

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();
    }
}
```

Decisions

```
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 journey!";
        }
        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();
    }
}
```

Methods

```
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

```

1  using System;
2
3  namespace _01_Complete
4  {
5      #region WhileIteration
6      class WhileIteration
7      {
8          #region Main
9          static void Main(string[] args)
10         {
11             bool displayMenu = true;
12             while (displayMenu)
13             {
14                 displayMenu = MainMenu();
15             }
16         }
17     #endregion
18     [MainMenu]
19     [Statics Methods]
20 }
21 #endregion
22
23 private static bool MainMenu()
24 {
25     Console.Clear();
26     Console.WriteLine("Choose an option");
27     Console.WriteLine("1) Print Numbers");
28     Console.WriteLine("2) Guessing Game");
29     Console.WriteLine("3) Exit");
30     string result = Console.ReadLine();
31     if (result=="1")
32     {
33         PrintNumbers();
34         return true;
35     }
36     else if (result=="2")
37     {
38         GuessingGame();
39         return true;
40     }
41     else if (result=="3")
42     {
43         return false;
44     }
45     else
46     {
47         return true;
48     }
49 }
50
51 private static void GuessingGame()
52 {
53     Console.Clear();
54     Console.WriteLine("Guessing game!!!");
55
56     Random myRandom = new Random();
57     int randomNumber = myRandom.Next(1, 11);
58     int guesses = 0;
59     bool incorrect = true;
60     do
61     {
62         Console.WriteLine("Guest a number between 1 and 10:");
63         string result = Console.ReadLine();
64         guesses++;
65         if (result == randomNumber.ToString())
66         {
67             incorrect = false;
68         }
69         else
70             Console.WriteLine("Wrong!");
71     } while (incorrect);
72     Console.WriteLine($"Correct! It took you {guesses} guesess.");
73     Console.ReadLine();
74 }
75
76 private static void PrintNumbers()
77 {
78     Console.Clear();
79     Console.WriteLine("Print numbers!");
80     Console.Write("Type a number:");
81     int result = int.Parse(Console.ReadLine());
82     int counter = 1;
83     string line = "";
84     while (counter < result + 1 )
85     {
86         Console.Write("{0:000}",counter);
87         counter++;
88         line = (counter == result+1) ? "\n" : "-";
89         Console.Write($"{line}");
90     }
91     Console.Write("Press any key to continue...");
92     Console.ReadLine();
93 }
94 #endregion
95

```

Trabajando con Cadenas de Caracteres

```
9     class WorkingWithStrings
10    {
11        static void Main(string[] args)
12        {
13            string myString = "my \"so called\" life";
14            myString = "What if I need a new \nline caracter";
15            myString = "Go to you c:\\\\ drive";
16            myString = @"Go to you c:\\drive";
17            myString = String.Format("{1}={1}", "first", "second");
18            myString = string.Format("{0:C}", 23403.03);
19            myString = string.Format("{0:N}", 123457890);
20            myString = string.Format("{0:P}", .234);
21            myString = string.Format("Phone Number:{0:(###) ###-###}", 
22                1234567890123);
23            /*string myString = "*/ //this is a test a string with extra spaces ";
24            myString = myString.Substring(6, 14);
25            myString = myString.ToUpper();
26            myString = myString.Replace(" ", "--");
27            myString = myString.Remove(6, 14);
28            myString = String.Format("Original length:{0} - lenght without spaces:{1}", myString.Length, myString.Trim().Length);
29            myString = "";
30            for (int i = 0; i < 100; i++)
31            {
32                //myString += String.Format("-- {0}", i.ToString());
33                myString += "-- " + i.ToString();
34            }
35
36            StringBuilder myString_SB = new StringBuilder();
37            int length = 100;
38            for (int i = 0; i < length; i++)
39            {
40                myString_SB.Append("--");
41                myString_SB.Append(i.ToString());
42            }
43
44            Console.WriteLine($"{myString_SB}");
45            Console.ReadLine();
46        }
47    }
48}
```

