Variables

```
class Variables
{
    static void Main(string[] args)
    {
        //int x;
        //int y;
        //x = 7;
        //y = x + 3;
        //Console.WriteLine(y);
        //Console.ReadLine();
        Console.WriteLine("What is your name?");
        Console.WriteLine("Type your first name:");
        string myFirstName;
        myFirstName = Console.ReadLine();
        Console.Write("Type your last name:");
        string myLastName = Console.ReadLine();
        Console.WriteLine($"Hello, {myFirstName} {myLastName} " +
            $"{"nice to see you!"}");
        Console.WriteLine("Hello {0} {1} {2} ", myFirstName, myLastName,
            "Nice to see you!");
        Console.ReadLine();
    }
}
```

Decisiones

```
class Decisions
{
    static void Main(string[] args)
    {
        Console.WriteLine("Billy, Big Giveaway");
        Console.Write("Choose a door: 1,2 or 3:");
        string userValue = Console.ReadLine();
        string message = "";
        if (userValue =="1")
        {
            message = " you won a new car!";
        }
        else if (userValue=="2")
        {
            message = " you won a new boat!";
        }
        else if (userValue=="3")
        {
            message = " you won a new jorney!";
        }
        else
        {
            message = " you did not win anything!";
        }
        Console.WriteLine($"You enter:{userValue},{message}");
        Console.ReadLine();
    }
}
```

Operadores y sentencias de expresión

```
class OperatorsExpresionStatements
{
    static void Main(string[] args)
    {
        //Variable declaration
        int x, y, a, b;
        //Assignment operator
        x = 3;
        y = 2;
        a = 1;
        b = 0;
        // There are many mathematical operators ...
        // Addition operator
        x = 3 + 4;
        //Substractin operator;
        x = 4 - 3;
        //Multiplication operator;
        x = 10 * 5;
        //Division operator
        x = 10 / 5;
        //order of opertions using parenthesis
        x = (x + y) * (a + b);
        // There area many operators used to evaluate values ...
        // Greater than operators
        if (x>y)
        {
            Console.WriteLine($"{x} is greater than {y}");
        }
        // Equality operator
        if (x==y)
        {
            Console.WriteLine($"{x} is equal to {y}");
        }
        // Less than operator
        if (x<y)
        {
            Console.WriteLine($"{x} is less than {y}");
        }
        // Greater or equal to operator
        if (x >= y)
        {
            Console.WriteLine($"{x} is greater or equal to {y}");
        }
        //Less than or equal to operator
        if (x<=y)
        {
            Console.WriteLine($"{x} is less than or equal to {y}");
        }

        // There are two "conditional" operators as well that an be used to ex
        // enhance an evalution. ..
        // ... and they can be combined together multiple times.
        // Conditional AND operator...
        if ((x>y) && (a>b))
        {
            Console.WriteLine($"({x}>{y}) && ({a}>{b})");
        }
        if ((x > y) || (a > b))
        {
            Console.WriteLine($"({x} > {y}) || ({a} > {b})");
        }

        // Also, here's the in-line conditional operator we learned about in
        //the previos lesson..
        string message = (x == 1) ? "Car" : "Boat";
        Console.WriteLine($"The value of x is {x}, so the answer is {message}")
        Console.ReadLine();
    }
}
```

Iterador For

```
class ForIteration
{
    static void Main(string[] args)
    {
        Console.Write("Choose a number between 0 to 10:");
        string y = Console.ReadLine();
        int number;
        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(i);
            if (Int32.TryParse(y,out number)) // Check the value entrance is
                //a number
            {
                if (number==i)
                    Console.WriteLine($"{number} == {i}, then was found it!");
            }
        }
        Console.ReadLine();

    }
}
```

Arrays

```
class UndestandingArrays
{
    static void Main(string[] args)
    {
        int number1 = 4;
        int number2 = 8;
        int number3 = 24;
        int number4 = 32;
        int number5 = 40;
        int number6 = 48;
        int number7 = 64;
        Console.WriteLine(number1);
        Console.WriteLine(number2);
        Console.WriteLine(number3);
        if (number1 == 4)
        {
            Console.WriteLine($"number is {number4}");
        }
        Console.WriteLine(number5);
        Console.WriteLine(number6);
        Console.WriteLine(number7);
        Console.ReadLine();

        int[] numbers = new int[8];
        // Option 1
        numbers[0] = 2;
        numbers[1] = 4;
        numbers[2] = 8;
        numbers[3] = 24;
        numbers[4] = 32;
        numbers[5] = 40;
        numbers[6] = 48;
        numbers[7] = 68; /**/
        // Option 2 better and cleanest
        numbers = new int[] { 2, 4, 8, 24, 32, 40, 48, 68 };

        foreach (var number in numbers)
        {
            Console.WriteLine($"The numbers is:{number}");
        }

        for (int i = 0; i < numbers.Length; i++)
        {
            Console.WriteLine($"The position in array[{i}] has the value " +
                $"{numbers[i]}");
        }
        string[] names = new string[] { "Eddie", "Alex", "Michael",
            "David Lee" };
        foreach (string name in names)
        {
            Console.WriteLine($"The name is:{name}");
        }


        string zig = "You can get what you want out of life " +
        " if you help enough other people get what they want.";

        char[] charArray = zig.ToCharArray();
        Array.Reverse(charArray);
        int colorActual = (int) Console.ForegroundColor;
        Console.WriteLine($"Cadena zig:{"\n"}{zig}");
        Console.ForegroundColor = ConsoleColor.DarkRed;

        foreach (char zigChar in charArray)
        {
            Console.Write(zigChar);
        }
        Console.ForegroundColor = (System.ConsoleColor) colorActual;

        string cadenaDeTexto = "Cadena de texto a volver de revez";
        Console.WriteLine($"{"\n"}CadenaDeTexto:{"\n"}{cadenaDeTexto}");
        char[] cadena = cadenaDeTexto.ToCharArray();
        Array.Reverse(cadena);

        Console.ForegroundColor = ConsoleColor.DarkGreen;
        foreach (var caracter in cadena)
        {
            Console.Write(caracter);
        }
        Console.ReadLine();
    }
}
```

