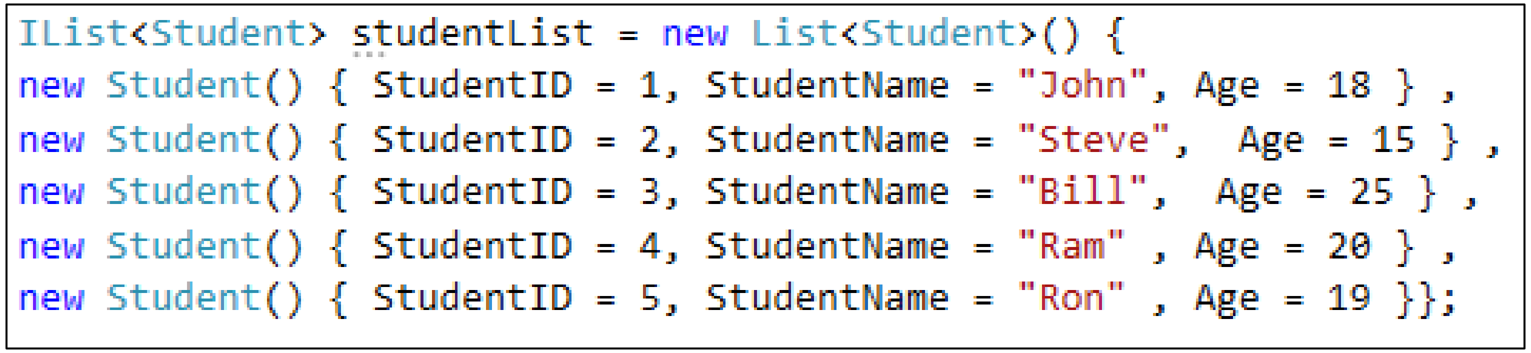


Points to Remember :

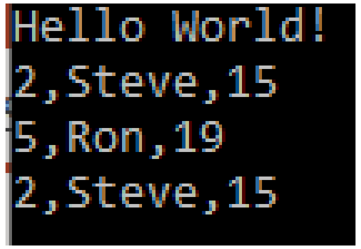
1) GroupBy & ToLookup return a collection that has a key and an inner collection based on a key field value.

2) The execution of GroupBy is deferred whereas that of ToLookup is immediate.

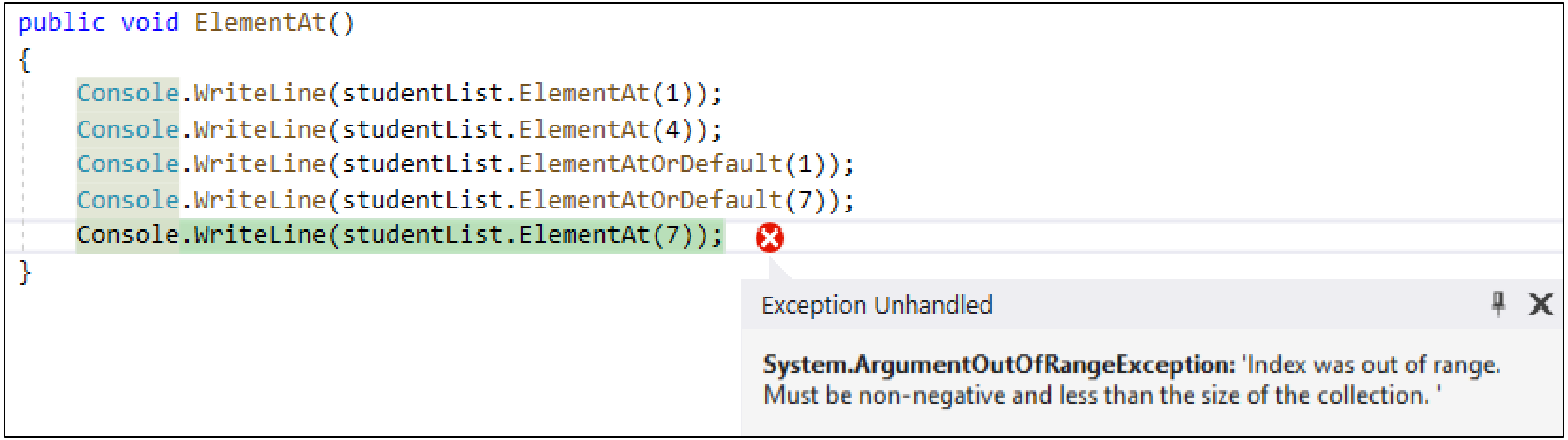
First/Last/Take/Skip/ElementAt



```
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 }};
```



```
Hello World!
2,Steve,15
5,Ron,19
2,Steve,15
```

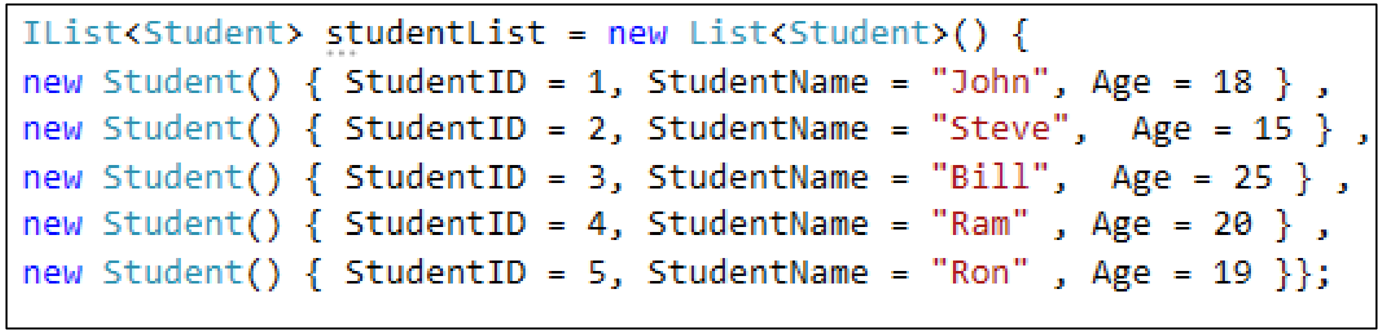


```
public void ElementAt()
{
    Console.WriteLine(studentList.ElementAt(1));
    Console.WriteLine(studentList.ElementAt(4));
    Console.WriteLine(studentList.ElementAtOrDefault(1));
    Console.WriteLine(studentList.ElementAtOrDefault(7));
    Console.WriteLine(studentList.ElementAt(7));
}
```

