LINQ (Language Integrated Query) - Detailed Notes

1. Introduction to LINQ

LINQ (Language Integrated Query) is a feature in C# that provides a unified approach to querying data from different data sources (such as collections, databases, XML, etc.). It brings SQL-like query capabilities directly into C# using strong typing and IntelliSense.

LINQ provides two main syntaxes:

1. Query Syntax: Similar to SQL.
2. Method Syntax: Uses extension methods (e.g., Where, Select).

2. LINQ using Query Syntax

Example 1: Query even numbers from a list

```
List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6 };

var evenNumbers = from n in numbers
                  where n % 2 == 0
                  select n;

foreach (var num in evenNumbers)
    Console.WriteLine(num);
```

3. LINQ using Method Syntax

Example 2: Select names starting with 'A' and sort them

```
List<string> names = new List<string> { "Aarav", "Anita", "Binod", "Alisha" };

var result = names.Where(n => n.StartsWith("A"))
                  .OrderBy(n => n);

foreach (var name in result)
    Console.WriteLine(name);
```

4. LINQ with Anonymous Types

Example 3: Select specific properties into anonymous types

```
var employees = new[] {
    new { Name = "Sam", Salary = 40000 },
    new { Name = "Rita", Salary = 55000 }
};

var highEarners = from e in employees
                  where e.Salary > 45000
                  select new { e.Name };

foreach (var emp in highEarners)
    Console.WriteLine(emp.Name);
```

5. LINQ with Complex Objects

Example 4: Filter and sort custom objects

```
class Product
{
    public string Name { get; set; }
    public double Price { get; set; }
}

List<Product> products = new List<Product>
{
    new Product { Name = "Laptop", Price = 70000 },
    new Product { Name = "Mouse", Price = 500 },
    new Product { Name = "Keyboard", Price = 1200 }
};

var affordable = products.Where(p => p.Price < 5000)
                         .OrderBy(p => p.Price);

foreach (var item in affordable)
    Console.WriteLine(item.Name + ": " + item.Price);
```

6. Grouping with LINQ

Example 5: Group products by category

```
var items = new[] {
    new { Name = "Pen", Category = "Stationery" },
    new { Name = "Notebook", Category = "Stationery" },
    new { Name = "Apple", Category = "Fruit" }
};

var groups = from i in items
             group i by i.Category;

foreach (var g in groups)
{
    Console.WriteLine("Category: " + g.Key);
    foreach (var item in g)
        Console.WriteLine(" - " + item.Name);
}
```

7. LINQ Aggregate Functions

Example 6: Use Sum, Average, Count

```
List<int> numbers = new List<int> { 10, 20, 30, 40 };

Console.WriteLine("Sum: " + numbers.Sum());
Console.WriteLine("Average: " + numbers.Average());
Console.WriteLine("Count: " + numbers.Count());
```

8. LINQ with Any() and All()

Example 7: Check conditions with Any and All

```
List<int> values = new List<int> { 2, 4, 6, 8 };

bool anyOdd = values.Any(v => v % 2 != 0);
bool allEven = values.All(v => v % 2 == 0);

Console.WriteLine("Any odd? " + anyOdd);
Console.WriteLine("All even? " + allEven);
```