Day 5

Delegates

Delegates

- delegate type is a type-safe object that “points to” a method or a list of methods that can be invoked later.
- Usually delegates are used for defining callbacks within applications.
- So delegate is a function pointer
- Delegate can point to static / instance methods
- When one delegate points to multiple functions at a time we call it as**multicast delegate**

Delegates

- While creating a delegate we need to specify the signature of the method(s)- parameter types and return types – that this delegate can point to.
- Eg

```
// taking two integers and returning an integer.
public delegate int BinaryOp(int x, int y);
```

Delegates available in DotNet Base Class Library

Action<> and Func<> Delegates

- Are defined under Sytem Namespcae
- Action<> delegate is a generic delegate that can point a method that takes up to 16 arguments and returns void.
- Func<> delegate is a generic delegate that can point a method that takes up to 16 arguments and returns a value.

Anonymous Method

- An anonymous method is inline unnamed method
- It is created using the delegate keyword and doesn’t require a name and return type.
- Hence we can say, an anonymous method has the only body without a name and, optional parameters.

Anonymous method examples

- deleg dobj = delegate
  { Console.WriteLine("Hello Anonymous"); };

- del d1 = delegate(int x, int y) { return x * y; };

Lambda expressions

- Lambda expression are used to create an anonymous function.
- Lambda declaration operator => is used to separate the lambda's parameter list from its body.
- A lambda expression can be of any of the following two forms:
  - Expression lambda that has an expression as its body:
  - Statement lambda that has a statement block as its body:

Expression lambda

- Expression lambdas has an expression on the right side of the => operator .
- Will have the following syntax

  (input-parameters) => expression

  – Eg:=   x=> x*x
- You enclose input parameters of a lambda expression in parentheses.
- Specify zero input parameters with empty parentheses
- If only 1 input parameter is there parenthesis is optional

  ( )=>Console.WriteLine("Hello")

  (a1,b1)=>a1+b1

Statement lambdas

- A statement lambda resembles an expression lambda except that its statements are enclosed in braces:

```
name =>
 {
  string greeting =“Good Morning”;
   Console.WriteLine(greeting);
 };
```

Lambda expressions and delegates

- Lambda expressions are anonymous function definitions.
- In order to call it, lambda must be assigned to a delegate

Eg:

```
Func<int,int> rr =x => x * x;
int output = rr(12);
```

Handling events

- An event is a message that a function can raise/generate.
- One or more functions can be created as event subscribers/listeners
- When event is generated then all subscribers will receive a notification indicating that the event occurred.

Event

- Event should be created as a member of a class
- The object/class that raises the event is called the *event sender.*
- The Event sender may not know which all classes/objects will be lisening to it.
- Event has an Invoke( ) method that is used for raising the event

Events & delegates

- Events will be using delegate.
- A delegate name must come in the declaration of the event
- That delegate is used for associating this event with the listener

Exception Handling

- C# language use following keywords -try, catch, finally - for handling exceptions and throw for throwing or raising exception
- A try block contains code that might be affected by an exception.
- Catch blocks are used to handle any resulting exceptions.
- One or more catch blocks can be there
- A finally block contains code that is run whether or not an exception is thrown in the try block.
- Finally block is optional
- Catch block is also optional , but if catch not present finally must be there

Catch Blocks

- A catch block can specify the type of exception to catch.
- The type specification is called an exception filter.
- Exception is an object of a class which is derived from System.Exception class.
- Multiple catch blocks with different exception classes can be chained together.
- The catch blocks are evaluated from top to bottom and the first catch block that specifies the exact type or a base class of the thrown exception is executed.

System.Exception Base Class

- All exceptions are classes which are derived from System.Exception class.

checked and unchecked statements

- The checked and unchecked statements specify the overflow-checking context for integer-type arithmetic operations and conversions.
- When integer arithmetic overflow occurs, the overflow-checking context defines what happens.

checked and unchecked statements

- In a checked context,
  - If overflow occurs in an expression with constants compile time error will occur
  - In an expression with variables a System.OverflowException will be thrown.
- In an unchecked context, no exception or error occurs. Instead smallest possible value for the destination will be set to the destination variable.
- Default behavior will be complier dependent and will be unchecked.