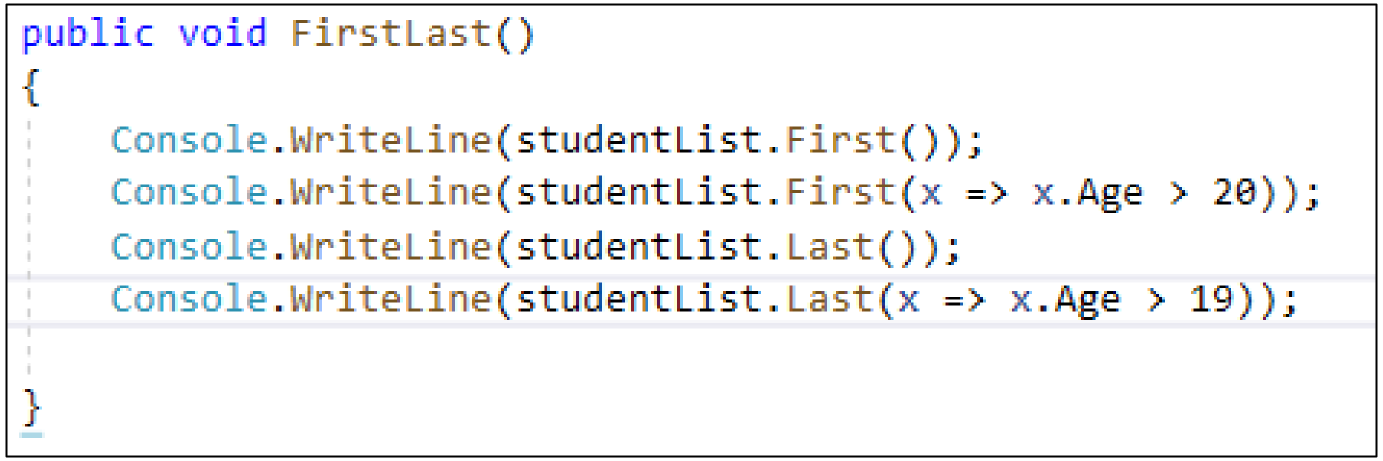
Exception Unhandled

System.ArgumentOutOfRangeException: 'Index was out of range. Must be non-negative and less than the size of the collection. '

First/Last/Take/Skip/ElementAt

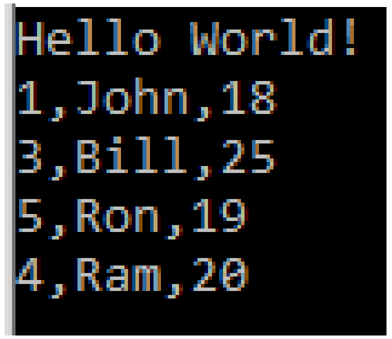


```csharp
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 }};
```



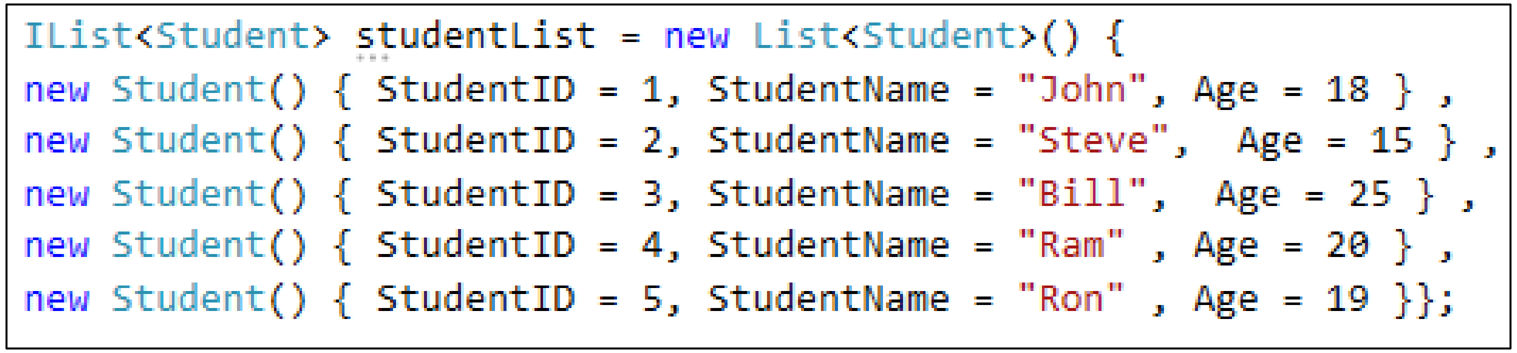
```csharp
public void FirstLast()
{
    Console.WriteLine(studentList.First());
    Console.WriteLine(studentList.First(x => x.Age > 20));
    Console.WriteLine(studentList.Last());
    Console.WriteLine(studentList.Last(x => x.Age > 19));

}
```