Activate

Fechas y Horas

```
8  namespace _01_Complete
9  {
10     class DatesAndTimes
11     {
12         static void Main(string[] args)
13         {
14             DateTime myValue = DateTime.Now;
15             Console.WriteLine(myValue.ToString());
16             int currentForegroundColor = (int)Console.ForegroundColor;
17             Console.ForegroundColor = ConsoleColor.DarkBlue;
18             Console.WriteLine($".ToShortDateString:{myValue.ToShortDateString()} " +
19                             $"\\nToShortTimeString:{myValue.ToShortTimeString()}");
20             Console.ForegroundColor = (ConsoleColor)currentForegroundColor;
21             Console.WriteLine($"ToString:{myValue.ToString()}" +
22                             $"\\nToLongDateString:{myValue.ToLongDateString()}" +
23                             $"\\nToLongTimeString:{myValue.ToLongTimeString()}");
24             Console.WriteLine($"AddDays:{myValue.AddDays(3).ToString()}" +
25                             $"\\nAddHours:{myValue.AddHours(3)}-");
26             Console.ForegroundColor = ConsoleColor.DarkRed;
27             Console.WriteLine($"Month:{myValue.Month}");
28             DateTime myBirthday = new DateTime(1968, 12, 31);
29             Console.WriteLine(myBirthday.ToShortDateString());
30             //Format of DateTime.Parse("dia/mes/año");
31             myBirthday = DateTime.Parse("31/12/1968");
32             //Represent a interval of time
33             TimeSpan myAge = DateTime.Now.Subtract(myBirthday);
34             Console.WriteLine(myAge.TotalDays);
35             string dateInput = "Dec 21, 2018"; // Esto no funciona si la cadena es "Dic 21, 2018"
36             DateTime parsedDate = DateTime.Parse(dateInput);
37             Console.WriteLine("{0}", parsedDate);
38             CultureInfo myCultureInfo = new CultureInfo("es-ES");
39             string mySpanishDate = "21 Diciembre 2018";
40             DateTime myDateTime = DateTime.Parse(mySpanishDate, myCultureInfo);
41             Console.WriteLine("{0}", myDateTime.ToShortDateString());
42             Console.ReadLine();
43         }
44     }
45 }
```

Classes

```
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
30
31           List<Car_SC> Cars = new List<Car_SC> { myCar,myCar1,myCar2};
32           foreach (Car_SC car in Cars)
33           {
34               Console.WriteLine($"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   // Remember "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_DL myCar = new Car_DL();
14             myCar.Make = "Oldsmobile";
15             myCar.Model = "Cutlas Supreme";
16             myCar.Year = 1986;
17             myCar.Color = "Silver";
18             Car_DL.MyMethod(); //Metodo statico que no necesita ser
19             //instanciado en un objeto para ser invocado
20             myCar = new Car_DL("Oldsmobile", "Cutlas Supreme", 1986,
21             "Silver");
22
23             Car_DL 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 exception of null object set
33             //Console.WriteLine("{0}-{1}-{2}-{3}", myOtherCar.Make,
34             //myOtherCar.Model, myOtherCar.Year, myOtherCar.Color);
35             Console.ReadLine();
36         }
37     class Car_DL
38     {
39         public Car_DL(string model, string make, int year, string color)
40         {
41             Model = model;
42             Make = make;
43             Year = year;
44             Color = color;
45         }
46         public Car_DL()
47         {
48             Make = "Nissan";
49         }
50         public string Model { get; set; }
51         public string Make { get; set; }
52         public int Year { get; set; }
53         public string Color { get; set; }
54         public static void MyMethod()
55         {
56             Console.WriteLine("Called the static MyMethod");
57             //Console.WriteLine(Car.Make); this is illegal!!!
58         }
59     }
60 }
61 }
```

