# Zasada działania LINQ

## 1) LINQ w wersji nie-ADO .NET (Nie generującej SQL)

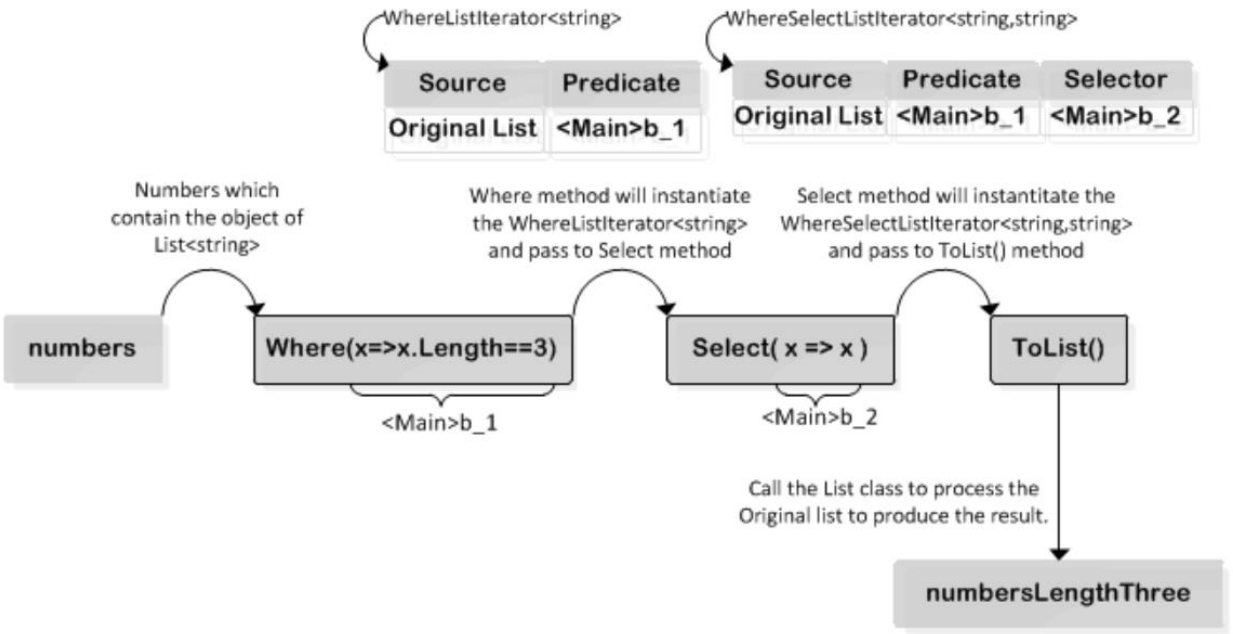
Typem z wracanym przez wszystkie wykorzystywane metody (Where, Select, …) jest IEnumerable – jest to interfejs, który ma metodę, która zwraca enumerator, który może iterować po kolekcji.

Prześledźmy działanie LINQ na przykładzie:

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

var numbersLengthThree =
    numbers.Where(x => x.Length == 3).Select(x => x).ToList();
```

Sposób działania dobrze opisuje poniższa grafika:



Kolejne kroki:

a. Kompilator tworzy metodę (<Main>b_1) używając do tego anonimowej metody, którą wpisaliśmy w Where, która jest predicate. Stworzoną metodę przechowa w MulticastDelegete i będzie kontynuował operację.
b. Zostanie wywołana metoda Where, która na wejściu otrzyma kolekcję numbers oraz instancję MulticastDelegete. Where zwraca iterator zawierający oryginalną listę oraz <Main>b_1 jako predicate. Stworzony iterator zostanie przekazany jako parametr do metody Select
c. Kompilator tworzy metodę (<Main>b_2) używając do tego anonimowej metody, którą wpisaliśmy w Select, która jest Selectorem. Stworzoną metodę przechowa w MulticastDelegete i będzie kontynuował operację.
d. W metodzie Select jako parametr otrzymamy iterator z punktu b. oraz Selector utworzony w c. i tutaj analogicznie jak w b. zwracaną wartością będzie kolejny iterator, który może zostać przekazany dalej.
e. Stworzony iterator trafia do metody ToList jako wejściowy parametr
f. Metoda ToList tworzy obiekt List, gdzie w konstruktorze zostanie stworzona nowa lista, gdzie za pomocą iteratora z punktu d. będą kopiowane kolejne elementy z oryginalnej listy w taki sposób, że oryginalna lista nie zostanie naruszona, a więc dopiero w tym miejscu zostaną wykonane wszystkie operacje jakie przekazaliśmy w parametrach i dopiero w tym miejscu powstanie wynikowa lista. Trzeba na to uważać szczególnie jeśli korzystamy z mechanizmu Closure, aby uniknąć niespodziewanych sytuacji związanych

ze zmianą wartości przekazywanej zmiennej, jak pokazane jest to na jednym ze slajdów.

Dzięki powyższemu mechanizmowi przetwarzanie LINQ jest leniwe, co odracza wykonanie operacji.

W podobny sposób wykonywane są operację dla pozostałych metod LINQ.

## 2) LINQ w wersji ADO .NET

Typem zwracanym przez wszystkie wykorzystywane metody w tej wersji LINQ jest IQuerable, który działa podobnie do IEnumerable, ale dodatkowo ma możliwość wygenerowania zapytań SQL.

Główna różnica polega na tym, że lambda przekazana przez użytkownika nie jest zamieniana na anonimową funkcję, ale za to tworzone jest z niej wyrażenie, które następnie jest dodawane do drzewa wyrażeń (Expression Tree) i każda kolejne metody (Where, Select, itp. …) przekazują między sobą IQuerable i dobudowują do drzewa kolejne wyrażenia.

Dzięki drzewom wyrażeń ADO .Net jest w stanie wygenerować optymalne zapytanie SQL. Gdyby zamiast wyrażeń tworzone były metody, to wygenerowanie SQL w ogóle nie byłoby możliwe, ponieważ ADO .Net „nie wiedziałby” jaką wewnętrzną strukturę ma przekazana lambda.