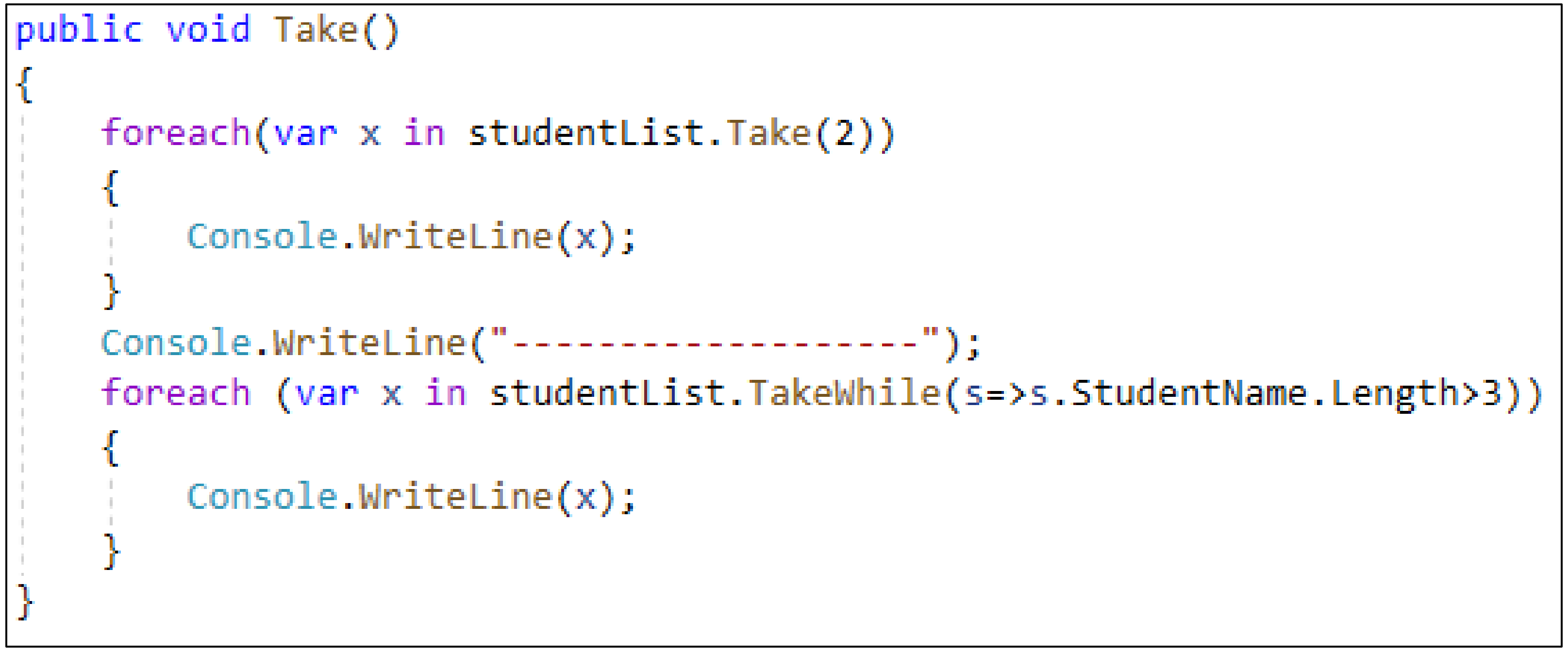


```
Hello World!
1,John,18
3,Bill,25
5,Ron,19
4,Ram,20
```

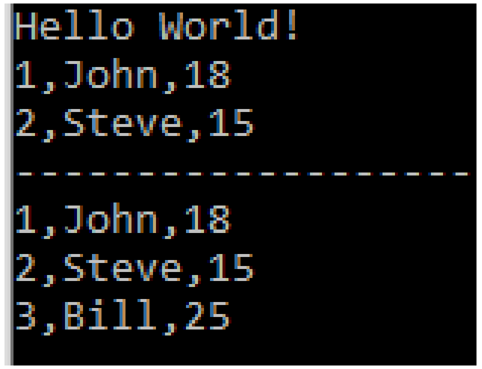
First/Last/Take/Skip/ElementAt



```
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 }};
```

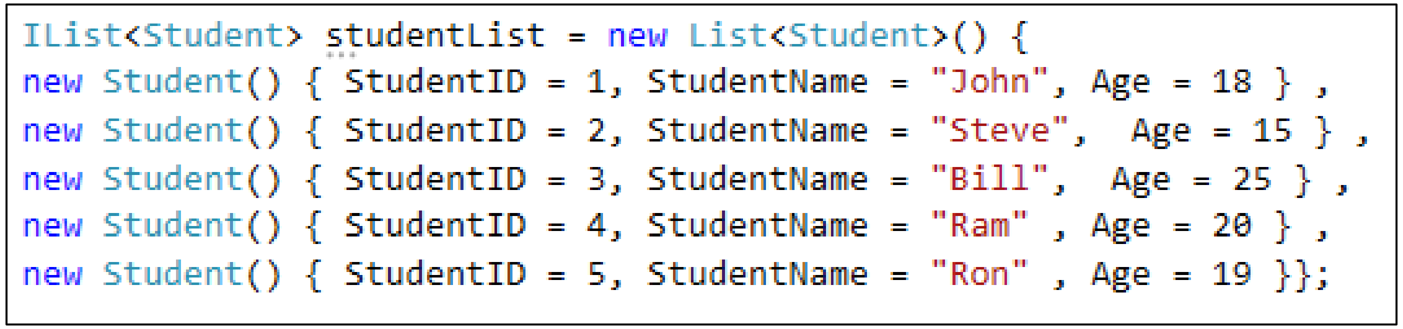


```
public void Take()
{
    foreach(var x in studentList.Take(2))
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("-------------------");
    foreach (var x in studentList.TakeWhile(s=>s.StudentName.Length>3))
    {
        Console.WriteLine(x);
    }
}
```

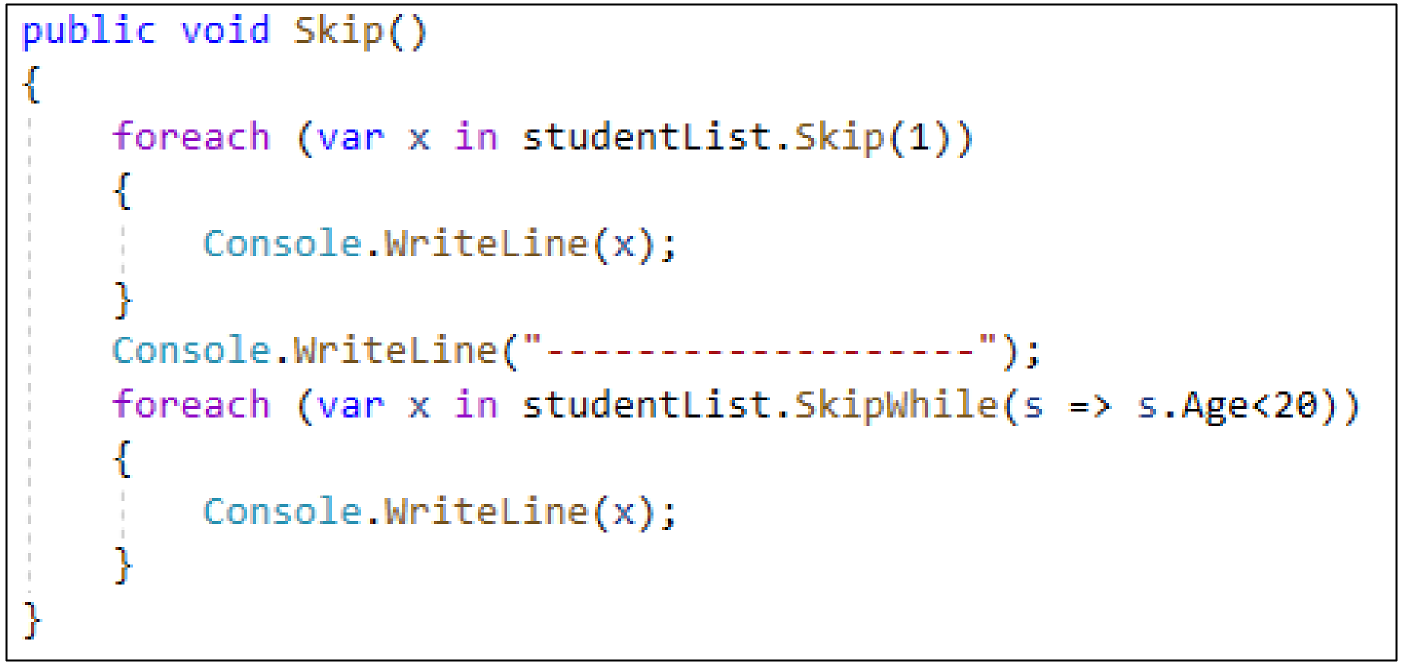


```
Hello World!
1,John,18
2,Steve,15
-------------------
1,John,18
2,Steve,15
3,Bill,25
```