Alcance de variables y clases	<pre> 1 using System; 2 namespace _01_Complete 3 { 4 class UndeUnderstandingScopes 5 { 6 public string k; 7 static void Main(string[] args) 8 { 9 string j = ""; 10 UndeUnderstandingScopes program = new UndeUnderstandingScopes(); 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 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 </pre>
Assemblies y NameSpaces	<pre> 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 </pre>

```
1  using ScrapeLibrary;      // Show how to do this project
2  // Don't forget to put "using ScrapeLibrary here!!!"
3  using System;
4  // Remember Add New in Solution '01-Complete
5  // Kind of project: Class Library (.Net Framework)
6  // Type: C# A project for creating a C# class library(.dll)
7  namespace _01_Complete
8  {
9
10    // Show all students The Reference in Solution Explorer and deadlight teh
11    // assemblies with referentiation in the projects.
12    class MyClient
13    {
14      static void Main(string[] args)
15      {
16        Scrape myScrape = new Scrape();
17        string value = myScrape.ScrapeWebpage("http://msdn.microsoft.com");
18        Console.WriteLine(value);
19        Console.ReadLine();
20      }
21    }
22    // Show all students the other project instead 01-Complete: ScrapeLibrary
23  }
24
25  using System.IO;
26  using System.Net;
27
28  namespace ScrapeLibrary
29  {
30    public class Scrape
31    {
32      public string ScrapeWebpage(string url)
33      {
34        return GetWebpage(url);
35      }
36
37      public string ScrapeWebpage(string url, string filepath)
38      {
39        string reply = GetWebpage(url);
40
41        File.WriteAllText(filepath, reply);
42        return reply;
43      }
44
45      private string GetWebpage(string url)
46      {
47        WebClient client = new WebClient();
48        string content = client.DownloadString(url);
49        content += "THAT'S ALL FOLKS!!!";
50        return content;
51      }
52    }
53  }
```

Trabajando con Colecciones (I)

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  namespace _01_Complete
5  {
6      class WorkingWithCollections
7      {
8          static void Main(string[] args)
9          {
10             Car car1 = new Car();
11             car1.Make = "Oldsmobile";
12             car1.Model = "Cutlas Supreme";
13             car1.VIN = "A1";
14             Car car3 = new Car
15             {
16                 Make = "GMC",
17                 Model = "Grand Voyager",
18                 VIN = "C1"
19             };
20             Car car2 = new Car
21             {
22                 Make = "Geo",
23                 Model = "Prism",
24                 VIN = "B2"
25             };
26             Book b1 = new Book();
27             b1.Author = "Robert Tabor";
28             b1.Title = "Microsoft .NET XML Web Services";
29             b1.ISBN = "0-000-00000-0";
30
31
32
33             ArrayList myArrayList = new ArrayList
34             {
35                 car1,
36                 car2,
37                 b1
38             };
39
40             foreach (Car car in myArrayList)
41             {
42                 Console.WriteLine(car.Make);
43             }
44             myArrayList.Remove(b1);
45             //color antes de la invocación;
46             int preColor = (int)Console.ForegroundColor;
47             //color antes de la invocación;
48             Console.ForegroundColor = ConsoleColor.DarkMagenta;
49             myArrayList.Add(car3);
50             foreach (Car car in myArrayList)
51             {
52                 Console.WriteLine(car.Make);
53             }
54             Console.ForegroundColor = (ConsoleColor) preColor;
55             //List<Car> myList = new List<Car>();
56             //myList.Add(car1);
57             //myList.Add(car2);
58
59             //List<T> Esto es equivalente a lo anterior y mucho mas eficiente
60             List<Car> myList = new List<Car>
```

Trabajando con colecciones (2)