Methodos

```
class HelperMethods
{
    static void Main(string[] args)
    {
        Console.WriteLine("The name Game");

        Console.Write("What's your first name?");
        string firstName = Console.ReadLine();

        Console.Write("What's your last name?");
        string lastName = Console.ReadLine();

        Console.Write("In what city where you born?");
        string city = Console.ReadLine();

        DisplayResult(firstName, lastName, city);
        Console.ForegroundColor = ConsoleColor.DarkMagenta;
        DisplayResult(ReverseString(firstName), ReverseString(lastName),
            ReverseString(city));
        Console.ReadLine();
    }
    private static void DisplayResult(string firstName, string lastName,
        string city)
    {
        Console.WriteLine($"Results:{firstName} {lastName} {city}");
    }
    private static string ReverseString(string message)
    {
        char[] messageArray = message.ToCharArray();
        Array.Reverse(messageArray);
        return String.Concat(messageArray);
    }
}
```

Iterador While

```
using System;

namespace _01_Complete
{
    #region WhileIteration
    class WhileIteration
    {
        #region Main
        static void Main(string[] args)
        {
            bool displayMenu = true;
            while (displayMenu)
            {
                displayMenu = MainMenu();
            }
        }
        #endregion
        MainMenu
        Statics Methods
    }
    #endregion
}
```

```
        private static bool MainMenu()
        {
            Console.Clear();
            Console.WriteLine("Choose an option");
            Console.WriteLine("1) Print Numbers");
            Console.WriteLine("2) Guessing Game");
            Console.WriteLine("3) Exit");
            string result = Console.ReadLine();
            if (result=="1")
            {
                PrintNumbers();
                return true;
            }
            else if (result=="2")
            {
                GuessingGame();
                return true;
            }
            else if (result=="3")
            {
                return false;
            }
            else
            {
                return true;
            }

        }
        private static void GuessingGame()
        {
            Console.Clear();
            Console.WriteLine("Guessing game!!");

            Random myRandom = new Random();
            int randomNumber = myRandom.Next(1, 11);
            int guesses = 0;
            bool incorrect = true;
            do
            {
                Console.WriteLine("Guest a nubmer between 1 and 10:");
                string result = Console.ReadLine();
                guesses++;
                if (result == randomNumber.ToString())
                {
                    incorrect = false;
                }
                else
                    Console.WriteLine("Wrong!");
            } while (incorrect);
            Console.WriteLine($"Correct! It took you {guesses} guesess.");
            Console.ReadLine();
        }
        private static void PrintNumbers()
        {
            Console.Clear();
            Console.WriteLine("Print numbers!");
            Console.Write("Type a number:");
            int result = int.Parse(Console.ReadLine());
            int counter = 1;
            string line = "";
            while (counter < result + 1 )
            {
                Console.Write("{0:000}",counter);
                counter++;
                line = (counter == result+1) ? "\n" : "-";
                Console.Write($"{line}");
            }
            Console.Write("Press any key to continue...");
            Console.ReadLine();
        }
        #endregion
    }
```

Trabajando con Cadenas de Caracteres

```
class WorkingWithStrings
{
    static void Main(string[] args)
    {
        string myString = "my \"so called\" life";
        myString = "What if I need a new \nline caracter";
        myString = "Go to you c:\\ drive";
        myString = @"Go to you c:\drive";
        myString = String.Format("{1}={1}", "first", "second");
        myString = string.Format("{0:C}", 23403.03);
        myString = string.Format("{0:N}", 123457890);
        myString = string.Format("{0:P}", .234);
        myString = string.Format("Phone Number:{0:(###) ###-####}",
            1234567890123);
        /*string myString = "*/ //this is a test a string with extra spaces " ;
        myString = myString.Substring(6, 14);
        myString = myString.ToUpper();
        myString = myString.Replace(" ", "--");
        myString = myString.Remove(6, 14);
        myString = String.Format("Original length:{0} - lenght without spaces:{1}", myString.Length, myString.Trim().Length);
        myString = "";
        for (int i = 0; i < 100; i++)
        {
            //myString += String.Format("-- {0}", i.ToString());
            myString += "--" + i.ToString();
        }

        StringBuilder myString_SB = new StringBuilder();
        int length = 100;
        for (int i = 0; i < length; i++)
        {
            myString_SB.Append("--");
            myString_SB.Append(i.ToString());
        }

        Console.WriteLine($"{myString_SB}" );
        Console.ReadLine();
    }
}
```