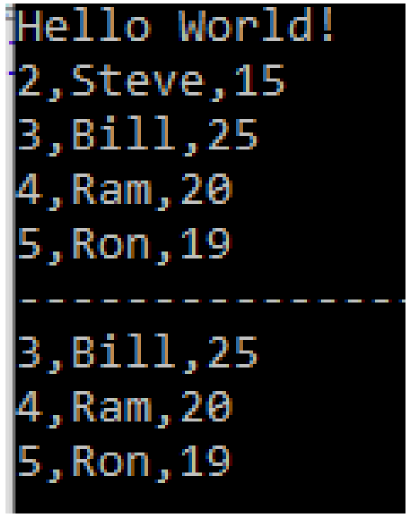
First/Last/Take/Skip/ElementAt



```
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 }};
```

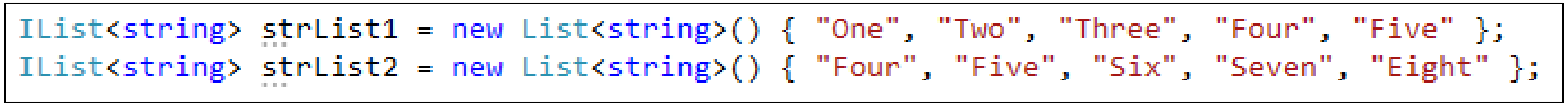


```
public void Skip()
{
    foreach (var x in studentList.Skip(1))
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("-------------------");
    foreach (var x in studentList.SkipWhile(s => s.Age<20))
    {
        Console.WriteLine(x);
    }
}
```

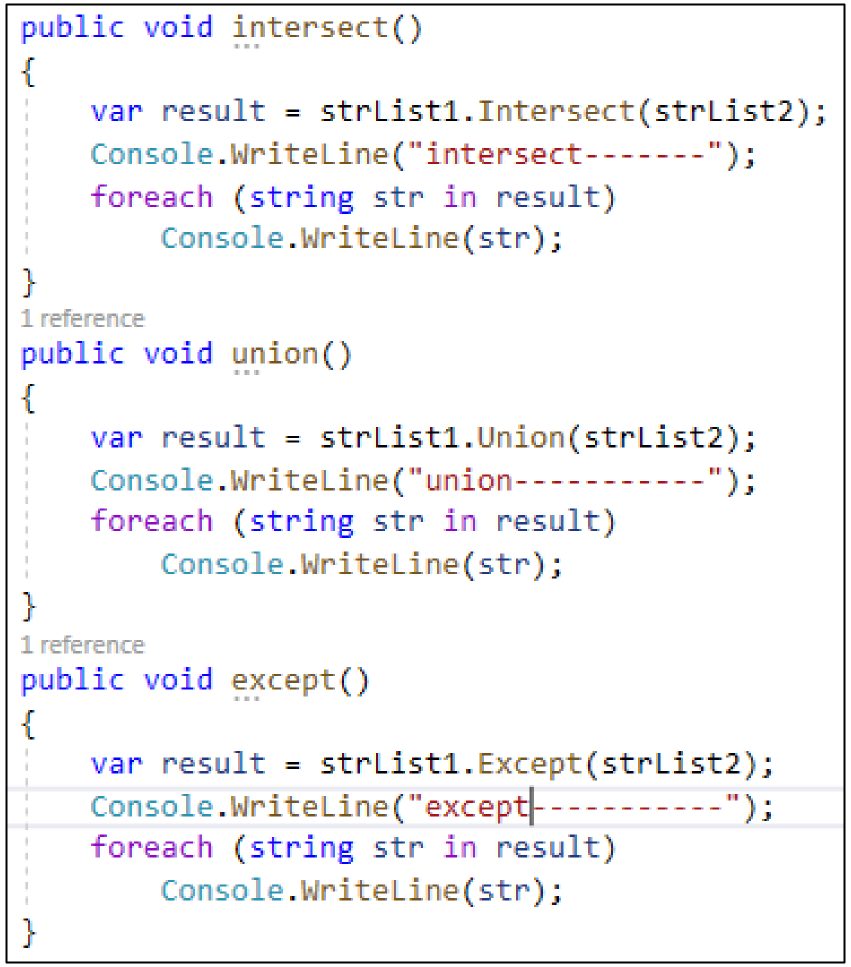


```
Hello World!
2,Steve,15
3,Bill,25
4,Ram,20
5,Ron,19
-----------------
3,Bill,25
4,Ram,20
5,Ron,19
```

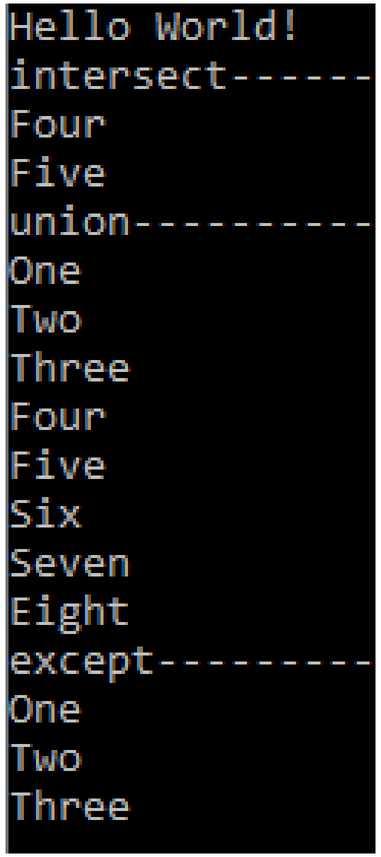
Set operators



```
IList<string> strList1 = new List<string>() { "One", "Two", "Three", "Four", "Five" };
IList<string> strList2 = new List<string>() { "Four", "Five", "Six", "Seven", "Eight" };
```