```
59         //List<T> Esto es equivalente a lo anterior y mucho mas eficiente
60         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             {
88                 new Car { Make = "Oldsmobile", Model = "Cutlas Supreme", VIN = "E5" }
89                 new Car { Make = "Nissan", Model = "Altima", VIN = "F6" }
90             };
91             // Collection initializer
92             //List<Car>
93             myList = new List<Car>()
94                 {
95                     new Car { Make = "Oldsmobile", Model = "Cutlas Supreme", VIN = "E5" }
96                     new Car { Make = "Nissan", Model = "Altima", VIN = "F6" }
97                 };
98                 Console.ReadLine();
99             }
100         class Car
101         {
102             public string VIN { get; set; }
103             public string Make { get; set; }
104             public string Model { get; set; }
105             public int Year { get; set; }
106             public double StickerPrice { get; set; }
107         }
108         class Book
109         {
110             public string Title { get; set; }
111             public string Author { get; set; }
112             public string ISBN { get; set; }
113         }
114     }
```

Entendiendo LINQ (1)

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4
5  namespace _01_Complete
6  {
7      class UnderstandingLINQ
8      {
9          static void Main(string[] args)
10         {
11             Car car1 = new Car()
12             {
13                 VIN = "A1",
14                 Make = "BMW",
15                 Model = "550i",
16                 StickerPrice = 55000,
17                 Year = 2009
18             };
19             List<Car> myCars = new List<Car>()
20             {
21                 car1,
22                 new Car() { VIN="B2", Make="Toyota", Model="4Runner",
23                             StickerPrice =35000, Year=2010},
24                 new Car() { VIN="C3", Make="BMW", Model = "745li",
25                             StickerPrice =75000, Year=2008},
26                 new Car() { VIN="D4", Make="Ford", Model="Escape",
27                             StickerPrice =25000, Year=2008},
28                 new Car() { VIN="E5", Make="BMW", Model="55i",
29                             StickerPrice =57000, Year=2010}
30             };
31
32             // LINQ query
33             var bmws = from car in myCars
34                         where car.Make == "BMW"
35                         && car.Year == 2010
36                         select car;
37             var orderedCars = from car in myCars
38                             orderby car.Year descending
39                             select car;
40
41             // LINQ method
42             bmws = myCars.Where(
43                 p => p.Make == "BMW" && p.Year == 2010);
44             orderedCars = myCars.OrderByDescending(p => p.Year);
45             var firstBMW = myCars.OrderByDescending(
46                 p => p.Year).First(p => p.Make == "BMW");
47             Console.WriteLine(firstBMW.VIN);
48             Console.WriteLine(myCars.TrueForAll(
49                 p => p.Year > 2007));
50             myCars.ForEach(p => p.StickerPrice -= 3000);
51             myCars.ForEach(p => Console.WriteLine("{0} {1:C}",
52                 p.VIN, p.StickerPrice));
53             Console.WriteLine(myCars.Exists(
54                 p => p.Model == "745li"));
55             Console.WriteLine(myCars.Sum(p => p.StickerPrice));
56             foreach (var car in orderedCars)
57             {
58                 Console.WriteLine("{0} {1}", car.Year,
59                                 car.Model, car.VIN);
60             }
61             orderedCars = myCars.OrderByDescending(p => p.Year);
62             Console.WriteLine(orderedCars.GetType());
63         }
64     }
65 }
```

Entiendo LINQ (2)

```
60      |
61      |
62      |
63      |
64      |
65      |
66      |
67      |
68      |
69      |
70      |
71      |
72      |
73      }
```

```
Console.WriteLine(orderedCars.GetType());
bmws = myCars.Where(
    p => p.Make == "BMW" && p.Year == 2010);
Console.WriteLine(bmws.GetType());
var newCars = from car in myCars
    where car.Make == "BMW"
    && car.Year == 2010
    select new { car.Make, car.Model };
Console.WriteLine(newCars.GetType());
Console.ReadLine();
```