Fechas y Horas

```
namespace _01_Complete
{
    class DatesAndTimes
    {
        static void Main(string[] args)
        {
            DateTime myValue = DateTime.Now;
            Console.WriteLine(myValue.ToString());
            int currentForegroundColor = (int)Console.ForegroundColor;
            Console.ForegroundColor = ConsoleColor.DarkBlue;
            Console.WriteLine($"ToShortDateString:{myValue.ToShortDateString()} " +
                $"\nToShortTimeString:{myValue.ToShortTimeString()}");
            Console.ForegroundColor = (ConsoleColor)currentForegroundColor;
            Console.WriteLine($"ToLongDateString:{myValue.ToLongDateString()}" +
                $"\nToLongTimeString:{myValue.ToLongTimeString()}");
            Console.WriteLine($"AddDays:{myValue.AddDays(3).ToLongTimeString()}" +
                $"\nAddHours:{myValue.AddHours(3)}-");
            Console.ForegroundColor = ConsoleColor.DarkRed;
            Console.WriteLine($"Month:{myValue.Month}");
            DateTime myBirthday = new DateTime(1968, 12, 31);
            Console.WriteLine(myBirthday.ToShortDateString());
            //Format of DateTime.Parse("dia/mes/año");
            myBirthday = DateTime.Parse("31/12/1968");
            //Represent a interval of time
            TimeSpan myAge = DateTime.Now.Subtract(myBirthday);
            Console.WriteLine(myAge.TotalDays);
            string dateInput = "Dec 21, 2018"; // Esto no funciona si la cadena es "Dic 21, 
            DateTime parsedDated = DateTime.Parse(dateInput);
            Console.WriteLine("{0}", parsedDated);
            CultureInfo myCultureInfo = new CultureInfo("es-ES");
            string mySpanishDate = "21 Diciembre 2018";
            DateTime myDateTime = DateTime.Parse(mySpanishDate, myCultureInfo);
            Console.WriteLine("{0}", myDateTime.ToShortDateString());
            Console.ReadLine();
        }
    }
}
```

Clases

```
6   class SimpleClasses
7   {
8       static void Main(string[] args)
9       {
10          Car_SC myCar = new Car_SC();
11          Car_SC myCar1 = new Car_SC();
12          Car_SC myCar2 = new Car_SC();
13
14          myCar.Color = "Red";
15          myCar1.Color = "Yellow";
16          myCar2.Color = "White";
17
18          myCar.Make = "Oldsmobile";
19          myCar1.Make = "Ford";
20          myCar2.Make = "GMC";
21
22          myCar.Year = 1970;
23          myCar1.Year = 1995;
24          myCar2.Year = 2015;
25
26          myCar.Model = "Cutlas Supreme";
27          myCar1.Model = "Mustang";
28          myCar2.Model = "K Series";
29
29
30          List<Car_SC> Cars = new List<Car_SC> { myCar,myCar1,myCar2};
31          foreach (Car_SC car in Cars)
32          {
33
34              Console.Write($"Make:{car.Make}-Model:{car.Model}-Year:" +
35                  $"{car.Year.ToString()}-" +
36                  $"Color:{car.Color.ToString()}-Market Value:");
37              Console.WriteLine("{0:C0}", car.DetermineMarketValue());
38          }
39          Console.ReadLine();
40
41      }
42  }
43  // Rembermer "prop" TAB-TAB define las propiedades de la clase
44  class Car_SC
45  {
46      public string Model{ get; set; }
47      public string Make { get; set; }
48      public int Year { get; set; }
49      public string Color { get; set; }
50
51      public decimal DetermineMarketValue()
52      {
53          decimal carValue;
54          if (Year > 2000)
55              carValue = 10000;
56          else
57              carValue = 2000;
58          return carValue;
59      }
60  }
```

Tiempo de vida de un Objeto

```
7   namespace _01_Complete
8   {
9       class ObjectLifetime
10      {
11          static void Main(string[] args)
12          {
13              Car_OL myCar = new Car_OL();
14              myCar.Make = "Oldsmobile";
15              myCar.Model = "Cutlas Supreme";
16              myCar.Year = 1986;
17              myCar.Color = "Silver";
18              Car_OL.MyMethod(); //Methodo statico que no necesita ser
19              //instanciado en un objeto para ser invocado
20              myCar = new Car_OL("Oldsmobile", "Cutlas Supreme", 1986,
21                  "Silver");
22
23              Car_OL myOtherCar = myCar;
24
25              //Car myOtherCar = new Car("Ford", "Escape", 2005, "White");
26              Console.WriteLine("{0}-{1}-{2}-{3}", myOtherCar.Make,
27                  myOtherCar.Model, myOtherCar.Year, myOtherCar.Color);
28              myOtherCar.Year = 1998;
29              Console.WriteLine("{0}-{1}-{2}-{3}", myCar.Make, myCar.Model,
30                  myCar.Year, myCar.Color);
31              //myOtherCar = null;
32              ////  This will cause an exceptin of null object set
33              //Console.WriteLine("{0}-{1}-{2}-{3}", myOtherCar.Make,
34              //myOtherCar.Model, myOtherCar.Year, myOtherCar.Color);
35              Console.ReadLine();
36          }
38      class Car_OL
39      {
40          public Car_OL(string model, string make, int year, string color)
41          {
42              Model = model;
43              Make = make;
44              Year = year;
45              Color = color;
46          }
47          public Car_OL()
48          {
49              Make = "Nissan";
50          }
51          public string Model { get; set; }
52          public string Make { get; set; }
53          public int  Year { get; set; }
54          public string Color { get; set; }
55          public static void MyMethod()
56          {
57              Console.WriteLine("Called the static MyMethod");
58              //Console.WriteLine(Car.Make); thi is illegal¡¡¡
59          }
60      }
61  }
```