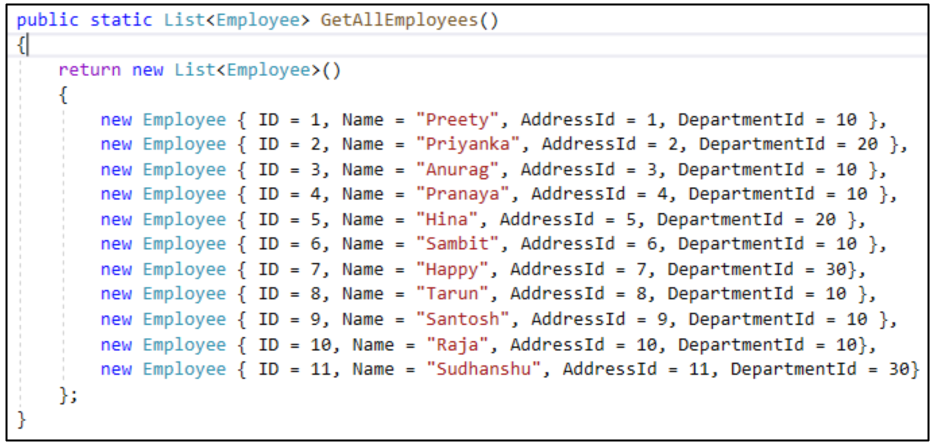


```
public void intersect()
{
    var result = strList1.Intersect(strList2);
    Console.WriteLine("intersect-------");
    foreach (string str in result)
        Console.WriteLine(str);
}
1 reference
public void union()
{
    var result = strList1.Union(strList2);
    Console.WriteLine("union----------");
    foreach (string str in result)
        Console.WriteLine(str);
}
1 reference
public void except()
{
    var result = strList1.Except(strList2);
    Console.WriteLine("except-----------");
    foreach (string str in result)
        Console.WriteLine(str);
}
```

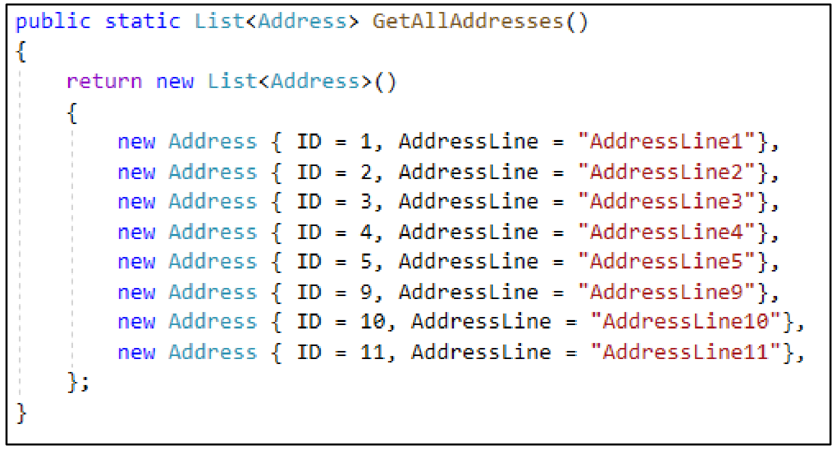


```
Hello World!
intersect------
Four
Five
union----------
One
Two
Three
Four
Five
Six
Seven
Eight
except---------
One
Two
Three
```