Enums y Switchs

```
1  using System;
2  using System.Collections.Generic;
3  // Starting showing the class and de enumerator, then the switch in the method
4  namespace _01_Complete
5  {
6      class EnumsAndSwitch
7      {
8          static void Main(string[] args)
9          {
10             List<Todo> all = new List<Todo>()
11             {
12                 new Todo { Description = "Task 1",
13                     EstimatedHours = 6, Status = Status.Completed },
14                 new Todo { Description = "Task 2",
15                     EstimatedHours = 2, Status = Status.InProgress },
16                 new Todo { Description = "Task 3",
17                     EstimatedHours = 8, Status = Status.NotStarted },
18                 new Todo { Description = "Task 4",
19                     EstimatedHours = 12, Status = Status.Deleted },
20                 new Todo { Description = "Task 5",
21                     EstimatedHours = 6, Status = Status.InProgress },
22                 new Todo { Description = "Task 6",
23                     EstimatedHours = 2, Status = Status.NotStarted },
24                 new Todo { Description = "Task 7",
25                     EstimatedHours = 14, Status = Status.NotStarted },
26                 new Todo { Description = "Task 8",
27                     EstimatedHours = 8, Status = Status.Completed },
28                 new Todo { Description = "Task 9",
29                     EstimatedHours = 8, Status = Status.InProgress },
30                 new Todo { Description = "Task 10",
31                     EstimatedHours = 8, Status = Status.Completed },
32                 new Todo { Description = "Task 11",
33                     EstimatedHours = 4, Status = Status.NotStarted },
34                 new Todo { Description = "Task 12",
35                     EstimatedHours = 10, Status = Status.Completed },
36                 new Todo { Description = "Task 13",
37                     EstimatedHours = 12, Status = Status.Deleted },
38                 new Todo { Description = "Task 14",
39                     EstimatedHours = 6, Status = Status.Completed }
40             };
41             Console.ForegroundColor = ConsoleColor.DarkRed;
42             PrintAssessment(all);
43             Console.ReadLine();
44         }
45
46         private static void PrintAssessment(List<Todo> allTask)
47         {
48             foreach (var todo in allTask)
49             {
50                 switch (todo.Status)
51                 {
52                     case Status.NotStarted:
53                         Console.ForegroundColor = ConsoleColor.Red;
54                         break;
55                     case Status.InProgress:
56                         Console.ForegroundColor = ConsoleColor.Green;
57                         break;
58                     case Status.OnHold:
59                         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  }
 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 Catch 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          }
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

```
1  using System;
2  namespace _01_Complete
3  {
4      class Ejemplo_Delegados
5      {
6          public delegate string delegado(string mensaje);
7          public delegate double delegado1(int a, int b);
8          public static void Main()
9          {
10             Ejemplo_Delegados o_delegado = new Ejemplo_Delegados();
11             delegado del = o_delegado.Mensaje_Delegado;
12             string mensajeResultado=string.Format($"ResultadoFinal-{del("llamada")}");
13             Console.WriteLine($"{mensajeResultado}-press... cualquier tecla...");
14             Console.ReadLine();
15             Cafetera cafetera = new Cafetera();
16             delegado1 del1 = cafetera.TiempoCpcion;
17             double tiempoCpcion = del1(5 , 4);
18             Console.WriteLine($"{tiempoCpcion}-es el tiempo de copción");
19             Console.ReadLine();
20         }
21         public string Mensaje_Delegado(string mensaje)
22         {
23             mensaje = String.Format($"{mensaje}:desde el delegado");
24             Console.WriteLine(mensaje);
25             return mensaje;
26         }
27     }
28     class Cafetera
29     {
30         public double TiempoCpcion(int inicio, int fin)
31         {
32             return (double) inicio + fin;
33         }
34     }
35 }
36 }
```

ListView from start