Alcance de variables y clases

```
1   using System;
2   namespace _01_Complete
3   {
4       class UndestandingScopes
5       {
6           public  string k;
7           static void Main(string[] args)
8           {
9               string j = "";
10              UndestandingScopes program = new UndestandingScopes();
11              for (int i = 0; i < 10; i++)
12              {
13                  j = i.ToString();
14                  program.k = i.ToString();
15                  Console.WriteLine(i);
16              }
17              //Console.WriteLine(i); This variable is invalid because is out "for
18              Console.WriteLine($"value of i:{j}");
19              Console.WriteLine($"value of k:{program.k}");
20              // Calling Car Class with its private and public methods
21              Car_US car = new Car_US();
22              Console.WriteLine("Calling car methods");
23              car.DoSomething();
24              Console.ReadLine();
25          }
26      }
27
27
28      class Car_US
29      {
30          public void DoSomething()
31          {
32              Console.WriteLine(HelperMethod());
33          }
34
35          private string HelperMethod()
36          {
37              return "Hello world!";
38          }
39      }
40
41  }
42
```

Asemblies y NameSpaces

```
5   namespace _01_Complete
6   {
7       class AssembliesAndNamespaces
8       {
9           static void Main(string[] args)
10          {
11              string text = string.Format("{0} {1}","A simple example to write ",
12                      "data into a file saved in disk");
13              System.Console.WriteLine(text);
14              File.WriteAllText(
15              @"C:\Exer\All\AssembliesAndNamespaces\WriteText.txt", text);
16              System.Console.ReadLine();
17
18              WebClient client = new WebClient();
19              string reply = client.DownloadString("http://msdn.microsoft.com");
20              Console.WriteLine(reply);
21              string path;
22              path = string.Format("{0}{1}",@"C:\Exer\All\AssembliesAndNamespaces\",
23                  "DownloadInformation.txt");
24
25              File.WriteAllText(path,reply);
26              Console.ReadLine();
27          }
28      }
29  }
30
```

```
using ScrapeLibrary;    // Show how to do this project
                        // Don't forget to put "using ScreapLibrary here!!!"
using System;
                        // Remember Add New in Solution '01-Complete
                        // Kind of project: Class Library (.Net Framework)
                        // Type: C# A project for creating a C# class library(.dll)
namespace _01_Complete
{

    // Show all students The Reference in Solucion Explorer and deadlight teh
    //aseemblyes with referentiation in the projects.
    class MyClient
    {
        static void Main(string[] args)
        {
            Scrape myScrape = new Scrape();
            string value = myScrape.ScrapeWebpage("http://msdn.microsoft.com");
            Console.WriteLine(value);
            Console.ReadLine();
        }
    }
    // Show all students the other project instead 01-Complete: ScrapeLibrary
}
```

```
using System.IO;
using System.Net;

namespace ScrapeLibrary
{
    public class Scrape
    {
        public string ScrapeWebpage(string url)
        {
            return GetWebpage(url);
        }

        public string ScrapeWebpage(string url, string filepath)
        {
            string reply = GetWebpage(url);

            File.WriteAllText(filepath, reply);
            return reply;
        }

        private string GetWebpage(string url)
        {
            WebClient client = new WebClient();
            string content = client.DownloadString(url);
            content += "THAT'S ALL FOLKS!!!";
            return content;
        }

    }
}
```

Trabajando con Colecciones (I)

```
using System;
using System.Collections;
using System.Collections.Generic;
namespace _01_Complete
{
    class WorkingWithCollections
    {
        static void Main(string[] args)
        {
            Car car1 = new Car();
            car1.Make = "Oldsmobile";
            car1.Model = "Cutlas Supreme";
            car1.VIN = "A1";
            Car car3 = new Car
            {
                Make = "GMC",
                Model = "Grand Voyager",
                VIN = "C1"
            };
            Car car2 = new Car
            {
                Make = "Geo",
                Model = "Prism",
                VIN = "B2"
            };
            Book b1 = new Book();
            b1.Author = "Robert Tabor";
            b1.Title = "Microsoft .NET XML Web Services";
            b1.ISBN = "0-000-00000-0";

            //ArrayLists are dynamically sized,
            //cool features sorting, remove items
            ArrayList myArrayList = new ArrayList
            {
                car1,
                car2,
                b1
            };

            foreach (Car car in myArrayList)
            {
                Console.WriteLine(car.Make);
            }
            myArrayList.Remove(b1);
            //color antes de la invocación;
            int preColor = (int)Console.ForegroundColor;
            //color antes de la invocación;
            Console.ForegroundColor = ConsoleColor.DarkMagenta;
            myArrayList.Add(car3);
            foreach (Car car in myArrayList)
            {
                Console.WriteLine(car.Make);
            }
            Console.ForegroundColor = (ConsoleColor) preColor;
            //List<Car> myList = new List<Car>();
            //myList.Add(car1);
            //myList.Add(car2);

            //List<T> Esto es equivalente a lo anteior y mucho mas eficiente
            List<Car> myList = new List<Car>
```