Anonymous types

- Anonymous types provide a convenient way to encapsulate a set of read-only properties into a single object without having to explicitly define a type.
- Eg:-
  var v = new { Amount = 108, Message = "Hello" };
- Here v is read-only. So v.Amount=110 will fail.

Extension Methods

- Is a feature introduced in .NET 3.5
- This feature allows to add new methods or properties to a class or structure, without modifying the original type

Defining Extension Methods

- Extension Methods must be defined within a static class
- Extension Method must be declared with the static keyword.
- All extension methods are marked by using the this keyword as a modifier on the first (and only the first) parameter of the method in question.
- The “this qualified” parameter represents the item being extended.

Importing & Invoking Extension Methods

- To use an extension method the namespace containing the extension method need to be imported with thre C# using keyword.
- The scope of extension methods are limited to the namespaces that define them or the namespaces that import them.

Extension Methods

- add new methods or properties to a class or structure, without modifying the original type in any direct manner.
- It is a very used feature to add functionality to an existing class(may be used many places in your code) without changing the class
- Extension methods eliminates the need to create child classes, inorder to add functionality

Extension Methods

- Extension methods must be static methods defined inside a static class.
- First parameter to extension method must be <u>this</u> keyword followed by <u>the name of the class you want to extend.</u>
- You can give additional parameters as per logic.
- Extension methods , though are static methods are invoked with the object of the base class

Partial Classes

- If the definition of a class is split into multiple source files(.cs) then each part is referred as a partial class
- All the parts available at compile time will form the final class.

Partial Classes

- All the parts must have the same accessibility, such as public, private, etc.
- If any part is declared abstract, then the whole type is considered abstract.
- If any part is declared sealed, then the whole type is considered sealed.
- If any part declares a base type, then the whole type inherits that class.
- Parts can specify different base interfaces, and the final type implements all the interfaces listed by all the partial declarations.

Partial Methods

- A partial class or struct may contain a partial method.
- One part of the class contains the signature of the method.
- Implementation can be defined in the same part or another part.