```

7   namespace ListView
8   {
9     class MyData
10    {
11      public string ItemName { get; set; }
12      public string Description { get; set; }
13    }
14  }
15  <Window x:Class="ListView.MainWindow"
16    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
17    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
18    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
19    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
20    xmlns:local="clr-namespace:ListView"
21    mc:Ignorable="d"
22    Title="MainWindow" Height="450" Width="800"
23  </Window>
24  <Grid>
25    <ListView x:Name="LvListaData" HorizontalAlignment="Left" Height="247"
26      Margin="122,89,0,0" VerticalAlignment="Top" Width="388">
27        <ListView.View>
28          <GridView>
29            <GridViewColumn Header="Item name" Width="120"
30              DisplayMemberBinding="{Binding ItemName}"></GridViewColumn>
31            <GridViewColumn Header="Description" Width="280"
32              DisplayMemberBinding="{Binding Description}"></GridViewColumn>
33          </GridView>
34        </ListView.View>
35      </ListView>
36      <Button Content="Button" HorizontalAlignment="Left" Margin="122,363,0,0"
37        VerticalAlignment="Top" Width="75" Click="Exit_ListView"/>
38    </Grid>
39  </Window>
40
41  using System.Collections.Generic;
42  using System.Windows;
43  namespace ListView
44  {
45    public partial class MainWindow : Window
46    {
47      public MainWindow()
48      {
49        InitializeComponent();
50        List<MyData> itemData = new List<MyData>();
51        itemData.Add(new MyData() { ItemName = "01", Description="Primera descripción" } );
52        itemData.Add(new MyData() { ItemName = "02", Description = "Primera descripción" } );
53        itemData.Add(new MyData() { ItemName = "04", Description = "Primera descripción" } );
54        itemData.Add(new MyData() { ItemName = "05", Description = "Primera descripción" } );
55        itemData.Add(new MyData() { ItemName = "06", Description = "Primera descripción" } );
56        itemData.Add(new MyData() { ItemName = "07", Description = "Primera descripción" } );
57        this.LvListaData.ItemsSource = itemData;
58      }
59      private void Exit_ListView(object sender, RoutedEventArgs e)
60      {
61        int k = 1;
62        k++;
63      }
64    }
65  }

```



The screenshot shows a Windows application window titled "GestionEmpleados.MainWindow". The window contains a grid layout with several controls: a ComboBox, a ListView with a GridView, and two buttons. The XAML code for this window is displayed in the code editor:

```
1 <Window x:Class="GestionEmpleados.MainWindow"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     xmlns:local="clr-namespace:GestionEmpleados"
7     mc:Ignorable="d"
8     Title="MainWindow" Height="450" Width="800">
9     <Grid Loaded="Grid_Loaded">
10        <!--<ComboBox Height="23" HorizontalAlignment="Stretch" Margin="40,16,42,0" x:Name="teachersList" VerticalAlignment="Top"/>
11        <ComboBox HorizontalAlignment="Left" Height="40" Margin="58,41,0,0" x:Name="CbSucursales" VerticalAlignment="Top" Width="703" SelectionChanged="CbSucursales_SelectionChanged" IsEnabled="true"/>
12        <ListView x:Name="LvEmpleados" HorizontalAlignment="Left" Height="230" Margin="59,95,0,0" VerticalAlignment="Top" Width="702" KeyDown="LvEmpleados_KeyDown">
13            <ListView.View>
14                <GridView>
15                    <GridViewColumn DisplayMemberBinding="{Binding EmployeeID}" Header="ID"/>
16                    <GridViewColumn DisplayMemberBinding="{Binding FirstName}" Header="Nombre"/>
17                    <GridViewColumn DisplayMemberBinding="{Binding LastName}" Header="Apellido"/>
18                </GridView>
19            </ListView.View>
20        </ListView>
21        <Button x:Name="SavalBBDD" Content="SalvarBBDD" HorizontalAlignment="Left" Height="36" Margin="61,354,0,0" VerticalAlignment="Top" Width="100" IsDefault="True" IsEnabled="False"/>
22        <Button x:Name="NuevoEmpleoVMI" Content="Nuevo Empleado desde VMI" HorizontalAlignment="Left" Height="36" Margin="11,354,0,0" VerticalAlignment="Top" Width="100" IsDefault="False" IsEnabled="False"/>
23    </Grid>
24
```