Trabajando con colecciones (2)

```
59              //List<T> Esto es equivalente a lo anteior y mucho mas eficiente
60              List<Car> myList = new List<Car>
61              {
62                  car1,
63                  car2
64              };
65              //myList.Add(b1);
66              foreach (Car car in myList)
67              {
68                  Console.WriteLine(car.Model);
69              }
70              // Dictionary<TKey, TValue>
71              Dictionary<string, Car> myDictionary = new Dictionary<string, Car>
72              {
73                  { car1.VIN, car1 },
74                  { car2.VIN, car2 }
75              };
76              Console.WriteLine(myDictionary["B2"].Make);
77              string[] names = { "Bob", "Steve", "Brian", "Chuck" };
78              // Object initializer
79              // No need for a Constructor
80              //Car
81                  car1 = new Car() { Make = "BMW", Model = "750li", VIN = "C3" };
82              //Car
83                  car2 = new Car() { Make = "Toyota", Model = "4Runner", VIN = "D4" }
84              // Collection initializer
85              //List<Car>
86                  myList = new List<Car>() {
87                  new Car { Make = "Oldsmobile", Model = "Cutlas Supreme", VIN = "E5"
88                  new Car { Make = "Nissan", Model = "Altima", VIN = "F6" }
```

```
84              // Collection initializer
85              //List<Car>
86                  myList = new List<Car>() {
87                  new Car { Make = "Oldsmobile", Model = "Cutlas Supreme", VIN = "E5"
88                  new Car { Make = "Nissan", Model = "Altima", VIN = "F6" }
89              };
90              Console.ReadLine();
91          }
92      }
93      class Car
94      {
95          public string VIN { get; set; }
96          public string Make { get; set; }
97          public string Model { get; set; }
98          public int Year { get; set; }
99          public double StickerPrice { get; set; }
100     }
101     class Book
102     {
103         public string Title { get; set; }
104         public string Author { get; set; }
105         public string ISBN { get; set; }
106     }
107 }
```

Entendiendo LINQ (1)

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace _01_Complete
{
    class UnderstandingLINQ
    {
        static void Main(string[] args)
        {
            Car car1 = new Car()
            {
                VIN = "A1",
                Make = "BMW",
                Model = "550i",
                StickerPrice = 55000,
                Year = 2009
            };
            List<Car> myCars = new List<Car>() {
                car1,
                new Car() { VIN="B2", Make="Toyota", Model="4Runner",
                    StickerPrice =35000, Year=2010},
                new Car() { VIN="C3", Make="BMW", Model = "745li",
                    StickerPrice =75000, Year=2008},
                new Car() { VIN="D4", Make="Ford", Model="Escape",
                    StickerPrice =25000, Year=2008},
                new Car() { VIN="E5", Make="BMW", Model="55i",
                    StickerPrice =57000, Year=2010}
            };

            // LINQ query
            var bmws = from car in myCars
                       where car.Make == "BMW"
                       && car.Year == 2010
                       select car;
            var orderedCars = from car in myCars
                              orderby car.Year descending
                              select car;
            // LINQ method
            bmws = myCars.Where(
                p => p.Make == "BMW" && p.Year == 2010);
            orderedCars = myCars.OrderByDescending(p => p.Year);
            var firstBMW = myCars.OrderByDescending(
                p => p.Year).First(p => p.Make == "BMW");
            Console.WriteLine(firstBMW.VIN);
            Console.WriteLine(myCars.TrueForAll(
                p => p.Year > 2007));
            myCars.ForEach(p => p.StickerPrice -= 3000);
            myCars.ForEach(p => Console.WriteLine("{0} {1:C}",
                p.VIN, p.StickerPrice));
            Console.WriteLine(myCars.Exists(
                p => p.Model == "745li"));
            Console.WriteLine(myCars.Sum(p => p.StickerPrice));
            foreach (var car in orderedCars)
            {
                Console.WriteLine("{0} {1}", car.Year,
                    car.Model, car.VIN);
            }
            orderedCars = myCars.OrderByDescending(p => p.Year);
            Console.WriteLine(orderedCars.GetType());
```

Entiendo LINQ (2)

```
60              Console.WriteLine(orderedCars.GetType());
61              bmws = myCars.Where(
62                  p => p.Make == "BMW" && p.Year == 2010);
63              Console.WriteLine(bmws.GetType());
64              var newCars = from car in myCars
65                            where car.Make == "BMW"
66                            && car.Year == 2010
67                            select new { car.Make, car.Model };
68              Console.WriteLine(newCars.GetType());
69              Console.ReadLine();
70          }
71      }
72  }
73
```

Enums y Switchs

```
using System;
using System.Collections.Generic;
// Starting showing the class and de enumerator, then the switch in the metho
namespace _01_Complete
{
    class EnumsAndSwitch
    {
        static void Main(string[] args)
        {
            List<Todo> all = new List<Todo>()
            {
                new Todo { Description = "Task 1",
                    EstimatedHours = 6, Status = Status.Completed },
                new Todo { Description = "Task 2",
                    EstimatedHours = 2, Status = Status.InProgress },
                new Todo { Description = "Task 3",
                    EstimatedHours = 8, Status = Status.NotStarted },
                new Todo { Description = "Task 4",
                    EstimatedHours = 12, Status = Status.Deleted },
                new Todo { Description = "Task 5",
                    EstimatedHours = 6, Status = Status.InProgress },
                new Todo { Description = "Task 6",
                    EstimatedHours = 2, Status = Status.NotStarted },
                new Todo { Description = "Task 7",
                    EstimatedHours = 14, Status = Status.NotStarted },
                new Todo { Description = "Task 8",
                    EstimatedHours = 8, Status = Status.Completed },
                new Todo { Description = "Task 9",
                    EstimatedHours = 8, Status = Status.InProgress },
                new Todo { Description = "Task 10",
                    EstimatedHours = 8, Status = Status.Completed },
                new Todo { Description = "Task 11",
                    EstimatedHours = 4, Status = Status.NotStarted },
                new Todo { Description = "Task 12",
                    EstimatedHours = 10, Status = Status.Completed },
                new Todo { Description = "Task 13",
                    EstimatedHours = 12, Status = Status.Deleted },
                new Todo { Description = "Task 14",
                    EstimatedHours = 6, Status = Status.Completed }
            };
            Console.ForegroundColor = ConsoleColor.DarkRed;
            PrintAssessment(all);
            Console.ReadLine();
        }