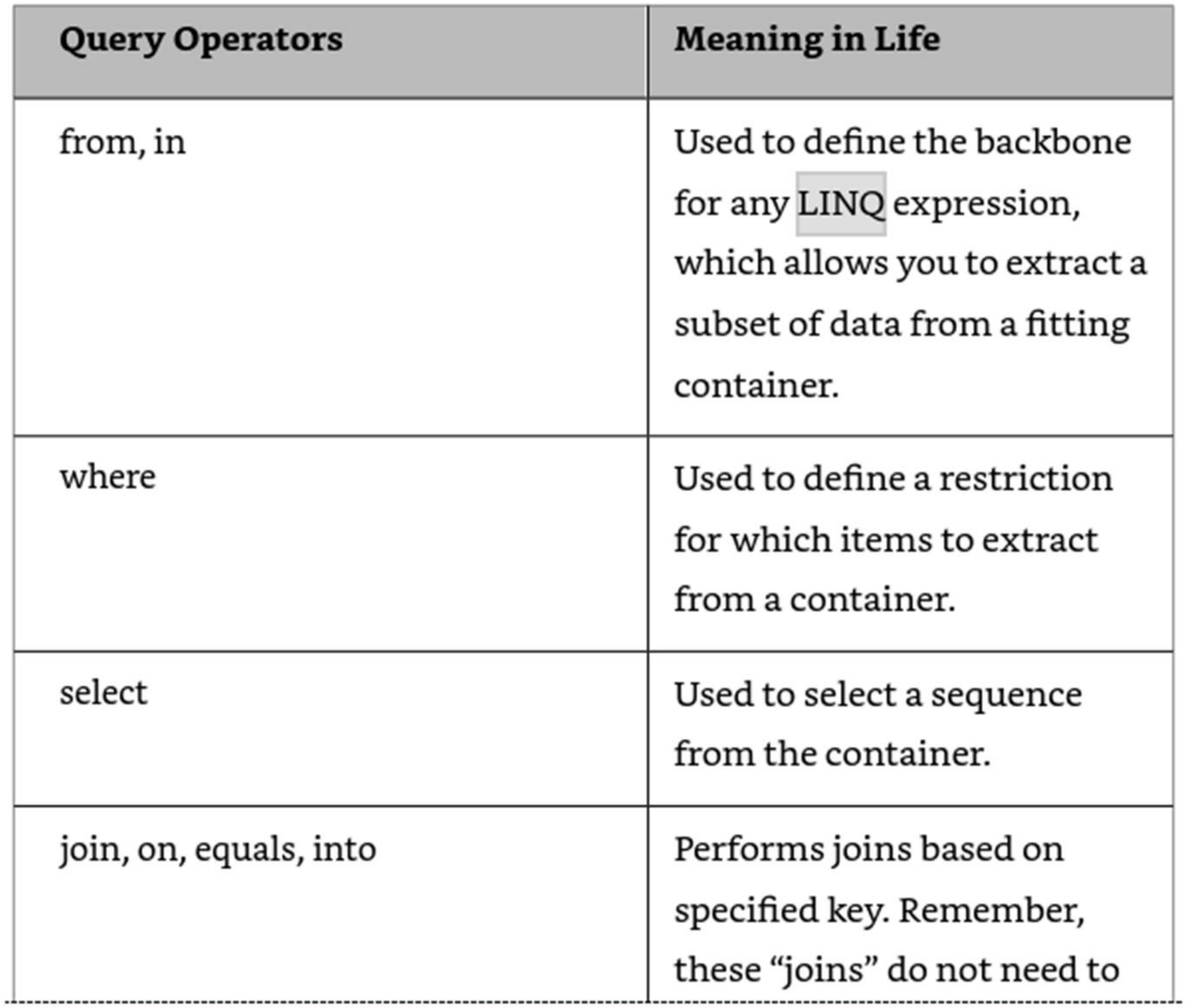
LINQ – Language Integrated Query

Language Integrated Query (LINQ)

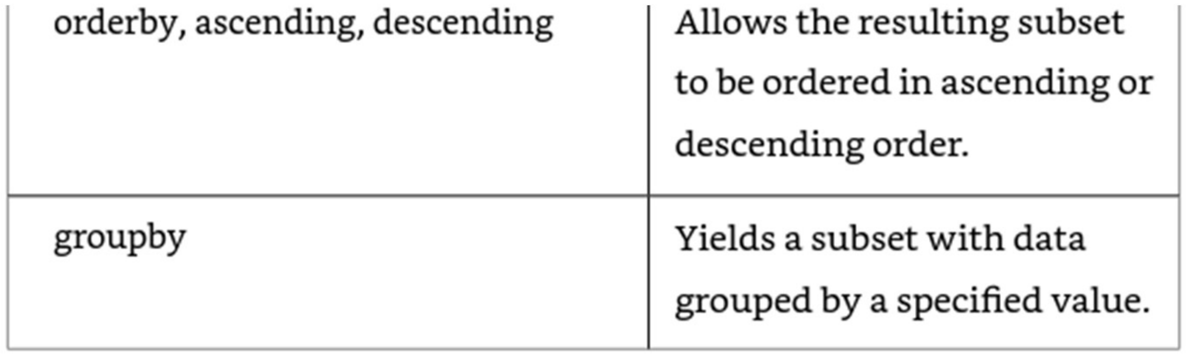
- LINQ is s set of technologies for getting query capabilities directly into the C# language.

Deferred Execution

- LINQ query they are not actually evaluated until you iterate over the resulting sequence.
- This feature of LINQ is termed deferred execution.
- But when a LINQ statement is selecting a single element (eg -First/FirstOrDefault, Single/SingleOrDefault, or any of the aggregation methods), the query is executed immediately.



Query Operators	Meaning in Life
from, in	Used to define the backbone for any LINQ expression, which allows you to extract a subset of data from a fitting container.
where	Used to define a restriction for which items to extract from a container.
select	Used to select a sequence from the container.
join, on, equals, into	Performs joins based on specified key. Remember, these “joins” do not need to



orderby, ascending, descending	Allows the resulting subset to be ordered in ascending or descending order.
groupby	Yields a subset with data grouped by a specified value.

PLINQ

- LINQ queries that are designed to run in parallel are termed PLINQ queries.