```

  MainWindow.xaml.cs  MainWindow.xaml
  GestionEmpleados
  1  using GestionEmpleados.Data;
  2  using System.Collections;
  3  using System.Collections.Generic;
  4  using System.Windows;
  5  using System.Windows.Controls;
  6  using System.Data;
  7  using System.Linq;
  8  using System;
  9
 10 namespace GestionEmpleados
 11 {
 12     /// <summary>
 13     /// Interaction logic for MainWindow.xaml
 14     /// </summary>
 15     public partial class MainWindow : Window
 16     {
 17         private EmpresaXYZEntities dBContext = null;
 18         private Branch sucursal = null;
 19         //private Employee employee = null;
 20         private IList<Employee> listaEmpleados = null;
 21         public MainWindow()
 22         {
 23             InitializeComponent();
 24         }
 25
 26         private void Grid_Loaded(object sender, RoutedEventArgs e)
 27         {
 28             dBContext = new EmpresaXYZEntities();
 29             IList<Branch> branch = (from a in dBContext.Branches
 30                                     orderby a.BranchName
 31                                     select a).ToList();
 32             CbSucursales.ItemsSource = branch;
 33             CbSucursales.DisplayMemberPath = "BranchName";
 34             CbSucursales.SelectedValuePath = "BranchID";
 35         }
 36
 37         private void CbSucursales_SelectionChanged(object sender, SelectionChangedEventArgs e)
 38         {
 39             sucursal = CbSucursales.SelectedItem as Branch;
 40             //this.dBContext.Entry<>
 41
 42             listaEmpleados = (from a in dBContext.Employees
 43                               where a.Branch == sucursal.BranchID
 44                               select a
 45                           ).ToList();
 46
 47             if (listaEmpleados.Count == 0)
 48                 MessageBox.Show($"La sucursal:{sucursal.BranchName} no tiene empleados", "Alerta",
 49                             MessageBoxButton.OK, MessageBoxImage.Warning);
 50             AtualizarListaEmpleados();
 51             //LvEmpleados.
 52
 53
 54         }
 55
 56         private void AtualizarListaEmpleados()
 57         {
 58             LvEmpleados.ItemsSource = null;
 59             LvEmpleados.ItemsSource = listaEmpleados;
 60             LvEmpleados.DisplayMemberPath = "FirstName";
 61
 62         }
 63
 64         private void LvEmpleados_KeyDown(object sender, System.Windows.Input.KeyEventArgs e)
 65         {
 66
 67             switch (e.Key)
 68             {
 69                 case System.Windows.Input.Key.Enter:
 70                     Employee empleado = LvEmpleados.SelectedItem as Employee;
 71                     EditarEmpleado(empleado);
 72                     break;
 73                 case System.Windows.Input.Key.Insert:
 74                     InsertNuevoEmpleado();
 75                     break;
 76                 case System.Windows.Input.Key.Delete:
 77                     empleado = LvEmpleados.SelectedItem as Employee;
 78                     BorreEmpleado(empleado);
 79                     break;
 80                 default:
 81                     break;
 82             }
 83         }
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 12510
 12511
 12512
 12513
 12514
 12515
 12516
 12517
 12518
 12519
 12520
 12521
 12522
 12523
 12524
 12525
 12526
 12527
 12528
 12529
 12530
 12531
 12532
 12533
 12534
 12535
 12536
 12537
 12538
 12539
 125310
 125311
 125312
 125313
 125314
 125315
 125316
 125317
 125318
 125319
 125320
 125321
 125322
 125323
 125324
 125325
 125326
 125327
 125328
 125329
 125330
 125331
 125332
 125333
 125334
 125335
 125336
 125337
 125338
 125339
 125340
 125341
 125342
 125343
 125344
 125345
 125346
 125347