        private static void PrintAssessment(List<Todo> allTask)
        {
            foreach (var todo in allTask)
            {
                switch (todo.Status)
                {
                    case Status.NotStarted:
                        Console.ForegroundColor = ConsoleColor.Red;
                        break;
                    case Status.InProgress:
                        Console.ForegroundColor = ConsoleColor.Green;
                        break;
                    case Status.OnHold:
                        Console.ForegroundColor = ConsoleColor.DarkRed;
```

Enum y Switch

```
60                          break;
61                      case Status.Completed:
62                          Console.ForegroundColor = ConsoleColor.Blue;
63                          break;
64                      case Status.Deleted:
65                          Console.ForegroundColor = ConsoleColor.Yellow;
66                          break;
67                      default:
68                          break;
69                  }
70                  Console.WriteLine(todo.Description);
71              }
72          }
73      }
74
75      class Todo
76      {
77          public string Description { get; set; }
78          public int EstimatedHours { get; set; }
79          public Status Status { get; set; }
80      }
81
82      enum Status
83      {
84          NotStarted,
85          InProgress,
86          OnHold,
87          Completed,
88          Deleted
89      }
```

Try Cash y Manejo de Excepciones

```
4   namespace _01_Complete
5   {
6       class HandlingExceptions
7       {
8           static void Main(string[] args)
9           {
10              try
11              {
12                  string content = File.ReadAllText(@"C:\Exer\All\Exampl.txt");
13                  Console.WriteLine(content);
14
15              }
16              catch (FileNotFoundException ex)
17              {
18                  Console.WriteLine("There was a problem!");
19                  Console.WriteLine("Make sure the name of the file is named correctly: Exampl.txt");
20              }
21              catch (DirectoryNotFoundException ex)
22              {
23                  Console.WriteLine("There was a problem!");
24                  Console.WriteLine(@"Make sure the directory C:\Lesson22 exists.");
25              }
26              catch (Exception ex)
27              {
28                  Console.WriteLine("There was a problem!");
29                  Console.WriteLine(ex.Message);
30              }
31              finally
32              {
33                  // Code to finalize
34                  // Setting objects to null
35                  // Closing database connections
36                  Console.WriteLine("Closing application now ...");
37              }
38              Console.ReadLine();
39          }
40      }
41  }
42
```

Eventos y Programación Orientada a Events

```
1   using System;
2   using System.Timers;
3
4   namespace _01_Complete
5   {
6       class TimerExample
7       {
8           static void Main(string[] args)
9           {
10              Timer myTimer = new Timer(2000);
11
12              myTimer.Elapsed += MyTimer_Elapsed;
13              myTimer.Elapsed += MyTimer_Elapsed1;
14
15              myTimer.Start();
16
17              Console.WriteLine("Press enter to remove the red event.");
18              Console.ReadLine();
19
20              myTimer.Elapsed -= MyTimer_Elapsed1;
21
22              Console.ReadLine();
23          }
24
25          private static void MyTimer_Elapsed(object sender, ElapsedEventArgs e)
26          {
27              Console.ForegroundColor = ConsoleColor.Red;
28              Console.WriteLine("Elapsed1: {0:HH:mm:ss.fff}", e.SignalTime);
29          }
30
31          private static void MyTimer_Elapsed1(object sender, ElapsedEventArgs e)
32          {
33              Console.ForegroundColor = ConsoleColor.White;
34              Console.WriteLine("Elapsed: {0:HH:mm:ss.fff}", e.SignalTime);
35          }
36      }
37  }
```

Delegados

```
using System;
namespace _01_Complete
{
    class Ejemplo_Delegados
    {
        public delegate string delegado(string mensaje);
        public delegate double delegado1(int a, int b);
        public static void Main()
        {
            Ejemplo_Delegados o_delegado = new Ejemplo_Delegados();
            delegado del = o_delegado.Mensaje_Delegado;
            string mensajeResultado=string.Format($"ResultadoFinal-{del("llamada")}");
            Console.WriteLine($"{mensajeResultado}-press... cualquier tecla...");
            Console.ReadLine();
            Cafetera cafetera = new Cafetera();
            delegado1 del1 = cafetera.TiempoCopcion;
            double tiempoCopción = del1(5 , 4);
            Console.WriteLine($"{tiempoCopción}-es el tiempo de copción");
            Console.ReadLine();
        }
        public string Mensaje_Delegado(string mensaje)
        {
            mensaje = String.Format($"{mensaje}:desde el delegado");
            Console.WriteLine(mensaje);
            return mensaje;
        }
    }
    class Cafetera
    {
        public double TiempoCopcion(int inicio, int fin)
        {
            return (double) inicio + fin;
        }