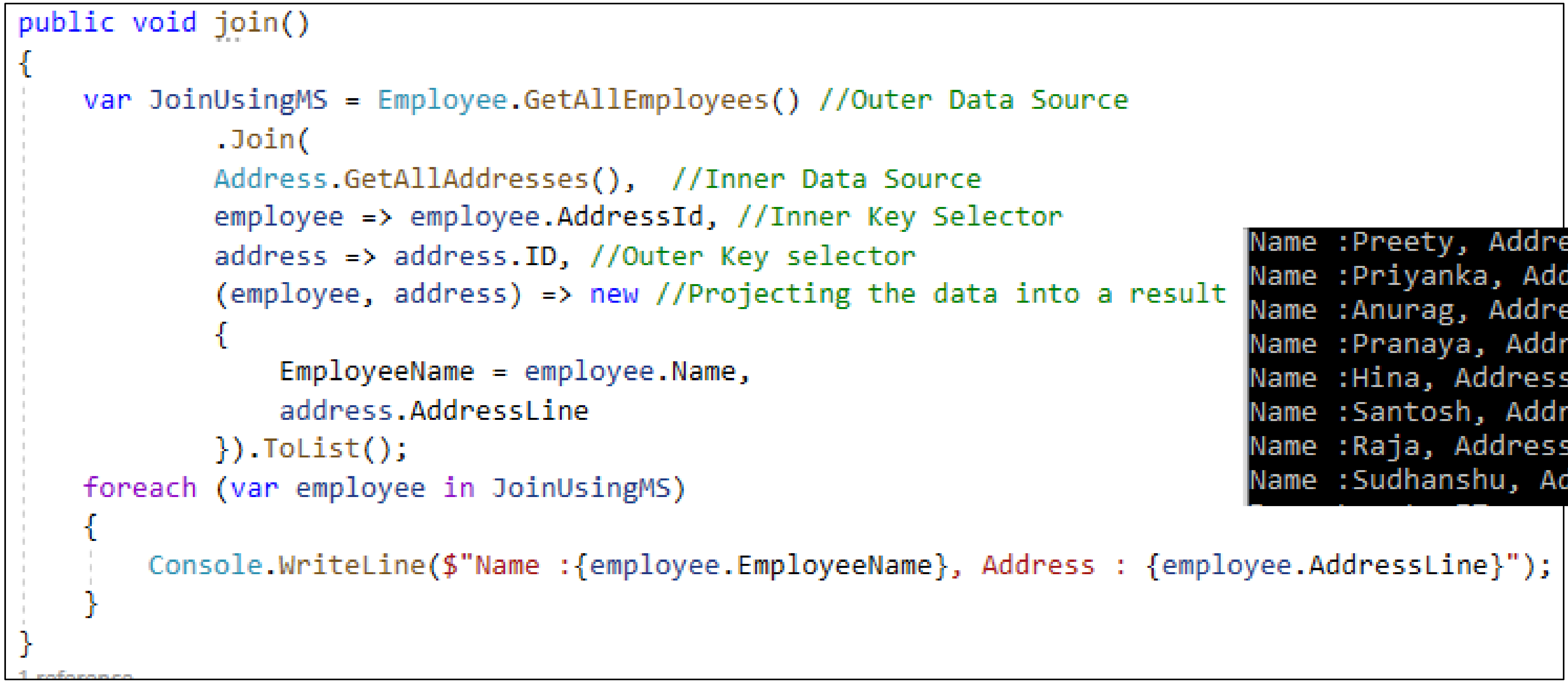
Join/GroupJoin



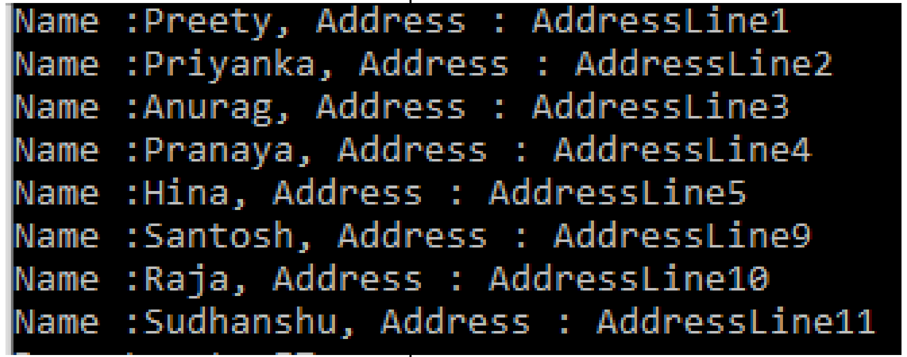
```
public static List<Employee> GetAllEmployees()
{
    return new List<Employee>()
    {
        new Employee { ID = 1, Name = "Preety", AddressId = 1, DepartmentId = 10 },
        new Employee { ID = 2, Name = "Priyanka", AddressId = 2, DepartmentId = 20 },
        new Employee { ID = 3, Name = "Anurag", AddressId = 3, DepartmentId = 10 },
        new Employee { ID = 4, Name = "Pranaya", AddressId = 4, DepartmentId = 10 },
        new Employee { ID = 5, Name = "Hina", AddressId = 5, DepartmentId = 20 },
        new Employee { ID = 6, Name = "Sambit", AddressId = 6, DepartmentId = 10 },
        new Employee { ID = 7, Name = "Happy", AddressId = 7, DepartmentId = 30},
        new Employee { ID = 8, Name = "Tarun", AddressId = 8, DepartmentId = 10 },
        new Employee { ID = 9, Name = "Santosh", AddressId = 9, DepartmentId = 10 },
        new Employee { ID = 10, Name = "Raja", AddressId = 10, DepartmentId = 10},
        new Employee { ID = 11, Name = "Sudhanshu", AddressId = 11, DepartmentId = 30}
    };
}
```



```
public static List<Address> GetAllAddresses()
{
    return new List<Address>()
    {
        new Address { ID = 1, AddressLine = "AddressLine1"},
        new Address { ID = 2, AddressLine = "AddressLine2"},
        new Address { ID = 3, AddressLine = "AddressLine3"},
        new Address { ID = 4, AddressLine = "AddressLine4"},
        new Address { ID = 5, AddressLine = "AddressLine5"},
        new Address { ID = 9, AddressLine = "AddressLine9"},
        new Address { ID = 10, AddressLine = "AddressLine10"},
        new Address { ID = 11, AddressLine = "AddressLine11"},
    };
}
```

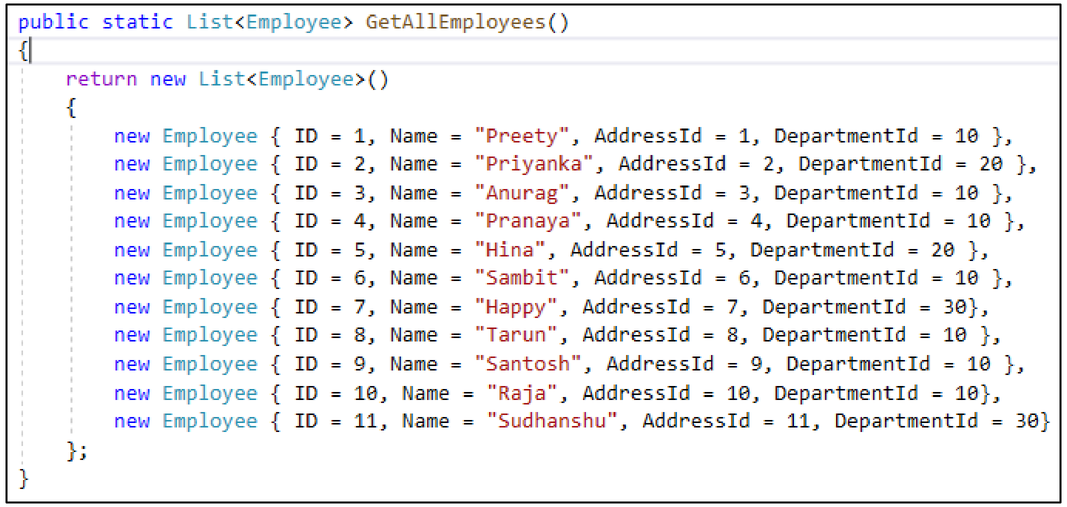


```
public void join()
{
    var JoinUsingMS = Employee.GetAllEmployees() //Outer Data Source
            .Join(
            Address.GetAllAddresses(),  //Inner Data Source
            employee => employee.AddressId, //Inner Key Selector
            address => address.ID, //Outer Key selector
            (employee, address) => new //Projecting the data into a result
            {
                EmployeeName = employee.Name,
                address.AddressLine
            }).ToList();
    foreach (var employee in JoinUsingMS)
    {
        Console.WriteLine($"Name :{employee.EmployeeName}, Address : {employee.AddressLine}");
    }
}
```