 125348
 125349
 125350
 125351
 125352
 125353
 125354
 125355
 125356
 125357
 125358
 125359
 125360
 125361
 125362
 125363
 125364
 125365
 125366
 125367
 125368
 125369
 125370
 125371
 125372
 125373
 125374
 125375
 125376
 125377
 125378
 125379
 125380
 125381
 125382
 125383
 125384
 125385
 125386
 125387
 125388
 125389
 125390
 125391
 125392
 125393
 125394
 125395
 125396
 125397
 125398
 125399
 1253100
 1253101
 1253102
 1253103
 1253104
 1253105
 1253106
 1253107
 1253108
 1253109
 1253110
 1253111
 1253112
 1253113
 1253114
 1253115
 1253116
 1253117
 1253118
 1253119
 1253120
 1253121
 1253122
 1253123
 1253124
 1253125
 1253126
 1253127
 1253128
 1253129
 1253130
 1253131
 1253132
 1253133
 1253134
 1253135
 1253136
 1253137
 1253138
 1253139
 1253140
 1253141
 1253142
 1253143
 1253144
 1253145
 1253146
 1253147
 1253148
 1253149
 1253150
 1253151
 1253152
 1253153
 1253154
 1253155
 1253156
 1253157
 1253158
 1253159
 12531510
 12531511
 12531512
 12531513
 12531514
 12531515
 12531516
 12531517
 12531518
 12531519
 12531520
 12531521
 12531522
 12531523
 12531524
 12531525
 12531526
 12531527
 12531528
 12531529
 12531530
 12531531
 12531532
 12531533
 12531534
 12531535
 12531536
 12531537
 12531538
 12531539
 125315310
 125315311
 125315312
 125315313
 125315314
 125315315
 125315316
 125315317
 125315318
 125315319
 125315320
 125315321
 125315322
 125315323
 125315324
 125315325
 125315326
 125315327
 125315328
 125315329
 125315330
 125315331
 125315332
 125315333
 125315334
 125315335
 125315336
 125315337
 125315338
 125315339
 125315340
 125315341
 125315342
 125315343
 125315344
 125315345
 125315346
 125315347
 125315348
 125315349
 125315350
 125315351
 125315352
 125315353
 125315354
 125315355
 125315356
 125315357
 125315358
 125315359
 125315360
 125315361
 125315362
 125315363
 125315364
 125315365
 125315366
 125315367
 125315368
 125315369
 125315370
 125315371
 125315372
 125315373
 125315374
 125315375
 125315376
 125315377
 125315378
 125315379
 125315380
 125315381
 125315382
 125315383
 125315384
 125315385
 125315386
 125315387
 125315388
 125315389
 125315390
 125315391
 125315392
 125315393
 125315394
 125315395
 125315396
 125315397
 125315398
 125315399
 1253153100
 1253153101
 1253153102
 1253153103
 1253153104
 1253153105
 1253153106
 1253153107
 1253153108
 1253153109
 1253153110
 1253153111
 1253153112
 1253153113
 1253153114
 1253153115
 1253153116
 1253153117
 1253153118
 1253153119
 1253153120
 1253153121
 1253153122
 1253153123
 1253153124
 1253153125
 1253153126
 1253153127
 1253153128
 1253153129
 1253153130
 1253153131
 1253153132
 1253153133
 1253153134
 1253153135
 1253153136
 1253153137
 1253153138
 1253153139
 1253153140
 1253153141
 1253153142
 1253153143
 1253153144
 1253153145
 1253153146
 1253153147
 1253153148
 1253153149
 1253153150
 1253153151
 1253153152
 1253153153
 1253153154
 1253153155
 1253153156
 1253153157
 1253153158
 1253153159
 1253153160
 1253153161
 1253153162
 1253153163
 1253153164
 1253153165
 1253153166
 1253153167
 1253153168
 1253153169
 1253153170
 1253153171
 1253153172
 1253153173
 1253153174
 1253153175
 1253153176
 1253153177
 1253153178
 1253153179
 1253153180
 1253153181
 1253153182
 1253153183
 1253153184
 1253153185
 1253153186
 1253153187
 1253153188
 1253153189
 1253153190
 1253153191
 1253153192
 1253153193
 1253153194
 1253153195
 1253153196
 1253153197
 1253153198
 1253153199
 1253153200
 1253153201
 1253153202
 1253153203
 1253153204
 1253153205
 1253153206
 1253153207
 1253153208
 1253153209
 1253153210
 1253153211
 1253153212
 1253153213
```

```

 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

```

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

```