    }
}
```

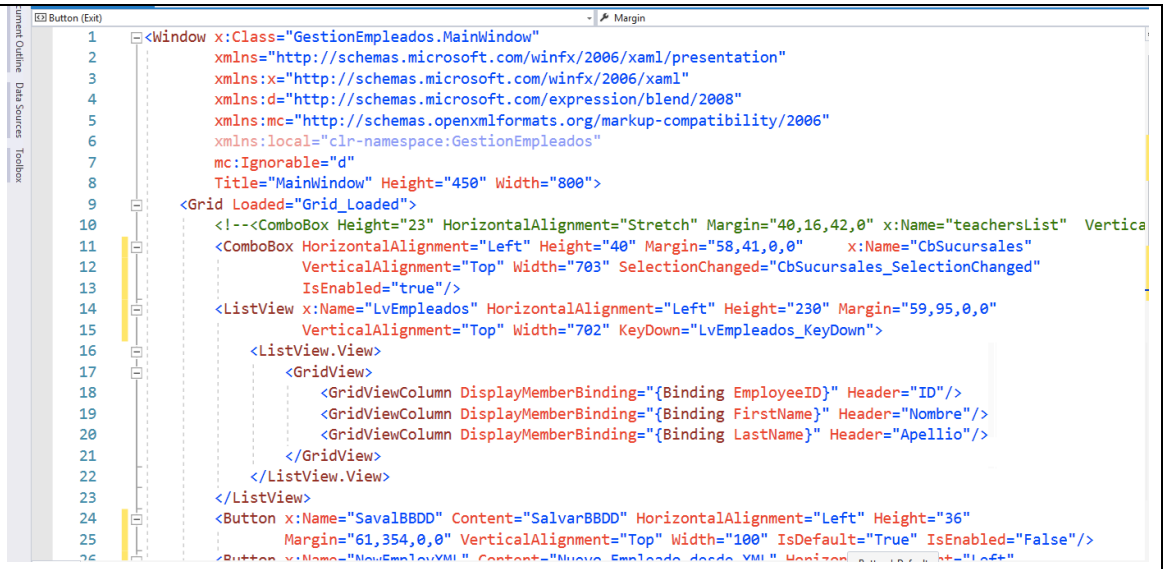
ListView from start

```
namespace ListView
{
    class MyData
    {
        public string ItemName { get; set; }
        public string Description { get; set; }
    }
}
```

```
<Window x:Class="ListView.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ListView"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
        <ListView x:Name="LvListaData" HorizontalAlignment="Left" Height="247"
                  Margin="122,89,0,0" VerticalAlignment="Top" Width="388">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Item name" Width="120"
                                    DisplayMemberBinding="{Binding ItemName}"></GridViewColumn>
                    <GridViewColumn Header="Description" Width="280"
                                    DisplayMemberBinding="{Binding Description}"></GridViewColumn>
                </GridView>
            </ListView.View>
        </ListView>
        <Button Content="Button" HorizontalAlignment="Left" Margin="122,363,0,0"
                VerticalAlignment="Top" Width="75" Click="Exit_ListView"/>

    </Grid>
</Window>
```

```
using System.Collections.Generic;
using System.Windows;
namespace ListView
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            List<MyData> itemData = new List<MyData>();
            itemData.Add(new MyData() { ItemName = "01", Description="Primera descripción" });
            itemData.Add(new MyData() { ItemName = "02", Description = "Primera descripción" });
            itemData.Add(new MyData() { ItemName = "04", Description = "Primera descripción" });
            itemData.Add(new MyData() { ItemName = "05", Description = "Primera descripción" });
            itemData.Add(new MyData() { ItemName = "06", Description = "Primera descripción" });
            itemData.Add(new MyData() { ItemName = "07", Description = "Primera descripción" });
            this.LvListaData.ItemsSource = itemData;
        }
        private void Exit_ListView(object sender, RoutedEventArgs e)
        {
            int k = 1;
            k++;
        }
    }
}
```




```
<Window x:Class="GestionEmpleados.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:GestionEmpleados"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid Loaded="Grid_Loaded">
        <!--<ComboBox Height="23" HorizontalAlignment="Stretch" Margin="40,16,42,0" x:Name="teachersList"  Vertica
        <ComboBox HorizontalAlignment="Left" Height="40" Margin="58,41,0,0"      x:Name="CbSucursales"
                  VerticalAlignment="Top" Width="703" SelectionChanged="CbSucursales_SelectionChanged"
                  IsEnabled="true"/>
        <ListView x:Name="LvEmpleados" HorizontalAlignment="Left" Height="230" Margin="59,95,0,0"
                  VerticalAlignment="Top" Width="702" KeyDown="LvEmpleados_KeyDown">
            <ListView.View>
                <GridView>
                    <GridViewColumn DisplayMemberBinding="{Binding EmployeeID}" Header="ID"/>
                    <GridViewColumn DisplayMemberBinding="{Binding FirstName}" Header="Nombre"/>
                    <GridViewColumn DisplayMemberBinding="{Binding LastName}" Header="Apellio"/>
                </GridView>
            </ListView.View>
        </ListView>
        <Button x:Name="SavalBBDD" Content="SalvarBBDD" HorizontalAlignment="Left" Height="36"
                Margin="61,354,0,0" VerticalAlignment="Top" Width="100" IsDefault="True" IsEnabled="False"/>
        <Button x:Name="NewEmployXML" Content="Nuevo Empleado desde XML" Horizon
```

```
using GestionEmpleados.Data;
using System.Collections;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Data;
using System.Linq;
using System;

namespace GestionEmpleados
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private EmpresaXYZEntities dBContext = null;
        private Branch sucursal = null;
        //private Employee employee = null;
        private IList<Employee> listaEmpleados = null;
        public MainWindow()
        {
            InitializeComponent();
        }

        private void Grid_Loaded(object sender, RoutedEventArgs e)
```

```
        private void Grid_Loaded(object sender, RoutedEventArgs e)
        {
            dBContext = new EmpresaXYZEntities();
            IList<Branch> branch = (from a in dBContext.Branches
                                    orderby a.BranchName
                                    select a).ToList();
            CbSucursales.ItemsSource = branch;
            CbSucursales.DisplayMemberPath = "BranchName";
            CbSucursales.SelectedValuePath = "BranchID";
        }
```