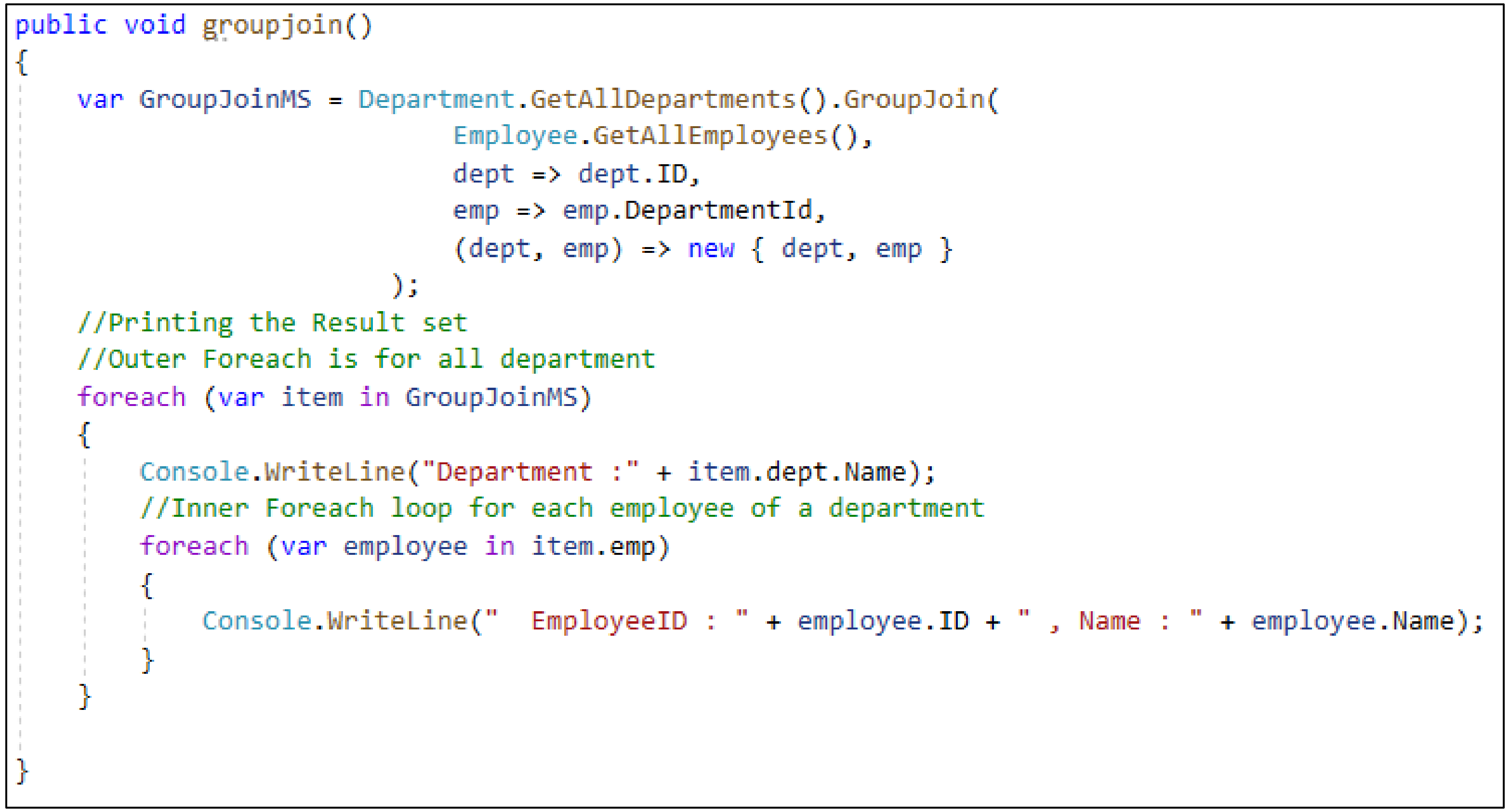


```
Name :Preety, Address : AddressLine1
Name :Priyanka, Address : AddressLine2
Name :Anurag, Address : AddressLine3
Name :Pranaya, Address : AddressLine4
Name :Hina, Address : AddressLine5
Name :Santosh, Address : AddressLine9
Name :Raja, Address : AddressLine10
Name :Sudhanshu, Address : AddressLine11
```

Join/GroupJoin

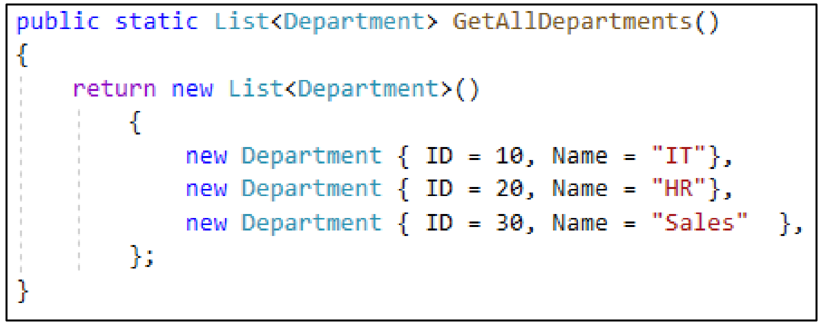


```
public static List<Employee> GetAllEmployees()
{
    return new List<Employee>()
    {
        new Employee { ID = 1, Name = "Preety", AddressId = 1, DepartmentId = 10 },
        new Employee { ID = 2, Name = "Priyanka", AddressId = 2, DepartmentId = 20 },
        new Employee { ID = 3, Name = "Anurag", AddressId = 3, DepartmentId = 10 },
        new Employee { ID = 4, Name = "Pranaya", AddressId = 4, DepartmentId = 10 },
        new Employee { ID = 5, Name = "Hina", AddressId = 5, DepartmentId = 20 },
        new Employee { ID = 6, Name = "Sambit", AddressId = 6, DepartmentId = 10 },
        new Employee { ID = 7, Name = "Happy", AddressId = 7, DepartmentId = 30},
        new Employee { ID = 8, Name = "Tarun", AddressId = 8, DepartmentId = 10 },
        new Employee { ID = 9, Name = "Santosh", AddressId = 9, DepartmentId = 10 },
        new Employee { ID = 10, Name = "Raja", AddressId = 10, DepartmentId = 10},
        new Employee { ID = 11, Name = "Sudhanshu", AddressId = 11, DepartmentId = 30}
    };
}
```

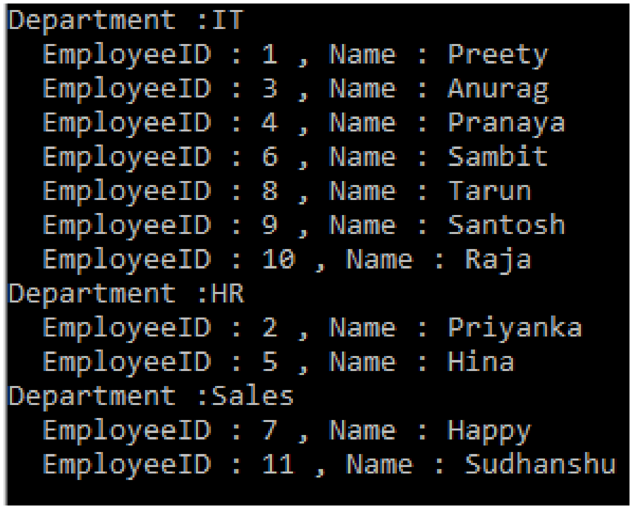


```
public void groupjoin()
{
    var GroupJoinMS = Department.GetAllDepartments().GroupJoin(
                            Employee.GetAllEmployees(),
                            dept => dept.ID,
                            emp => emp.DepartmentId,
                            (dept, emp) => new { dept, emp }
                        );
    //Printing the Result set
    //Outer Foreach is for all department
    foreach (var item in GroupJoinMS)
    {
        Console.WriteLine("Department :" + item.dept.Name);
        //Inner Foreach loop for each employee of a department
        foreach (var employee in item.emp)
        {
            Console.WriteLine("  EmployeeID : " + employee.ID + " , Name : " + employee.Name);
        }
    }

}
```

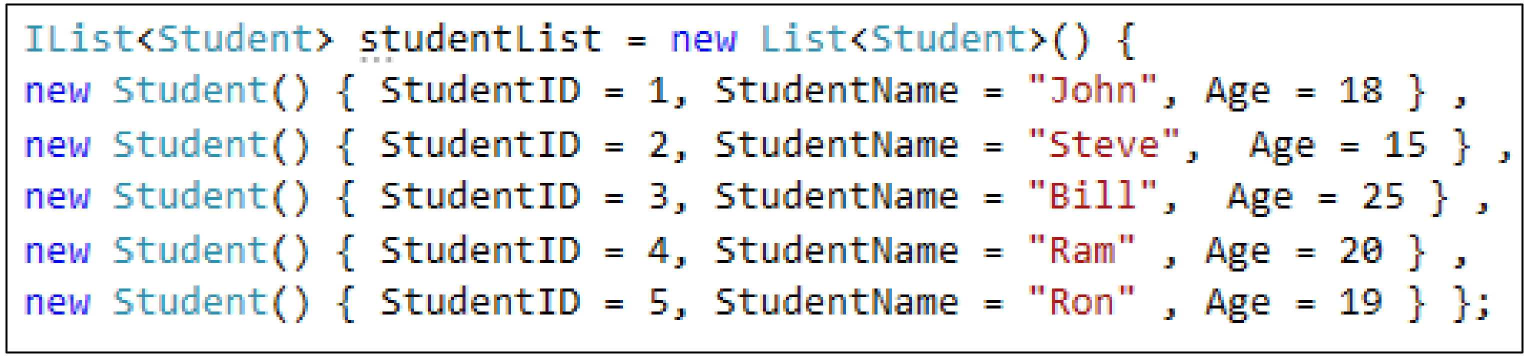


```
public static List<Department> GetAllDepartments()
{
    return new List<Department>()
        {
            new Department { ID = 10, Name = "IT"},
            new Department { ID = 20, Name = "HR"},
            new Department { ID = 30, Name = "Sales"  },
        };
}
```

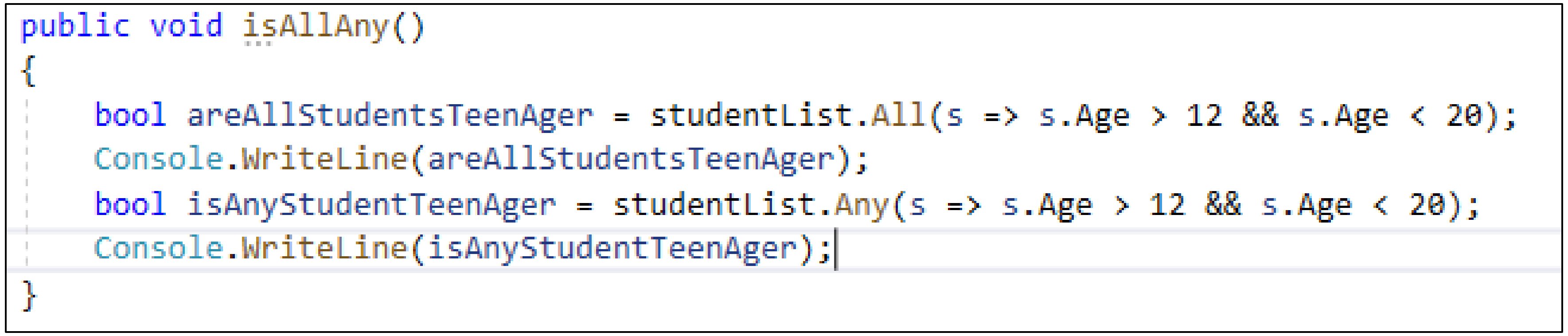


```
Department :IT
  EmployeeID : 1 , Name : Preety
  EmployeeID : 3 , Name : Anurag
  EmployeeID : 4 , Name : Pranaya
  EmployeeID : 6 , Name : Sambit
  EmployeeID : 8 , Name : Tarun
  EmployeeID : 9 , Name : Santosh
  EmployeeID : 10 , Name : Raja
Department :HR
  EmployeeID : 2 , Name : Priyanka
  EmployeeID : 5 , Name : Hina
Department :Sales
  EmployeeID : 7 , Name : Happy
  EmployeeID : 11 , Name : Sudhanshu
```

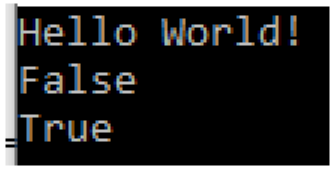
All/Any



```
IList<Student> studentList = new List<Student>() {
new Student() { StudentID = 1, StudentName = "John", Age = 18 } ,
new Student() { StudentID = 2, StudentName = "Steve",  Age = 15 } ,
new Student() { StudentID = 3, StudentName = "Bill",  Age = 25 } ,
new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
new Student() { StudentID = 5, StudentName = "Ron" , Age = 19 } };
```



```
public void isAllAny()
{
    bool areAllStudentsTeenAger = studentList.All(s => s.Age > 12 && s.Age < 20);
    Console.WriteLine(areAllStudentsTeenAger);
    bool isAnyStudentTeenAger = studentList.Any(s => s.Age > 12 && s.Age < 20);
    Console.WriteLine(isAnyStudentTeenAger);
}
```



```
Hello World!
False
True
```