```
        }

        private void CbSucursales_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            sucursal = CbSucursales.SelectedItem as Branch;
            //this.dBContext.Entry<>

            listaEmpleados = (from a in dBContext.Employees
                              where a.Branch == sucursal.BranchID
                              select a
                              ).ToList();

            if (listaEmpleados.Count == 0)
                MessageBox.Show($"La sucursal:{sucursal.BranchName} no tiene empleados", "Alerta",
                    MessageBoxButton.OK, MessageBoxImage.Warning);
            AtualizarListaEmpleados();
            //LvEmpleados.



        }
```

```
        private void AtualizarListaEmpleados()
        {
            LvEmpleados.ItemsSource = null;
            LvEmpleados.ItemsSource = listaEmpleados;
            LvEmpleados.DisplayMemberPath = "FirstName";

        }

        private void LvEmpleados_KeyDown(object sender, System.Windows.Input.KeyEventArgs e)
        {

            switch (e.Key)
            {
                case System.Windows.Input.Key.Enter:
                    Employee empleado = LvEmpleados.SelectedItem as Employee;
                    EditarEmpleado(empleado);
                    break;
                case System.Windows.Input.Key.Insert:
                    InsertNuevoEmpleado();
                    break;
                case System.Windows.Input.Key.Delete:
                    empleado = LvEmpleados.SelectedItem as Employee;
                    BorreEmpleado(empleado);
                    break;
                default:
                    break;
```

```
85
86          private void BorreEmpleado(Employee empleado)
87          {
88              MessageBoxResult response = MessageBox.Show(
89                 String.Format("Remove {0} {1}", empleado.FirstName , empleado.LastName),
90                 "Confirm", MessageBoxButton.YesNo, MessageBoxImage.Question,
91                 MessageBoxResult.No);
92              // If the user clicked Yes, remove the empleado from the database
93              if (response == MessageBoxResult.Yes)
94              {
95                  listaEmpleados.Remove(empleado);
96                  dBContext.Employees.Remove(empleado);
97                  SavalBBDD.IsEnabled = true;
98                  AtualizarListaEmpleados();
99
100             }
101         }
102
103         private void InsertNuevoEmpleado()
104         {
105             // Use the StudentsForm to get the details of the student from the user
106             Empleados sf = new Empleados();
107
108             // Set the title of the form to indicate which class the student will be added to (the class for the cur
109             sf.Title = $"Nuevo empleado para la sucursal:{sucursal.BranchName}";
110
111             // Display the form and get the details of the new student
112             if (sf.ShowDialog().Value)
113             {
114                 // When the user closes the form, retrieve the details of the student from the form
115                 // and use them to create a new Student object
116                 Employee newEmpleado = new Employee();
117                 newEmpleado.FirstName = sf.Nombre_Empleado.Text;
118                 newEmpleado.LastName = String.Empty;
119                 newEmpleado.Branch = Int32.Parse(sf.ID_Sucursal.Text);
120                 DateTime vente = DateTime.Parse("1/1/1998");
121                 newEmpleado.DateOfBirth = vente;
122                 //newEmpleado.DateOfBirth = DateTime.Parse(sf.dateOfBirth.Text);
123
124                 // Assign the new student to the current teacher
125                 sucursal.Employees.Add(newEmpleado);
126
127                 // Add the student to the list displayed on the form
128                 listaEmpleados.Add(newEmpleado);
129                 AtualizarListaEmpleados();
130
131
132                 // Enable saving (changes are not made permanent until they are written back to the database)
133                 SavalBBDD.IsEnabled = true;
134             }
135         }
136
137         private void EditarEmpleado(Employee empleado)
138         {
139             // Use the empleadosForm to display and edit the details of the empleado
140             Empleados em = new Empleados();
141
142             // Set the title of the form and populate the fields on the form with the details of the empleado
143             em.Title = "Edit empleado Details";
144             em.Nombre_Empleado.Text = empleado.FirstName ;
145             em.IDEmpleado.Text =  empleado.EmployeeID.ToString();
146             em.ID_Sucursal.Text = empleado.Branch.ToString();
147             // Display the form
148             if (em.ShowDialog().Value)
149             {
150                 // When the user closes the form, copy the details back to the empleado
151                 empleado.FirstName = em.Nombre_Empleado.Text;
152                 empleado.LastName = String.Empty;
153                 empleado.Branch = Int32.Parse(em.ID_Sucursal.Text);
154                 DateTime vente = DateTime.Parse("1/1/1998") ;
155                 empleado.DateOfBirth = vente;
156                 AtualizarListaEmpleados();
157                 // Enable saving (changes are not made permanent until they are written back to the database)
158                 SavalBBDD.IsEnabled = true;
159             }
160         }
161     }
162 }
```

Empleados.xaml.cs | Empleados.xaml | MainWindow.xaml.cs* | MainWindow.xaml*

GestionEmpleados | GestionEmpleados.Empleados | Empleados()

```
26
27          private void Button_Click(object sender, RoutedEventArgs e)
28          {
29              // Indicate that the data is valid
30              this.DialogResult = true;
31          }
32      }
33  }
```
